KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Optimisations and applications of fully homomorphic encryption

Jiayi Kang

Supervisors:
Prof. dr. ir. Frederik Vercauteren
Prof. dr. Nigel P. Smart
Dr. Ilia Iliashenko

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

November 2025

Optimisations and applications of fully homomorphic encryption

Jiayi KANG

Examination committee:
Prof. em. dr. ir. Jean-Pierre Celis, chair
Prof. dr. ir. Frederik Vercauteren, supervisor
Prof. dr. Nigel P. Smart, supervisor
Dr. Ilia Iliashenko, supervisor
Prof. dr. Wim Veys
Dr. Wouter Castryck
Prof. dr. Jean-Sebastien Coron
(University of Luxembourg)
Prof. dr. Eleftheria Makri
(Leiden University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

November 2025

Popularized Abstract

In the modern digital world, individuals and organisations increasingly rely on cloud services to store and process their data. While cloud computing provides convenient access to powerful resources, it also raises serious privacy concerns. Since data is transmitted to and handled by external servers, they inevitably gain access to potentially sensitive client information. In practice, this may include medical records, financial transactions, or personal data, whose exposure creates risks of inadvertent leakage and data misuse. Can we benefit from cloud computing without sacrificing privacy?

An affirmative answer is provided by fully homomorphic encryption (FHE), a cryptographic technology that allows computations on encrypted data. With FHE, a cloud server can perform the required operations without ever seeing the underlying information. The computation results are returned in an encrypted form to the client, who can decrypt it using a secret key. In this way, client data remains private throughout the entire process.

Despite its strong privacy guarantees, FHE faces two main challenges. The first challenge is *efficiency*, since computing on encrypted data is much slower than computing on the original data. The second challenge is *integrity*, since the client may not be convinced that the server performs the correct computation.

In this thesis, we address these challenges in three aspects. First, we optimise oblivious algorithms, the basic building blocks for FHE computations, thereby accelerating a range of FHE procedures and applications. Second, we build practical applications of FHE, including privacy-preserving machine learning, secure analysis of genetic data, and secure generation of cryptographic proofs. Finally, we introduce blind proofs, a framework that ensures the integrity of outsourced computations. We further show that this approach achieves practical performance.

Gepopulariseerde Samenvatting

In de moderne digitale wereld vertrouwen individuen en organisaties steeds vaker op clouddiensten om hun gegevens op te slaan en te verwerken. Hoewel cloudberekening gemakkelijke toegang biedt tot krachtige computers, roept het ook ernstige privacybezorgdheden op. Omdat gegevens worden verzonden naar en verwerkt door externe servers, krijgen deze onvermijdelijk toegang tot mogelijk gevoelige informatie van cliënten. In de praktijk kan dit medische dossiers, financiële transacties of persoonlijke gegevens omvatten, waarvan de blootstelling risico's creëert op onbedoeld lekken en misbruik van gegevens. Kunnen we profiteren van cloudberekening zonder onze privacy op te offeren?

Een positief antwoord wordt geboden door volledig homomorfe encryptie (FHE), een cryptografische technologie die berekeningen op versleutelde gegevens mogelijk maakt. Met FHE kan een cloudserver de vereiste bewerkingen uitvoeren zonder ooit de onderliggende informatie te zien. De resultaten van de berekeningen worden in versleutelde vorm teruggestuurd naar de cliënt, die deze kan ontsleutelen met een geheime sleutel. Op deze manier blijft de privacy van de cliëntgegevens gedurende het hele proces gewaarborgd.

Ondanks de sterke privacygaranties wordt FHE geconfronteerd met twee belangrijke uitdagingen. De eerste uitdaging is *efficiëntie*, aangezien berekeningen op versleutelde gegevens veel trager zijn dan berekeningen op de oorspronkelijke gegevens. De tweede uitdaging is *integriteit*, omdat de cliënt er mogelijk niet van overtuigd is dat de server de juiste berekening uitvoert.

In dit proefschrift pakken wij deze uitdagingen op drie vlakken aan. Ten eerste optimaliseren wij oblivious algoritms, de fundamentele bouwstenen voor FHE-berekeningen, waardoor een reeks FHE-procedures en -toepassingen wordt versneld. Ten tweede ontwikkelen wij praktische toepassingen van FHE, waaronder privacybeschermende machine learning, veilige analyse van genetische gegevens en veilige generatie van cryptografische bewijzen. Tot slot introduceren

wij blind proofs, een raamwerk dat de integriteit van uitbestede berekeningen waarborgt. Wij tonen bovendien aan dat deze benadering in de praktijk efficiënte prestaties behaalt.

Abstract

In today's digital age, cloud storage and computing have become indispensable. Resource-constrained clients such as individuals and small organisations increasingly rely on powerful servers to store, manage and process their data. However, outsourcing data to external servers leads to significant privacy concerns, particularly when dealing with sensitive information such as medical records, financial transactions, or personal data.

Fully homomorphic encryption (FHE) is a cryptographic technique that allows computation over encrypted data. In secure outsourcing with FHE, a client sends encrypted data to a server, which can perform requested computations without accessing the original data. The server returns the resulting ciphertexts, which the client can decrypt to obtain the final output.

Despite its strong privacy guarantees, the practical adoption of FHE is limited by two main challenges: *efficiency*, which arises from the substantial performance overhead of FHE; and *integrity*, which stems from the lack of mechanisms to verify the correctness of the outsourced computation.

In this thesis, we contribute to addressing these challenges in three aspects.

First, we optimise oblivious algorithms for use in FHE, achieving improvements in key performance metrics and accelerating both bootstrapping and a range of applications.

Second, we build efficient privacy-preserving information systems based on FHE. These include (i) two private machine learning protocols, the k-nearest neighbour algorithm and decision tree evaluation, (ii) SQUID, a secure system for storing and analysing genotype-phenotype data, and (iii) a protocol for securely delegating zero-knowledge proof generation.

Third, we construct verifiable secure delegation of computation through FHE techniques. We provide the notion of blind proofs to provide integrity guarantees and demonstrate its practicality using blind zkSNARKs, a concrete instantiation of blind proofs.

Beknopte samenvatting

In het digitale tijdperk van vandaag zijn cloudopslag en -berekening onmisbaar geworden. Cliënten met beperkte rekenkracht en opslagcapaciteit, zoals individuen en kleine organisaties, vertrouwen steeds meer op krachtige servers om hun gegevens op te slaan, te beheren en te verwerken. Het uitbesteden van gegevens aan externe servers leidt echter tot aanzienlijke privacybezorgdheden, vooral bij het omgaan met gevoelige informatie zoals medische dossiers, financiële transacties of persoonlijke gegevens.

Volledig homomorfe encryptie (FHE) is een cryptografische techniek die berekeningen op versleutelde gegevens mogelijk maakt. Bij veilig uitbesteden met FHE stuurt een cliënt versleutelde gegevens naar een server, die de gevraagde berekeningen kan uitvoeren zonder toegang te hebben tot de oorspronkelijke gegevens. De server stuurt de resulterende cijferteksten terug, die de cliënt kan ontsleutelen om de uiteindelijke uitvoer te verkrijgen.

Ondanks de sterke privacygaranties wordt de praktische adoptie van FHE beperkt door twee belangrijke uitdagingen: *efficiëntie*, voortkomend uit de aanzienlijke overhead van FHE; en *integriteit*, voortvloeiend uit het gebrek aan mechanismen om de correctheid van de uitbestede berekening te verifiëren.

In dit proefschrift dragen wij bij aan het aanpakken van deze uitdagingen op drie vlakken.

Ten eerste optimaliseren wij oblivious algorithms voor gebruik in FHE, waarmee verbeteringen worden bereikt in belangrijke prestatie-indicatoren en zowel bootstrapping als een reeks toepassingen worden versneld.

Ten tweede bouwen wij efficiënte privacybeschermende informatiesystemen gebaseerd op FHE. Deze omvatten (i) twee private machine learning-protocollen, het k-nearest neighbour algoritme en decision tree-evaluatie, (ii) SQUID, een veilig systeem voor het opslaan en analyseren van genotype-fenotypegegevens, en (iii) een protocol voor het veilig delegeren van de generatie van zero-knowledge-bewijzen.

Ten derde construeren wij verifieerbare veilige delegatie van berekeningen door middel van FHE-technieken. Wij introduceren het concept van blind proofs om integriteitsgaranties te bieden en tonen de praktische haalbaarheid aan via blind zkSNARKs, een concrete instantiatie van blind proofs.

List of Abbreviations

k-NN k-nearest neighbors. 8, 9, 11, 50

CRT Chinese remainder theorem. 14

FHE fully homomorphic encryption. 4
FRI fast Reed-Solomon interactive oracle proofs of proximity. 28

IOP interactive oracle proof. 28, 51

LWE learning with errors. 19

MLaaS Machine learning as a service. 3, 52
MLWE module learning with errors. 22
MSIS module short integer solution. 22

OMR oblivious message retrieval. 52

PIR private information retrieval. 49, 52
PoD proof of decryption. 47
PPML privacy-preserving machine learning. 8, 45, 49

R1CS rank-1 constraint system. 27, 51
RLWE ring learning with errors. 21
RS Reed-Solomon. 28

SIMD single-instruction multiple-data. 6
SIS short integer solution. 18

VC verifiable computation. 28, 43
vCOED verifiable computation over encrypted data. 43, 45, 46
vFHE verifiable fully homomorphic encryption. 7, 43, 45, 51

zk-SNARK zero-knowledge succinct non-interactive arguments of knowledge. 27
zkDel zero-knowledge proof delegation. 8, 54

Contents

Part I

Prelude

Chapter 1

Introduction

In today's digital age, cloud storage and computing have become increasingly popular, with individuals and small organisations relying on external servers to store, manage and process their data, driving a market worth hundreds of billions of dollars [133]. This trend is driven by a fundamental imbalance: client devices, such as personal smartphones, computers, as well as small-scale corporate workstations, often have limited storage and computational capacity, whereas the volume of data generated and collected continues to grow rapidly. Consequently, it is natural for clients to *outsource* storage and computation to more powerful servers, whose resources can be flexibly scaled on demand.

This outsourcing paradigm is widely employed in a variety of practical applications.

- *Remote storage.* A client uploads files to a cloud server, which maintains them over time and returns the requested files to the client.
- *Querying and analysis of stored data.* Beyond storage, a client can request the server to search or analyse data. For example, a client organisation that stores a large database on the cloud may want to know its key statistics. Downloading the entire database for local analysis can be inefficient or even infeasible for a resource-constrained client. Instead, a more practical solution is that the cloud server performs computation directly on the stored data and returns a report of the queried statistics to the client.
- *Machine learning as a service (MLaaS).* Clients can also outsource the evaluation of machine learning models to a cloud server. In this setting, a client sends input features to the server, which is processed by the server using a pre-trained model. Then the resulting output is returned to the client. The rise of generative AI services gives a good example, where

clients provide text inputs and receive answers generated entirely on the server side.

While cloud storage and computing provide significant convenience, they also raise critical privacy concerns. Since data must be transmitted to and processed by cloud servers, they inevitably gain access to potentially sensitive client information. In practice, this may include medical records, financial transactions, or personal data, whose exposure creates risks of inadvertent leakage and data misuse. The tension between the need for outsourcing and the need to preserve clients' privacy motivates research into secure cloud storage and computing.

To provide data both at rest and in transit, a basic cryptographic technique called *encryption* can be used. Encryption transforms *plaintext* data into *ciphertexts*, and its reverse process, *decryption*, recovers the original plaintext using a *secret key*. If an encryption scheme is secure, then recovering the plaintext from the ciphertext without the secret key is computationally infeasible, as it would require solving the underlying hard mathematical problem.

Therefore, in the *remote storage* use case, a client can encrypt its original data into ciphertexts and store ciphertexts on the cloud server. When the data is needed, the server returns the ciphertext, and the client decrypts it locally with the secret key. In this way, the security of the encryption scheme guarantees that neither the cloud server nor an adversary monitoring the communication channel can learn anything about the client's private data.

However, in use cases of *querying and analysis of stored data* and *MLaaS*, traditional encryption schemes do not support computation over encrypted data. This leads to the need for advanced cryptographic techniques that allow computation over encrypted data without decrypting it. This thesis focuses on fully homomorphic encryption (FHE).

FHE. FHE is a class of encryption schemes that allows computations over ciphertexts without knowing the secret key. This interesting property enables the cloud server not only to store ciphertexts, but also to evaluate desired functions on ciphertexts. Therefore, this makes the cloud server an *evaluator*. As an example, given FHE ciphertexts $\mathtt{ct}[m_1]$ and $\mathtt{ct}[m_2]$ that encrypt plaintext messages m_1 and m_2, and a function f, the cloud server can perform the following computation

$$\mathrm{Eval}_f : \{\mathtt{ct}[m_1], \mathtt{ct}[m_2]\} \to \mathtt{ct}[f(m_1, m_2)]$$

which outputs an FHE ciphertext $\mathtt{ct}[f(m_1, m_2)]$ that encrypts $f(m_1, m_2)$. Crucially, the security of the FHE scheme guarantees that the cloud server cannot learn the original data (m_1, m_2) or the computation result $f(m_1, m_2)$. The client can then use the secret key to recover $f(m_1, m_2)$ from the resulting ciphertext.

In FHE, the function f to be evaluated can be an arbitrary computation, thanks to the seminal bootstrapping idea proposed by Gentry [83]. Each FHE ciphertext contains a *noise* component, which must remain below a pre-defined threshold; otherwise, the decryption procedure cannot correctly remove the noise and recover the underlying plaintext. Since the noise grows with elementary homomorphic operations such as additions and multiplications, evaluating complex functions eventually requires a procedure to *refresh* the ciphertext noise. This is achieved through bootstrapping, which homomorphically evaluates the decryption function using a *bootstrapping key*. The bootstrapping key consists of an encryption of the secret key, but it does not leak information about the secret key itself under the widely believed circular security assumption. This technique enables FHE to support arbitrary computations on encrypted data while preserving the confidentiality of the underlying messages.

Several practical FHE schemes have since been developed, including BGV [34], BFV [33, 68], GBFV [81], TFHE [51], FHEW [63], CKKS [48], and FINAL [31]. This thesis focuses on BGV/BFV/GBFV and TFHE/FHEW, and does not cover CKKS and FINAL.

Other secure outsourcing methods compared to FHE. Searchable symmetric encryption (SSE) [57, 139] enables efficient keyword search over encrypted data stored on a remote cloud server. To support this functionality, a client stores an encrypted index structure on the server, along with the encrypted data. Then, the client can perform keyword queries by generating query tokens, which the server evaluates against encrypted indices to identify the matching documents without learning the keyword itself.

However, SSE schemes fail to protect against information leakage through (1) access pattern, which determines if certain records are consistently accessed, and (2) search pattern, which indicates if and when an encrypted query is repeated. These leakages can inadvertently expose properties of datasets, thereby compromising privacy. In contrast, FHE prevents such leakages since the access pattern remains uniform and FHE ciphertexts are randomised, or more formally, FHE attains the IND-CPA (indistinguishability under chosen plaintext attack) security.

Multiparty computation (MPC) [19, 87, 137, 148] enables multiple parties to jointly compute a function over their private inputs while revealing nothing beyond the final output. In a secure outsourcing scenario, a client can split its input into secret shares and distribute them among several cloud servers. These servers then jointly perform the desired computation. However, the privacy of this approach relies on the assumption that these servers do not collude; otherwise, they can reconstruct the client's input. This strong non-collusion

assumption makes this MPC solution less practical in real-world outsourcing settings.

If a client wants to avoid the non-collusion assumption by using a single cloud server in MPC, then the client can also participate as a computing party. This approach, however, requires the client to compute and interact with the server in multiple rounds. In contrast, FHE allows a client to fully delegate the computation task: the client simply sends the encrypted input and later receives the encrypted output, without the need for online interaction.

Trusted execution environment (TEE) leverages dedicated hardware to enable secure computation inside protected domains, which cannot be accessed outside of the TEE. In a secure outsourcing scenario, a client sends encrypted data to a remote machine with a TEE, which loads it to the protected memory, decrypts it, and performs computation in the clear. While TEE incurs a much lower computation overhead compared to FHE, its security requires trust in the hardware of the remote server rather than purely cryptographic guarantees. Vulnerabilities have been discovered in implementations by major vendors such as Intel and ARM, raising concerns about their security guarantees in practice [41, 103].

1.1 Challenges in FHE

While FHE offers strong privacy guarantees, its broader adoption is constrained by two primary bottlenecks. The first is the *efficiency* bottleneck, as FHE incurs significant overhead in computation, communication and storage. The second is the *integrity* bottleneck, stemming from the lack of mechanisms to verify the correctness of the outsourced computation. We elaborate on both challenges below.

Communication efficiency. In secure outsourcing, a client sends encrypted messages to the server. However, ciphertexts are often much larger than the original messages, e.g. the encryption of 1 bit can expand to several kilobytes. This high ciphertext-to-plaintext expansion factor results in a significant *communication* overhead.

Fortunately, FHE schemes, such as BGV, BFV, GBFV, and CKKS, support *packing*, where multiple messages are encoded into a single ciphertext. This not only reduces the number of ciphertexts to be transmitted but also allows homomorphic evaluation on the packed messages in a single-instruction multiple-data (SIMD) manner, reducing both communication and computation costs.

Another useful technique to reduce the client-to-server communication overhead is transciphering [124], which increases computation costs as a tradeoff. In this approach, the client uses a classical symmetric scheme Π to encrypt messages, producing ciphertexts with an expansion factor close to one, which are then sent to the server. The server evaluates the decryption of Π homomorphically to obtain FHE encryptions of the messages. The resulting FHE ciphertexts are then used for the evaluation of the desired function.

Computation efficiency. The *computation* overhead in FHE remains substantial even with packing. On the one hand, homomorphic operations on ciphertexts are orders of magnitude slower than their plaintext equivalents. For example, a homomorphic multiplication is more than 10,000 times slower than a plaintext multiplication [96]. To mitigate this cost, a major line of research focuses on hardware acceleration [26,27,79,140], which can be up to three orders of magnitude faster than CPUs.

On the other hand, developing efficient FHE programs remains difficult. An important part of the inefficiency is the amplification in computation complexity when a plaintext program is converted into a program operating on the corresponding FHE ciphertexts. For example, in the homomorphic evaluation of the if-else paradigm, each conditional statement needs to be executed. By extension, when traversing a binary tree, the full tree is touched instead of a single path. In other words, the data secrecy guaranteed by FHE comes at the cost of increased computational complexity, even if the computation overhead of each FHE operation would be low.

Integrity. When a client outsources the computation of f on input m to a server, the client expects to receive the correct result $f(m)$. In secure outsourcing with FHE, the client sends an encryption of m to the server, which performs Eval_f and returns a resulting ciphertext $\mathtt{ct}$. While the decryption of $\mathtt{ct}$ is expected to be $f(m)$, this is not guaranteed. In other words, FHE ensures data privacy but does not provide the integrity of the outsourced computation.

In verifiable fully homomorphic encryption (vFHE), integrity is addressed by proving that the server executed Eval_f honestly. However, current vFHE constructions are far from being practical: proving even a single homomorphic multiplication can take several minutes [98]. Moreover, ensuring the correctness of Eval_f also requires proving non-arithmetic maintenance operations, which further increases the performance overhead.

The challenge of *communication efficiency* is addressed in our recent work [97], which is not included in this thesis. This thesis focuses on *computation efficiency* and *integrity*.

1.2 Contributions

This thesis contributes to the advancement of FHE in three main areas:

(i) Optimising oblivious algorithms for use in FHE. Oblivious algorithms execute a fixed sequence of operations independent of the inputs, allowing for a direct translation into their homomorphic analogue without increasing the computational complexity. Therefore, oblivious algorithms serve as building blocks for FHE procedures (e.g. bootstrapping) and applications. In this thesis, we optimise various oblivious algorithms with respect to key performance measures, including algorithmic complexity and multiplicative depth (i.e. the number of consecutive multiplications).

(ii) Building efficient privacy-preserving information systems based on FHE. This thesis presents two privacy-preserving machine learning (PPML) protocols, the k-nearest neighbors (k-NN) algorithm and the decision tree evaluation, achieving concrete efficiency improvements over the state of the art. To protect sensitive genomic information, we develop SQUID, a secure and efficient system for storing and analysing genotype-phenotype data. Moreover, we use FHE for secure zero-knowledge proof delegation (zkDel), and demonstrate its practical performance.

(iii) Constructing verifiable secure delegation of computation through FHE techniques. In secure outsourcing with FHE, the decryption of the server output `ct` is not guaranteed to yield the desired $f(m)$: the server may deviate from the homomorphic protocol, or the noise in `ct` may exceed the threshold for a correct decryption. vFHE addresses the first issue by proving ciphertext relations, and the performance remains impractical. We introduce an alternative approach, the *blind proof* framework, to address both integrity issues by proving plaintext relations. We further demonstrate its practicality using blind zkSNARKs, a concrete instantiation of blind proofs.

Below, we provide an overview of the works presented in this thesis, as well as other works completed during the same period.

On Polynomial Functions Modulo p^e and Faster Bootstrapping for Homomorphic Encryption [78] This work presents a systematic study of polynomial functions modulo p^e, investigating their existence, computation and equivalence of polynomial representations, among other aspects. The developed theory is applied to improve digit extraction, an oblivious procedure that forms the performance bottleneck in BGV/BFV bootstrapping. Specifically, we found sparse representations for the digit extraction function and proposed a new method to decompose digit extraction into multiple stages, each of which can

be evaluated with a polynomial of low degree. Altogether, these optimisations yield speedups of up to 2.8× for digit extraction and 2.6× for BGV/BFV bootstrapping.

Published in EUROCRYPT 2023, also appears in Chapter 6.

Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification [54] This work revisits oblivious Top-k algorithms, where the k smallest (or largest) elements are selected from d elements using a fixed sequence of comparators. By combining existing constructions with new improvements, we propose a new Top-k network with complexity $O(d \log^2 k)$ in general and $O(d \log k)$ for small $k \ll \sqrt{d}$. We integrate this optimised Top-k network with TFHE to build a secure, non-interactive k-NN classifier that is asymptotically faster than the best prior work with $O(d^2)$ complexity. The concrete efficiency also shows a significant improvement of up to 47x for $d = 1000$.

Published in SAC 2024, also appears in Chapter 7.

Faster Private Decision Tree Evaluation for Batched Input from Homomorphic Encryption [55] This work focuses on private decision tree evaluation on batched inputs, where a client sends a batch of encrypted feature vectors and a cloud server performs homomorphic inferences against a decision tree model stored as plaintexts. We propose two novel methods for the key component, batched ciphertext-plaintext comparisons, and adapt an existing tree traversal method to achieve optimal client-to-server communication. Overall, our batched decision tree protocols outperform the prior works for large batch sizes, achieving up to a 17× speedup in a batch size of 16384.

Published in SCN 2024, also appears in Chapter 8.

SQUiD: ultra-secure storage and analysis of genetic data for the advancement of precision medicine [28] This work presents Secure Queriable Database (SQUiD), a secure and scalable framework designed to store and query genotype-phenotype databases in the cloud using homomorphic encryption. Our approach incorporates a set of optimisations to reduce storage overhead, accelerate query processing, and support queries from multiple users. We demonstrate that SQUiD can efficiently execute a wide range of queries on large-scale genotype-phenotype datasets, including queries on a filtered cohort such as minor allele frequency (MAF) and polygenic risk scores (PRS). These results highlight the potential of SQUiD as a valuable tool for privacy-preserving genetic data storage and analysis, contributing to the advancement of precision medicine.

Published in Genome Biology 2024, also appears in Chapter 9.

Blind zkSNARKs for Private Proof Delegation and Verifiable Computation over Encrypted Data [72] While a zkSNARK prover computes $\mathsf{Prove}_f(m, y)$ to convince the verifier that $y = f(m)$, this work formalises the blind zkSNARKs framework, where a prover only has access to $\mathtt{ct}[y]$ and $\mathtt{ct}[f(m)]$ and homomorphically evaluates the proving procedure (i.e. compute $\mathsf{Eval}_{\mathsf{Prove}_f}$) to convince the verifier that the underlying plaintexts satisfy the relation defined by f. We instantiate blind zkSNARKs with Fractal, leveraging GBFV for large-field arithmetic and incorporating several optimisations, e.g. a packing-friendly 2D-NTT protocol. With a 32× parallelisation speedup on a powerful server, the prover runtime for 2^{20} R1CS constraints is estimated to be under 20 minutes. Additionally, we achieve public verifiability by appending a zero-knowledge proof of decryption (PoD), enabling another compelling application of blind zkSNARKs: zero-knowledge proof delegation (zkDel). Our PoD construction is highly efficient, allowing the decryption of around 2500 ciphertexts to be proven in under 2 seconds on a single thread.

Published in Communications in Cryptology 2025, also appears in Chapter 10.

Other works

On the Overflow and p-adic Theory Applied to Homomorphic Encryption [29] This work studies the overflow phenomenon when performing integer and rational arithmetic using modular arithmetic over $\mathbb{Z}/q\mathbb{Z}$. While previous works avoid overflows by restricting supported circuits in the homomorphic evaluation, we discuss two possibilities of tolerating them. Firstly, we explain that when input messages and the final result are well-bounded, intermediate values can be arbitrarily large without affecting output correctness. This kind of overflow is called pseudo-overflow and does not need to be avoided. Secondly, we observe that for a prime-power modulus $q = p^r$, overflow errors are small in the p-adic norm. Therefore, incorporating a p-adic encoding to BGV/BFV allows the evaluation of circuits that are up to 2× deeper under the same ciphertext parameters, and output errors are bounded by p^{-r} in the p-adic norm.

Published as a short paper in CSCML 2024.

Pirouette: Query Efficient Single-Server PIR [97] This paper presents Pirouette, a private information retrieval protocol with low query size. Specifically, the query consists of only one component of a high-precision LWE ciphertext, from which we design a modular procedure to extract the inputs for the subsequent homomorphic computation. For a database of 2^{25} records, the query size is just 36B. Moreover, if the query seed is set once by the server, then the query size drops to only 32 bits, resulting in an exceptionally low expansion factor of $32/25 = 1.28$. This represents a novel approach to

reduce the client-to-server communication, which provides a lower server-side computational overhead than transciphering and is applicable for many other FHE-based applications.

Full version available as preprint.

FRIttata: Distributed Proof Generation of FRI-based SNARKs [146] This paper presents FRIttata, the first distributively computable SNARK for general circuits that is transparent and plausibly post-quantum secure. The construction builds on a distributed FRI protocol, whose security properties such as the knowledge soundness are inherited from the FRI protocol [20]. We investigate and implement three instantiations of distributed FRI, each offering different trade-offs between proof size, communication overhead, and verifier efficiency.

Full version available as preprint.

1.3 Outline

This thesis consists of two parts. Part I provides a comprehensive overview and sets the stage for the forthcoming content in the second part. To begin with, Chapter 2 covers necessary preliminaries on mathematics, including algebra and lattices, and on cryptography, including zero-knowledge proofs and verifiable computation. Next, Chapter 3 provides an extensive overview of FHE, specifically tailored to align with our contributions. Then, Chapter 4 introduces two frameworks designed to provide data integrity alongside the privacy guarantees offered by FHE. Finally, Chapter 5 concludes this dissertation by discussing the main results and outlining directions for future work.

Part II presents five publications, contributing both to the theoretical foundations of FHE (Chapter 6 on bootstrapping, Chapter 10 on plaintext verifiability) and to the practical applications of FHE (Chapter 7 on k-NN, Chapter 8 on decision trees, Chapter 9 on genome query, and Chapter 10 on proof delegation). The chapters are presented in chronological order, reflecting the research's evolution from a focus on efficiency to one that also encompasses integrity.

Chapter 2

Preliminaries

2.1 Basic notation

Given a positive integer a, let $[a]$ denote the set $\{1, 2, \ldots, a\}$ and $[a]_0$ denote the set $\{0, 1, \ldots, a\}$. We use uppercase bold letters such as $\mathbf{U}$ for matrices, lowercase letters such as u for row vectors and $\overrightarrow{u}$ for column vectors. The inner product between two vectors of the same type is denoted as $\langle \cdot, \cdot \rangle$. Elements in polynomial rings are written in lowercase bold letters such as $\boldsymbol{u}$ or $u(X)$. Following the convention used in Magma, we use $R \,!\, a$ to denote the coercion of an element a into the ring R.

For a positive integer q, let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the ring of integers modulo q, where the representative interval is by default $(-\frac{q}{2}, \frac{q}{2}]$. Furthermore, let $\mathbb{Z}_q^\times$ denote the multiplicative group of integers modulo q, and $|\mathbb{Z}_q^\times| = \varphi(q)$ with $\varphi(\cdot)$ the Euler totient function. For $x \in \mathbb{Z}$, we denote the centered reduction modulo q by $[x]_q \in \mathbb{Z}_q$. Let $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the flooring and ceiling functions, respectively, and let $\lfloor \cdot \rceil$ denote the rounding function that rounds half up. All this notation extends to elements in polynomial rings coefficient-wise.

For $\boldsymbol{g} = \sum_{i=0}^{n-1} g_i X^i \in \mathbb{Q}[X]$, let $\overrightarrow{g}$ denote its coefficient vector $[g_0 \; g_1 \; \ldots \; g_{n-1}]^\intercal \in \mathbb{Q}^n$, then its coefficient-wise norms are $\|\boldsymbol{g}\|_p = \|\overrightarrow{g}\|_p$, e.g.

$$\|\boldsymbol{g}\|_1 = \sum_{i=0}^{n-1} |g_i|, \quad \|\boldsymbol{g}\|_2 = \left(\sum_{i=0}^{n-1} g_i^2 \right)^{\frac{1}{2}}, \quad \|\boldsymbol{g}\|_\infty = \max\{|g_i|\}_{i \in [n-1]_0}.$$

Given a probability distribution χ, let $a \leftarrow \chi$ denote that a is sampled from χ. Given a finite set S, let $a \xleftarrow{\$} S$ denote that a is sampled uniformly at random from S. Let $D_{\mathbb{Z}^n, \sigma}$ denote the discrete Gaussian distribution, where each $\overrightarrow{x} \in \mathbb{Z}^n$

is assigned with a probability proportional to $\exp(-\pi\|\overrightarrow{x}\|^2/\sigma^2)$. Due to the orthonormality of the basis, the discrete Gaussian $D_{\mathbb{Z}^n,\sigma}$ is simply the product distribution of n independent copies of $D_{\mathbb{Z},\sigma}$, also referred to as the *spherical* Gaussian distribution.

2.2 Preliminaries on algebra

2.2.1 Ideals and quotient rings

Let $(R, +, \cdot)$ denote a commutative ring with identity 1.

Definition 2.2.1. A subset $I \subseteq R$ is an *ideal* if $(I, +)$ is a subgroup of the additive group $(R, +)$ and $RI \subseteq I$, i.e. $x \in R$ and $y \in I$ implies their product $xy \in I$.

From an ideal $I \subseteq R$, the *quotient ring* is defined as $R/I = \{x + I : x \in R\}$.

Definition 2.2.2. Let S be a subset of R. The ideal I generated by S is defined as $I = \langle S \rangle = \{\sum_{i=1}^{n} x_i y_i : x_i \in R, y_i \in S, n \geq 1\}$, and S is a *generating set* of the ideal I.

Definition 2.2.3. An ideal is *finitely generated* if it has a finite generating set, and *principal* if it can be generated by a single element.

Definition 2.2.4. An ideal $P \subseteq R$ is a *prime ideal* if $\forall xy \in P$ for some elements $x, y \in R$, then $x \in P$ or $y \in P$.

Given ideals I, J in R, new ideals can be constructed from the sum $I + J = \{x+y : x \in I, y \in I\}$ and the product $I \cdot J = \{\sum_{i=1}^{n} x_i y_i : x_i \in I, y_i \in I, n \geq 1\}$. The sum $I + J$ is the smallest ideal containing both I and J.

Definition 2.2.5. Two ideals I and J in R are coprime if $I+J=R$.

Theorem 2.2.1 (Chinese remainder theorem (CRT))**.** *Let $I_1, \ldots, I_n$ be pairwise coprime ideals in R, and let $I = \prod_{j=1}^{n} I_j$ denote their product. Then*

$$CRT : R/I \to \prod_{j=1}^{n} R/I_j,\ x \bmod I \mapsto (x \bmod I_1, \ldots, x \bmod I_n).$$

is an isomorphism of rings.

2.2.2 Cyclotomic fields and rings

For a positive integer $\mathfrak{m}$, denote $e^{2\pi i/\mathfrak{m}}$ as $\eta_{\mathfrak{m}}$ and denote the $\mathfrak{m}$-th cyclotomic polynomial as $\Phi_{\mathfrak{m}}(X) = \prod_{k\in\mathbb{Z}_{\mathfrak{m}}^{\times}}(X - \eta_{\mathfrak{m}}^{k}) \in \mathbb{Z}[X]$, which is an irreducible polynomial over $\mathbb{Q}$ of degree $\mathfrak{n} = \varphi(\mathfrak{m})$. Specifically, when $\mathfrak{m}$ is a power-of-two, $\Phi_{\mathfrak{m}}(X) = X^{\mathfrak{m}/2} + 1$. Another important fact about cyclotomic polynomials is that $\Phi_{\mathfrak{m}}(X) = \Phi_{\mathrm{rad}(\mathfrak{m})}(X^{m/\mathrm{rad}(\mathfrak{m})})$, where $\mathrm{rad}(\mathfrak{m})$ is the product of all distinct primes dividing $\mathfrak{m}$.

The $\mathfrak{m}$-th cyclotomic number field is denoted as $\mathcal{K}_{\mathfrak{m}} = \mathbb{Q}[X]/(\Phi_{\mathfrak{m}}(X))$, and the $\mathfrak{m}$-th cyclotomic ring is $\mathcal{R}_{\mathfrak{m}} = \mathbb{Z}[X]/(\Phi_{\mathfrak{m}}(X))$. Note that $\mathcal{K}_{\mathfrak{m}}$ is an $\mathfrak{n}$-dimensional vector space over $\mathbb{Q}$, and the basis $\{1, X, X^2, \ldots, X^{\mathfrak{n}-1}\}$ is known as *the power basis*. The Galois group $\mathrm{Gal}(\mathcal{K}_{\mathfrak{m}}/\mathbb{Q})$ contains $\mathfrak{n}$ automorphisms of $\mathcal{K}_{\mathfrak{m}}$ that fix $\mathbb{Q}$

$$\tau_j : \mathcal{K}_{\mathfrak{m}} \to \mathcal{K}_{\mathfrak{m}},\ a(X) \mapsto a(X^j),\ j \in \mathbb{Z}_{\mathfrak{m}}^{\times}.$$

The $\mathbb{Q}$-linear trace and the (field) norm are two maps from $\mathcal{K}_{\mathfrak{m}}$ to $\mathbb{Q}$ defined as

$$\mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}}/\mathbb{Q}}(a(X)) = \sum_{j\in\mathbb{Z}_{\mathfrak{m}}^{\times}} \tau_j(a(X)), \quad \mathrm{N}_{\mathcal{K}_{\mathfrak{m}}/\mathbb{Q}}(a(X)) = \prod_{j\in\mathbb{Z}_{\mathfrak{m}}^{\times}} \tau_j(a(X)),$$

for any $a(X) \in \mathcal{K}_{\mathfrak{m}}$, which further induce functions $\mathrm{Tr}, \mathrm{N} : \mathcal{R}_{\mathfrak{m}} \to \mathbb{Z}$. The norm of a non-zero ideal $I \subseteq \mathcal{R}_{\mathfrak{m}}$ is its index as an additive subgroup in $\mathcal{R}_{\mathfrak{m}}$, i.e. $\mathrm{N}(I) = |\mathcal{R}_{\mathfrak{m}}/I|$. For any non-zero $\boldsymbol{\alpha} \in \mathcal{R}_{\mathfrak{m}}$, the norm of the ideal $I = \langle\boldsymbol{\alpha}\rangle \subseteq \mathcal{R}_{\mathfrak{m}}$ satisfies $\mathrm{N}(I) = |\mathrm{N}(\boldsymbol{\alpha})|$.

Towers of cyclotomics. Let $\zeta_{\mathfrak{m}}$ denote an element of order $\mathfrak{m}$. For any $\mathfrak{m}'$ dividing $\mathfrak{m}$, the $\mathfrak{m}'$-th cyclotomic field $\mathcal{K}_{\mathfrak{m}'} = \mathbb{Q}(\zeta_{\mathfrak{m}'})$ can be viewed as a subfield of $\mathcal{K}_{\mathfrak{m}} = \mathbb{Q}(\zeta_{\mathfrak{m}})$, under the embedding that identifies $\zeta_{\mathfrak{m}'}$ as $\zeta_{\mathfrak{m}}^{\mathfrak{m}/\mathfrak{m}'}$. The $\mathcal{K}_{\mathfrak{m}'}$-linear trace function $\mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}}/\mathcal{K}_{\mathfrak{m}'}}$ can be defined as

$$\mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}}/\mathcal{K}_{\mathfrak{m}'}}(\boldsymbol{a}) = \sum_{i=1 \bmod \mathfrak{m}'} \tau_i(\boldsymbol{a}).$$

The transitivity of the trace in the field extension $\mathcal{K}_{\mathfrak{m}}/\mathcal{K}_{\mathfrak{m}'}/\mathbb{Q}$ gives $\mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}}/\mathbb{Q}} = \mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}'}/\mathbb{Q}} \circ \mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}}/\mathcal{K}_{\mathfrak{m}'}}$, which similarly holds for the norm. Moreover, for every $\mathcal{K}_{\mathfrak{m}'}$-linear function $L : \mathcal{K}_{\mathfrak{m}} \to \mathcal{K}_{\mathfrak{m}'}$, there exists an element $\boldsymbol{r} \in \mathcal{K}_{\mathfrak{m}}$ such that $L(\boldsymbol{a}) = \mathrm{Tr}_{\mathcal{K}_{\mathfrak{m}}/\mathcal{K}_{\mathfrak{m}'}}(\boldsymbol{r} \cdot \boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{K}_{\mathfrak{m}}$.

The powerful basis. Let $\mathfrak{m} = \prod_{i=1}^{k} \mathfrak{m}_i$ denote the factorization of $\mathfrak{m}$ into prime powers, there exists an isomorphism between $\mathcal{K}_{\mathfrak{m}} = \mathbb{Q}[X]/(\Phi_{\mathfrak{m}}(X))$ and the tensor product $\otimes_i \mathcal{K}_{\mathfrak{m}_i} = \mathbb{Q}[X_1, \ldots, X_k]/(\Phi_{\mathfrak{m}_1}(X_1), \ldots, \Phi_{\mathfrak{m}_k}(X_k))$ defined by

$$\otimes_i \mathcal{K}_{\mathfrak{m}_i} \to \mathcal{K}_{\mathfrak{m}}, \quad \sum_{\mathbf{j}} a_{\mathbf{j}} \cdot \mathbf{X}^{\mathbf{j}} \mapsto \sum_{\mathbf{j}} a_{\mathbf{j}} \cdot f(\mathbf{X}^{\mathbf{j}}),$$

where $\mathbf{j} = (j_1, \ldots, j_k)$ for $0 \leq j_i \leq \varphi(\mathfrak{m}_i)$, and $\mathbf{X}^{\mathbf{j}} := \prod_{i=1}^{k} X_i^{j_i}$ denotes a basis element in the tensor product, which is mapped to $f(\mathbf{X}^{\mathbf{j}}) := \prod_{i=1}^{k} X^{\mathfrak{m}/\mathfrak{m}_i \cdot j_i}$ in $\mathcal{K}_{\mathfrak{m}}$. Therefore, the set $\{\mathbf{X}^{\mathbf{j}}\}$ forms another basis for $\mathcal{K}_{\mathfrak{m}}$ that does not coincide with the power basis, typically referred to as *the powerful basis* [115].

Rotation matrix and expansion factor. Given $\boldsymbol{g}, \boldsymbol{f} \in \mathcal{R}_{\mathfrak{m}}$ and the product $\boldsymbol{h} = \boldsymbol{g} \cdot \boldsymbol{f} \in \mathcal{R}_{\mathfrak{m}}$, their coefficient representations satisfy $\overrightarrow{h} = \mathsf{Rot}_{\mathfrak{m}}(\boldsymbol{g}) \cdot \overrightarrow{f}$, where $\mathsf{Rot}_{\mathfrak{m}}(\boldsymbol{g}) = \left[\begin{array}{cccc} | & | & & | \\ \overrightarrow{g_{(0)}} & \overrightarrow{g_{(1)}} & \cdots & \overrightarrow{g_{(\mathfrak{n}-1)}} \\ | & | & & | \end{array}\right] \in \mathbb{Z}^{\mathfrak{n} \times \mathfrak{n}}$ and $\boldsymbol{g_{(i)}} = X^i \cdot g(X) \bmod \Phi_{\mathfrak{m}}(X)$. To bound the coefficient growth in such multiplication, the expansion factor of the ring $\mathcal{R}_{\mathfrak{m}}$ is defined as

$$\delta_{\mathfrak{m}} = \sup\left\{ \frac{\|\boldsymbol{g} \cdot \boldsymbol{f} \bmod \Phi_{\mathfrak{m}}\|_{\infty}}{\|\boldsymbol{g}\|_{\infty} \cdot \|\boldsymbol{f}\|_{\infty}} \mid \boldsymbol{g}, \boldsymbol{f} \in \mathbb{Z}[X] \setminus 0 \text{ and } \deg(\boldsymbol{g}), \deg(\boldsymbol{f}) \leq (\mathfrak{n}-1) \right\}.$$

Let $EF_{\mathfrak{m}}$ be the lowest number such that $\left\|\overrightarrow{g_{(i)}}\right\|_{\infty} \leq EF_{\mathfrak{m}} \cdot \|\boldsymbol{g}\|_{\infty}$ for any $i \in [\mathfrak{n}-1]_0$ and $\boldsymbol{g} \in \mathcal{R}_{\mathfrak{m}}$, then $\delta_{\mathfrak{m}} \leq \mathfrak{n} \cdot EF_{\mathfrak{m}}$. Specifically, when $\mathfrak{m}$ is a power-of-two, $EF_{\mathfrak{m}} = 1$ and $\delta_{\mathfrak{m}} = \mathfrak{n}$.

Canonical embedding. While coefficient norms provide a straightforward measure of sizes for polynomials in $\mathcal{K}_{\mathfrak{m}}$, analyzing the norm growth upon multiplication requires the expansion factor $\delta_{\mathfrak{m}}$, which depends heavily on the polynomial modulus $\Phi_{\mathfrak{m}}(X)$ and often results in loose bounds. This leads to the broad use of the canonical norm $\|\cdot\|^{\text{can}}$ [56, 58, 86, 114, 115], defined from the following canonical embedding σ into $\mathbb{C}^{\mathfrak{n}}$

$$\sigma : \mathcal{K}_{\mathfrak{m}} \to \mathbb{C}^{\mathfrak{n}} \colon a(X) \mapsto \{a(\eta_{\mathfrak{m}}^{j})\}_{j \in \mathbb{Z}_{\mathfrak{m}}^{\times}}.$$

Let s_1 and s_2 denote the number of real embeddings and the number of pairs of complex embeddings, so $\mathfrak{n} = s_1 + 2s_2$. Furthermore, we order the embeddings such that $\{\sigma_j\}_{j \in [s_1]}$ are the real embeddings and $\sigma_{s_1+s_2+j} = \overline{\sigma_{s_1+j}}$ for $j \in [s_2]$. Then the image of the canonical embedding, $\sigma(\mathcal{K}_{\mathfrak{m}})$ is in the following space

$$H_{\mathfrak{m}} = \left\{(x_1, \cdots, x_{\mathfrak{n}}) \in \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} \mid x_{s_1+s_2+j} = \overline{x_{s_1+j}},\ \forall j \in [s_2]\right\} \subseteq \mathbb{C}^{\mathfrak{n}}.$$

This space $H_{\mathfrak{m}}$, when equipped with the inner product induced by $\mathbb{C}^{\mathfrak{n}}$, is isomorphic to $\mathbb{R}^{\mathfrak{n}}$ [114]. Additionally, it is also equipped with the ℓ_p norm induced from $\mathbb{C}^{\mathfrak{n}}$, hence the *canonical norm* is defined as $\|\boldsymbol{a}\|_p^{\text{can}} = \|\sigma(\boldsymbol{a})\|_p, \forall \boldsymbol{a} \in \mathcal{K}_{\mathfrak{m}}$.

Lemma 2.2.1 (Adapted from [58])**.** *For all $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{K}_{\mathfrak{m}}$, the following properties are satisfied*

- $\|\boldsymbol{a}\|_{\infty}^{\text{can}} \leq \|\boldsymbol{a}\|_1$,

- $\|\boldsymbol{a}\|_\infty \leq c_m \cdot \|\boldsymbol{a}\|_\infty^{\text{can}}$, *where* c_m *is a constant determined by* m,
- $\|\boldsymbol{a}+\boldsymbol{b}\|_\infty^{\text{can}} \leq \|\boldsymbol{a}\|_\infty^{\text{can}} + \|\boldsymbol{b}\|_\infty^{\text{can}}$,
- $\|\boldsymbol{a}\cdot\boldsymbol{b}\|_p^{\text{can}} \leq \|\boldsymbol{a}\|_\infty^{\text{can}} \cdot \|\boldsymbol{b}\|_p^{\text{can}}$.

Specifically, $c_{\mathfrak{m}} = 1$ for power-of-two $\mathfrak{m}$, and for $\mathfrak{m} = p_1^{e_1}\cdots p_k^{e_k}$ for different primes $p_1, \ldots, p_k$, if $p_1 \cdots p_k \leq 400$ then $c_{\mathfrak{m}} \leq 8.6$ [58].

Additional notation. For a non-zero element $\boldsymbol{t} \in \mathcal{R}_{\mathfrak{m}}$, we denote the quotient ring of $\mathcal{R}_{\mathfrak{m}}$ modulo $\boldsymbol{t}$ as $\mathcal{R}_{\mathfrak{m},\boldsymbol{t}} = \mathcal{R}_{\mathfrak{m}} / \boldsymbol{t}\,\mathcal{R}_{\mathfrak{m}}$. In this thesis, we often consider a special case for a non-zero integer q and the quotient ring $\mathcal{R}_{\mathfrak{m},q} = \mathcal{R}_{\mathfrak{m}} / q\,\mathcal{R}_{\mathfrak{m}}$. Notation for coefficient vectors and norms in $\mathcal{R}_{\mathfrak{m}}$ naturally extend to $\mathcal{R}_{\mathfrak{m},q}$ using representatives in $(-\frac{q}{2}, \frac{q}{2}]$.

2.2.3 Linear algebra

Let R denote a commutative ring and $R[X]_{<d}$ denote the R-module of polynomials of degree less than d.

Definition 2.2.6. Let $\boldsymbol{f} = \sum_{i=0}^{d_f} f_i X^i, \boldsymbol{g} = \sum_{i=0}^{d_g} g_i X^i \in R[X]$ denote two polynomials, their *Sylvester map* is the R-module homomorphism

$$S_{R,\boldsymbol{f},\boldsymbol{g}} : R[X]_{<d_{\boldsymbol{g}}} \oplus R[X]_{<d_{\boldsymbol{f}}} \to R[X]_{<(d_{\boldsymbol{g}}+d_{\boldsymbol{f}})} : (\boldsymbol{r},\boldsymbol{s}) \mapsto \boldsymbol{r}\cdot\boldsymbol{f} + \boldsymbol{s}\cdot\boldsymbol{g}.$$

The *Sylvester matrix* is the corresponding representation in the power basis

$$M(S_{R,\boldsymbol{f},\boldsymbol{g}}) = \begin{bmatrix} f_{d_{\boldsymbol{f}}} & f_{d_{\boldsymbol{f}}-1} & \cdots & f_0 & & & \\ & & & \ddots & & & \\ & & & f_{d_{\boldsymbol{f}}} & f_{d_{\boldsymbol{f}}-1} & \cdots & f_0 \\ g_{d_{\boldsymbol{g}}} & g_{d_{\boldsymbol{g}}-1} & \cdots & \cdots & g_0 & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & g_{d_{\boldsymbol{g}}} & g_{d_{\boldsymbol{g}}-1} & \cdots & \cdots & g_0 \end{bmatrix} \in R[X]^{(d_{\boldsymbol{g}}+d_{\boldsymbol{f}})\times(d_{\boldsymbol{g}}+d_{\boldsymbol{f}})}.$$

When $R = \mathbb{Q}$ (or any field), the monic polynomial of the lowest degree in $\text{Image}(S_{\mathbb{Q},\boldsymbol{f},\boldsymbol{g}})$ is $\gcd(\boldsymbol{f},\boldsymbol{g})$. For coprime $\boldsymbol{f},\boldsymbol{g} \in \mathbb{Q}[X]$, $\gcd(\boldsymbol{f},\boldsymbol{g}) = 1 \in \text{Image}(S_{\mathbb{Q},\boldsymbol{f},\boldsymbol{g}})$. Consequently, for $R = \mathbb{Z}$ and coprime $\boldsymbol{f},\boldsymbol{g} \in \mathbb{Z}[X]$, the non-zero polynomials of the lowest degree in $\text{Image}(S_{\mathbb{Z},\boldsymbol{f},\boldsymbol{g}})$ are constants.

Definition 2.2.7. Let $\boldsymbol{f},\boldsymbol{g} \in \mathbb{Z}[X]$ be coprime polynomials. The *congruence number* $\text{Con}(\boldsymbol{f},\boldsymbol{g})$ is the smallest positive integer in $\text{Image}(S_{\mathbb{Z},\boldsymbol{f},\boldsymbol{g}})$.

The congruence number $\text{Con}(\boldsymbol{f},\boldsymbol{g})$ for $\boldsymbol{f},\boldsymbol{g} \in \mathbb{Z}[X]$ can be derived in two equivalent methods.

- view $\boldsymbol{f}, \boldsymbol{g}$ as polynomials in $\mathbb{Q}[X]$, find $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{Q}[X]$ such that $\boldsymbol{r} \cdot \boldsymbol{f} + \boldsymbol{s} \cdot \boldsymbol{g} = 1$, then the least common multiple of the denominator of $\boldsymbol{r}$ and $\boldsymbol{s}$ is $\mathrm{Con}(\boldsymbol{f}, \boldsymbol{g})$.
- transform $M(S_{\mathbb{Z},\boldsymbol{f},\boldsymbol{g}})$ into its row echelon form $M'(S_{\mathbb{Z},\boldsymbol{f},\boldsymbol{g}})$ using row operations, then $\mathrm{Con}(\boldsymbol{f}, \boldsymbol{g})$ is the bottom-right entry of $M'(S_{\mathbb{Z},\boldsymbol{f},\boldsymbol{g}})$.

Let $p = \mathrm{Con}(\boldsymbol{f}, \boldsymbol{g})$. The last non-zero row of the reduction of $M'(S_{\mathbb{Z},\boldsymbol{f},\boldsymbol{g}})$ modulo p gives (the coefficients of) a non-trivial common divisor of $\boldsymbol{f}$ and $\boldsymbol{g}$ modulo p, i.e. $\gcd(\mathbb{Z}_p[X]!\ \boldsymbol{f}, \mathbb{Z}_p[X]!\ \boldsymbol{g})$.

2.3 Lattices and hard lattice problems

An n-dimensional lattice $\mathcal{L}$ is a discrete additive subgroup of the vector space $\mathbb{R}^n$, generated as

$$\mathcal{L} = \left\{ \sum_{i=1}^{k} x_i \cdot \overrightarrow{b_i} \mid x_i \in \mathbb{Z} \right\},$$

where the vectors $\left\{\overrightarrow{b_i}\right\}_{i\in[k]}$ are linearly independent and $k \leq n$ is the rank of $\mathcal{L}$. The lattice generated by $\mathbf{B} = \left\{\overrightarrow{b_i}\right\}_{i\in[k]}$ is denoted as $\mathcal{L}(\mathbf{B})$.

2.3.1 The short integer solution problem

The short integer solution (SIS) problem was first introduced by Ajtai [2] in 1996.

Definition 2.3.1 ($\mathsf{SIS}_{n,m,q,B}$)**.** Given $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ (typically $n \ll m$), the $\mathsf{SIS}_{n,m,q,B}$ problem is to find a non-zero $\overrightarrow{z} \in \mathbb{Z}^m$ such that $\mathbf{A} \cdot \overrightarrow{z} = \overrightarrow{0} \bmod q$ and $\|\overrightarrow{z}\|_2 \leq B$.

If $B \geq q$, the SIS problem admits a trivial solution $[q, 0, \ldots, 0]^\intercal$. Furthermore, if $B \geq \sqrt{m} q^{n/m}$, the SIS problem is guaranteed to have a solution, as shown in [122, Lemma 5.2] using the pigeon-hole principle. Therefore, to avoid considering SIS problems that are trivially simple or vacuously hard, most works consider bounds $B \in [\sqrt{m} q^{n/m}, q)$ [99, 111, 122].

Lattice reduction. The SIS problem is equivalent to finding a short non-zero vector in an m-dimensional q-ary lattice

$$\Lambda_q^{\perp}(\mathbf{A}) \coloneqq \{\overrightarrow{z} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \overrightarrow{z} = \overrightarrow{0} \bmod q\}.$$

Therefore, the computational hardness of SIS can be estimated via lattice reductions. As reported in [74, 123], the lattice reduction algorithms are able to find vectors in $\Lambda_q^{\perp}(\mathbf{A})$ of length

$$\min\left\{q, \left(\det\left(\Lambda_q^{\perp}(\mathbf{A})\right)\right)^{1/m} \cdot \delta^m\right\} = \min\left\{q, q^{n/m} \cdot \delta^m\right\}, \tag{2.1}$$

where δ is the root-Hermite factor depending on the reduction algorithm, and the relation $\det\left(\Lambda_q^{\perp}(\mathbf{A})\right) = q^n$ holds with high probability since the number of elements of $\mathbb{Z}_q^m$ that belong to $\Lambda_q^{\perp}(\mathbf{A})$ is q^{m-n} with high probability.

When Equation (2.1) is considered as a function of m, the minimum value $2^{2\sqrt{n \log q \log \delta}}$ is achieved when $m = \sqrt{n \log q / \log \delta}$. Therefore, for lattices with larger m, one can efficiently reduce it to this optimal dimension by fixing some of the coordinates to 0, which allows finding a non-zero vector in $\Lambda_q^{\perp}(\mathbf{A})$ of length

$$\min\left\{q, 2^{2\sqrt{n \log q \log \delta}}\right\}. \tag{2.2}$$

Hardness estimation. The computational hardness of SIS can be estimated using lattice reduction, where one computes a sufficiently small root-Hermite factor δ such that the output vector with the length of Equation (2.2) is adequately short. Using the NTL library [138], Lindner and Peikert [104] estimated that setting $\log_2(\delta) \simeq 1.8/(\lambda + 110)$ offers λ-bit security, i.e. $\delta \simeq 1.0052$ is suggested for 128 bit security. Recent lattice-based cryptographic constructions [65, 67] use Albrecht et al.'s lattice estimator [3], adopting $\delta \simeq 1.0045$ for 128-bit security.

2.3.2 The learning with errors problem

The learning with errors (LWE) problem was introduced by Regev [135] in 2005, parametrized by a dimension n, an integer modulus q, and an error distribution χ over $\mathbb{Z}$.

Definition 2.3.2 (LWE distribution $\mathcal{D}_{\overrightarrow{s},\chi}$)**.** Given $\overrightarrow{s} \in \mathbb{Z}_q^n$, the LWE distribution $\mathcal{D}_{\overrightarrow{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing a uniformly random $\mathsf{a} \in \mathbb{Z}_q^n$ and error $e \leftarrow \chi$, and outputting $(\mathsf{a}, b = \mathsf{a} \cdot \overrightarrow{s} + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

We can represent $m \geq 1$ LWE instances with the same secret $\overrightarrow{s} \in \mathbb{Z}_q^n$ in the following matrix form

$$(\mathbf{A}, \overrightarrow{b} = \mathbf{A} \cdot \overrightarrow{s} + \overrightarrow{e} \bmod q),$$

where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\overrightarrow{b}, \overrightarrow{e} \in \mathbb{Z}_q^m$. Then we describe the two versions of the LWE problem.

Definition 2.3.3 (Search $\mathsf{LWE}_{n,m,q,\chi}$)**.** The search $\mathsf{LWE}_{n,m,q,\chi}$ problem is to recover $\overrightarrow{s} \in \mathbb{Z}_q^n$ from m LWE instances $(\mathbf{A}, \overrightarrow{b} = \mathbf{A} \cdot \overrightarrow{s} + \overrightarrow{e} \bmod q)$ sampled according to the distribution $\mathcal{D}_{\overrightarrow{s},\chi}$.

Definition 2.3.4 (Decision $\mathsf{LWE}_{n,m,q,\chi}$)**.** The decision $\mathsf{LWE}_{n,m,q,\chi}$ problem is to distinguish m LWE instances $(\mathbf{A}, \overrightarrow{b} = \mathbf{A} \cdot \overrightarrow{s} + \overrightarrow{e} \bmod q)$ sampled according to the distribution $\mathcal{D}_{\overrightarrow{s},\chi}$, from $(\mathbf{A}, \overrightarrow{b}) \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.

The reduction from Search-LWE to Decision-LWE is straightforward, as the recovered secret $\overrightarrow{s}$ allows the direct computation of the error $\overrightarrow{e}$. The converse reduction is provided in [135, Lemma 4.2], establishing the equivalence of Decision-LWE and Search-LWE.

Solving LWE from SIS. The Decision-LWE can be transformed into a SIS instance that finds a non-zero small vector $\overrightarrow{z} \in \mathbb{Z}_q^m$ such that

$$\overrightarrow{z}^\intercal \cdot \mathbf{A} = \mathbf{0} \bmod q.$$

- If $(\mathbf{A}, \overrightarrow{b}) \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, then $\langle \overrightarrow{z}, \overrightarrow{b} \rangle$ is also uniformly random.
- If $(\mathbf{A}, \overrightarrow{b}) \leftarrow \mathcal{D}_{\overrightarrow{s},\chi}$, then

$$\langle \overrightarrow{z}, \overrightarrow{b} \rangle = \langle \overrightarrow{z}^\intercal \cdot \mathbf{A}, \overrightarrow{s} \rangle + \langle \overrightarrow{z}, \overrightarrow{e} \rangle = \langle \overrightarrow{z}, \overrightarrow{e} \rangle \bmod q. \tag{2.3}$$

If $\overrightarrow{z}$ is sufficiently short, then Equation (2.3) is small since $\overrightarrow{e}$ is small.

For the error distribution $\chi = D_{\mathbb{Z},\sigma}$, Lindner and Peikert [104] demonstrated that the advantage ϵ of distinguishing Equation (2.3) from random is close to $\exp\left(-\pi(\|\overrightarrow{z}\| \cdot \frac{\sigma}{q})^2\right)$. This implies that achieving a distinguishing advantage ϵ requires finding a vector of length $\alpha \cdot (q/\sigma)$, where $\alpha = \sqrt{\ln(1/\epsilon)/\pi}$. Since a lattice reduction with root-Hermite factor δ allows to find a short vector with the length of Equation (2.2), secure LWE parameters should satisfy

$$\alpha \cdot \frac{q}{\sigma} < 2^{2\sqrt{n \log q \log \delta}}. \tag{2.4}$$

This inequality in Equation (2.4) provides an insight into the relation between secure LWE parameters. In particular, for a fixed lattice dimension n, increasing the modulus q beyond a certain bound necessitates also increasing the standard deviation σ to maintain security.

Hardness estimation. The concrete security of LWE instances for given parameter sets is best estimated using Albrecht et al.'s lattice estimator [3],

which considers various attacks. These include the above `dual` lattice attack, `primal` lattice attacks such as `primal-usvp` and `primal-bdd`, as well as well as `hybrid` attacks that augment lattice attacks with combinatorial methods, notably `dual-hybrid` and `primal-hybrid` attacks.

2.3.3 The Ring-LWE problem

The ring learning with errors (RLWE) problem, introduced by Lyubashevsky et al. [114,115], extends the LWE problem to algebraic rings. These works introduce the family of *ideal lattices*, which consists of lattices of the form $\sigma(I) \subset H_{\mathfrak{m}}$ for canonical embedding σ and ideals $I \subset \mathcal{R}_{\mathfrak{m}}$. While the original definition of RLWE in [114, 115] is in the "dual" form, we present the "non-dual" form, which is equivalent to the dual form up to the choice of error distribution [61, 131]. Following the notation in Section 2.2.2, we write $\mathcal{R}_{\mathfrak{m},q} = \mathbb{Z}[X]/\left(\Phi_{\mathfrak{m}}(X), q\right)$, and $\mathfrak{n} = \varphi(\mathfrak{m})$ is the polynomial degree.

Definition 2.3.5 (RLWE distribution $\mathcal{D}_{\boldsymbol{s},\chi}$)**.** Given $\boldsymbol{s} \in \mathcal{R}_{\mathfrak{m},q}$ and an error distribution χ over $\mathcal{R}_{\mathfrak{m},q}$, the RLWE distribution $\mathcal{D}_{\boldsymbol{s},\chi}$ over $\mathcal{R}^2_{\mathfrak{m},q}$ is sampled by choosing a uniformly random $\boldsymbol{a} \in \mathcal{R}_{\mathfrak{m},q}$ and an error $\mathbf{e} \leftarrow \chi$, and outputting $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \mathbf{e} \bmod q) \in \mathcal{R}^2_{\mathfrak{m},q}$.

Compared to LWE, RLWE provides better efficiency and amortized storage. The efficiency comes from operating over ring elements in $\mathcal{R}$, enabling faster multiplications via the fast Fourier transform (FFT). In terms of storage, a single RLWE sample contains the same information as $\mathfrak{n}$ LWE samples with the same secret vector $\overrightarrow{s}$. An RLWE sample is $2\mathfrak{n}\log q$ bits, whereas the size of $\mathfrak{n}$ LWE samples is $(\mathfrak{n}^2 + \mathfrak{n})\log q$ bits. Therefore, storing $\mathfrak{n}$ LWE samples incurs a higher storage cost. As a remark, if we restrict our attention to fresh ciphertexts, then the random components (the $\mathbf{A}$ part in LWE and the $\boldsymbol{a}$ part in RLWE) can be generated from a short PRNG seed. As such, the storage cost reduces to $\mathfrak{n}\log q$ plus the seed size for both LWE and RLWE cases.

Next, we describe the two versions of the RLWE problem.

Definition 2.3.6 (Search $\mathsf{RLWE}_{\mathfrak{m},m,q,\chi}$)**.** The search $\mathsf{RLWE}_{\mathfrak{m},m,q,\chi}$ problem is to recover $\boldsymbol{s} \in \mathcal{R}_{\mathfrak{m},q}$ from m RLWE instances $\{(\boldsymbol{a}_i, \boldsymbol{b}_i = \boldsymbol{a}_i \cdot \boldsymbol{s} + \mathbf{e} \bmod q)\}_{i\in[m]}$ sampled according to the RLWE distribution $\mathcal{D}_{\boldsymbol{s},\chi}$.

Definition 2.3.7 (Decision $\mathsf{RLWE}_{\mathfrak{m},m,q,\chi}$)**.** The decision $\mathsf{RLWE}_{\mathfrak{m},m,q,\chi}$ problem is to distinguish m RLWE instances $\{(\boldsymbol{a}_i, \boldsymbol{b}_i = \boldsymbol{a}_i \cdot \boldsymbol{s} + \mathbf{e} \bmod q)\}_{i\in[m]}$ sampled according to the RLWE distribution $\mathcal{D}_{\boldsymbol{s},\chi}$, from $\left\{(\boldsymbol{a}_i, \boldsymbol{b}_i) \xleftarrow{\$} \mathcal{R}^2_{\mathfrak{m},q}\right\}_{i\in[m]}$.

To date, no known attack on RLWE problems exploits its additional ring structure compared to LWE. Therefore, the security of an RLWE instance is

estimated by transforming it to a corresponding LWE instance with the same dimension $n = \mathfrak{n}$, modulus q and an appropriate error distribution.

As in the LWE problem, the error term can be sampled according to an n-dimensional spherical Gaussian distribution in the embedding space $H_{\mathfrak{m}}$. When these errors are sampled coefficient-wise, the distribution needs to be transformed from $H_{\mathfrak{m}}$ to $\mathcal{R}_{\mathfrak{m}}$ [114, 131]. A special case is when $\mathfrak{m}$ is a power of two, then the transformed distribution is still an $\mathfrak{n}$-dimensional spherical Gaussian distribution, albeit with a different standard deviation.

2.3.4 Module-SIS and Module-LWE problems

The module short integer solution (MSIS) and module learning with errors (MLWE) problems were formally introduced by Langlois and Stehle [99] in 2015. A module is an algebraic structure that generalizes rings and vector spaces. The MSIS and MLWE problems are based on *module lattices*, i.e. finitely generated modules over $\mathcal{R}_{\mathfrak{m}}$, generalizing both arbitrary lattices and ideal lattices.

Definition 2.3.8 ($\mathsf{MSIS}_{\mathfrak{m},n,m,q,B}$)**.** Given $\mathbf{A} \leftarrow \mathcal{R}_{\mathfrak{m},q}^{n \times m}$, the $\mathsf{MSIS}_{\mathfrak{m},n,m,q,B}$ problem is to find a non-zero $\overrightarrow{\boldsymbol{z}} \in \mathcal{R}_{\mathfrak{m}}^{m}$ such that $\mathbf{A} \cdot \overrightarrow{\boldsymbol{z}} = \overrightarrow{\mathbf{0}} \bmod q$ and $\left\| \overrightarrow{\boldsymbol{z}} \right\|_2 \leq B$.

The MSIS problem is a variant of SIS equipped with block matrices. Specifically, the matrix $\mathbf{A} \leftarrow \mathcal{R}_{\mathfrak{m},q}^{n \times m}$ is equivalent to an $\mathfrak{n}n \times \mathfrak{n}m$ dimensional matrix over $\mathbb{Z}_q$, where each $\mathfrak{n} \times \mathfrak{n}$ block is the rotation matrix of some element in $\mathcal{R}_{\mathfrak{m},q}$.

Definition 2.3.9 ($\mathsf{MLWE}_{\mathfrak{m},n,m,q,\chi}$)**.** Given $\overrightarrow{\boldsymbol{s}} \in \mathcal{R}_{\mathfrak{m},q}^{n}$ and an error distribution χ over $\mathcal{R}_{\mathfrak{m},q}^{m}$, the $\mathsf{MLWE}_{\mathfrak{m},n,m,q,\chi}$ problem is to distinguish $(\mathbf{A}, \mathbf{A} \cdot \overrightarrow{\boldsymbol{s}} + \overrightarrow{\boldsymbol{e}})$ for $\mathbf{A} \leftarrow \mathcal{R}_{\mathfrak{m},q}^{m \times n}$ and error vector $\overrightarrow{\boldsymbol{e}} \leftarrow \chi$, from $(\mathbf{A}, \overrightarrow{\boldsymbol{b}}) \leftarrow \mathcal{R}_{\mathfrak{m},q}^{m \times n} \times \mathcal{R}_{\mathfrak{m},q}^{m}$.

Similarly, the MLWE problem is an LWE problem equipped with $\mathfrak{n}m \times \mathfrak{n}n$ dimensional matrices with $\mathfrak{n} \times \mathfrak{n}$ dimensional blocks. Since no known attack on the MSIS and MLWE problems exploits their additional block structure, the hardness of MSIS and MLWE can be evaluated by reducing them to their SIS and LWE counterparts. The security of SIS and LWE are explained in Section 2.3.1 and 2.3.2, and can be estimated using Albrecht et al.'s lattice estimator [4].

2.4 Zero-knowledge proofs

A *proof system* is a protocol in which a prover $\mathcal{P}$ wishes to convince a verifier $\mathcal{V}$ that a given statement is true. The concept of *zero-knowledge* proofs, introduced

by Goldwasser et al. [88], adds a crucial requirement that the proof reveals nothing about the private input of $\mathcal{P}$, thereby protecting the prover's privacy.

Mathematically, a relation $\mathfrak{R}$ is defined as a set of instance-witness pairs $(\mathbb{x}, \mathbb{w})$, and the corresponding language is defined as $\mathfrak{L}_{\mathfrak{R}} = \{\mathbb{x} \mid \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathfrak{R}\}$. The instance $\mathbb{x}$ is an input to both $\mathcal{P}$ and $\mathcal{V}$, while the witness $\mathbb{w}$ is known only to $\mathcal{P}$. The prover $\mathcal{P}$ then generates a proof π to convince the verifier that there exists (or $\mathcal{P}$ knows) a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathfrak{R}$. The prover $\mathcal{P}$ sending π instead of $\mathbb{w}$ to the verifier $\mathcal{V}$ not only allows to achieve the zero-knowledge property, but also enables improved efficiency: π can be much smaller than $\mathbb{w}$, and verifying π can be faster than validating $(\mathbb{x}, \mathbb{w}) \in \mathfrak{R}$ directly.

Informally, a zero-knowledge proof protocol needs to satisfy the following properties:

- **Completeness**. If a statement is true, then the prover can convince the verifier.
- **Soundness**. If a statement is false, an honest verifier always rejects the proof except with negligible probability.
- **Zero-knowledge**. The proof does not reveal anything but the truth of the statement, in particular it does not reveal the prover's witness $\mathbb{w}$.

If the prover wants to prove the knowledge of the associated witness $\mathbb{w}$, a stronger notion of soundness is required:

- **Knowledge-soundness**. If the verifier accepts the proof, there exists a polynomial-time extractor algorithm with access to the prover that outputs the witness $\mathbb{w}$.

2.4.1 Lattice-based zero-knowledge proofs

Lattice-based Σ-protocols. Σ-protocols are three-round interactive proofs between a prover that knows an instance-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathfrak{R}$ and a verifier who only knows the instance $\mathbb{x}$, as illustrated in Figure 2.1.

A Σ-protocol is *public-coin* if the challenge value c is a public and random string, which is not derived in a complex or private manner. A Σ-protocol is expected to satisfy *completeness*, *honest-verifier zero-knowledge (HVZK)*, and *special soundness*, as formally defined in [15, Section 2.1]. In the Random Oracle (RO) model [18], applying the Fiat-Shamir transform [69] to a public coin Σ-protocol with these properties yields a non-interactive zero-knowledge (NIZK) proof or argument of knowledge.

A classic example of Σ-protocols is the discrete logarithm (DL) based Schnorr protocol [136], which achieves perfect completeness, honest-verifier zero-knowledge, and special soundness with a negligible error in a single protocol

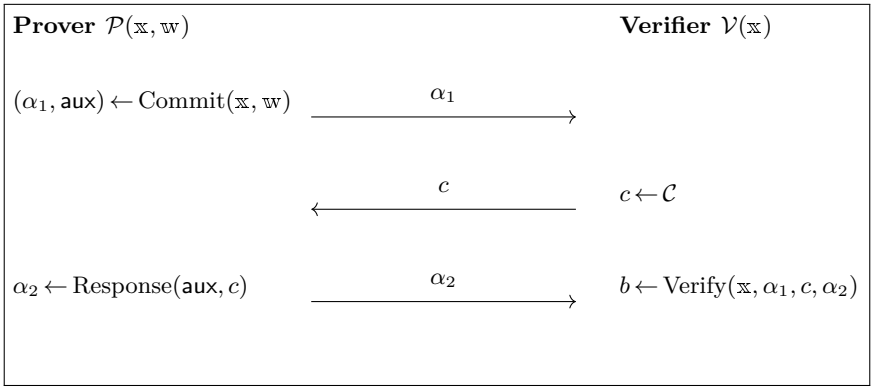


Figure 2.1: Diagram of a Σ-protocol.

run. In contrast, the behavior of lattice-based Σ-protocols deviates in two key aspects. First, the perfect completeness is relaxed: an honest prover can fail with probability at most α due to the use of *rejection sampling* [110,111], where an honest prover sometimes has to abort the protocol to achieve zero-knowledge. Second, lattice-based Σ-protocols typically exhibit a soundness gap, meaning a soundness extractor recovers a witness for a broader relation $\mathfrak{R}' \supset \mathfrak{R}$ instead of $\mathfrak{R}$. Due to these deviations, lattice-based Σ-protocols are sometimes referred to as Σ′-protocols [24] in the literature. Below, we present a typical example of a lattice-based Σ-protocol to illustrate these aspects in detail.

Let $\mathcal{R}$ denote the integer ring $\mathbb{Z}$ or a polynomial ring $\mathcal{R}_\mathfrak{m}$ and define $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. For suitable parameters, the matrix $\mathbf{A} \leftarrow \mathcal{R}_q^{n \times m}$ defines a one-way function that maps a small vector $\overrightarrow{s} \in \mathcal{R}^m$ to $\overrightarrow{t} \coloneqq \mathbf{A} \cdot \overrightarrow{s} \in \mathcal{R}_q^n$. The hardness of inverting this function relies on the SIS problem (when $\mathcal{R} = \mathbb{Z}$) or the MSIS problem (when $\mathcal{R} = \mathcal{R}_\mathfrak{m}$). For a given bound B, we define the relation

$$\mathfrak{R} = \left\{ (\overrightarrow{t}, \overrightarrow{s}, 1) \in \mathcal{R}_q^n \times \mathcal{R}^m \times \mathcal{R} \mid \mathbf{A} \cdot \overrightarrow{s} = \overrightarrow{t} \bmod q, \|\overrightarrow{s}\| \leq B \right\},$$

and its corresponding Σ-protocol is as follows.

1. The prover samples $\overrightarrow{y}$ from a suitable distribution D over $\mathcal{R}_q^m$ and commits to $\overrightarrow{y}$ by sending $\overrightarrow{w} = \mathbf{A} \cdot \overrightarrow{y} \bmod q$ to the verifier.
2. The verifier sends a challenge $c \leftarrow \mathcal{C}$ to the prover.
3. The prover computes $\overrightarrow{z} = c \cdot \overrightarrow{s} + \overrightarrow{y}$ and sends it to the verifier with probability $D(\overrightarrow{z}) / \left(M \cdot D'_{\overrightarrow{s}}(\overrightarrow{z})\right)$, where $D'_{\overrightarrow{s}}(\cdot)$ denotes the distribution generated by first picking $\overrightarrow{y} \leftarrow D$ and adding it to $c \cdot \overrightarrow{s}$ for $c \leftarrow \mathcal{C}$, M is a parameter such that $D(\overrightarrow{a}) \leq M \cdot D'_{\overrightarrow{s}}(\overrightarrow{a}), \forall \overrightarrow{a} \in \mathcal{R}^m$; otherwise aborts.

4. The verifier checks if $\mathbf{A} \cdot \overrightarrow{z} = c \cdot \overrightarrow{t} + \overrightarrow{w}$ and $\|\overrightarrow{z}\| \leq B'$, where $B' > B$ is determined by the distribution D.

For the protocol to be zero-knowledge, the prover's response $\overrightarrow{z}$ should not depend on the witness $\overrightarrow{s}$. This is achieved via the rejection sampling in Step 3, where the resulting distribution of $\overrightarrow{z}$ is

$$D'_{\overrightarrow{s}}(\overrightarrow{z}) \cdot D(\overrightarrow{z}) / \left(M \cdot D'_{\overrightarrow{s}}(\overrightarrow{z})\right) = D(\overrightarrow{z})/M,$$

and the expected number of runs needed to output a sample is M.

Furthermore, the soundness extractor of this protocol only extracts a witness for the relaxed relation

$$\mathfrak{R}' = \left\{ (\overrightarrow{t}, \overrightarrow{s}, \overline{c}) \in \mathcal{R}_q^n \times \mathcal{R}^m \times \mathcal{R} \mid \mathbf{A} \cdot \overrightarrow{s} = \overline{c} \cdot \overrightarrow{t} \bmod q, \|\overrightarrow{s}\| \leq 2B', \overline{c} \in \overline{\mathcal{C}} \right\},$$

where $\overline{\mathcal{C}}$ is the set of differences of elements of $\mathcal{C}$ except for 0 and $B' > B$ is a larger bound. Such an *approximate* proof with a soundness gap is sufficient for applications such as digital signatures [16, 62, 91, 110, 111] and commitment schemes [2, 17, 25], the latter of which will be elaborated on next. On the other hand, applications such as lattice-based verifiable encryptions [112] benefit from *exact* proofs [113], which will be briefly described after covering commitments.

Lattice-based commitment schemes. In line with [8, 66], we define commitment schemes as follows.

Definition 2.4.1 (Commitment scheme)**.** A commitment scheme $\mathcal{CT} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Open})$ includes the following probabilistic polynomial-time algorithms:

- $\mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$: for a given security parameter λ, it returns public parameters pp, which define a message space $\mathcal{S}_M$, a randomness space $\mathcal{S}_R$ and a commitment space $\mathcal{S}_C$.
- $\mathcal{CT}.\mathsf{Com}_{\mathsf{pp}}(m, r)$: for a given message $m \in \mathcal{S}_M$ and some randomness $r \leftarrow \mathcal{S}_R$, it returns a commitment $\mathtt{C}_m$.
- $\mathcal{CT}.\mathsf{Open}_{\mathsf{pp}}(m, r, \mathtt{C})$: for a given tuple $(m, r, \mathtt{C}) \in \mathcal{S}_M \times \mathcal{S}_R \times \mathcal{S}_C$, it returns either acc or rej.

In lattice-based commitment schemes such as the Ajtai commitment [2], the BDLOP commitment [17], and the combined ABDLOP commitment [113], the opening algorithm Open is *relaxed*. Specifically, Open takes a relaxation factor $\overline{c} \in \overline{\mathcal{C}}$ as input, in addition to the commitment $\mathtt{C}$ and the message-randomness pair (m, r).

Let R denote the $(2d)$-th cyclotomic ring $\mathbb{Z}[X]/\left(X^d + 1\right)$ where d is a power-of-two, and $\mathsf{R}_q \coloneqq \mathsf{R}/q\mathsf{R}$. We describe the ABDLOP commitment with the

challenge space

$$\mathcal{C}_{k,\eta} = \left\{ \boldsymbol{c} \in \mathsf{R}_q : \ \sigma_{-1}(\boldsymbol{c}) = \boldsymbol{c} \text{ and } \sqrt[2k]{\left\|\boldsymbol{c}^{2k}\right\|_1} \leq \eta \right\},$$

which is exponentially large in the security parameter λ for soundness purposes.

- $\mathsf{ABDLOP}.\mathsf{KeyGen}(1^\lambda)$: the public parameters pp are generated as

$$\mathsf{pp} = (\mathbf{A_1}, \mathbf{A_2}, \mathbf{B}) \leftarrow \mathsf{R}_q^{\omega \times m_1} \times \mathsf{R}_q^{\omega \times m_2} \times \mathsf{R}_q^{u \times m_2}.$$

- $\mathsf{ABDLOP}.\mathsf{Com}_{\mathsf{pp}}(\overrightarrow{\boldsymbol{s_1}}, \overrightarrow{\boldsymbol{m}}, \overrightarrow{\boldsymbol{s_2}})$: to commit to a small message $\overrightarrow{\boldsymbol{s_1}} \in \mathsf{R}_q^{m_1}$ where $\left\|\overrightarrow{\boldsymbol{s_1}}\right\| \leq \alpha$ and an arbitrarily large message $\overrightarrow{\boldsymbol{m}} \in \mathsf{R}_q^{u}$, one samples a small randomness $\overrightarrow{\boldsymbol{s_2}} \leftarrow \chi^{m_2}$ where χ is a distribution over R_q with bounded infinity norm ν and computes

$$\begin{bmatrix} \overrightarrow{\boldsymbol{t_A}} \\ \overrightarrow{\boldsymbol{t_B}} \end{bmatrix} = \begin{bmatrix} \mathbf{A_1} \\ \mathbf{0} \end{bmatrix} \cdot \overrightarrow{\boldsymbol{s_1}} + \begin{bmatrix} \mathbf{A_2} \\ \mathbf{B} \end{bmatrix} \cdot \overrightarrow{\boldsymbol{s_2}} + \begin{bmatrix} \mathbf{0} \\ \overrightarrow{\boldsymbol{m}} \end{bmatrix} \mod q.$$

  As such, the ABDLOP scheme not only allows the commitment of large messages $\overrightarrow{\boldsymbol{m}}$ as in the BDLOP commitment, but also compresses small messages $\overrightarrow{\boldsymbol{s_1}}$ as in the Ajtai commitment.
- $\mathsf{ABDLOP}.\mathsf{Open}_{\mathsf{pp}}\left((\overrightarrow{\boldsymbol{s_1}}, \overrightarrow{\boldsymbol{m}}, \overrightarrow{\boldsymbol{s_2}}), [\overrightarrow{\boldsymbol{t_A}}\ \overrightarrow{\boldsymbol{t_B}}]^\intercal, \bar{c}\right)$: check if the following relations are satisfied

$$\mathsf{ABDLOP}.\mathsf{Com}_{\mathsf{pp}}\left(\overrightarrow{\boldsymbol{s_1}}, \overrightarrow{\boldsymbol{m}}, \overrightarrow{\boldsymbol{s_2}}\right) = \begin{bmatrix} \overrightarrow{\boldsymbol{t_A}} \\ \overrightarrow{\boldsymbol{t_B}} \end{bmatrix}$$

$$\left\|\bar{c}\overrightarrow{\boldsymbol{s_1}}\right\|_2 \leq B_1 \text{ and } \left\|\bar{c}\overrightarrow{\boldsymbol{s_2}}\right\|_2 \leq B_2,$$

  where $B_1 = B_1(\alpha)$ and $B_2 = B_2(\nu)$ are pre-determined constants.

As shown in [32, 113], the ABDLOP commitment is hiding and binding under the MLWE and MSIS assumptions.

Lemma 2.4.1. *If the* $\mathsf{MLWE}_{2d,m_2-(\omega+u),\omega+u,q,\chi}$ *problem is hard, then* ABDLOP *is computationally hiding. If* $\mathsf{MSIS}_{2d,\omega,m_1+m_2,4\eta\sqrt{B_1^2+B_2^2}}$ *is hard, then* ABDLOP *is computationally binding with respect to the relaxed openings.*

Exact lattice-based proofs. [113] develops *exact* lattice-based commit-and-prove protocols based on the ABDLOP commitment scheme, which proves relations between committed messages as well as their exact norms (or bounds of norms). This proof is realized as a specific application of a more general protocol that proves knowledge of constant coefficients of quadratic relations over polynomial rings for committed messages.

To illustrate, consider the simple example of proving $\|\boldsymbol{s}\| = \beta$. Let σ denote the automorphism in R that maps $a(X)$ to $a(X^{-1})$, then $\|\boldsymbol{s}\|^2$ corresponds to the constant coefficient of $s(X) \cdot \sigma(s(X))$. Given $\boldsymbol{z} = \boldsymbol{c} \cdot \boldsymbol{s} + \boldsymbol{y}$ where the challenge $\boldsymbol{c}$ is invariant under σ, then a verifier can compute

$$\sigma(\boldsymbol{z}) \cdot \boldsymbol{z} - \boldsymbol{c}^2 \cdot \beta^2 = \left(\sigma(\boldsymbol{s}) \cdot \boldsymbol{s} - \beta^2\right) \cdot \boldsymbol{c}^2 + \boldsymbol{g_1} \cdot \boldsymbol{c} + \boldsymbol{g_0}, \tag{2.5}$$

where $\boldsymbol{g_1} = \sigma(\boldsymbol{s}) \cdot \boldsymbol{y} + \boldsymbol{s} \cdot \sigma(\boldsymbol{y})$ and $\boldsymbol{g_0} = \sigma(\boldsymbol{y}) \cdot \boldsymbol{y}$ are committed by the prover prior to receiving $\boldsymbol{c}$. Equation (2.5) is a quadratic equation in the variable $\boldsymbol{c}$, hence if the prover proves the constant coefficient of the polynomial

$$\sigma(\boldsymbol{z}) \cdot \boldsymbol{z} - \boldsymbol{c}^2 \cdot \beta^2 - \boldsymbol{g_1} \cdot \boldsymbol{c} - \boldsymbol{g_0}$$

vanishes, then it implies that $\|\boldsymbol{s}\|^2 = \beta^2 \bmod q$ holds with high probability. Finally, to ensure there is no wraparound modulo q, i.e. $\|\boldsymbol{s}\|^2 = \beta^2$, the approximate range proof technique [84] is integrated. Proving $\|\boldsymbol{s}\| \leq \beta$ can be similarly transformed into vanishing constant proofs of quadratic relations, and we refer to [113, Section 5.2] for more details.

2.4.2 zk-SNARKs

In the class of zero-knowledge proofs, a particularly interesting concept is zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK), which is:

- **succinct**: the proof size is polynomial in the security parameter, and the verifier's computation time is polynomial in the security parameter and the instance size,
- **non-interactive**: the proof consists of a single message sent from the prover to the verifier, without rounds of interactions,
- **argument of knowledge**: a computationally bounded prover cannot generate a valid proof without knowing the witness $\mathbb{w}$,
- **zero-knowledge**: the verifier does not learn anything about the statement beyond its validity.

In zk-SNARKs, a popular representation for arithmetic circuits is a rank-1 constraint system (R1CS). An R1CS instance is specified by three $m \times n$ matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ with entries from a field $\mathbb{F}$ and is satisfiable if and only if there exists a vector $\overrightarrow{z} = [1, \mathbb{x}, \mathbb{w}] \in \mathbb{F}^n$ such that

$$(\mathbf{A} \cdot \overrightarrow{z}) \circ (\mathbf{B} \cdot \overrightarrow{z}) = \mathbf{C} \cdot \overrightarrow{z},$$

where $\circ$ stands for the component-wise multiplication. At a high level, [21, 49] proves the satisfiability of an R1CS instance via two sub-checks:

- **Lincheck**: given vectors $\overrightarrow{z}, \mathbf{A}, \overrightarrow{a}, \mathbf{B}, \overrightarrow{b}, \mathbf{C}, \overrightarrow{c}$, check whether

$$\mathbf{A} \cdot \overrightarrow{z} = \overrightarrow{a},\ \mathbf{B} \cdot \overrightarrow{z} = \overrightarrow{b},\ \text{and } \mathbf{C} \cdot \overrightarrow{z} = \overrightarrow{c}.$$

- **Rowcheck**: given vectors $\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c} \in \mathbb{F}^m$, check whether $\overrightarrow{a} \circ \overrightarrow{b} = \overrightarrow{c}$.

The design of interactive oracle proof (IOP) for Linchecks and Rowchecks only admits trivial protocols where all the entries are queried. To achieve sublinear query complexity, the vectors $\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}, \overrightarrow{z}$ need to be encoded via some error-correcting code. The works [21, 49] use the Reed-Solomon (RS) codes [134], and these checks are eventually translated into rational constraints [49] on RS-encodings. As an example, define polynomials f_a, f_b, f_c that interpolate vectors $\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}$ over some evaluation domain H. Let Z_H denote a polynomial that vanishes on H that satisfies $Z_H(a) = 0 \Leftrightarrow a \in H$. Then the rowcheck $\overrightarrow{a} \circ \overrightarrow{b} = \overrightarrow{c}$ holds if and only if Z_H divides the polynomial $f_a \cdot f_b - f_c$, i.e. the rational expression

$$Q \coloneqq (f_a \cdot f_b - f_c)/Z_H$$

is a polynomial of bounded degree. In [21, 49], the low-degree properties are tested via the fast Reed-Solomon interactive oracle proofs of proximity (FRI) [20] protocol.

2.5 Verifiable computation

Verifiable computation (VC), introduced by [82,130], allows a client with limited computational resources to delegate the evaluation of a function f on several inputs while retaining the ability to verify that the computation of f has been carried out correctly on each input. We present its formal definition adapted from [130] as follows.

Definition 2.5.1 (Verifiable computation)**.** A verifiable computation scheme $\mathcal{VC} = (\mathsf{KeyGen}, \mathsf{Compute}, \mathsf{Verify})$ consists of the following algorithms:

- $\mathcal{VC}.\mathsf{KeyGen}(1^\lambda, f)$: given the security parameter λ and the function f to compute, it returns a public evaluation key EK_f and a public verification key VK_f.
- $\mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_f}(x)$: given the evaluation key EK_f and input x, it returns $y = f(x)$ and a proof π_y of y's correctness.
- $\mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_f}(x, y, \pi_y)$: given the verification key VK_f, x, y and the proof π_y, it returns acc if $y = f(x)$, and rej otherwise.

Correctness. A verifiable computation scheme $\mathcal{VC}$ is correct for functions in $\mathcal{F}$ if for any $f \in \mathcal{F}$, and $x \in \text{Domain}(f)$, the following probability

$$\Pr\left[\ \mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_f}(x, y, \pi_y) = \mathsf{acc} \ \middle|\ \begin{array}{c}(\mathsf{EK}_f, \mathsf{VK}_f) \leftarrow \mathcal{VC}.\mathsf{KeyGen}(1^\lambda, f)\\ (y, \pi_y) \leftarrow \mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_f}(x)\end{array}\ \right]$$

is no lower than $1 - \mathsf{negl}(\lambda)$.

Security. A verifiable computation scheme $\mathcal{VC}$ is secure for a function f, if for any adversary $\mathcal{A}$ that runs in probabilistic polynomial time, the following probability

$$\Pr\left[\ \begin{array}{c}\mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_f}(\hat{x}, \hat{y}, \hat{\pi}_{\hat{y}}) = \mathsf{acc}\\ \wedge\ \hat{y} \neq f(\hat{x})\end{array}\ \middle|\ \begin{array}{c}(\mathsf{EK}_f, \mathsf{VK}_f) \leftarrow \mathcal{VC}.\mathsf{KeyGen}(1^\lambda, f)\\ (\hat{x}, \hat{y}, \hat{\pi}_{\hat{y}}) \leftarrow \mathcal{A}(\mathsf{EK}_f, \mathsf{VK}_f)\end{array}\ \right]$$

is $\mathsf{negl}(\lambda)$.

Outsourceability. A verifiable computation scheme $\mathcal{VC}$ is outsourceable for a function f, if (1) $\mathcal{VC}.\mathsf{KeyGen}(1^\lambda, f)$ is a one-time operation whose cost is amortised over many computations, and (2) for any x, y, π_y, the time required to compute $\mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_f}(x, y, \pi_y)$ is $o(T)$, where T is the time required to compute $f(x)$.

As a further extension, a delegated function f may take inputs from both parties: the client's input x and the server's input w. A VC scheme is *zero-knowledge* if the client learns nothing about w beyond the computation output $y = f(x, w)$. zk-SNARKs provide a concrete instantiation of such zero-knowledge VC.

Chapter 3

Fully homomorphic encryption

3.1 Overview of FHE

As introduced in Chapter 1, FHE is an encryption scheme that allows computations over encrypted data. More formally, we define a secret-key HE scheme with plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$ as follows [35, 47, 83, 92].

Definition 3.1.1 (Homomorphic Encryption)**.** An HE scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ includes the following polynomial-time algorithms:

- $\mathcal{E}.\mathsf{KeyGen}(1^\lambda)$: given the security parameter λ, it returns a secret key sk and a public evaluation key evk.
- $\mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}(\{m_i\}_{i\in[r]})$: given the secret key sk and plaintexts $\{m_i\}_{i\in[r]} \in \mathcal{P}^r$, it returns ciphertexts $\{\mathtt{ct}_i\}_{i\in[r]} \in \mathcal{C}^r$, which can also be denoted as $\{\mathtt{ct}[m_i]\}_{i\in[r]}$.
- $\mathcal{E}.\mathsf{Dec}_{\mathsf{sk}}(\{\mathtt{ct}_i\}_{i\in[r]})$: given the secret key sk and ciphertexts $\{\mathtt{ct}_i\}_{i\in[r]} \in \mathcal{C}^r$, it returns plaintexts $\{m_i\}_{i\in[r]} \in \mathcal{P}^r$.
- $\mathcal{E}.\mathsf{Eval}_{\mathsf{evk}}(f, \{\mathtt{ct}_i\}_{i\in[\ell]})$: given the public evaluation key evk, a function $f : \mathcal{P}^\ell \to \mathcal{P}^r$ and a set of ciphertexts $\{\mathtt{ct}_i\}_{i\in[\ell]} \in \mathcal{C}^\ell$, it returns ciphertexts $\{\mathtt{ct}'_i\}_{i\in[r]} \in \mathcal{C}^r$.

Correctness. An HE scheme $\mathcal{E}$ is correct for functions in $\mathcal{F}$ if for any $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, function $f \in \mathcal{F}$, plaintexts $\{m_i\}_{i\in[\ell]}$ and their encryptions $\{\mathtt{ct}_i\} \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}(\{m_i\})$, the relation

$$\mathcal{E}.\mathsf{Dec}_{\mathsf{sk}}\left(\mathcal{E}.\,\mathsf{Eval}_{\mathsf{evk}}(f, \mathtt{ct}_1, \ldots, \mathtt{ct}_\ell)\right) = f(m_1, \ldots, m_\ell),$$

holds with probability no lower than $1 - \mathsf{negl}(\lambda)$.

CPA security. An HE scheme $\mathcal{E}$ is IND-CPA secure if for any probabilistic polynomial-time adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathcal{A}^{\mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}(\cdot)}(1^\lambda, \mathsf{evk}) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}(0)}(1^\lambda, \mathsf{evk}) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where the probability is over $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, the coins of $\mathcal{E}.\mathsf{Enc}$ and the coins of the adversary $\mathcal{A}$.

3.2 RLWE ciphertexts, encodings and operations

Given a cyclotomic ring $\mathcal{R}_{\mathfrak{m},q} = \mathbb{Z}[X]/(\Phi_\mathfrak{m}(X), q)$ with polynomial degree $\mathfrak{n} = \varphi(\mathfrak{m})$ and an error distribution χ over $\mathcal{R}_{\mathfrak{m},q}$, an RLWE encryption of $\boldsymbol{m} \in \mathcal{R}_{\mathfrak{m},q}$ under the secret key $\boldsymbol{s} \in \mathcal{R}$ is defined as

$$\mathsf{RLWE}^{\mathfrak{n},q}_{\boldsymbol{s}}(m) = (\boldsymbol{b}, \boldsymbol{a}) = (-\boldsymbol{a}\cdot\boldsymbol{s} + \boldsymbol{m} + \boldsymbol{e} \bmod q,\ \boldsymbol{a}) \in \mathcal{R}^2_{\mathfrak{m},q},$$

where $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_{\mathfrak{m},q}$ and $\boldsymbol{e} \leftarrow \chi$. The message $\boldsymbol{m} \in \mathcal{R}_{\mathfrak{m},q}$ is masked additively using the pair $(-\boldsymbol{a}\cdot\boldsymbol{s} + \boldsymbol{e} \bmod q, -\boldsymbol{a})$, which is indistinguishable from uniform under the hardness assumption of the $\mathsf{RLWE}_{\mathfrak{m},1,q,\chi}$ problem.

3.2.1 RLWE encodings and message packings

BGV and BFV encoding. For a given plaintext modulus $t \ll q$, let $\mathcal{R}_{\mathfrak{m},t} = \mathbb{Z}[X]/(\Phi_\mathfrak{m}(X), t)$ denote the plaintext space and $\Delta = q/t$ denote a scaling factor. The BGV [34] and BFV [33,68] homomorphic encryption schemes provide two approaches to encrypt a message $\overline{\boldsymbol{m}} \in \mathcal{R}_{\mathfrak{m},t}$:

$$\mathsf{BGV}^{\mathfrak{n},q}_{s}(\overline{\boldsymbol{m}}) = (\boldsymbol{b}, \boldsymbol{a}) = (-\boldsymbol{a}\cdot\boldsymbol{s} + \overline{\boldsymbol{m}} + t\cdot\boldsymbol{e} \bmod q,\ \boldsymbol{a}) \in \mathcal{R}^2_{\mathfrak{m},q}$$

$$\mathsf{BFV}^{\mathfrak{n},q}_{s}(\overline{\boldsymbol{m}}) = (\boldsymbol{b}, \boldsymbol{a}) = (-\boldsymbol{a}\cdot\boldsymbol{s} + \lfloor\Delta\cdot\overline{\boldsymbol{m}}\rceil + \boldsymbol{e} \bmod q,\ \boldsymbol{a}) \in \mathcal{R}^2_{\mathfrak{m},q}.$$

In other words, in the encoded messages $(\boldsymbol{b} + \boldsymbol{a}\cdot\boldsymbol{s} \bmod q)$, the plaintext $\overline{\boldsymbol{m}}$ resides in the least significant bits in BGV and the most significant bits (i.e. scaled by Δ) in BFV. Therefore, given an RLWE ciphertext $\mathsf{ct} = (\boldsymbol{c_0}, \boldsymbol{c_1})$, its BGV decryption is $[[\boldsymbol{c_0} + \boldsymbol{c_1}\cdot\boldsymbol{s}]_q]_t$ and its BFV decryption is $\lfloor\frac{t}{q}[\boldsymbol{c_0} + \boldsymbol{c_1}\cdot\boldsymbol{s}]_q\rceil$.

As a remark, when $\gcd(t, q) = 1$, we can transform between BGV and BFV ciphertexts as in [5, Appendix A]. Given $\mathsf{BGV}(\overline{\boldsymbol{m}})$, multiplication by $(t^{-1} \bmod q)$ maps it to $\mathsf{BFV}(-\overline{\boldsymbol{m}}/q)$. Conversely, given $\mathsf{BFV}(\overline{\boldsymbol{m}})$, multiplication by t maps it to $\mathsf{BGV}(-q\cdot\overline{\boldsymbol{m}})$.

Message packing in BGV and BFV. When the plaintext modulus $t = p$ is a prime and $\gcd(p, \mathfrak{m}) = 1$, the polynomial $\Phi_{\mathfrak{m}}(X)$ splits modulo p into ℓ irreducible factors of same degree d

$$\Phi_{\mathfrak{m}}(X) = F_1(X) \cdots F_\ell(X),$$

where d is the order of p modulo $\mathfrak{m}$, and $\ell = \mathfrak{n}/d$. Hence, CRT (Theorem 2.2.1) leads to the following isomorphism

$$\mathcal{R}_{\mathfrak{m},p} = \mathbb{Z}[X]/\big(\Phi_{\mathfrak{m}}(X), p\big) \to \prod_{i=1}^{\ell} \mathbb{Z}[X]/\big(p, F_i(X)\big)$$

$$\mu(X) \mapsto \prod_{i=1}^{\ell} \left(\mu(X) \bmod F_i(X)\right),$$

where each component $\mathbb{Z}[X]/\big(p, F_i(X)\big)$ is a finite field $\mathbb{F}_{p^d}$ referred to as a *slot*. Using the inverse CRT, ℓ values in $\mathbb{F}_{p^d}$ can be packed into a single plaintext in $\mathcal{R}_{\mathfrak{m},p}$. As such, arithmetic operations on plaintexts naturally apply to all slots, enabling SIMD (single-instruction multiple-data) operations.

In real-world FHE applications, messages are normally small integers rather than elements of an extension field $\mathbb{F}_{p^d}$ with $d > 1$. To maximise the packing capacity for such messages, the plaintext modulus t is commonly chosen as a prime p satisfying $p = 1 \bmod \mathfrak{m}$ to provide $\mathfrak{n}$ SIMD slots, all with extension degree $d = 1$. For example, in our private decision tree evaluation (Chapter 8) and encrypted database querying (Chapter 9), we choose parameters $\big(\mathfrak{m} = 2^{15}, t = 65537\big)$ and $(\mathfrak{m} = 99928, t = 99929)$, respectively.

GBFV encoding. In the GBFV [81] homomorphic encryption scheme, the plaintext modulus $t \ll q$ is generalised into a polynomial $\boldsymbol{t} = t(X)$ where $\|t(X)\|^{\text{can}} \ll q$. The use of a polynomial modulus was first proposed in [34] and implemented in a special case in [46].

The resulting plaintext space is $\mathcal{R}_{\mathfrak{m},\boldsymbol{t}} = \mathbb{Z}[X]/\left(\Phi_{\mathfrak{m}}(X), \boldsymbol{t}\right)$, and for $\boldsymbol{m} \in \mathcal{R}_{\mathfrak{m},\boldsymbol{t}}$, its canonical representative in $\mathcal{R}_{\mathfrak{m}}$ is computed using the following $\mathsf{Flatten}_{\boldsymbol{t}}$ function:

$$\mathsf{Flatten}_{\boldsymbol{t}} : \mathcal{R}_{\mathfrak{m},\boldsymbol{t}} \to \mathcal{R}_{\mathfrak{m}} : \ \boldsymbol{m} \mapsto \boldsymbol{t} \cdot \left[\frac{\boldsymbol{m}}{\boldsymbol{t}}\right]_1,$$

where $[\cdot]_1$ denotes the coefficient-wise centered reduction of polynomials with non-integral coefficients, as introduced in Section 2.1. Let $\Delta = q/t(X) \in \mathcal{K}_m$ denote the scaling factor, the GBFV encryption of $\overline{\boldsymbol{m}} \in \mathcal{R}_{\mathfrak{m},\boldsymbol{t}}$ is

$$\mathsf{GBFV}_s^{\mathfrak{n},q}(\overline{\boldsymbol{m}}) = (\boldsymbol{b}, \boldsymbol{a}) = (-\boldsymbol{a} \cdot \boldsymbol{s} + \lfloor \Delta \cdot \overline{\boldsymbol{m}} \rceil + \boldsymbol{e} \bmod q, \ \boldsymbol{a}) \in \mathcal{R}_{\mathfrak{m},q}^2,$$

where $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_{\mathfrak{m},q}$ and $\boldsymbol{e} \leftarrow \chi$. As such, given an RLWE ciphertext $\mathsf{ct} = (\boldsymbol{c_0}, \boldsymbol{c_1})$, its GBFV decryption $\lfloor \frac{1}{\Delta}[\boldsymbol{c_0} + \boldsymbol{c_1} \cdot \boldsymbol{s}]_q \rceil$ recovers the canonical representative of the encoded plaintext.

Message packing in GBFV. For a generalized $\boldsymbol{t} = t(X)$, the corresponding modulus is $p = \mathrm{Con}\left(\Phi_{\mathfrak{m}}(X), t(X)\right)$ (see Definition 2.2.7). Since $p \in \left(\Phi_{\mathfrak{m}}(X), t(X)\right)$, there exists $\beta(X) \in \mathbb{Z}[X]$ such that $p = t(X) \cdot \beta(X) \bmod \Phi_{\mathfrak{m}}(X)$, which implies

$$p\mathcal{R}_{\mathfrak{m}} = \boldsymbol{t}\mathcal{R}_{\mathfrak{m}} \cdot \boldsymbol{\beta}\mathcal{R}_{\mathfrak{m}}. \tag{3.1}$$

When the modulus p is a prime and $\gcd(p, \mathfrak{m}) = 1$, the ideal $p\mathcal{R}_{\mathfrak{m}}$ splits into a product of ℓ prime ideals, analogous to the BFV and BGV settings:

$$p\mathcal{R}_{\mathfrak{m}} = \prod_{i=1}^{\ell} \big(p, F_i(X)\big), \tag{3.2}$$

where each F_i is an irreducible polynomial modulo p of degree d (the order of p modulo $\mathfrak{m}$), and $\ell = \mathfrak{n}/d$. Combining Equations (3.1) and (3.2), the ideal $\boldsymbol{t}\mathcal{R}_{\mathfrak{m}}$ can be factorized as

$$\boldsymbol{t}\mathcal{R}_{\mathfrak{m}} = \prod_{i \in I} \big(p, F_i(X)\big) = \big(t'(X), p\big), \tag{3.3}$$

where $I \subset [\ell]$ is an index set of cardinality ℓ', and $t'(X) = \prod_{i \in I} F_i(X)$ is the greatest common divisor of $\Phi_{\mathfrak{m}}(X)$ and $t(X)$ modulo p, i.e. $\gcd(\mathbb{Z}_p[X]!\ \Phi_{\mathfrak{m}},\ \mathbb{Z}_p[X]!\ \boldsymbol{t})$. The polynomial $t'(X)$ can also be derived from the Sylvester matrix, as explained in Section 2.2.3.

To summarise, the ideal $I = (\Phi_{\mathfrak{m}}(X), t(X))$ in $\mathbb{Z}[X]$ satisfies

$$I = (\Phi_{\mathfrak{m}}(X), t(X)) = (\Phi_{\mathfrak{m}}(X), t(X), p) = (t'(X), p) \subset \mathbb{Z}[X],$$

hence CRT (Theorem 2.2.1) leads to the following isomorphism

$$\begin{aligned} \mathcal{R}_{\mathfrak{m}}/t\mathcal{R}_{\mathfrak{m}} = \mathbb{Z}[X]/\left(t'(X), p\right) &\to \prod_{i \in I} \mathbb{Z}[X]/\big(p, F_i(X)\big) \\ \mu(X) &\mapsto \prod_{i \in I} \left(\mu(X) \bmod F_i(X)\right), \end{aligned}$$

providing ℓ' SIMD slots in one plaintext element, and each slot is isomorphic to $\mathbb{F}_{p^d}$.

A particular simple case is when $t'(X) = t(X) \bmod p$, i.e. $t(X)$ divides $\Phi_{\mathfrak{m}}(X)$ over $\mathbb{Z}_p[X]$, as presented in the following lemma [81].

Lemma 3.2.1. *Let $r = rad(\mathfrak{m})$ denote the radical of $\mathfrak{m}$ and consider $0 < k < \mathfrak{n} = \varphi(\mathfrak{m})$ such that $k \mid (\mathfrak{m}/r)$. Let $t(X) = X^k - b$ for an integer b and $p = \Phi_r\left(b^{\mathfrak{m}/(rk)}\right)$. If p is a prime number and does not divide $\mathfrak{m}$, then the plaintext space $\mathcal{R}_{\mathfrak{m},t(X)}$ provides $\ell' = k/d$ SIMD slots where d is the order of p modulo $\mathfrak{m}$, each isomorphic to $\mathbb{F}_{p^d}$.*

The following example presents GBFV parameters whose resulting modulus p is the Goldilocks prime $\Phi_6(2^{32}) = 2^{64} - 2^{32} + 1$, a popular parameter for the base-field size in zero-knowledge proofs [132, 141, 149] as well as in our blind zkSNARK works (Chapter 9).

Example 3.2.1 (Goldilock prime and its extensions [81])**.** *We present the following parameter sets to obtain SIMD slots over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ for $p = 2^{64} - 2^{32} + 1$.*

Base field encoding. *Consider the following family of parameters indexed by $0 \le i \le 5,\ \ 6 \le j \le 16$:*

$$\left\{\mathfrak{m} = 3 \cdot 2^j, \quad t(X) = X^k - b \text{ with } k = 2^{i+j-6} \text{ and } b = 2^{2^i}\right\}. \tag{3.4}$$

Since $p = \Phi_{rad(\mathfrak{m})}\left(b^{\mathfrak{m}/(rad(\mathfrak{m})\cdot k)}\right)$ and $p \bmod \mathfrak{m} = 1$, Lemma 3.2.1 proves that the parameter indexed by (i, j) in (3.4) provides 2^{i+j-6} SIMD slots, each isomorphic to $\mathbb{F}_p$.

Quadratic extensions. *Consider the following family of parameters indexed by $0 \le i \le 5,\ \ 6 \le j \le 14$:*

$$\left\{\mathfrak{m} = 7 \cdot 3 \cdot 2^j, \quad t(X) = X^k - b \text{ with } k = 7 \cdot 2^{i+j-6} \text{ and } b = 2^{2^i}\right\}. \tag{3.5}$$

Following Section 2.2.2, the following isomorphism

$$\mathbb{Z}[X_1, X_2]/\left(\Phi_{3\cdot 2^j}(X_1), \Phi_7(X_2)\right) \cong \mathbb{Z}[X]/\left(\Phi_{7\cdot 3\cdot 2^j}(X)\right)$$

$$\sum_\ell a_\ell X_1^{\ell_1} X_2^{\ell_2} \mapsto \sum_\ell a_\ell X^{\ell_1 \cdot 7 + \ell_2 \cdot 3 \cdot 2^j}$$

maps X_1 to X^7. Therefore,

$$\begin{aligned}
&\mathbb{Z}[X]/\left(\Phi_{7\cdot 3\cdot 2^j}(X), X^{7\cdot 2^{i+j-6}} - 2^{2^i}\right) \\
&\cong \mathbb{Z}[X_1, X_2]/\left(\Phi_{3\cdot 2^j}(X_1), X_1^{2^{i+j-6}} - 2^{2^i}, \Phi_7(X_2)\right) \\
&\cong \mathbb{Z}[X_1, X_2]/\left(p, X_1^{2^{i+j-6}} - 2^{2^i}, \Phi_7(X_2)\right) \\
&\cong \mathbb{Z}[X_1]/\left(p, X_1^{2^{i+j-6}} - 2^{2^i}\right) \otimes \mathbb{Z}[X_2]/\left(p, \Phi_7(X_2)\right).
\end{aligned}$$

Since the ring $\mathbb{Z}[X_1]/\left(p, X_1^{2^{i+j-6}} - 2^{2^i}\right)$ *provides* 2^{i+j-6} *SIMD slots that are isomorphic to* $\mathbb{F}_p$ *(as in the base field encoding) and the ring* $\mathbb{Z}[X_2]/(p, \Phi_7(X_2))$ *provides 3 SIMD slots that are isomorphic to* $\mathbb{F}_{p^2}$ *(as* $\Phi_7(X_2)$ *has degree 6 and the order of p modulo 7 is 2), their tensor product provides* $3 \cdot 2^{i+j-6}$ *SIMD slots, each isomorphic to* $\mathbb{F}_{p^2}$*.*

3.2.2 Homomorphic operations on RLWE ciphertexts

Let $\overrightarrow{g} = [g_0, g_1, \ldots, g_{\ell-1}]^\intercal \in \mathbb{Z}^\ell$ denote a gadget vector in a single-radix system [34] or a mixed-radix system [94]. The *gadget decomposition* map $\overrightarrow{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^\ell$ satisfies $\langle \overrightarrow{g}^{-1}(a), \overrightarrow{g} \rangle = a \bmod q$ for any $a \in \mathbb{Z}_q$, which naturally extends coefficient-wise to $\overrightarrow{\boldsymbol{g}}^{-1} : \mathcal{R}_{\mathfrak{m},q} \to \mathcal{R}_{\mathfrak{m}}^\ell$.

Below we outline the homomorphic operations on RLWE ciphertexts as instantiated in the BFV/GBFV schemes, including addition (Add), multiplication (Mult), automorphism (Aut), key switching (KeySwitch) and modulus switching (ModSwitch).

- $\mathsf{Add}(\mathsf{ct}, \mathsf{ct}')$: given two RLWE ciphertexts $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_{\mathfrak{m},q}^2$ and $\mathsf{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1') \in \mathcal{R}_{\mathfrak{m},q}^2$, return

$$(\boldsymbol{c}_0 + \boldsymbol{c}_0', \boldsymbol{c}_1 + \boldsymbol{c}_1') \in \mathcal{R}_{\mathfrak{m},q}^2.$$

- $\mathsf{Add}(\mathsf{ct}, \boldsymbol{m})$: given ciphertext $\mathsf{ct} \in \mathcal{R}_{\mathfrak{m},q}^2$ and a message $\boldsymbol{m} \in \mathcal{R}_{\mathfrak{m},\boldsymbol{t}}$, compute a noiseless BFV/GBFV encryption

$$\mathsf{ct}' = (\lfloor \Delta \cdot \boldsymbol{m} \rceil, 0) \in \mathcal{R}_{\mathfrak{m},q}^2,$$

  and return $\mathsf{Add}(\mathsf{ct}, \mathsf{ct}')$.
- $\mathsf{KeySwitch}(\mathsf{ct}, \mathsf{ksk}_{\boldsymbol{s}\to\boldsymbol{s}'})$: given an RLWE ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_{\mathfrak{m},q}^2$ under the secret key $\boldsymbol{s}$ and a key-switching key $\mathsf{ksk}_{\boldsymbol{s}\to\boldsymbol{s}'} = (\overrightarrow{\boldsymbol{r_0}}, \overrightarrow{\boldsymbol{r_1}}) \in \mathcal{R}_{\mathfrak{m},q}^{\ell\times 2}$ that consists of ℓ RLWE encryptions of $\boldsymbol{s} \cdot g_i$ under $\boldsymbol{s}'$, return

$$(\boldsymbol{c}_0, \boldsymbol{0}) + \left(\langle \overrightarrow{\boldsymbol{g}}^{-1}(\boldsymbol{c_1}), \overrightarrow{\boldsymbol{r_0}} \rangle, \langle \overrightarrow{\boldsymbol{g}}^{-1}(\boldsymbol{c_1}), \overrightarrow{\boldsymbol{r_1}} \rangle\right).$$

- $\mathsf{Mult}(\mathsf{ct}, \mathsf{ct}', \mathsf{ksk}_{\boldsymbol{s^2}\to\boldsymbol{s}})$: given two RLWE ciphertexts $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_{\mathfrak{m},q}^2$, $\mathsf{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1') \in \mathcal{R}_{\mathfrak{m},q}^2$, and a key-switching key $\mathsf{ksk}_{\boldsymbol{s^2}\to\boldsymbol{s}} \in \mathcal{R}_{\mathfrak{m},q}^{\ell\times 2}$, compute

$$\mathsf{ct}'' = (\lfloor (\boldsymbol{c}_0 \cdot \boldsymbol{c}_0')/\Delta \rceil, \lfloor (\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0')/\Delta \rceil) \in \mathcal{R}_{\mathfrak{m},q}^2$$

$$\mathsf{ct}_{tmp} = (0, \lfloor (\boldsymbol{c}_1 \cdot \boldsymbol{c}_1')/\Delta \rceil) \in \mathcal{R}_{\mathfrak{m},q}^2,$$

  and return $\mathsf{Add}\left(\mathsf{ct}'', \mathsf{KeySwitch}(\mathsf{ct}_{tmp}, \mathsf{evk})\right)$.

- $\mathsf{Mult}(\mathsf{ct}, \boldsymbol{m})$: given an RLWE ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}^2_{\mathfrak{m},q}$ and a message $\boldsymbol{m} \in \mathcal{R}_{\mathfrak{m},\boldsymbol{t}}$, return

$$(\hat{\boldsymbol{m}} \cdot \boldsymbol{c}_0, \hat{\boldsymbol{m}} \cdot \boldsymbol{c}_1) \in \mathcal{R}^2_{\mathfrak{m},q},$$

  where $\hat{\boldsymbol{m}} = \mathsf{Flatten}(\boldsymbol{m})$ is used to control the noise growth.
- $\mathsf{Aut}(\mathsf{ct}, \tau, \mathsf{ksk}_{\tau(\boldsymbol{s})\to\boldsymbol{s}})$: given an RLWE ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}^2_{\mathfrak{m},q}$, an automorphism $\tau \in \mathcal{G}$ where

$$\mathcal{G} = \{\tau_i \in \mathrm{Gal}(\mathcal{K}_\mathfrak{m}/\mathbb{Q}) \mid \tau_i(\boldsymbol{t}) \in \boldsymbol{t}\mathcal{R}_\mathfrak{m}\},$$

  and a key-switching key $\mathsf{ksk}_{\tau(\boldsymbol{s})\to\boldsymbol{s}} \in \mathcal{R}^{\ell\times 2}_{\mathfrak{m},q}$, compute

$$\mathsf{ct}_{tmp} = ((\tau(\boldsymbol{t})/\boldsymbol{t}) \cdot \tau(\boldsymbol{c}_0), (\tau(\boldsymbol{t})/\boldsymbol{t}) \cdot \tau(\boldsymbol{c}_1)) \in \mathcal{R}^2_{\mathfrak{m},q},$$

  and return $\mathsf{KeySwitch}(\mathsf{ct}_{tmp}, \mathsf{ksk}_{\tau(\boldsymbol{s})\to\boldsymbol{s}})$. In particular, in BFV, $\boldsymbol{t} = t$ is an integer, hence $\tau_i(t) = t$ for any $\tau_i \in \mathrm{Gal}(\mathcal{K}_\mathfrak{m}/\mathbb{Q})$. In such cases, each τ_i induces a valid automorphism on $\mathcal{R}_{\mathfrak{m},t}$, i.e. $\mathcal{G} = \mathrm{Gal}(\mathcal{K}_\mathfrak{m}/\mathbb{Q})$.
  Furthermore, for GBFV with a binomial modulus (as described in Lemma 3.2.1), an explicit expression of

$$\mathcal{G} = \{\tau_i \in \mathrm{Gal}(\mathcal{K}_\mathfrak{m}/\mathbb{Q}) \mid i = 1 \bmod \mathfrak{m}/k\}$$

  is given and proven in [81].
- $\mathsf{ModSwitch}(\mathsf{ct}, q')$: given an RLWE ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}^2_{\mathfrak{m},q}$, return

$$\left(\lfloor \frac{q'}{q}\boldsymbol{c}_0 \rceil, \lfloor \frac{q'}{q}\boldsymbol{c}_1 \rceil\right) \in \mathcal{R}^2_{\mathfrak{m},q'}.$$

3.2.3 Basic procedures on RLWE ciphertexts

Ring switching. Ring switching [85] is a technique that converts RLWE ciphertexts from the ring $\mathcal{R}_{\mathfrak{m},q}$ to the ring $\mathcal{R}_{\mathfrak{m}',q}$, where the corresponding number field $\mathcal{K}_{\mathfrak{m}'}$ is a subfield of $\mathcal{K}_\mathfrak{m}$. This conversion relies on the fact that for any $\mathcal{K}_{\mathfrak{m}'}$-linear function $L : \mathcal{K}_\mathfrak{m} \to \mathcal{K}_{\mathfrak{m}'}$, there exists an element $\boldsymbol{r} \in \mathcal{K}_\mathfrak{m}$ such that $L(\boldsymbol{a}) = \mathrm{Tr}_{\mathcal{K}_\mathfrak{m}/\mathcal{K}_{\mathfrak{m}'}}(\boldsymbol{r} \cdot \boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{K}_\mathfrak{m}$. We define ring switching as follows.

- $\mathsf{RingSwitch}_{\mathfrak{m}\to\mathfrak{m}'}$: given a ciphertext $\mathsf{RLWE}^{\mathfrak{n},q}_{\boldsymbol{s}}(\boldsymbol{m})$, return $\mathsf{RLWE}^{\mathfrak{n}',q}_{\boldsymbol{s}'}(\boldsymbol{m}')$ where $\boldsymbol{m}'$ contains a subset of the messages encoded in $\boldsymbol{m}$.

Ring switching for *slot-wise encoded* RLWE ciphertexts was introduced in [85] and further applied in [81] and in Chapter 10. On the other hand, [13, 36] leveraged ring switching in the context of *coefficient-wise encoding*, particularly for rings with power-of-two cyclotomic orders $\mathfrak{m}$. In practice, ring switching is an effective method for reducing the server-to-client communication in FHE-based

applications. While the server may perform homomorphic computations over a large ring $\mathcal{R}_{\mathfrak{m},q}$, the client is often interested in only a subset of the encoded messages in the resulting ciphertext. In such cases, ring switching allows the server to map the ciphertext to a smaller ring $\mathcal{R}_{\mathfrak{m}',q}$, effectively compressing it before transmission.

GBFV-to-BFV conversion and packing. Consider GBFV and BFV ciphertexts defined over the same lattice dimension $\mathfrak{n}$ and ciphertext modulus q, but with different plaintext moduli: $t(X)$ for GBFV and $p = \mathrm{Con}\left(\Phi_{\mathfrak{m}}\left(X\right), t(X)\right)$ for BFV. Geelen and Vercauteren [81] introduced efficient techniques for almost noise-free conversions between these two schemes, as well as packing and unpacking methods based on automorphisms in the quotient group $\mathrm{Gal}(\mathcal{K}_{\mathfrak{m}}/\mathbb{Q})/\mathcal{G}$. Following the notation in Section 3.2.1, let ℓ and $\ell' \mid \ell$ denote the number of SIMD slots in BFV and GBFV, respectively.

In GBFV-to-BFV conversion, a GBFV ciphertext is converted to a BFV ciphertext, where the ℓ slots contain ℓ' messages from the GBFV slots and $\ell - \ell'$ irrelevant messages. In GBFV-to-BFV packing, ℓ/ℓ' GBFV ciphertexts are packed in a single BFV ciphertext, where the ℓ slots contain all the messages from the GBFV slots. The reverse procedures are BFV-to-GBFV conversion, where only ℓ' out of ℓ slots are extracted to a GBFV ciphertext, and BFV-to-GBFV unpacking, which maps 1 BFV ciphertext to ℓ/ℓ' GBFV ciphertexts, each recovering ℓ' slots out of ℓ.

These packing and unpacking techniques are particularly useful in reducing communication overhead in GBFV-based applications including blind zkSNARKs in Chapter 10. For instance, when a client needs to upload a large amount of input data, it can transmit BFV ciphertexts. The server then unpacks them into GBFV ciphertexts for evaluation. Similarly, the server can pack multiple GBFV ciphertexts into a single BFV ciphertext before sending results back to the client, minimizing the communication cost.

Bootstrapping. As discussed in Sections 2.3.3 and 3.2.1, RLWE ciphertexts inherently contain a noise component, which is crucial for ensuring security. This noise grows with homomorphic operations, and once it exceeds a certain threshold, correct decryption is not guaranteed. Bootstrapping [83] is therefore introduced to reduce the noise by homomorphically decrypting ciphertexts.

We focus below on the bootstrapping procedure for BFV, as BGV bootstrapping is nearly identical [80]. For GBFV, bootstrapping consists of the same building blocks as BFV bootstrapping [81], and additionally incorporates GBFV-to-BFV conversion for single-ciphertext bootstrapping and GBFV-to-BFV packing for batch bootstrapping. Given a BFV ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ with plaintext

modulus p^r and ciphertext modulus p^e, its decryption amounts to computing

$$\boldsymbol{w} \leftarrow [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{p^e}, \tag{3.6}$$

$$\boldsymbol{m} \leftarrow \left[\lfloor \boldsymbol{w}/p^{e-r} \rceil\right]_{p^r}. \tag{3.7}$$

Therefore, general bootstrapping [93] consists of the following steps:

- inner product, which homomorphically evaluates Equation (3.6),
- linear transformation, which moves the noisy coefficients of the encrypted plaintext into SIMD slots,
- digit extraction, which performs the slot-wise rounding procedure that corresponds to Equation (3.7),
- inverse linear transformation, which moves the noise-free slots of the encrypted plaintexts back into the coefficients.

For ciphertexts with sparsely packed slots, thin bootstrapping [45] provides a performance-optimized variant that uses similar steps arranged in a different order. Our work in Chapter 6 presents an optimized digit extraction procedure based on the theory of polynomial functions, yielding performance improvements of up to 2.6× for general bootstrapping and up to 2.0× for thin bootstrapping.

3.3 LWE ciphertexts, encodings and operations

Given an LWE dimension n, integer modulus q and an error distribution χ over $\mathbb{Z}$, an LWE encryption of $m \in \mathbb{Z}_q$ under the secret key $\overrightarrow{s} \in \mathbb{Z}^n$ is defined as

$$\mathsf{LWE}^{n,q}_{\overrightarrow{s}}(m) = (b, \mathsf{a}) = (-\mathsf{a} \cdot \overrightarrow{s} + m + e \bmod q,\ \mathsf{a}) \in \mathbb{Z}_q^{n+1},$$

where $\mathsf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \leftarrow \chi$. Intuitively, the message $m \in \mathbb{Z}_q$ is masked additively using $(-\mathsf{a} \cdot \overrightarrow{s} + e \bmod q, \mathsf{a})$, which is indistinguishable from uniform under the hardness assumption of the $\mathsf{LWE}_{n,1,q,\chi}$ problem.

Message encoding. For a given plaintext modulus $t \ll q$, let $\Delta = \lfloor q/t \rfloor$ denote a scaling factor. Encrypting a plaintext $\overline{m} \in \mathbb{Z}_t$ amounts to computing $\mathsf{LWE}^{n,q}_{\overrightarrow{s}}(m)$, where $m = \Delta \cdot \overline{m} \bmod q$ is the encoded message. In other words, the resulting LWE ciphertext is

$$\mathsf{LWE}^{n,q}_{\overrightarrow{s}}(\overline{m}, \Delta) = (b, \mathsf{a}) = (-\mathsf{a} \cdot \overrightarrow{s} + \Delta \cdot \overline{m} + e \bmod q,\ \mathsf{a}) \in \mathbb{Z}_q^{n+1}$$

where $\mathsf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \leftarrow \chi$. Conversely, an LWE ciphertext (b, a) is decrypted as $\lfloor \frac{t}{q}[b + \mathsf{a} \cdot \overrightarrow{s}]_q \rceil$ to recover the encoded message.

3.3.1 Homomorphic operations on LWE ciphertexts

While bootstrapping is an expensive procedure in RLWE-based FHE schemes, it serves as a basic building block for LWE-based FHE schemes, TFHE [51] and FHEW [63]. Such LWE bootstrapping procedures not only refresh noise but also allow the evaluation of an arbitrary function, known as accumulator-based bootstrapping [6, 73, 121]. Below we outline the homomorphic operations used in this thesis, including modulus switching ($\mathsf{ModSwitch}$), LWE-to-RLWE and RLWE-to-LWE key switching ($\mathsf{KeySwitch}$), blind rotation ($\mathsf{BlindRotate}$), sample extraction ($\mathsf{SampleExtract}$), and bootstrapping ($\mathsf{Bootstrap}$).

- $\mathsf{ModSwitch}(\mathsf{ct}, q')$: given an LWE ciphertext $\mathsf{ct} = [b, a_0, \ldots, a_{n-1}] \in \mathbb{Z}_q^{n+1}$, return
$$\left(\lfloor \frac{q'}{q} b \rceil, \lfloor \frac{q'}{q} a_0 \rceil, \ldots, \lfloor \frac{q'}{q} a_{n-1} \rceil\right) \in \mathbb{Z}_{q'}^{n+1}.$$
- $\mathsf{nLWEtoRLWE}(\{\mathsf{ct}_j\}, \mathsf{ksk}_{\overrightarrow{s} \to \boldsymbol{s'}})$: given n LWE ciphertexts $\{\mathsf{ct}_j = (b_j, \mathsf{a}_j)\}_{j\in[n-1]_0}$ and a key-switching key $\mathsf{ksk}_{\overrightarrow{s} \to \boldsymbol{s'}} = \{\mathsf{ksk}_i\}_{i\in[n-1]_0}$ where each ksk_i consists of ℓ RLWE encryptions $\{\mathsf{RLWE}^{\mathfrak{n},q}_{\boldsymbol{s'}}(\overrightarrow{s}[i] \cdot g_k)\}_{k\in[\ell-1]_0}$, vertically stack the i-th entry as
$$\boldsymbol{b} = \sum_{j\in[0,n]} b_j \cdot X^j \quad \boldsymbol{a}_i = \sum_{j\in[0,n]} \mathsf{a}_j[i] \cdot X^j,$$
then compute
$$\mathsf{ct}^{(i)} = \left(\langle \overrightarrow{\boldsymbol{g}}^{-1}(\boldsymbol{a}_i), \mathsf{ksk}_i[0]\rangle,\ \langle \overrightarrow{\boldsymbol{g}}^{-1}(\boldsymbol{a}_i), \mathsf{ksk}_i[1]\rangle\right),\ \forall i \in [n-1]_0,$$
and return
$$\left(\boldsymbol{b} + \sum_{i\in[0,n-1]} \mathsf{ct}^{(i)}[0], \sum_{i\in[0,n-1]} \mathsf{ct}^{(i)}[1]\right) \in \mathcal{R}^2_{\mathfrak{m},q}.$$
This method described above follows from [51], and an alternative construction using automorphisms is given in [44].
- $\mathsf{LWEtoLWE}(\mathsf{ct}, \mathsf{ksk}_{\overrightarrow{s} \to \overrightarrow{s}'})$: given an LWE ciphertext $\mathsf{ct} = (b, \mathsf{a}) \in \mathbb{Z}_q^{n+1}$ under the secret key $\overrightarrow{s}$ and a key-switching key $\mathsf{ksk}_{\overrightarrow{s} \to \overrightarrow{s}'} = \{\mathsf{ksk}_i\}_{i\in[n-1]_0}$ where each ksk_i consists of ℓ LWE encryptions $\left\{\mathsf{LWE}^{n',q}_{\overrightarrow{s}'}(\overrightarrow{s}[i] \cdot g_k)\right\}_{k\in[\ell-1]_0}$, then compute
$$\mathsf{ct}^{(i)} = \left(\langle \overrightarrow{g}^{-1}(a_i), \mathsf{ksk}_i[0]\rangle,\ \ldots,\ \langle \overrightarrow{g}^{-1}(a_i), \mathsf{ksk}_i[n']\rangle\right),\ \forall i \in [n-1]_0,$$
and return
$$\left(b + \sum_{i\in[n-1]_0} \mathsf{ct}^{(i)}[0],\ \ldots,\ \sum_{i\in[n-1]_0} \mathsf{ct}^{(i)}[n']\right) \in \mathbb{Z}_q^{n'+1}.$$

- $\mathsf{SampleExtract}(\mathsf{ct}, k)$: given an RLWE ciphertext $\mathsf{RLWE}_{\boldsymbol{s}}^{\mathfrak{n},q} = (\boldsymbol{b}, \boldsymbol{a})$ where $\mathfrak{n}$ is a power of two and an index $k \in [\mathfrak{n}-1]_0$, compute

$$\mathsf{a}[i] = \begin{cases} \boldsymbol{a}[k-i] & \text{if } k-i < 0 \\ -\boldsymbol{a}[k-i+\mathfrak{n}] & \text{if } k-i \geq 0 \end{cases}, \quad \forall i \in [\mathfrak{n}-1]_0,$$

  and return $(\boldsymbol{b}[k], \mathsf{a}) \in \mathbb{Z}_q^{\mathfrak{n}+1}$.
- $\mathsf{BlindRotate}(\mathsf{ct}, \mathsf{bsk}, \mathsf{acc})$: given an LWE ciphertext $\mathsf{ct} = \mathsf{LWE}_{\overrightarrow{s}}^{n,q}(m) = (b, \mathsf{a})$ where q is typically a power of two $\mathfrak{n}$ or $2\mathfrak{n}$, a bootstrapping key bsk that encrypts the secret key $\overrightarrow{s}$, and an accumulator $\mathsf{acc} = \mathsf{RLWE}^{\mathfrak{n},Q}(T(X))$ (either a noiseless RLWE encryption or an output of $\mathsf{nLWEtoRLWE}$ as in our k-NN work (Chapter 7)), the blind rotation performed in the AP [6] or the GINX [73] style returns

$$\mathsf{RLWE}^{\mathfrak{n},Q}\left(T(X) \cdot X^{\varphi(\mathsf{ct}) \bmod 2\mathfrak{n}}\right),$$

  where $\varphi(\mathsf{ct}) \coloneqq b - \mathsf{a} \cdot \overrightarrow{s}$.
- $\mathsf{Bootstrap}(\mathsf{ct}, \mathsf{bsk}, T(X))$: given an LWE ciphertext $\mathsf{ct} = (b, \mathsf{a}) \in \mathbb{Z}_q^{n+1}$ of modulus q, bootstrapping key bsk that encrypts the secret key $\overrightarrow{s}$, and a test polynomial $T(X) \in \mathcal{R}_{\mathfrak{m},q}$ where $\mathfrak{m}$ is a power of two, compute

$$\mathsf{ct}' = \mathsf{ModSwitch}(\mathsf{ct}, 2\mathfrak{n}) \in \mathbb{Z}_{2\mathfrak{n}}^{n+1},$$

$$\mathsf{acc} = (T(X), \mathbf{0}) \in \mathcal{R}_{\mathfrak{m},q}^2,$$

$$\mathsf{ct}'' = \mathsf{BlindRotate}(\mathsf{ct}', \mathsf{bsk}, \mathsf{acc}) \in \mathcal{R}_{\mathfrak{m},q}^2,$$

  and return $\mathsf{SampleExtract}(\mathsf{ct}'', 0) \in \mathbb{Z}_q^{\mathfrak{n}+1}$. Note that the output LWE ciphertext can be key-switched back to the original LWE dimension n using $\mathsf{LWEtoLWE}$, or used to build the accumulator via $\mathsf{nLWEtoRLWE}$ for a subsequent functional bootstrapping.

3.3.2 Basic procedures on LWE ciphertexts

Augmented Comparators. In machine learning applications, messages are often associated with class labels. In an augmented comparator, given ciphertexts $\mathsf{c_0} = \mathsf{LWE}(m_0)$ and $\mathsf{c_1} = \mathsf{LWE}(m_1)$, along with their labels $\mathsf{c_0}' = \mathsf{LWE}(m_0')$ and $\mathsf{c_1}' = \mathsf{LWE}(m_1')$, it returns LWE encryptions of:

- $\min(m_0, m_1)$ and its corresponding label m_i' with $i = \arg\min(m_0, m_1)$,
- $\max(m_0, m_1)$ and its corresponding label m_i' with $i = \arg\max(m_0, m_1)$.

The procedure begins by homomorphically computing the difference $\mathsf{LWE}_{\overrightarrow{s}}(m)$, where $m = m_0 - m_1$. This ciphertext encrypts a positive number if $m_1 < m_0$.

Next, we construct two accumulators via the $\mathsf{nLWEtoRLWE}$ procedure

$$\mathsf{acc}^{(m)} = \mathsf{RLWE}(\sum_{i=0}^{\mathfrak{n}/2-1} m_1 \cdot X^i - \sum_{i=\mathfrak{n}/2}^{\mathfrak{n}-1} m_0 \cdot X^i),$$

$$\mathsf{acc}^{(\ell)} = \mathsf{RLWE}(\sum_{i=0}^{\mathfrak{n}/2-1} m_1' \cdot X^i - \sum_{i=\mathfrak{n}/2}^{\mathfrak{n}-1} m_0' \cdot X^i).$$

Assuming $m \in (-t/4, t/4)$, the maximum message ct_m and its label ct_ℓ can be homomorphically obtained via

$$\mathsf{ct} = \mathsf{ModSwitch}(\mathsf{LWE}_{\vec{s}}(m), 2\mathfrak{n}) \in \mathbb{Z}_{2\mathfrak{n}}^{n+1},$$

$$\mathsf{ct}^{(k)} = \mathsf{BlindRotate}(\mathsf{ct}, \mathsf{bsk}, \mathsf{acc}^{(k)}) \in \mathcal{R}_{\mathfrak{m},q}^2,\ k = m, \ell,$$

$$\mathsf{ct}_k = \mathsf{SampleExtract}(\mathsf{ct}^{(k)}, 0) \in \mathbb{Z}_q^{\mathfrak{n}+1},\ k = m, \ell.$$

The minimum element can be computed as $\min(m_0, m_1) = m_0 + m_1 - \max(m_0, m_1)$ and similarly for its label.

3.4 Discussion

As noted in Section 1, this thesis does not cover CKKS and FINAL schemes. We summarise the characteristics of ideal applications for TFHE/FHEW, BGV/BFV and GBFV in terms of plaintext precision, desired SIMD capacity and desired functions in Table 3.1.

Properties	TFHE/FHEW	BGV/BFV	GBFV
Plaintext precision	Low	Moderate	High
Desired SIMD capacity	No	High	Moderate
Desired Functions	Arbitrary	Arithmetic computations with low multiplicative depth	
Applied in Part II	k-NN (Ch.7)	Batched PDTE (Ch.8), SQUiD (Ch.9)	Blind Fractal (Ch.10)

Table 3.1: Characteristics of ideal applications for different FHE schemes.

TFHE/FHEW supports the evaluation of arbitrary functions via programmable bootstrapping [50]. However, in practice, the plaintext precision is limited

to a small precision of 4 or 5 bits. To handle messages of high precision, programmable bootstrapping can be applied recursively [97, 107], but this introduces a substantial performance overhead.

In contrast, RLWE-based FHE schemes rely on building blocks such as Add and Mult, which increase noise in ciphertexts. Since RLWE-based bootstrapping is an expensive procedure, it is common to use BGV/BFV/GBFV in a *levelled* manner, i.e. HE parameters are chosen to provide a large enough noise budget so that the entire computation can be completed without bootstrapping. Larger HE parameters incur higher costs in terms of computation, communication, and storage. To improve overall efficiency, it is desirable to control the noise growth and therefore to use smaller HE parameters.

More precisely, the noise consumption in a homomorphic computation is mainly determined by two factors: the multiplicative depth (i.e. the number of consecutive multiplications), and the noise consumption in a single Mult. Because of the former, it is important to optimise the circuit depth for RLWE-based FHE schemes. The latter is proportional to the plaintext modulus p in BGV/BFV and $\|t(X)\|^{\text{can}}$ in GBFV, so the practical plaintext precision $\log_2(p)$ is approximately up to 16 bits in BGV/BFV. On the other hand, GBFV can support large precisions (e.g. 64 bits) with a low multiplicative noise consumption, at the cost of fewer SIMD slots.

Chapter 4

Verifiable computation over encrypted data

When a client outsources the computation of a function f to an untrusted cloud server, three key concerns arise. The first is *efficiency*: on the client side, the client should perform fewer operations than executing f locally; and on the server side, the computation should not incur an impractical computation overhead. The second is *privacy*: the client wants to ensure that the server does not learn its inputs, outputs, or any intermediate values during the computation. FHE, as discussed in earlier sections, addresses these two concerns by providing efficient computation over encrypted data. The third concern is *integrity*: the client wants to ensure that the result returned by the server is correct. This challenge is addressed through VC, as introduced in Section 2.5.

Verifiable computation over encrypted data (vCOED) addresses both *privacy* and *integrity*. Below, we formally define vCOED. Later in Section 4.1 and Section 4.2, we introduce two vCOED instantiations by combining FHE and VC: vFHE and blind proofs. We then discuss their *efficiency*, along with other relevant aspects, in Section 4.3.

Definition 4.0.1 (Verifiable computation over encrypted data)**.** A scheme for verifiable computation over encrypted data $\mathcal{VE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Compute}, \mathsf{Verify})$ consists of the following algorithms:

- $\mathcal{VE}.\mathsf{KeyGen}(1^\lambda, f)$: given the security parameter λ and the function f, it returns two pairs of keys: $(\mathsf{sk}, \mathsf{evk})$ for secure computation and $(\overline{\mathsf{EK}}_f, \overline{\mathsf{VK}}_f)$ for verification.
- $\mathcal{VE}.\mathsf{Enc}_{\mathsf{sk}}(x)$: given a secret key sk and the input x, it returns a public encryption $\mathtt{ct}[x]$.

- $\mathcal{VE}.\mathsf{Compute}_{(\mathsf{evk},\overline{\mathsf{EK}}_f)}(\mathtt{ct}[x])$: given public keys $(\mathsf{evk},\overline{\mathsf{EK}}_f)$ and a public encryption $\mathtt{ct}[x]$, it returns a ciphertext $\mathtt{ct}[y]$ and a proof π that $y = f(x)$.
- $\mathcal{VE}.\mathsf{Verify}_{(\mathsf{sk},\overline{\mathsf{VK}}_f)}(\mathtt{ct}[x],\mathtt{ct}[y],\pi)$: given the secret key sk, the verification key $\overline{\mathsf{VK}}_f$, ciphertexts $\mathtt{ct}[x]$, $\mathtt{ct}[y]$, and a proof π, it returns acc and $y = f(x)$ or rej.

Correctness. A scheme for verifiable computation over encrypted data $\mathcal{VE}$ is correct for functions in $\mathcal{F}$ if for any $f \in \mathcal{F}$, and $x \in \mathrm{Domain}(f)$, the following probability

$$\Pr\left[\begin{array}{l|c} \mathcal{VE}.\mathsf{Verify}_{(\mathsf{sk},\overline{\mathsf{VK}}_f)}(\mathtt{ct}[x],\mathtt{ct}[y],\pi) = (\mathsf{acc}, f(x)) & \begin{array}{c}(\mathsf{sk},\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f) \leftarrow \mathcal{VE}.\mathsf{KeyGen}(1^\lambda, f)\\ \mathtt{ct}[x] \leftarrow \mathcal{VE}.\mathsf{Enc}_{\mathsf{sk}}(x)\\ (\mathtt{ct}[y],\pi) \leftarrow \mathcal{VE}.\mathsf{Compute}_{(\mathsf{evk},\overline{\mathsf{EK}}_f)}(\mathtt{ct}[x])\end{array}\end{array}\right]$$

is no lower than $1 - \mathsf{negl}(\lambda)$.

Security. A scheme for verifiable computation over encrypted data $\mathcal{VE}$ is secure for a function f, if for any adversary $\mathcal{A}$ that runs in probabilistic polynomial time, the following probability

$$\Pr\left[\begin{array}{l|c} \begin{array}{c}\mathcal{VE}.\mathsf{Verify}_{(\mathsf{sk},\overline{\mathsf{VK}}_f)}(\mathtt{ct}[x],\mathtt{ct}[\hat{y}],\hat{\pi}) = (\mathsf{acc},\hat{y})\\ \wedge\ \hat{y} \neq f(x)\end{array} & \begin{array}{c}(\mathsf{sk},\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f) \leftarrow \mathcal{VE}.\mathsf{KeyGen}(1^\lambda, f)\\ x \leftarrow \mathcal{A}(\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f)\\ \mathtt{ct}[x] \leftarrow \mathcal{VE}.\mathsf{Enc}_{\mathsf{sk}}(x)\\ \mathtt{ct}[\hat{y}],\hat{\pi} \leftarrow \mathcal{A}(\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f)\end{array}\end{array}\right]$$

is $\mathsf{negl}(\lambda)$.

Privacy. A scheme for verifiable computation over encrypted data $\mathcal{VE}$ preserves privacy if it offers IND-CPA security, i.e. for any f and any probabilistic polynomial-time adversary $\mathcal{A}$, it holds that

$$\Big|\Pr\left[\mathcal{A}^{\mathcal{VE}.\mathsf{Enc}_{\mathsf{sk}}(\cdot)}(1^\lambda,\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{VE}.\mathsf{Enc}_{\mathsf{sk}}(0)}(1^\lambda,\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f) = 1\right]\Big| \leq \mathsf{negl}(\lambda),$$

where the probability is over $(\mathsf{sk},\mathsf{evk},\overline{\mathsf{EK}}_f,\overline{\mathsf{VK}}_f) \leftarrow \mathcal{VE}.\mathsf{KeyGen}(1^\lambda, f)$, the coins of $\mathcal{VE}.\mathsf{Enc}$ and the coins of the adversary $\mathcal{A}$.

Outsourceability. A scheme for verifiable computation over encrypted data $\mathcal{VE}$ is outsourceable for a function f, if (1) $\mathcal{VE}.\mathsf{KeyGen}(1^\lambda, f)$ is a one-time operation whose cost is amortised over many computations, and (2) for any $\mathtt{ct}[x],\mathtt{ct}[y],\pi$, the time required to compute $\mathcal{VE}.\mathsf{Verify}_{(\mathsf{sk},\overline{\mathsf{VK}}_f)}(\mathtt{ct}[x],\mathtt{ct}[y],\pi)$ is $o(T)$, where T is the time required to compute $f(x)$.

It is possible to make vCOED schemes *publicly verifiable*. In this setting, the verifier does not hold sk and needs assistance from a secret key holder, who decrypts ciphertexts and generates proofs of decryptions.

Furthermore, vCOED schemes can be extended to achieve prover privacy [9, 30, 71, 75, 150]. This extension considers functions of the form $f(x, w)$, where an additional input w is held by the server and remains secret to the verifier. The verifier will accept $(\mathtt{ct}[x], \mathtt{ct}[y], \pi)$ if and only if there exists a w such that $y = f(x, w)$. This is particularly useful when the server provides PPML as a service and wants to keep the secret machine learning model hidden from the client.

To construct vCOED schemes, vFHE applies proof systems to FHE schemes, necessitating a VC scheme that verifies the homomorphic evaluation of a function f. In contrast, blind proofs apply FHE schemes to proof systems, necessitating an FHE scheme capable of evaluating the verifiable computation homomorphically. When zkSNARKs are used as a concrete instantiation of VC, the resulting blind proofs are referred to as blind zkSNARKs.

4.1 Verifiable fully homomorphic encryption

In vFHE, the correctness of homomorphic evaluations on ciphertexts is verified using a VC scheme. The term vFHE was introduced in [98], while its construction originates from earlier works [30, 70, 71, 75]. Since then, the idea has been further developed through a series of follow-up studies [10, 39, 105, 152]. Below, we adapt the description in [70] and formally define vFHE as a vCOED scheme.

Definition 4.1.1 (Verifiable Fully Homomorphic Encryption)**.** A vFHE scheme $\mathcal{VE}^v$ is a vCOED scheme built upon an FHE scheme $\mathcal{E}$ and a VC scheme $\mathcal{VC}$. It consists of the following algorithms:

- $\mathcal{VE}^v.\mathsf{KeyGen}(1^\lambda, f)$: Run $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{EK}_{homf}, \mathsf{VK}_{homf}) \leftarrow \mathcal{VC}.\mathsf{KeyGen}\left(1^\lambda, \mathcal{E}.\mathsf{Eval}_{\mathsf{evk}}(f, \cdot)\right)$, where $(\mathsf{EK}_{homf}, \mathsf{VK}_{homf})$ are keys output by the VC scheme $\mathcal{VC}$ that verifies the homomomorphic evaluation of f. Then return $(\mathsf{sk}, \mathsf{evk})$ and $(\mathsf{EK}_{homf}, \mathsf{VK}_{homf})$.
- $\mathcal{VE}^v.\mathsf{Enc}_{\mathsf{sk}}(x)$: Return $\mathtt{ct}[x] \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}(x)$.
- $\mathcal{VE}^v.\mathsf{Compute}_{(\mathsf{evk}, \mathsf{EK}_{homf})}(\mathtt{ct}[x])$: Return $(\mathtt{ct}[y], \pi_{\mathtt{ct}[y]}) \leftarrow \mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_{homf}}(\mathtt{ct}[x])$.
- $\mathcal{VE}^v.\mathsf{Verify}_{(\mathsf{sk}, \mathsf{VK}_{homf})}(\mathtt{ct}[x], \mathtt{ct}[y], \pi_{\mathtt{ct}[y]})$: Run $b \leftarrow \mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_{homf}}(\mathtt{ct}[x], \mathtt{ct}[y], \pi_{\mathtt{ct}[y]})$. If $b = \mathsf{acc}$, then return acc and $\mathcal{E}.\mathsf{Dec}_{\mathsf{sk}}(\mathtt{ct}[y])$; otherwise, return rej.

4.2 Blind proofs

In blind proofs, verifiable computation is evaluated homomorphically, then the proof can be verified in plaintext after decryption. This idea was first proposed in [76], and the practicality of the blind FRI protocol is demonstrated in [9]. Our work (Chapter 10) further demonstrated the practicality of the blind zkSNARK, instantiated with the GBFV scheme and the Fractal zero-knowledge proof protocol. Below, we give a formal definition of blind proofs as a vCOED scheme.

Definition 4.2.1 (Blind proofs)**.** A blind proof $\mathcal{VE}^b$ is a vCOED scheme built upon an FHE scheme $\mathcal{E}$ and a VC scheme $\mathcal{VC}$. It consists of the following algorithms:

- $\mathcal{VE}^b.\mathsf{KeyGen}(1^\lambda, f)$: Run $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{EK}_f, \mathsf{VK}_f) \leftarrow \mathcal{VC}.\mathsf{KeyGen}\left(1^\lambda, f\right)$, where $(\mathsf{EK}_f, \mathsf{VK}_f)$ are keys output by the VC scheme $\mathcal{VC}$ that verifies the function f. Return $(\mathsf{sk}, \mathsf{evk})$ and $(\mathsf{EK}_f, \mathsf{VK}_f)$.
- $\mathcal{VE}^b.\mathsf{Enc}_{\mathsf{sk}}(x)$: Return $\mathtt{ct}[x] \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}(x)$.
- $\mathcal{VE}^b.\mathsf{Compute}_{(\mathsf{evk},\mathsf{EK}_f)}(\mathtt{ct}[x])$: Return $(\mathtt{ct}[y], \mathtt{ct}[\pi_y]) \leftarrow \mathcal{E}.\,\mathsf{Eval}_{\mathsf{evk}}(\mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_f}, \mathtt{ct}[x])$.
- $\mathcal{VE}^b.\mathsf{Verify}_{(\mathsf{sk},\mathsf{VK}_f)}(\mathtt{ct}[x], \mathtt{ct}[y], \mathtt{ct}[\pi_y])$: Run $(x, y, \pi_y) \leftarrow \mathcal{E}.\mathsf{Dec}_{\mathsf{sk}}(\mathtt{ct}[x], \mathtt{ct}[y], \mathtt{ct}[\pi_y])$, and $b \leftarrow \mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_f}(x, y, \pi_y)$. If $b = \mathsf{acc}$, then return acc and y; otherwise, return rej.

Our instantiation of blind zkSNARKs in Chapter 10 includes a slight modification to the definition above. Realizing $\mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_f}$ with Fractal involves multiple hash function computations, hence the homomorphic evaluation $\mathcal{E}.\,\mathsf{Eval}_{\mathsf{evk}}(\mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_f}, \mathtt{ct}[x])$ is extremely expensive (if not impractical). Therefore, we evaluate $\mathcal{VC}.\mathsf{Compute}_{\mathsf{EK}_f}$ only up to the BCS compilation [22]. As such, the output of $\mathcal{VE}^b.\mathsf{Compute}_{(\mathsf{evk},\mathsf{EK}_f)}(\mathtt{ct}[x])$ should also contain ciphertexts $\mathtt{ct}[q]$ that encrypt responses to the simulated queries, which will be required later for the verification $\mathcal{VC}.\mathsf{Verify}_{\mathsf{VK}_f}$.

4.3 Discussion

In previous sections, we introduced two instantiations of vCOED, vFHE and blind proofs, both designed to ensure privacy and integrity. We summarise all primitives that support outsourced computation in Table 4.1.

Regarding privacy, the verification of blind proofs includes an FHE decryption step prior to VC verification, and is therefore vulnerable to reaction attacks [42, 151]. Specifically, when the verifier signals to the prover whether verification passes, one bit of information about the plaintext/secret key may be leaked [9].

System	Privacy	Integrity	Efficiency
FHE	+	−	+
VC	−	+	+
vFHE	+	+	−
Blind proofs	+	+	+

Table 4.1: Comparison of computation outsourcing primitives. Privacy: − - no input/output privacy; + - secure against honest-but-curious servers, but vulnerable to limited leakage via reaction attacks; + - secure against such reaction attacks. Integrity: − - no integrity; +- with moderate integrity; + - with integrity. Efficiency (relative scale): −- inefficient; +- moderately efficient; +- efficient.

In contrast, vFHE remains secure against such attacks, as its verification process first checks the VC proof and only proceeds with FHE decryption if that verification succeeds.

Regarding integrity, blind proofs validate relations at the plaintext level, whereas vFHE schemes validate relations at the ciphertext level. Consequently, if the correctness of the underlying FHE scheme $\mathcal{E}$ is not guaranteed, e.g when the noise in the output ciphertext exceeds the threshold for correct decryption, then an honest execution of $\mathcal{VE}.\mathsf{Compute}$ would still be accepted in the vFHE verification, even though its decryption does not equal $f(x)$. In contrast, a blind proof scheme would correctly reject such cases, thereby offering stronger integrity guarantees.

Regarding efficiency, state-of-the-art vFHE constructions [10] can verifiably outsource computations involving hundreds of plaintext constraints in a runtime similar to blind zkSNARKs with 2^{20} constraints. This reveals an efficiency gap of approximately four orders of magnitude in favor of blind proofs. The reasons for this gap are twofold: first, building VC on ciphertexts leads to a significant circuit size blowup; and second, proving non-arithmetic operations such as modswitching and keyswitching further introduces additional performance bottlenecks.

As a final remark, achieving public verifiability in both vFHE and blind proofs requires the assistance of the secret key holder. The secret key holder decrypts and gives proof of decryption (PoD) for $(\mathtt{ct}[y], y)$ in vFHE and for $(\{ct[x], \mathtt{ct}[y], \mathtt{ct}[\pi_y]\}, \{x, y, \pi_y\})$ in blind proofs. Moreover, as explained in Section 4.2, our instantiation of blind zkSNARKs also requires $(\{\mathtt{ct}[q]\}, \{q\})$ as part of the verification inputs. In this case, the secret key holder also needs to give proofs of decryption for these pairs. Fortunately, the PoD construction from

lattice-based zero-knowledge proofs in Chapter 10 supports batching, allowing multiple ciphertext-plaintext pairs to be verified with a single PoD.

Chapter 5

Conclusion and future work

5.1 Conclusion

Since we started this research in 2021, FHE has gained increasing recognition as a viable tool for privacy-preserving applications. This growing interest is reflected in substantial progress across both theoretical and practical aspects.

On the theoretical side, there have been notable advances in improving core FHE components such as bootstrapping [14, 77, 78, 90, 100, 116, 117, 127, 144, 145] and homomorphic matrix multiplication [12, 129]. Researchers have also explored fundamental trade-offs, such as those between input precision and packing capacity in the GBFV scheme [81].

On the practical side, efficient FHE-based solutions have been proposed for various applications including private information retrieval (PIR) [36, 59, 95, 97, 102, 109, 120] and PPML [11, 53, 54, 55, 118]. Most notably, FHE has recently been deployed in real-world systems, e.g. Apple has incorporated FHE into its ecosystem to support privacy-preserving applications [7].

While FHE research has primarily focused on offering *privacy*, there is growing interest in addressing *integrity*, i.e. ensuring the verifiability of outsourced computations. Several recent works aim to augment FHE with integrity guarantees [9, 10, 39, 72, 75, 76, 98, 105, 150, 152], though typically at a significant cost to *efficiency*.

In line with these achievements, our contribution consists of accelerating bootstrapping, applying FHE to various applications such as private genome queries and PPML, and demonstrating the feasibility of achieving integrity through blind zkSNARKs. Together, these efforts aim to bring FHE closer to a

modular solution that simultaneously ensures *efficiency*, *privacy*, and *integrity*. We discuss the broader impact of our works and draw the following conclusions.

Exploring underlying mathematical structures enables optimisations in FHE. Our work in Chapter 6 systematises the theory of polynomial functions modulo p^e, with a particular focus on the existence of null polynomials, i.e. non-zero polynomials that evaluate to zero in every point. This mathematical theory has led to substantial improvements in the digit removal procedure, a main bottleneck in BGV, BFV and GBFV bootstrapping.

- In Chapter 6, we proposed four techniques for leveraging null polynomials to improve BGV/BFV bootstrapping with a small plaintext modulus p, including a function composition method that introduces null polynomials over a subset.
- Later, Ma et al. [116] applied null polynomials over a subset (referred to as local null polynomials in their work) to accelerate BGV/BFV bootstrapping with a large plaintext modulus p. Their construction fully exploits the sparsity of inputs without relying on function composition.
- The large-p bootstrapping trick developed in [116] was later adopted by Geelen and Vercauteren in [81] to realise bootstrapping for GBFV.

Beyond the relevance to FHE, the study of polynomial functions has also proven useful in other areas. For instance, the theory has been applied to optimise algebraic attacks on symmetric primitives defined over composite modulus, as demonstrated in [89].

For suitable use cases, FHE provides a viable approach to constructing efficient privacy-preserving systems. Our work in Chapter 7 constructs a secure non-interactive k-NN classifier from FHE.

- Built on TFHE, we considered inputs with 4-bit precision and significantly improved upon the best previous approach [153], achieving up to a $47\times$ speedup. This improvement is primarily due to our optimized oblivious Top-k algorithm, which reduces the complexity from $O(d^2)$ to $O(d \log^2 k)$.
- Later, Azogagh et al. [11] used *blind array access* to achieve a Top-k algorithm with improved complexity $O(d + k)$. Realising blind array access with TFHE blind rotations, their approach reports a $4\times$ speedup over ours.
- Interestingly, Apple has also deployed a private nearest neighbour search using FHE in a real-world setting [7]. Their system corresponds to a special case of k-NN with $k = 1$ with 8-bit input precision. Comparing such real-world deployments with the above research prototypes of parameterized k and 4-bit precision in terms of costs and accuracy would be valuable.

Our work in Chapter 8 develops private decision tree evaluation protocols optimised for batched inputs. Based on BGV/BFV, our construction takes full advantage of the SIMD capacity to efficiently pack multiple input feature vectors for evaluation on the same decision tree. This is particularly useful in use cases such as privacy-preserving credit scoring, where a bank outsources a decision tree model and evaluates it for multiple applicants without revealing their profiles. Our work outperforms the best prior work [118] from batch sizes of around 1,000. With performance gains increasing at larger scales, it reaches up to a 17× speedup at a batch size of 16,384.

In Chapter 9, the BGV/BFV SIMD capacity is exploited in a different way. We propose SQUiD, a secure queryable database for storing and analyzing genotype-phenotype data. Traditional data access models, such as those recommended by initiatives such as dbGaP and UK Biobank [1], typically involve researchers downloading encrypted data, decrypting the data locally, and then analysing the data in plaintext. In contrast, SQUiD enables the server to store the encrypted database in BGV/BFV SIMD slots, supporting secure data processing such as cohort creation and data aggregation. The privacy guarantee and practical usability of SQUiD have led to its consideration as a solution for managing psychiatric patient data at the New York Genome Center.

Blind proofs offer a promising direction to achieve integrity in FHE-based applications. Before we started the work in Chapter 10, vFHE was the only known approach to achieve data integrity in FHE-based applications. While there have been ongoing improvements to vFHE, its performance remains limited due to inherent challenges such as proving non-arithmetic operations.

In 2024, an alternative approach to ensuring integrity has emerged [9, 72, 76], which is generalised to blind proofs in this thesis. Our work in Chapter 10 demonstrated the feasibility of blind zkSNARKs by showing that a computation represented by 2^{20} R1CS constraints can be proven within 20 minutes, assuming a 32× speedup through parallelization. This represents an improvement of four orders of magnitude over the performance of existing vFHE techniques.

Later, faster instantiations of blind zkSNARKs were proposed in subsequent work [150], where the use of the Spartan Polynomial IOP and the Brakedown/Ligero polynomial commitment scheme brings an additional 8× speedup. While these results highlight the significant promise of blind zkSNARKs, both our work and [150] are based on synthetic benchmarks that evaluate proof generation for a generic function f represented with 2^{20} R1CS constraints.

Looking ahead, to realise the full potential of blind proofs, it is essential to move beyond generic benchmarks and explore concrete applications with well-defined

functions f. Several such candidate applications are discussed in the next section on future work.

5.2 Future work

Secure applications with large payloads using GBFV. FHE is a widely used technique for securely retrieving (encrypted) payloads stored in a public server. The desired payload is retrieved according to an encrypted index given by a client in the context of PIR, or according to the result of oblivious message detection in oblivious message retrieval (OMR). In practical constructions for PIR [106] and OMR [108], payloads are stored in SIMD slots and are homomorphically retrieved using the BFV scheme. However, slot sizes in BFV are typically small to support homomorphic evaluation of sufficiently deep circuits without bootstrapping. When the payload size is larger than the slot size, the payload is split into small blocks stored in multiple ciphertexts. This leads to a computation overhead proportional to the number of blocks. To address this overhead, GBFV presents an attractive alternative that supports larger payloads in SIMD slots while maintaining a reasonable noise growth during computation.

FairProof for privacy-preserving machine learning. As MLaaS becomes increasingly common in high-stakes domains such as hiring, lending, or healthcare, it has sparked serious concerns about model fairness [142], leading to growing distrust among individuals affected by ML-based decisions [64]. In particular, individuals want assurance that the same model is used consistently across all evaluations and that no one is subjected to discrimination through the use of different models. To rebuild trust, the model owner can prove the evaluation of a committed ML model using zero-knowledge proofs. For example, a recent work on Fairproof [147] enables model owners to issue publicly verifiable certificates while ensuring model confidentiality.

In parallel, client privacy is another critical requirement in MLaaS, especially when input features contain sensitive personal or financial data. As is shown in Chapter 7 and Chapter 8, FHE provides a compelling solution: clients send their encrypted feature vectors to the server, which homomorphically evaluates the model and returns the encrypted prediction to the client. While this preserves input privacy, it does not guarantee that the server is applying the *same* model to all users. In fact, there is still a risk that the server may selectively apply different models based on client metadata.

Achieving both fairness and privacy remains a challenging and interesting open problem, and its solution may use a combination of FHE and zero-knowledge

proofs. Blind proofs (Section 4.2) represent a promising direction: the server not only evaluates an ML model homomorphically on encrypted inputs, but also generates a blind proof that the decrypted output corresponds to the evaluation of a previously committed model.

Verifiable private information retrieval. In verifiable databases (VDB) [23,40], a client outsources storage of a database DB of size $\mathcal{N}$ to an untrusted server, and later retrieves a record $\mathsf{DB}[i]$ of the desired index $i \in [\mathcal{N}]$. To prevent the server from tampering with the data, the server should not only send the response $\mathsf{DB}[i]$ but also a proof that the response was computed correctly.

A simple way to ensure data integrity is that a client signs each pair $(j, \mathsf{DB}[j]), j \in [\mathcal{N}]$ before outsourcing DB to the server, and then the server is requested to output the record together with its valid signature. While this approach suffices for static databases, it is not sufficient for updatable databases, since the client cannot revoke signatures given to the server for the previous values. This leads to various solutions on accumulators [37, 38, 125], authenticated data structures [126, 128], verifiable computation [23], and vector commitments [40].

Another important aspect of database outsourcing is preserving client privacy, as the queried index i may reveal sensitive information. To address this, PIR protocols ensure that the server does not learn the index i in plaintext, with FHE being a widely used primitive in such constructions.

Combining data integrity with client privacy is therefore essential. This has been explored in [52] for authenticated private information retrieval and in [60] by adding verifiability to the SimplePIR protocol [95]. However, both works rely on digests of a large size $\mathcal{O}\left(\sqrt{N}\right)$, which can be impractical for large databases. Furthermore, both works only consider a static database setting. Reducing the digest size and supporting efficient updates in verifiable PIR remain important open directions for future work. Blind proofs (Section 4.2), which combine verifiable computation (supporting data updates) and FHE (supporting PIR), represent a promising primitive for addressing these challenges.

Public verifiability for encrypted database queries. In SQUiD (Chapter 9), doctors or researchers that are authenticated by the data owner (e.g. NIH) can query an encrypted genotype-phenotype database stored on a public cloud. The cloud server can securely generate a cohort of genetically similar patients and compute key metrics such as polygenic risk scores (PRS) for this cohort of patients. The resulting encrypted outputs can then be decrypted by the authorised querier and used to support diagnosis and personalised treatment decisions.

To increase trust and transparency, it is desirable if the broader public, such as the patient whose condition the query concerns, can verify the results. This entails two key aspects. Firstly, data integrity, as discussed above for unencrypted databases, ensures that the database has not been tampered with. Secondly, patients are typically not authorised by the data owner, hence cannot reproduce queries themselves for validation. Therefore, the patient should be able to verify the correctness of the query result without possessing the doctor's secret key. This necessitates public verifiability, which can be achieved by having the doctor append a proof of decryption (Chapter 10) alongside the result.

Private proof delegation without proof of decryption. In zkDel instantiated with blind zkSNARKs (Chapter 10), FHE ciphertexts are committed in the BCS compilation [22] and used to derive randomness for the simulated verifier. Encrypted responses $\{\mathtt{ct}[q_i]\}$ to the simulated verifier are included in the proof. A designated verifier holding the secret key can decrypt these ciphertexts, perform the plaintext verification and ensure that the randomness is correctly derived from the committed ciphertexts. To enable public verifiability, a secret key holder needs to decrypt $\{\mathtt{ct}[q_i]\}$, append $\{q_i\}$ together with PoDs for these ciphertext-plaintext pairs $\{\mathtt{ct}[q_i], q_i\}$ to the proof. As such, a public verifier also needs to verify the PoD in addition.

This raises a natural question: is it possible to perform zkDel using FHE without generating and verifying PoDs? One possible direction is to homomorphically evaluate the Merkle tree in the BCS compilation, allowing the randomness to be derived from plaintexts. Developing FHE-friendly hash functions and evaluating Merkle trees efficiently using FHE remain open challenges for future work.

Security of vCOED. A formal analysis of the security guarantees provided by vCOED also remains an important open problem. Walter [143] introduced the semi-active (SA) security model for vFHE, which can be extended to security against verified chosen-ciphertext attacks (vCCA) [119] by incorporating additional ciphertext checks. However, both models do not consider the chosen plaintext attacks with a decryption oracle (CPA-D) attack [43, 47, 101], hence the impact of CPA-D attacks on the SA and vCCA security models requires further investigation. Furthermore, the only known security result for blind proofs suggests a potential leakage of 1 bit of information about the client's message or secret key. Developing a comprehensive security model for blind proofs is therefore a promising and necessary direction for further research.

Bibliography

[1] UK biobank data access guide. `https://uk-biobank.gitbook.io/data-access-guide`, 2023. Last accessed: 11 Dec 2024.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[3] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[4] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015.

[5] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, August 2013.

[6] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Berlin, Heidelberg, August 2014.

[7] Apple Machine Learning Research. Combining Machine Learning and Homomorphic Encryption in the Apple Ecosystem. `https://machinelearning.apple.com/research/homomorphic-encryption`, 2024. Accessed: 2025-07-29.

[8] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1467–1481. ACM Press, November 2023.

[9] Diego F. Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez. HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part V*, volume 15488 of *Lecture Notes in Computer Science*, pages 302–334, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[10] Shahla Atapoor, Karim Baghery, Hilder V. L. Pereira, and Jannik Spiessens. Verifiable FHE via lattice-based SNARKs. *CiC*, 1(1):24, 2024.

[11] Sofiane Azogagh, Marc-Olivier Killijian, and Félix Larose-Gervais. A non-comparison oblivious sort and its application to private k-NN. Cryptology ePrint Archive, Report 2024/1894, 2024.

[12] Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé. Plaintext-ciphertext matrix multiplication and FHE bootstrapping: Fast and fused. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 387–421. Springer, Cham, August 2024.

[13] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. HERMES: Efficient ring packing using MLWE ciphertexts and application to transciphering. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 37–69. Springer, Cham, August 2023.

[14] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, and Damien Stehlé. Bootstrapping bits with CKKS. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 94–123. Springer, Cham, May 2024.

[15] Karim Baghery, Noah Knapen, Georgio Nicolas, and Mahdi Rahimi. Pre-constructed publicly verifiable secret sharing and applications. In Marc Fischlin and Veelasha Moonsamy, editors, *Applied Cryptography and Network Security - 23rd International Conference, ACNS 2025, Munich, Germany, June 23-26, 2025, Proceedings, Part I*, volume 15825 of *Lecture Notes in Computer Science*, pages 89–119. Springer, 2025.

[16] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, Cham, February 2014.

[17] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385. Springer, Cham, September 2018.

[18] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[19] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[20] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[21] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019.

[22] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

[23] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 111–131. Springer, Berlin, Heidelberg, August 2011.

[24] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 551–572. Springer, Berlin, Heidelberg, December 2014.

[25] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments

from learning with errors over rings. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 305–325. Springer, Cham, September 2015.

[26] Jonas Bertels, Michiel Van Beirendonck, Furkan Turan, and Ingrid Verbauwhede. Hardware acceleration of FHEW. In Maksim Jenihhin, Hana Kubátová, Nele Metens, Jaan Raik, Foisal Ahmed, and Jan Belohoubek, editors, *26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2023, Tallinn, Estonia, May 3-5, 2023*, pages 57–60. IEEE, 2023.

[27] Jonas Bertels, Hilder V. L. Pereira, and Ingrid Verbauwhede. FINAL bootstrap acceleration on FPGA using dsp-free constant-multiplier ntts. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(3):293–316, 2025.

[28] Jacob Blindenbach, Jiayi Kang, Seungwan Hong, Caline Karam, Thomas Lehner, and Gamze Gürsoy. SQUiD: ultra-secure storage and analysis of genetic data for the advancement of precision medicine. *Genome Biology*, 25(314), 2024.

[29] Jacob A. Blindenbach, Jung Hee Cheon, Gamze Gürsoy, and Jiayi Kang. On the overflow and p-adic theory applied to homomorphic encryption. In Shlomi Dolev, Michael Elhadad, Miroslaw Kutylowski, and Giuseppe Persiano, editors, *Cyber Security, Cryptology, and Machine Learning - 8th International Symposium, CSCML 2024, Be'er Sheva, Israel, December 19-20, 2024, Proceedings*, volume 15349 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 2024.

[30] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Cham, May 2021.

[31] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 188–215. Springer, Cham, December 2022.

[32] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 384–417. Springer, Cham, August 2023.

[33] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran

Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Berlin, Heidelberg, August 2012.

[34] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[35] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Cryptology ePrint Archive, Report 2011/344, 2011.

[36] Alexander Burton, Samir Jordan Menon, and David J. Wu. Respire: High-rate PIR for databases with small records. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1463–1477. ACM Press, October 2024.

[37] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, Berlin, Heidelberg, March 2009.

[38] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Berlin, Heidelberg, August 2002.

[39] Ignacio Cascudo, Anamaria Costache, Daniele Cozzo, Dario Fiore, Antonio Guimarães, and Eduardo Soria-Vazquez. Verifiable computation for approximate homomorphic encryption schemes. Cryptology ePrint Archive, Paper 2025/286, 2025.

[40] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Berlin, Heidelberg, February / March 2013.

[41] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. In *2020 IEEE Symposium on Security and Privacy*, pages 1416–1432. IEEE Computer Society Press, May 2020.

[42] Bhuvnesh Chaturvedi, Anirban Chakraborty, Ayantika Chatterjee, and Debdeep Mukhopadhyay. A practical full key recovery attack on TFHE and FHEW by inducing decryption errors. Cryptology ePrint Archive, Report 2022/1563, 2022.

[43] Marina Checri, Renaud Sirdey, Aymen Boudguiga, and Jean-Paul Bultel. On the practical CPA^D security of "exact" and threshold FHE schemes and libraries. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 3–33. Springer, Cham, August 2024.

[44] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21International Conference on Applied Cryptography and Network Security, Part I*, volume 12726 of *LNCS*, pages 460–479. Springer, Cham, June 2021.

[45] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 315–337. Springer, Cham, April / May 2018.

[46] Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia. High-precision arithmetic in homomorphic encryption. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 116–136. Springer, Cham, April 2018.

[47] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the IND-CPA$^{\mathrm{D}}$ security of exact FHE schemes. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 2505–2519. ACM Press, October 2024.

[48] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Cham, December 2017.

[49] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Cham, May 2020.

[50] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Berlin, Heidelberg, December 2016.

[51] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.

[52] Simone Colombo, Kirill Nikitin, Henry Corrigan-Gibbs, David J. Wu, and Bryan Ford. Authenticated private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 3835–3851. USENIX Association, August 2023.

[53] Kelong Cong, Debajyoti Das, Jeongeun Park, and Hilder V. L. Pereira. SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 563–577. ACM Press, November 2022.

[54] Kelong Cong, Robin Geelen, Jiayi Kang, and Jeongeun Park. Revisiting oblivious top-k selection with applications to secure k-nn classification. In Maria Eichlseder and Sébastien Gambs, editors, *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part I*, volume 15516 of *Lecture Notes in Computer Science*, pages 3–25. Springer, 2024.

[55] Kelong Cong, Jiayi Kang, Georgio Nicolas, and Jeongeun Park. Faster private decision tree evaluation for batched input from homomorphic encryption. In Clemente Galdi and Duong Hieu Phan, editors, *SCN 24, Part II*, volume 14974 of *LNCS*, pages 3–23. Springer, Cham, September 2024.

[56] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340. Springer, Cham, February / March 2016.

[57] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 79–88. ACM Press, October / November 2006.

[58] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Berlin, Heidelberg, August 2012.

[59] Alex Davidson, Gonçalo Pestana, and Sofía Celi. FrodoPIR: Simple, scalable, single-server private information retrieval. *PoPETs*, 2023(1):365–383, January 2023.

[60] Leo de Castro and Keewoo Lee. VeriSimplePIR: Verifiability in SimplePIR at no online cost for honest servers. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024.

[61] Léo Ducas and Alain Durmus. Ring-LWE in polynomial rings. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 34–51. Springer, Berlin, Heidelberg, May 2012.

[62] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Berlin, Heidelberg, August 2013.

[63] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Berlin, Heidelberg, April 2015.

[64] Cynthia Dwork and Martha Minow. Distrust of artificial intelligence: Sources & responses from computer science & law. *Daedalus*, 151(2):309–321, 2022.

[65] Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 484–517. Springer, Cham, August 2023.

[66] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146. Springer, Cham, August 2019.

[67] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. MatRiCT$^+$: More efficient post-quantum private blockchain payments. In *2022 IEEE Symposium on Security and Privacy*, pages 1281–1298. IEEE Computer Society Press, May 2022.

[68] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[69] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor,

CRYPTO'86, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[70] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 844–855. ACM Press, November 2014.

[71] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Cham, May 2020.

[72] Mariana Gama, Emad Heydari Beni, Jiayi Kang, Jannik Spiessens, and Frederik Vercauteren. Blind zkSNARKs for private proof delegation and verifiable computation over encrypted data. *IACR Communications in Cryptology*, 2(3), 2025.

[73] Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 528–558. Springer, Berlin, Heidelberg, May 2016.

[74] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Berlin, Heidelberg, April 2008.

[75] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.

[76] Sanjam Garg, Aarushi Goel, and Mingyuan Wang. How to prove statements obliviously? In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 449–487. Springer, Cham, August 2024.

[77] Robin Geelen. Revisiting the slot-to-coefficient transformation for BGV and BFV. *CiC*, 1(3):37, 2024.

[78] Robin Geelen, Ilia Iliashenko, Jiayi Kang, and Frederik Vercauteren. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 257–286. Springer, Cham, April 2023.

[79] Robin Geelen, Michiel Van Beirendonck, Hilder V. L. Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios D. Dimou, Ingrid Verbauwhede, Frederik Vercauteren, and David W. Archer. BASALISC: Programmable hardware accelerator for BGV fully homomorphic encryption. *IACR TCHES*, 2023(4):32–57, 2023.

[80] Robin Geelen and Frederik Vercauteren. Bootstrapping for BGV and BFV revisited. *J. Cryptol.*, 36(2):12, 2023.

[81] Robin Geelen and Frederik Vercauteren. Fully homomorphic encryption for cyclotomic prime moduli. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 366–397, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[82] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Berlin, Heidelberg, August 2010.

[83] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[84] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, Cham, May / June 2022.

[85] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Ring switching in BGV-style homomorphic encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 19–37. Springer, Berlin, Heidelberg, September 2012.

[86] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Berlin, Heidelberg, August 2012.

[87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[88] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[89] Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, Morten Øygarden, Håvard Raddum, and Qingju Wang. Cryptanalysis of symmetric primitives over rings and a key recovery attack on rubato. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 305–339. Springer, Cham, August 2023.

[90] Antonio Guimarães, Hilder V. L. Pereira, and Barry van Leeuwen. Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VI*, volume 14443 of *LNCS*, pages 3–35. Springer, Singapore, December 2023.

[91] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Berlin, Heidelberg, September 2012.

[92] Shai Halevi. Homomorphic encryption. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 219–276. Springer International Publishing, 2017.

[93] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670. Springer, Berlin, Heidelberg, April 2015.

[94] Shai Halevi and Victor Shoup. Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481, 2020.

[95] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 3889–3905. USENIX Association, August 2023.

[96] Wonkyung Jung, Eojin Lee, Sangpyo Kim, Keewoo Lee, Namhoon Kim, Chohong Min, Jung Hee Cheon, and Jung Ho Ahn. HEAAN demystified: Accelerating fully homomorphic encryption through architecture-centric analysis and optimization. *CoRR*, abs/2003.04510, 2020.

[97] Jiayi Kang and Leonard Schild. Pirouette: Query efficient single-server PIR. Cryptology ePrint Archive, Paper 2025/680, 2025.

[98] Christian Knabenhans, Alexander Viand, Antonio Merino-Gallardo, and Anwar Hithnawi. vfhe: Verifiable fully homomorphic encryption. In Flávio Bergamaschi, Anamaria Costache, and Kurt Rohloff, editors, *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Salt Lake City, UT, USA, October 14-18, 2024*, pages 11–22. ACM, 2024.

[99] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

[100] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 227–256. Springer, Cham, April 2023.

[101] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 648–677. Springer, Cham, October 2021.

[102] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz. Hintless single-server private information retrieval. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part IX*, volume 14928 of *LNCS*, pages 183–217. Springer, Cham, August 2024.

[103] Mengyuan Li, Yuheng Yang, Guoxing Chen, Mengjia Yan, and Yinqian Zhang. SoK: Understanding design choices and pitfalls of trusted execution environments. In Jianying Zhou, Tony Q. S. Quek, Debin Gao, and Alvaro A. Cárdenas, editors, *ASIACCS 24*. ACM Press, July 2024.

[104] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Berlin, Heidelberg, February 2011.

[105] Fengrun Liu, Haofei Liang, Tianyu Zhang, Yuncong Hu, Xiang Xie, Haisheng Tan, and Yu Yu. HasteBoots: Proving FHE bootstrapping in seconds. Cryptology ePrint Archive, Paper 2025/261, 2025.

[106] Jian Liu, Jingyu Li, Di Wu, and Kui Ren. PIRANA: Faster multi-query PIR via constant-weight codes. In *2024 IEEE Symposium on Security and Privacy*, pages 4315–4330. IEEE Computer Society Press, May 2024.

[107] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 130–160. Springer, Cham, December 2022.

[108] Zeyu Liu and Eran Tromer. Oblivious message retrieval. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 753–783. Springer, Cham, August 2022.

[109] Ming Luo, Feng-Hao Liu, and Han Wang. Faster FHE-based single-server private information retrieval. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1405–1419. ACM Press, October 2024.

[110] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.

[111] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012.

[112] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 293–323. Springer, Cham, April / May 2017.

[113] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Cham, August 2022.

[114] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010.

[115] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Berlin, Heidelberg, May 2013.

[116] Shihe Ma, Tairong Huang, Anyu Wang, and Xiaoyun Wang. Accelerating BGV bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. In Marc

Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 403–432. Springer, Cham, May 2024.

[117] Shihe Ma, Tairong Huang, Anyu Wang, and Xiaoyun Wang. Faster BGV bootstrapping for power-of-two cyclotomics through homomorphic NTT. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part I*, volume 15484 of *Lecture Notes in Computer Science*, pages 143–175, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[118] Rasoul Akhavan Mahdavi, Haoyan Ni, Dimitry Linkov, and Florian Kerschbaum. Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 2945–2958. ACM Press, November 2023.

[119] Mark Manulis and Jérôme Nguyen. Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 63–93. Springer, Cham, May 2024.

[120] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, high-rate single-server PIR via FHE composition. In *2022 IEEE Symposium on Security and Privacy*, pages 930–947. IEEE Computer Society Press, May 2022.

[121] Daniele Micciancio and Yuriy Polyakov. Bootstrapping in FHEW-like cryptosystems. Cryptology ePrint Archive, Report 2020/086, 2020.

[122] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[123] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[124] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 113–124. ACM, 2011.

[125] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, Berlin, Heidelberg, February 2005.

[126] Kobbi Nissim and Moni Naor. Certificate revocation and certificate update. In Aviel D. Rubin, editor, *USENIX Security 98*. USENIX Association, January 1998.

[127] Hiroki Okada, Rachel Player, and Simon Pohmann. Homomorphic polynomial evaluation using galois structure and applications to BFV bootstrapping. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VI*, volume 14443 of *LNCS*, pages 69–100. Springer, Singapore, December 2023.

[128] Charalampos Papamanthou and Roberto Tamassia. Time and space efficient algorithms for two-party authenticated data structures. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS 07*, volume 4861 of *LNCS*, pages 1–15. Springer, Berlin, Heidelberg, December 2008.

[129] Jai Hyun Park. Ciphertext-ciphertext matrix multiplication: Fast for large matrices. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VIII*, volume 15608 of *Lecture Notes in Computer Science*, pages 153–180, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[130] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

[131] Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 411–430. Springer, Cham, August / September 2016.

[132] Plonky2. `https://github.com/0xPolygonZero/plonky2`, 2021.

[133] Precedence Research. Cloud computing market size, growth, report 2025. `https://www.precedenceresearch.com/cloud-computing-market`, 2025. Accessed: 2025-08-25.

[134] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[135] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[136] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990.

[137] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[138] Victor Shoup. Number theory library 5.5.2 NTL for C++. `http://www.shoup.net/ntl`.

[139] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000.

[140] Michiel van Beirendonck, Jan-Pieter D'Anvers, Furkan Turan, and Ingrid Verbauwhede. FPT: A fixed-point accelerator for torus fully homomorphic encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 741–755. ACM Press, November 2023.

[141] Miden VM. `https://github.com/0xMiden/miden-vm`, 2022.

[142] S. Wallarchive and H. Schellmannarchive. Linkedin's job-matching ai was biased. the company's solution? more ai. `https://www.wsj.com/articles/linkedin-algorithm-artificial-intelligence-bias-job-recruiting-11624944661`, June 2021. Accessed: 2025-06-27.

[143] Michael Walter. What have SNARGs ever done for FHE? Cryptology ePrint Archive, Report 2024/1207, 2024.

[144] Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: Faster and smaller. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 342–372. Springer, Cham, May 2024.

[145] Binwu Xiang, Jiang Zhang, Yi Deng, Yiran Dai, and Dengguo Feng. Fast blind rotation for bootstrapping FHEs. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 3–36. Springer, Cham, August 2023.

[146] Hua Xu, Mariana Gama, Emad Heydari Beni, and Jiayi Kang. FRIttata: Distributed proof generation of FRI-based SNARKs. Cryptology ePrint Archive, Paper 2025/1285, 2025.

[147] Chhavi Yadav, Amrita Roy Chowdhury, Dan Boneh, and Kamalika Chaudhuri. Fairproof : Confidential and certifiable fairness for neural networks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[148] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[149] Risc zero. `https://github.com/risc0/risc0`, 2022.

[150] Xinxuan Zhang, Ruida Wang, Zeyu Liu, Binwu Xiang, Yi Deng, and Xianhui Lu. FHE-SNARK vs. SNARK-FHE: From analysis to practical verifiable computation. Cryptology ePrint Archive, Report 2025/302, 2025.

[151] Zhenfei Zhang, Thomas Plantard, and Willy Susilo. Reaction attack on outsourced computing with fully homomorphic encryption schemes. In Howon Kim, editor, *Information Security and Cryptology - ICISC 2011 - 14th International Conference, Seoul, Korea, November 30 - December 2, 2011. Revised Selected Papers*, volume 7259 of *Lecture Notes in Computer Science*, pages 419–436. Springer, 2011.

[152] Zhelei Zhou, Yun Li, Yuchen Wang, Zhaomin Yang, Bingsheng Zhang, Cheng Hong, Tao Wei, and Wenguang Chen. ZHE: efficient zero-knowledge proofs for HE evaluations. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 3328–3346. IEEE, 2025.

[153] Martin Zuber and Renaud Sirdey. Efficient homomorphic evaluation of k-NN classifiers. *PoPETs*, 2021(2):111–129, April 2021.

Part II

Publications

Chapter 6

On Polynomial Functions Modulo p^e and Faster Bootstrapping for Homomorphic Encryption

Publication Data

Geelen, R., Iliashenko, I., Kang, J., and Vercauteren, F. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In *EUROCRYPT 2023, Part III* (Apr. 2023), C. Hazay and M. Stam, Eds., vol. 14006 of *LNCS*, Springer, Cham, pp. 257–286

Contribution

I formalized the theory of polynomial functions and improved the digit extraction procedure in collaboration with other authors.

On Polynomial Functions Modulo p^e and Faster Bootstrapping for Homomorphic Encryption

Robin Geelen[1], Ilia Iliashenko[2], Jiayi Kang[1], and Frederik Vercauteren[1]

[1] imec-COSIC, KU Leuven, Belgium
[2] CipherMode Labs, USA

Abstract. In this paper, we perform a systematic study of functions $f : \mathbb{Z}_{p^e} \to \mathbb{Z}_{p^e}$ and categorize those functions that can be represented by a polynomial with integer coefficients. More specifically, we cover the following properties: necessary and sufficient conditions for the existence of an integer polynomial representation; computation of such a representation; and the complete set of equivalent polynomials that represent a given function.

As an application, we use the newly developed theory to speed up bootstrapping for the BGV and BFV homomorphic encryption schemes. The crucial ingredient underlying our improvements is the existence of null polynomials, i.e. non-zero polynomials that evaluate to zero in every point. We exploit the rich algebraic structure of these null polynomials to find better representations of the digit extraction function, which is the main bottleneck in bootstrapping. As such, we obtain sparse polynomials that have 50% fewer coefficients than the original ones. In addition, we propose a new method to decompose digit extraction as a series of polynomial evaluations. This lowers the time complexity from $\mathcal{O}(\sqrt{pe})$ to $\mathcal{O}(\sqrt{p}\sqrt[4]{e})$ for digit extraction modulo p^e, at the cost of a slight increase in multiplicative depth. Overall, our implementation in `HElib` shows a significant speedup of a factor up to 2.6 over the state-of-the-art.

1 Introduction

Homomorphic encryption (HE) allows computations on encrypted data without knowledge of the secret key. In the past 15 years, there have been tremendous improvements in HE protocols, both in speed and applicability. In spite of these efforts, homomorphic encryption remains extremely slow compared to unencrypted computations and further speedups are required.

Homomorphic computations are typically realized as arithmetic circuits, i.e. sequences of additions and multiplications that implement a desired functionality. In the lattice-based schemes BGV [6] and BFV [5, 10], these operations are performed over (extensions of) $\mathbb{Z}_{p^e}$, where p is a prime number and e is a positive integer.[1] Functions on $\mathbb{Z}_{p^e}$ have rather interesting properties. First, only a limited class of functions can be described by polynomials with integer coefficients, the so-called *polyfunctions* (short for polynomial functions). Second, the polynomials that represent a given polyfunction are always non-unique and we can therefore try to find the polynomial representation that is most efficient to evaluate homomorphically.

An example application that can benefit from the study of polyfunctions is *bootstrapping* – the ciphertext refreshing procedure that enables unbounded fully homomorphic encryption. This procedure is necessary because lattice-based schemes include a noise term that grows when we evaluate an arithmetic circuit. Bootstrapping reduces the noise back to a lower level, which enables further evaluation of homomorphic additions and multiplications. Since its introduction by Gentry in 2009 [11], the latency and throughput of bootstrapping were improved several orders of magnitude in many subsequent works [8, 13, 15], but it remains the main bottleneck to achieve fully homomorphic encryption.

1.1 Related Work

Polyfunctions. Research into polyfunctions has a long history. Already in 1921, Kemper [17] studied elementary structures of polyfunctions over $\mathbb{Z}_m$ for a composite integer m. This early research represents polynomials in the monomial basis $\{X^i\}_{i=0,1,\ldots}$. However, since the mid-1960s, much of the literature [7, 9, 16, 23] started to use the falling factorial basis $\{X \cdot (X-1) \cdot \ldots \cdot (X-i)\}_{i=0,1,\ldots}$. The reason for this shift is that the falling factorial polynomials almost directly give rise to non-trivial *null polynomials* (i.e. polynomials that by definition evaluate to zero in every point when interpreted modulo some prime power p^e).

Null polynomials result in equivalent representations of the same polyfunction $f\colon \mathbb{Z}_{p^e} \to \mathbb{Z}_{p^e}$. Specifically, two polynomials $F(X), H(X) \in \mathbb{Z}[X]$ represent the same function f if and only if their difference $F(X) - H(X)$ is a null polynomial. Equivalently, the set of all possible representations of f is obtained as $F(X) + \mathcal{O}_{p^e}$, where $\mathcal{O}_{p^e}$ is the set of all null polynomials modulo p^e. In other words, there exists a one-to-one correspondence between polyfunctions

[1] Some protocols for secure multi-party computation [3] also work over $\mathbb{Z}_{p^e}$, which makes our study of polyfunctions even more widely applicable. However, improvements in multi-party computation are not the direct focus of this paper.

and collections of equivalent polynomials:

$$\text{polyfunction } f : \mathbb{Z}_{p^e} \to \mathbb{Z}_{p^e} \iff F(X) + \mathcal{O}_{p^e}.$$

Bootstrapping. The first bootstrapping procedure for BGV was proposed by Gentry et al. [13] for encryption of bits, and this method was later improved by subsequent research. The most relevant works for this paper are from Alperin-Sheriff and Peikert [1], which is generalized by Halevi and Shoup [15]; and from Chen and Han [8]. Alperin-Sheriff/Peikert/Halevi/Shoup proposed a bootstrapping method that works for the more general plaintext space $\mathbb{Z}_{p^e}$. Their technique relies on a "digit removal" procedure, which involves repeated homomorphic evaluation of the *lifting polynomial* and has degree p^{e-1} in total. Chen/Han introduced an additional *digit extraction polynomial* (sometimes called the *lowest digit retain polynomial*) that has a much lower degree equal to $(p-1) \cdot (e-1) + 1$. Lower degrees are typically favored in homomorphic encryption.

In practice, polynomial evaluations account for most of the computational cost of bootstrapping: in the implementation of `HElib`, they are altogether $3\times$ to $50\times$ more expensive than all other operations combined [15]. This situation is exactly the same for BGV and BFV, because both schemes have an identical bootstrapping procedure.

1.2 Our Contributions

The aim of this paper is to further develop the theory of polyfunctions with a focus on cryptographic applications. New insights in these polyfunctions allow us to significantly accelerate HE bootstrapping.

Polyfunctions. In the first part of the paper (Section 3), we study polyfunctions modulo p^e. This includes the following:

- In Section 3.1, we study the complete set of null polynomials modulo p^e (denoted by $\mathcal{O}_{p^e}$) as to obtain the set of all equivalent polyfunction representations. A novel element of our approach is also restricting $\mathcal{O}_{p^e}$ to contain only polynomials of bounded degree. When doing so, the resulting set forms a lattice structure, and we can find small-coefficient representations by solving the closest vector problem in this lattice. This is interesting in homomorphic encryption, because small coefficients lead to less noise growth.
- In Section 3.3, we extend Newton interpolation from the real numbers to $\mathbb{Z}_{p^e}$. Our method always returns a polynomial representation of the

lowest degree when given a polyfunction as input. When given a function that is not a polyfunction, our method can detect this and returns an error.
- In Section 3.5, we discuss several properties of polyfunctions that are especially relevant for HE bootstrapping. In particular, we consider the class of even and odd polyfunctions that satisfy respectively $f(-a) = f(a) \pmod{p^e}$ and $f(-a) = -f(a) \pmod{p^e}$ for $a \in \mathbb{Z}$. We show that each such function can be represented by a sparse polynomial with only even- or odd-exponent terms. Evaluating such a sparse representation is asymptotically cheaper by a factor of $\sqrt{2}$.

Bootstrapping. In the second part of the paper (Sections 4 and 5), we apply the newly developed theory to speed up BGV and BFV bootstrapping. The most expensive component of bootstrapping, both in degree and execution time, is evaluation of the digit extraction polynomial. In order to accelerate it, we apply the following improvements:

- We propose multiple methods to obtain better representations of the digit extraction function. First, we show that this function is either even or odd, and can therefore be represented as a polynomial with only 50% of the coefficients. Second, we propose a new technique to decompose digit extraction in multiple stages. Let g_e be the digit extraction function modulo p^e, then we write it as $g_e = g_{e,e'} \circ g_{e'}$. In our algorithm, both $g_{e'}$ and $g_{e,e'}$ are evaluated using polynomials of much smaller degree than the direct approach. As a consequence, we lower the time complexity for digit extraction from $\mathcal{O}(\sqrt{pe})$ to $\mathcal{O}(\sqrt{p}\sqrt[4]{e})$, at the cost of $\lceil \log_2 p \rceil$ increase in multiplicative depth.
- In order to fully benefit from the optimized digit extraction polynomials, we revise the digit removal procedure of Chen/Han [8]. Our improved algorithm utilizes the digit extraction polynomial exclusively, without relying on the lifting polynomial. We implemented our new bootstrapping algorithm in `HElib`, and observe that it is up to 2.6 times faster than the state-of-the-art. Our code is made publicly available.[2]

[2]See `https://github.com/KULeuven-COSIC/Bootstrapping_Polyfunctions`.

Table 1: Examples of $\nu_2(n!)$

n	1	2	3	4	5	6	7	8	9	10
$\nu_2(n!)$	0	1	1	3	3	4	4	7	7	8

Table 2: Examples of $\mu(2^e)$

e	1	2	3	4	5	6	7	8	9	10
$\mu(2^e)$	2	4	4	6	8	8	8	10	12	12

2 Preliminaries

2.1 Notations

For prime p and integer exponent $e \geqslant 1$, the set of functions from $\mathbb{Z}_{p^e}$ to itself is denoted by $\mathcal{F}_{p^e}$. Moreover, we write the evaluation of a polynomial $F(X)$ at $X = a$ as $F(a)$ or sometimes $F(X)|_{X=a}$.

Let $\nu_p(\cdot)$ denote the *p-adic valuation* function defined as

$$\nu_p(m) = \begin{cases} \max\{k \in \mathbb{N} : p^k \mid m\} & \text{if } m \neq 0 \\ \infty & \text{if } m = 0. \end{cases}$$

It generalizes to the rational numbers as $\nu_p(m/n) = \nu_p(m) - \nu_p(n)$, and we call a rational number *p-integral* if its p-adic valuation is non-negative. Let $\mu(\cdot)$ denote the *Smarandache* function defined as

$$\mu(k) = \min\{i \in \mathbb{N} : k \mid i!\}.$$

Observe that $\nu_p(\cdot)$ and $\mu(\cdot)$ are complementary in some sense. Specifically, it follows directly from the above definitions that $\mu(p^e)$ is the smallest integer for which $\nu_p(\mu(p^e)!) \geqslant e$. A few example instances of $\nu_p(n!)$ and $\mu(p^e)$ for $p = 2$ are listed in Tables 1 and 2.

2.2 Newton Interpolation over $\mathbb{R}$

The Falling Factorial Basis. The Newton interpolation method relies on the so-called *falling factorial polynomials*. Those polynomials are indexed by an integer $i \geqslant 0$ and defined as

$$(X)_i = \prod_{k=0}^{i-1} (X - k) \in \mathbb{Z}[X],$$

where by definition we set $(X)_0 = 1$. When reduced modulo p^e, these polynomials exhibit very specific properties that will be studied later in this paper.

Let $\mathbb{P}_n \subseteq \mathbb{Z}[X]$ be the set of polynomials of degree at most n. Obviously, the set $\{X^i \mid 0 \leqslant i \leqslant n\}$ forms a basis for $\mathbb{P}_n$ when seen as a module over $\mathbb{Z}$. We refer to it as the monomial basis. Similarly, also the set $\{(X)_i \mid 0 \leqslant i \leqslant n\}$ forms a basis for $\mathbb{P}_n$, known as the falling factorial basis.

Newton Interpolation. Consider a collection of $n+1$ data points $(i, y_i) \in \mathbb{R}^2$ for $i = 0, \ldots, n$.[3] Using Newton interpolation, we can find a polynomial $F(X)$ of degree at most n that interpolates these data points. Concretely, write the polynomial $F(X) \in \mathbb{R}[X]$ in the format

$$F(X) = c_0 + c_1(X)_1 + c_2(X)_2 + \ldots + c_n(X)_n. \tag{1}$$

Then we can uniquely determine the falling factorial coefficients c_i such that

$$F(i) = y_i, \quad \forall 0 \leqslant i \leqslant n.$$

The coefficients can be computed from forward differences, as introduced in the following definition.

Definition 2.1. The *i-th forward difference* of a function $f\colon \mathbb{R} \to \mathbb{R}$, evaluated at $j \in \mathbb{Z}$, is recursively defined as

$$\Delta^i f(j) = \begin{cases} f(j) & \text{if } i = 0 \\ \Delta^{i-1} f(j+1) - \Delta^{i-1} f(j) & \text{if } i > 0. \end{cases}$$

We will now apply these forward differences to a polynomial $F(X)$. Note that we slightly abuse notation and consider a polynomial as a function in X. As shown in Figure 1, the value of $\Delta^i F(X)|_{X=j}$ for $i, j = 0, 1, \ldots, n$ can be derived from Definition 2.1. Each element in this triangle is defined as $\alpha_{i,j} = \Delta^i F(X)|_{X=j}$, and computed as the difference between the element above and the element above left. We only show rows for $i = 0, \ldots, n$, because all following rows are zero for a polynomial of degree n. This is easily seen by computing

$$\begin{aligned} \Delta\,(X)_i &= (X+1)X \cdot \ldots \cdot (X-i+2) - X(X-1) \cdot \ldots \cdot (X-i+1) \\ &= i(X)_{i-1}, \end{aligned} \tag{2}$$

and using the result in Equation (1). Note that Equation (2) is the analogue of taking the derivative of the monomial X^i.

[3] In a more general version, we could consider the data points (x_i, y_i). For our purpose, however, it is sufficient to choose $x_i = i$.

The coefficients of the interpolating polynomial $F(X)$ can now be computed as $c_i = \alpha_{i,0} = \Delta^i F(X)|_{X=0}/i!$. This result is achieved by taking the i-th forward difference of both sides of Equation (1), and again filling in Equation (2). This leads to the interpolating polynomial

$$F(X) = \alpha_{0,0} + \alpha_{1,0}(X)_1 + \frac{\alpha_{2,0}}{2!}(X)_2 + \ldots + \frac{\alpha_{n,0}}{n!}(X)_n. \tag{3}$$

Note the analogy with the Taylor series of a function.

Finally, the following relations are useful:

$$\alpha_{0,j} = \sum_{v=0}^{j} \binom{j}{v} \alpha_{v,0}, \tag{4a}$$

$$\alpha_{i,0} = \sum_{v=0}^{i} (-1)^{i+v} \binom{i}{v} \alpha_{0,v}. \tag{4b}$$

These equations establish a relationship between the elements in the first row and the diagonal of Figure 1.

$\alpha_{0,0}$	$\alpha_{0,1}$	$\alpha_{0,2}$	$\cdots$	$\alpha_{0,n}$
	$\alpha_{1,0}$	$\alpha_{1,1}$	$\cdots$	$\alpha_{1,n-1}$
		$\alpha_{2,0}$	$\cdots$	$\alpha_{2,n-2}$
			$\ddots$	$\vdots$
				$\alpha_{n,0}$

Figure 1: Evaluation of forward differences with $\alpha_{i,j} = \Delta^i F(X)|_{X=j}$.

The above theory generalizes directly to polynomial rings over any field. However, the subject of this paper is polynomials over $\mathbb{Z}_{p^e}$, which is not a field in general.

2.3 Polyfunctions Modulo p^e

Definition 2.2. Let $f \in \mathcal{F}_{p^e}$ be a function from $\mathbb{Z}_{p^e}$ to itself. If there exists a polynomial $F(X) \in \mathbb{Z}[X]$ that satisfies $F(a) = f(a) \pmod{p^e}$ for all $a \in \mathbb{Z}$, then f is a *polyfunction modulo* p^e and $F(X)$ is a *representation* of f.[4]

As a corollary of the theory in Section 2.2, all functions from the field $\mathbb{F}_p$ to itself are polyfunctions. A unique representation of degree less than p is obtained

[4]We define the evaluation of a function $f \in \mathcal{F}_{p^e}$ at an integer a in the natural way, by implicitly converting a to its residue class modulo p^e.

by starting from all data points and applying Newton interpolation. However, the situation is different for functions modulo p^e: first, not all functions are described by integer polynomials modulo p^e, regardless of the degree; second, polyfunctions always have a non-unique representation of the lowest degree due to the existence of *null polynomials* [16, 19, 21, 23].

Null Polynomials Modulo p^e. We define a null polynomial as follows.

Definition 2.3. An element $O(X) \in \mathbb{Z}[X]$ is called a *null polynomial modulo* p^e if the function $f \in \mathcal{F}_{p^e}$ that it represents maps every element to zero. In other words, we have that $O(a) = 0 \pmod{p^e}$ for all $a \in \mathbb{Z}$.

Observe that the evaluation of the falling factorial polynomial $(X)_i$ at *any* integer is divisible by $i!$. Hence it is a null polynomial modulo p^e if $\nu_p(i!) \geqslant e$. Also the other direction holds: if $(X)_i$ is a null polynomial modulo p^e, then evaluating it at $X = i$ gives $(X)_i\,|_{X=i} = i!$, and therefore $\nu_p(i!) \geqslant e$. Following the notation defined earlier, we find that the smallest possible value of i for which $(X)_i$ is a null polynomial modulo p^e, is equal to $i = \mu(p^e)$.

2.4 Lattices

Definition 2.4. The set $\mathcal{L} \subseteq \mathbb{R}^n$ is a *lattice* if there exist $\mathbb{R}$-linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$ such that

$$\mathcal{L} = \left\{ \sum_{i=1}^{k} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}.$$

The set of vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ constitute a *basis*, and k is called the *rank*. A lattice is called *q-ary* for an integer q if $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$.

For a lattice vector $\mathbf{v} \in \mathcal{L}$, the length $\|\mathbf{v}\|$ denotes its Euclidean norm (2-norm). We will rely on the closest vector problem (CVP):

Definition 2.5 (Closest vector problem (exact form))**.** Consider a lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and a vector $\mathbf{t} \in \mathbb{R}^n$, CVP asks to recover a lattice vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{v}\| = \min_{\mathbf{y} \in \mathcal{L}} \|\mathbf{t} - \mathbf{y}\|$.

Lattices have been studied extensively in cryptography due to the conjectured intractability of certain lattice problems, such as the shortest vector problem (SVP) and the closest vector problem (CVP). The hardness of these problems is used as the security foundation of many cryptosystems, including the BGV and BFV schemes. However, we will use lattices for a different reason, namely the study of polynomial representations with small coefficients.

2.5 Homomorphic Encryption

We are interested in homomorphic encryption schemes that support arithmetic circuits over $\mathbb{Z}_{p^e}$. In the literature, those schemes are known as BGV [6] and BFV [5, 10]. Both schemes have the same interface, and only differ from each other in terms of the underlying implementation.

Homomorphic Operations. Next to the usual key generation, encryption and decryption, homomorphic encryption schemes have two extra procedures to evaluate additions and multiplications over the ciphertexts that they encrypt. Both procedures can either take two ciphertexts, or one ciphertext and one plaintext. Moreover, there is one special division operation, which takes a ciphertext that encrypts a message m known to be divisible by p. It outputs a new ciphertext that encrypts m/p, but under plaintext modulus p^{e-1} instead of p^e. This operation fails if the input message is not divisible by p.

Plaintext Batching. BGV and BFV can batch multiple elements of $\mathbb{Z}_{p^e}$ per plaintext [22]. Specifically, the plaintext ring is isomorphic to $\mathbb{Z}_{p^e}^{\ell}$, where addition and multiplication are defined component-wise. Each copy of $\mathbb{Z}_{p^e}$ is called a *plaintext slot*, and can be operated on homomorphically and in parallel. This is sometimes referred to as SIMD operations due to the resemblance in parallel computing architectures.

The above explanation is actually a special case of a more general technique. Given a polynomial $F(X) \in \mathbb{Z}[X]$ that is irreducible modulo p, we can define the Galois ring $E = \mathbb{Z}_{p^e}[X]/(F(X)) \supseteq \mathbb{Z}_{p^e}$. The plaintext rings of BGV and BFV are then isomorphic to E^{ℓ}, again with component-wise addition and multiplication. We refer to this more general version as fully packed slots. If the slots are restricted to encode elements from the subring $\mathbb{Z}_{p^e}$ (like explained above), then they are called sparsely packed.

Bootstrapping. Every HE ciphertext contains a special component called the *noise*. When evaluating homomorphic additions and multiplications, the noise gets larger depending on the complexity of the involved operations. The decryption function removes the noise, but only works correctly if the noise is small enough (depending on the chosen scheme parameters).

To enable circuits that consist of an unlimited number of additions and multiplications, we need a method to reduce the ciphertext noise without decrypting directly. This is achieved via bootstrapping. The idea is to decrypt a ciphertext *homomorphically* by evaluating the scheme's own decryption circuit. This reduces noise and allows further evaluation of additions and multiplications.

Bootstrapping comes in two variants: the slots of the encrypted message can either be fully packed or sparsely packed. We refer to the first situation as general bootstrapping, and the second one as thin bootstrapping. Finally, we emphasize that BGV and BFV have an identical bootstrapping procedure. All optimizations for one scheme therefore carry over to the other one immediately.

3 Systematic Study of Polyfunctions

3.1 Null Polynomials

The set of null polynomials modulo p^e can be described in the falling factorial basis. This was already noticed by Singmaster [21] who proved the general structure of this set. We formulate an adapted version in the following theorem, where we additionally take into account null polynomials of bounded degree. Our theorem is proven based on the same outline as Singmaster's proof.

Theorem 3.1. *A polynomial $O(X) \in \mathbb{Z}[X]$ is a null polynomial modulo p^e of degree at most n if and only if there exist $a_0, \dots a_n \in \mathbb{Z}$ such that*

$$O(X) = \sum_{i=0}^{n} a_i \cdot O_i(X), \quad \textit{with} \quad O_i(X) = p^{\max(e-\nu_p(i!),0)} \cdot (X)_i \,. \tag{5}$$

In this equation, the exponent of p equals 0 if $i \geqslant \mu(p^e)$.

Proof. ($\Leftarrow$) As already pointed out in Section 2.3, the evaluation of $(X)_i$ at any integer is divisible by $p^{\nu_p(i!)}$. Therefore, each term in Equation (5) evaluated at any integer is divisible by $p^{\max(e,\nu_p(i!))} \geqslant p^e$. Since each term is a null polynomial modulo p^e, so is their linear combination.

($\Rightarrow$) We prove the following assertion for $0 \leqslant m \leqslant n+1$ by applying induction on m:

$$O(X) = \sum_{i=m}^{n} b_i \cdot (X)_i + \sum_{i=0}^{m-1} a_i \cdot O_i(X), \tag{6}$$

for some $a_i, b_i \in \mathbb{Z}$.

The base case $m = 0$ is trivial since the second sum is empty, and the first sum amounts to writing a polynomial in the falling factorial basis. It is therefore possible to find appropriate constants b_i that satisfy Equation (6).

Now suppose that Equation (6) was established for some $m < n+1$, that is

$$O(X) = b_m \cdot (X)_m + \sum_{i=m+1}^{n} b_i \cdot (X)_i + \sum_{i=0}^{m-1} a_i \cdot O_i(X).$$

Evaluating both sides at $X = m$ gives

$$0 = O(m) = b_m \cdot m! \pmod{p^e}.$$

Taking the p-adic valuation of the right-hand side gives

$$\nu_p(b_m \cdot m!) = \nu_p(b_m) + \nu_p(m!) \geqslant e \implies \nu_p(b_m) \geqslant e - \nu_p(m!).$$

The constants b_i are integers, so it follows that $\nu_p(b_m) \geqslant \max(e - \nu_p(m!), 0)$. We can therefore write $b_m = a_m \cdot p^{\max(e-\nu_p(m!),0)}$ for some $a_m \in \mathbb{Z}$, which results in

$$O(X) = \sum_{i=m+1}^{n} b_i \cdot (X)_i + \sum_{i=0}^{m} a_i \cdot O_i(X).$$

This expression replaces m by $m+1$ in Equation (6) and thereby completes the induction. The final result follows by setting $m = n + 1$ in Equation (6). □

Corollary 3.1. *Each null polynomial modulo p^e of degree $n < \mu(p^e)$ is divisible by $p^{e-\nu_p(n!)}$, where divisibility is defined in the polynomial ring $\mathbb{Z}[X]$. Therefore, all monic null polynomials have degree at least $\mu(p^e)$.*

Corollary 3.2. *The set of all null polynomials modulo p^e is obtained directly from Theorem 3.1 by allowing an arbitrarily large (but finite) degree n.*

The Null Lattice. Adopting the notation from Equation (5), the set of null polynomials of degree at most n is given by

$$\mathcal{O}_{p^e}^{(n)} = \left\{ \sum_{i=0}^{n} a_i \cdot O_i(X) \mid a_i \in \mathbb{Z} \right\} \subseteq \mathbb{P}_n.$$

When considering polynomials as coefficient vectors, it can easily be seen that the above set forms a p^e-ary lattice with basis vectors $O_i(X)$. For convenience of notation, we will not make a difference between polynomials and lattice vectors: the set $\mathcal{O}_{p^e}^{(n)}$ inherits all properties from Section 2.4, including the norm.

3.2 Cosets of Equivalent Polynomials

A representation $F(X)$ of a polyfunction f is never unique. That is, given a null polynomial $O(X)$, we can construct an equivalent polynomial $H(X) = F(X) + O(X)$ that represents the same polyfunction. The set of all representations of a polyfunction forms the coset $F(X)+\mathcal{O}_{p^e}$. Moreover, the set of all representations of degree at most n forms the coset $F(X) + \mathcal{O}_{p^e}^{(n)}$ (assuming that $\deg(F) \leqslant n$).

As explained in Section 2.3, $(X)_{\mu(p^e)}$ is a monic null polynomial modulo p^e, which implies that we can always divide by it (using Euclidean division) to obtain a representation of degree less than $\mu(p^e)$. This proves the following lemma.

Lemma 3.1 (Small degree representation [16])**.** *Each polyfunction $f \in \mathcal{F}_{p^e}$ has a representation of degree strictly less than $\mu(p^e)$.*

Although Euclidean division always returns a representation of degree less than $\mu(p^e)$, it is not necessarily minimized. In order to guarantee the lowest possible degree, one has to consecutively divide by $O_i(X)$ for $i = \mu(p^e), \ldots, 0$. This leads to the canonical representation of Keller and Olson [16].

Theorem 3.2 (Canonical representation [16])**.** *Let $f \in \mathcal{F}_{p^e}$ be a polyfunction, then there exists a unique canonical representation*

$$F(X) = \sum_{i=0}^{\mu(p^e)-1} c_i (X)_i$$

with $0 \leqslant c_i < p^{e-\nu_p(i!)}$.

From Theorem 3.2, we can compute the number of canonical representations, which is equal to the number of polyfunctions modulo p^e. This is done by adding the total number of possibilities for the coefficients c_i, which gives

$$\operatorname{vol}\left(\mathcal{O}_{p^e}^{(\mu(p^e)-1)}\right) = \exp_p\left(\sum_{k=0}^{\mu(p^e)-1} e - \nu_p(k!)\right) = \exp_p\left(\sum_{k=1}^{e} \mu(p^k)\right), \quad (7)$$

where $\operatorname{vol}(\cdot)$ denotes the volume of a lattice and $\exp_p(\cdot)$ the exponential function with base p. The first equality highlights the one-to-one correspondence between polyfunctions and equivalent representations of degree less than $\mu(p^e)$, obtained as cosets modulo the null lattice. The second equality was proven by Specker et al. [23] and not repeated here for brevity.

Note that, although a canonical representative can be chosen in a unique manner, it is not necessarily the most convenient polynomial to evaluate homomorphically. Later we study the digit extraction polynomial in FHE bootstrapping, where we take a different representative than the canonical choice.

Finally, we compare the number of functions in $\mathcal{F}_{p^e}$ to the number of polyfunctions from Equation (7). Since a function is uniquely determined by its input-output pairs, the total number of functions equals $(p^e)^{p^e} = p^{e \cdot p^e}$. This expression is typically much larger than Equation (7) for $e \geqslant 2$, so only very few functions are representable by polynomials.

Example 3.1. *There are $2^{8 \cdot 2^8} \approx 10^{617}$ functions in $\mathcal{F}_{2^8}$, while only $2^{50} \approx 10^{15}$ of them are polyfunctions as computed from Equation* (7)*.*

3.3 Existence of Polynomial Representation

In this section, we examine whether a given function $f \in \mathcal{F}_{p^e}$ is a polyfunction or not. We extend the Newton interpolation method to functions modulo p^e, and return a representation of the lowest degree if f is a polyfunction.

Consider a function $f \in \mathcal{F}_{p^e}$ that is defined by p^e data points $(i, f(i)) \in \mathbb{Z}_{p^e}^2$ for $i = 0, \ldots, p^e - 1$. We will now use *reduced* forward differences, which are similar to the regular forward difference defined earlier, but include an extra reduction modulo p^e in the set $\{0, \ldots, p^e - 1\}$.

Definition 3.1. The *reduced i-th forward difference* of a function $f \in \mathcal{F}_{p^e}$, evaluated at $j \in \mathbb{Z}$, is defined as

$$\overline{\Delta^i} f(j) = \Delta^i f(j) \pmod{p^e}.$$

The values $\overline{\Delta^i} f(j)$ for $i = 0, 1, \ldots, \mu(p^e)$ and $j = 0, \ldots, p^e - 1$ can be derived from Definition 3.1. This is shown in Figure 2, where $\alpha_{i,j} = \overline{\Delta^i} F(X)|_{X=j}$.

The relations in Section 2.2 such as Equations (2), (4a), and (4b) still hold modulo p^e. Moreover, we will show later that the interpolating polynomial from Equation (3) is also valid for polyfunctions over $\mathbb{Z}_{p^e}$.

$$\begin{array}{ccccccc}
\alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,\mu(p^e)} & \cdots & \alpha_{0,p^e-1} \\
 & \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,\mu(p^e)-1} & \cdots & \alpha_{1,p^e-2} \\
 & & \alpha_{2,0} & \cdots & \alpha_{2,\mu(p^e)-2} & \cdots & \alpha_{2,p^e-3} \\
 & & & \ddots & \vdots & & \vdots \\
 & & & & \alpha_{\mu(p^e),0} & \cdots & \alpha_{\mu(p^e),p^e-\mu(p^e)-1}
\end{array}$$

Figure 2: Evaluation of reduced forward differences with $\alpha_{i,j} = \overline{\Delta^i} F(X)|_{X=j}$.

Polynomial Representation. In order to examine if a function $f \in \mathcal{F}_{p^e}$ is a polyfunction, we introduce a new lemma.

Lemma 3.2. *Let $F(X) \in \mathbb{Q}[X]$ be a polynomial of degree less than $\mu(p^e)$ with evaluation function f. Then f with all evaluations interpreted modulo p^e is a polyfunction if and only if the coefficients of $F(X)$ are p-integral.*

Proof. ($\Leftarrow$) If the coefficients of $F(X)$ are p-integral, then it can be coerced into $\mathbb{Z}[X]$ by replacing all denominators by their multiplicative inverse modulo p^e.

($\Rightarrow$) Since f is a polyfunction, there exists a representation $H(X) \in \mathbb{Z}[X]$ of degree less than $\mu(p^e)$. The polynomial $O(X) = H(X) - F(X)$ also has degree less than $\mu(p^e)$, and its evaluation function modulo p^e is zero. Writing

$$O(X) = \sum_{i=0}^{\mu(p^e)-1} \frac{a_i}{b_i} \cdot (X)_i,$$

where a_i/b_i is a fraction in simplest form, it suffices to prove $\nu_p(b_i) = 0$ for all i. Assume on the contrary that $\nu_p(b_i) = \max_j\{\nu_p(b_j)\} = c > 0$, then $p^c \cdot O(X)$ can be coerced into a null polynomial modulo p^{c+e}. Since the degree of this null polynomial is strictly less than $\mu(p^e) \leqslant \mu(p^{c+e})$, it follows from Corollary 3.1 that

$$p^c \cdot \frac{a_i}{b_i} = 0 \pmod{p}.$$

From $\nu_p(b_i) = c$, it follows directly that $a_i = 0 \pmod{p}$. Hence both a_i and b_i are divisible by p, which contradicts the fact that a_i/b_i is in its simplest form. □

Remark 3.1. *A polynomial $F(X) \in \mathbb{Q}[X]$ with non-p-integral coefficients can still represent a (poly)function $f \in \mathcal{F}_{p^e}$, for example if its degree is at least $\mu(p^e)$. However, it is not directly possible to evaluate such a function homomorphically.*

Now we introduce a simple way to decide whether a given function $f \in \mathcal{F}_{p^e}$ is a polyfunction, relying on the reduced forward differences from Figure 2.

Theorem 3.3. *A function $f \in \mathcal{F}_{p^e}$ is a polyfunction if and only if the following two criteria are satisfied:*

1. *For all $i < \mu(p^e)$, we have $\nu_p(\alpha_{i,0}) \geqslant \nu_p(i!)$. Note that $\alpha_{i,0}$ are the diagonal elements of Figure 2.*
2. *All elements in the last row of Figure 2 are zero.*

Proof. ($\Leftarrow$) Consider the polynomial

$$F(X) = \sum_{i=0}^{p^e-1} \frac{\alpha_{i,0}}{i!} \cdot (X)_i \in \mathbb{Q}[X]. \tag{8}$$

Following Equation (3), this polynomial interpolates f in all data points. Now we are given that all elements in the last row of Figure 2 are zero, thus so are all values in the next row (which is not displayed). Hence $\alpha_{i,0} = 0$ for all $i \geqslant \mu(p^e)$. Therefore, we can terminate the summation of Equation (8) earlier and get

$$F(X) = \sum_{i=0}^{\mu(p^e)-1} \frac{\alpha_{i,0}}{i!} \cdot (X)_i \in \mathbb{Q}[X]. \tag{9}$$

Now it remains to prove that the coefficients of $F(X)$ are p-integral, and then the result follows immediately from Lemma 3.2. Considering Equation (9), this is trivial since we are given that $\nu_p(\alpha_{i,0}) \geqslant \nu_p(i!)$ for all $i < \mu(p^e)$.

($\Rightarrow$) If f is a polyfunction, it has a representation $F(X)$ of degree less than $\mu(p^e)$. Hence it follows from Equation (2) that $\overline{\Delta^{\mu(p^e)}}F(X)$ is zero in every point, which proves the second criterion of the theorem.

Consider again the polynomial of Equation (9). Following the same line of reasoning as in the first part of this proof, it is a representation of f modulo p^e. According to Lemma 3.2, we know that $F(X)$ must have p-integral coefficients, which implies $\nu_p(\alpha_{i,0}) \geqslant \nu_p(i!)$ for all $i < \mu(p^e)$. □

Interestingly, Equation (9) gives a polynomial representation $F(X)$ obtained by Newton interpolation restricted to $\{0, 1, \ldots, \mu(p^e) - 1\}$, i.e. only information about $f(i)$ for $i < \mu(p^e)$ has been used. Condition 1 of Theorem 3.3 can be interpreted as restricting the coefficients of $F(X)$ to p-integral values. Condition 2 is a consistency requirement: $F(a) = f(a) \pmod{p^e}$ for each $a \in \{\mu(p^e), \ldots, p^e - 1\}$. Finally, we note that also different interpolation methods could be used.

Corollary 3.3. *If f is a polyfunction, then Equation* (9) *gives a representation of the lowest degree.*

Proof. It was already proven that the polynomial $F(X)$ from Equation (9) can be coerced into a representation in $\mathbb{Z}[X]$, so it remains to show that its degree is minimal. Suppose that n is the largest integer such that $\alpha_{n,0} \neq 0$, and assume on the contrary that there exists a representation $H(X)$ whose degree is less than n. Then $O(X) = F(X) - H(X)$ is a null polynomial modulo p^e, with leading monomial $(\alpha_{n,0}/n!) \cdot X^n$. It follows from Corollary 3.1 that

$$\frac{\alpha_{n,0}}{n!} = 0 \pmod{p^{e-\nu_p(n!)}},$$

and thus $\alpha_{n,0} = 0 \pmod{p^e}$, leading to a contradiction. □

3.4 Bit and Digit Extraction Function

As an example, we apply the previously developed theory to the bit extraction function – a polyfunction that is useful in the part about FHE bootstrapping.

Example 3.2. *Let $g_e \in \mathcal{F}_{2^e}$ be the bit extraction function defined as*

$$g_e : \mathbb{Z}_{2^e} \to \mathbb{Z}_{2^e} : a \mapsto a \pmod 2,$$

0	1	0	$\cdots$	1	$\cdots$	0	$\cdots$	1
	1	-1	$\cdots$	1	$\cdots$	-1	$\cdots$	1
		-2	$\cdots$	2	$\cdots$	-2	$\cdots$	2
			$\ddots$	$\vdots$		$\vdots$		$\vdots$
				$(-2)^{i-1}$	$\cdots$	-2^{i-1}	$\cdots$	2^{i-1}
					$\ddots$	$\vdots$		$\vdots$
						$(-2)^{\mu(2^e)-1}$	$\cdots$	$2^{\mu(2^e)-1}$

Figure 3: Forward differences of the bit extraction function.

where reduction modulo 2 is done in the set $\{0,1\}$*. Its forward differences are shown in Figure 3, which should be closely compared to Figure 2. The reduced forward differences are computed via reduction modulo* 2^e*. It can easily be verified in Table 1 that the diagonal elements* $\alpha_{i,0} = (-2)^{i-1}$ *satisfy* $\nu_2(\alpha_{i,0}) \geqslant \nu_2(i!)$*, and that all elements on the last row are congruent to zero modulo* 2^e*. Therefore, the bit extraction function is a polyfunction.*

Following Corollary 3.3, the polynomial

$$G_e(X) = \sum_{i=1}^{e} \frac{(-2)^{i-1}}{i!} \cdot (X)_i \tag{10}$$

is a representation of g_e *of the lowest degree. It follows that there does not exist a bit extraction polynomial of degree less than* e*. Finally, the complete set of representations is easily obtained as* $G_e(X) + \mathcal{O}_{2^e}$*.*

More generally, we define the digit extraction function modulo p^e for any prime p from its balanced digit decomposition. Denote the balanced digits of $w \in \mathbb{Z}_{p^e}$ by $w_i \in \{-(p-1)/2, \ldots, (p-1)/2\}$ such that

$$w = \sum_{i=0}^{e-1} w_i p^i,$$

then we define the map $g_e \in \mathcal{F}_{p^e}$ as

$$g_e \colon \mathbb{Z}_{p^e} \to \mathbb{Z}_{p^e} \colon w \mapsto w_0.$$

Analogously to the previous example, we can show that g_e is a polyfunction and obtain a representation of the lowest degree. In the general case, there does not exist a digit extraction polynomial of degree less than $(p-1)(e-1)+1$. The complete set of representations is obtained by adding $\mathcal{O}_{p^e}$.

3.5 Further Properties of Polyfunctions

Not every function is a polyfunction modulo p^e. For example, the function

$$f(a) = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$$

is not a polyfunction for $e > 1$, because it is not *congruence preserving*. More specifically, all polyfunctions satisfy the following lemma.

Lemma 3.3 (Congruence preservation [7, 9, 16, 17])**.** *Let f be a polyfunction modulo p^e, then for any $a \in \mathbb{Z}$, we have*

$$f(a + p^k) = f(a) \pmod{p^k}, \quad \forall k \leqslant e. \tag{11}$$

Proof. Let $F(X)$ be a representation of f. Since a polynomial is built from additions and multiplications only, we know that

$$F(a + p^k) = F(a) \pmod{p^k}.$$

Since $p^k \mid p^e$, we can directly replace F by f. This completes the proof. □

Congruence preservation is not a sufficient condition to be a polyfunction [4]. In Section 3.3 - Theorem 3.3, we derived a necessary and sufficient condition for a function to be a polyfunction based on reduced forward differences [17], which is consistent with the analytical characterization by Carlitz [7] and further leads to a representation of the lowest degree.

We can also give a sufficient but unnecessary condition for a function to be a polyfunction. A function that satisfies

$$f(a + p) = f(a) \pmod{p^e}, \quad \forall a \in \mathbb{Z}, \tag{12}$$

is said to have period p and is always a polyfunction. A representation can be derived as follows.

Lemma 3.4 (Adapted from [14])**.** *The polynomial $U(X) = 1 - X^{\varphi(p^e)}$ satisfies the following property modulo p^e:*

$$\forall a \in \mathbb{Z}\colon U(a) = \begin{cases} 1 & \textit{if } p \mid a \\ 0 & \textit{otherwise,} \end{cases}$$

where $\varphi(\cdot)$ is Euler's totient function.

A representation for a function f with period p is

$$F(X) = \sum_{k=0}^{p-1} f(k) \cdot U(X - k), \tag{13}$$

from which we can construct the set of complete representations $F(X) + \mathcal{O}_{p^e}$. A well-known example is the digit extraction function.

As shown by the next example, having period p is a sufficient, but not a necessary condition for a function to be a polyfunction.

Example 3.3. *As computed in Example 3.1, there are* $2^{8 \cdot 2^8} \approx 10^{617}$ *functions in* $\mathcal{F}_{2^8}$*, while only* $2^{50} \approx 10^{15}$ *of them are polyfunctions. From these polyfunctions, only* $2^{8 \cdot 2} \approx 10^5$ *have period* 2*.*

Even and Odd Polyfunctions. We construct a new lemma to find sparse representations of even and odd polyfunctions.

Lemma 3.5. *Let* $f \in \mathcal{F}_{p^e}$ *be an even (resp. odd) polyfunction, that is,* $f(-a) = f(a) \pmod{p^e}$ *(resp.* $f(-a) = -f(a) \pmod{p^e}$*) for* $a \in \mathbb{Z}$*. Moreover, assume that* f *has a degree-*n *representation. Then the following holds:*

- *If* p *is an odd prime, then* f *has a representation* $F(X)$ *of degree at most* n*, which contains only even (resp. odd) exponents.*
- *If* $p = 2$*, and we consider* f *modulo* p^{e-1} *instead of* p^e*, then it has a representation* $F(X)$ *of degree at most* n*, which contains only even (resp. odd) exponents.*

Proof. Consider a representation $H(X) \in \mathbb{Z}[X]$ of f that has degree equal to n. Due to the evenness (resp. oddness) of f, the polynomial $H'(X) = H(-X)$ (resp. $H'(X) = -H(-X)$) is an equivalent representation of f.

Now we consider the integer polynomial

$$F(X) = \frac{H(X) + H'(X)}{2}, \tag{14}$$

which contains only even (resp. odd) exponents and has degree at most n. By evaluating Equation (14) in any $a \in \mathbb{Z}$, we see that $F(a) = f(a) \pmod{p^e}$ for an odd prime p, and $F(a) = f(a) \pmod{p^{e-1}}$ for $p = 2$. Hence $F(X)$ is also a representation of f, and it can easily be checked that it contains only even (resp. odd) exponents. □

4 Faster Bootstrapping for BGV and BFV

This section explains our improved bootstrapping techniques for BGV and BFV, leveraging the observations from the first part of the paper. Both general and thin bootstrapping involve two important components: the linear transformations and digit removal. We do not propose adaptations to the linear transformations, and leave them unchanged in the implementation. Our improvements are inside the digit removal step, and follow from the polyfunctions theory. Since digit removal is $3\times$ to $50\times$ more expensive than the linear transformations [15], any speedup leads to an almost equal effect in the entire bootstrapping procedure.

4.1 Cost Model

Amdahl's law [2] states that the speedup gained by optimizing a single part of an algorithm is limited to the fraction of time that the improved part is used. In order to accelerate digit removal, we must therefore concentrate on the slowest and most commonly used FHE operations. The true bottleneck of digit removal is *non-scalar multiplication*, i.e. multiplication of two ciphertexts. For an example parameter set with ring dimension $N = 2^{16}$, non-scalar multiplication in `HElib` is $7\times$ more expensive than its scalar counterpart.

An approach that follows our cost model is the baby-step/giant-step algorithm for evaluating a set of polynomials with scalar coefficients in a common non-scalar point [12, 20]. It can asymptotically evaluate m degree-n polynomials with $2\sqrt{mn}$ non-scalar multiplications. Therefore, our implementation uses this algorithm for polynomial evaluation.

Although not the focus of this paper, digit removal is also costly in terms of multiplicative depth (which is by definition the maximal number of multiplications encountered in each possible input-output path). Our approach accelerates bootstrapping without significantly affecting the multiplicative depth of digit removal. This is achieved by exclusive use of low-degree polynomials.

4.2 Digit Removal Algorithm

The digit removal procedure removes the v least significant digits of its input $w \in \mathbb{Z}_{p^e}$ for a given prime number p and $v < e$. Formally, for odd p, denote the balanced digits of $w \in \mathbb{Z}_{p^e}$ by $w_i \in \{-(p-1)/2, \ldots, (p-1)/2\}$ such that

$$w = \sum_{i=0}^{e-1} w_i p^i.$$

Digit removal is then defined as the map

$$w \mapsto \left\lfloor \frac{w}{p^v} \right\rceil = \sum_{i=v}^{e-1} w_i p^{i-v}.$$

In other words, it consecutively scales and rounds the input. This is necessary in bootstrapping to remove the noise. To evaluate the procedure homomorphically, it is written as a series of polynomial evaluations and divisions.

Note that the balanced digit representation only exists for odd prime numbers. If $p = 2$, we need to consider the digits in $\{0, 1\}$, which causes the output of digit removal to be $\lfloor w/p^v \rfloor$. However, bootstrapping requires a rounding operation instead of flooring. This can be fixed by applying the simple equality $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$ just before digit removal.

Existing Digit Removal Algorithms. Digit removal uses the following notation: we write $w_{i,j}$ for *any* integer of which the least significant digit is w_i, and the next j least significant digits are all zeros. Formally, this means that $w_{i,j} = w_i \pmod{p^{j+1}}$. Moreover, we require two well-known polynomials:

- The *lifting polynomial* $F_e(X) \in \mathbb{Z}[X]$ satisfies $F_e(w_{i,j}) = w_{i,j+1}$ for $j \leqslant e$. In other words, it allows us to compute a valid $w_{i,j+1}$ from any given $w_{i,j}$ by zeroing one extra digit.
- The *digit extraction polynomial* $G_e(X) \in \mathbb{Z}[X]$ satisfies $G_e(w_{i,0}) = w_{i,e-1}$, which allows us to compute a valid $w_{i,e-1}$ from any given $w_{i,0}$ by zeroing $e - 1$ extra digits. In other words, it is a representation of the digit extraction function g_e introduced in Section 3.3.

It can be proven that the above polynomials always exist [8]. Their degrees are respectively p and $(e - 1)(p - 1) + 1$.

The high-level idea of digit removal is to use the lifting polynomial and/or digit extraction polynomial to extract the least significant digit of the input w. The result is then subtracted from w and divided by p, and this is repeated until enough digits are removed. A suitable choice of lifting and digit extraction polynomials ensures a low multiplicative depth of the resulting procedure. Digit removal is visualized in the trapezoid of Figure 4 for an example parameter set of $e = 5$ and $v = 3$. The procedure works as follows:

- We start from $w_{0,0} = w$ in the first row, and then compute the numbers on its right via a series of polynomial evaluations. The choice of polynomials depends on the chosen algorithm, and is explained later.
- In the second row, we first compute $w_{1,0} = (w - w_{0,1})/p$ and then repeat the same procedure from the first row.
- In the last row, we similarly compute $w_{2,0} = ((w - w_{0,2})/p - w_{1,1})/p$ and again repeat the same procedure from the first and second row.

- The result is obtained as $(((w - w_{0,4})/p - w_{1,3})/p - w_{2,2})/p$. This is omitted from the figure.

In summary, the first digit of each row is computed by subtracting the digits on the same diagonal and dividing by p. All other digits are obtained via a series of polynomial evaluations, starting from the first digit in its row. Finally, the result is obtained by subtracting the last digit of each row from the input and dividing by p.

$$\begin{array}{ccccccccc} w_{0,0} & \to & w_{0,1} & \to & w_{0,2} & \to & w_{0,3} & \to & w_{0,4} \\ & \swarrow & & \nearrow & & & & \nearrow & \\ w_{1,0} & \to & w_{1,1} & \to & w_{1,2} & \to & w_{1,3} & & \\ & \swarrow & & & & \nearrow & & & \\ w_{2,0} & \to & w_{2,1} & \to & w_{2,2} & & & & \end{array}$$

Figure 4: Visualization of digit removal for $e = 5$ and $v = 3$.

Until now, we have only specified operations *between rows*, which are identical for all methods that we will discuss (including our own). Existing digit removal procedures differ in how they compute digits *within the same row*. Two different methods have been proposed for this: one from Alperin-Sheriff/Peikert [1] which was generalized by Halevi/Shoup [15], and a second one from Chen/Han [8].

Alperin-Sheriff/Peikert/Halevi/Shoup Digit Removal. This procedure computes each number in the trapezoid (except for the first one in each row) by applying the lifting polynomial to the number on its left. In other words, we use the identity $w_{i,j+1} = F_e(w_{i,j})$. The cost is dominated by $ev - v(v+1)/2$ evaluations of the lifting polynomial. The degree of this procedure is roughly p^{e-1}.

Chen/Han Digit Removal. This procedure computes the last number of each row by applying the digit extraction polynomial to the first number of the same row. In other words, we use the identity $w_{i,j} = G_{j+1}(w_{i,0})$. All other digits are computed identical to the Alperin-Sheriff/Peikert/Halevi/Shoup procedure; but note that some digits are not used and can therefore be omitted. In the example of Figure 4, we do not need to compute $w_{0,3}$, $w_{1,2}$ and $w_{2,1}$.

The cost of Chen/Han digit removal is dominated by $v(v-1)/2$ evaluations of the lifting polynomial and v evaluations of the digit extraction polynomial. However, its main advantage is in degree, which is roughly rp^v with $r = e - v$. During bootstrapping, the parameter r represents the precision of the plaintext space, i.e. plaintexts are computed modulo p^r. As such, the Chen/Han procedures has asymptotically lower degree than Alperin-Sheriff/Peikert/Halevi/Shoup for high-precision plaintext spaces.

4.3 Faster Digit Removal

In the following sections, we apply five changes to the original Chen/Han digit removal procedure. The first adaptation relates to digit removal itself, and is a better method to evaluate the polynomials of each row. The other improvements follow from polyfunctions theory.

Adapted Row Computation. As already mentioned earlier, digit removal can be analyzed row per row, where Alperin-Sheriff/Peikert and Halevi/Shoup take a different approach than Chen/Han. In contrast to both these versions, we propose a method that uses the digit extraction polynomial exclusively, without relying on the lifting polynomial. Specifically, we compute each element of the trapezoid by applying a suitable digit extraction polynomial to the first element in the same row. This has two advantages: firstly, all polynomials can be evaluated *simultaneously* using the baby-step/giant-step technique. Due to the $2\sqrt{mn}$ complexity, this is beneficial over evaluating all polynomials separately. Secondly, this method also works well in conjunction with our next optimization (finding a more efficient representation of the digit extraction polynomial).

In instantiating our method, we need to avoid depth increase of the resulting procedure as much as possible. In particular, we have to be careful with the path along the evaluated circuit of largest depth, referred to as the *critical path*. Any depth increase in the critical path causes a corresponding depth increase in the entire procedure. The critical path of Chen/Han digit removal is depicted in Figure 5. It runs via the vertical dimension first, because the depth grows linearly there and logarithmically in the horizontal dimension.

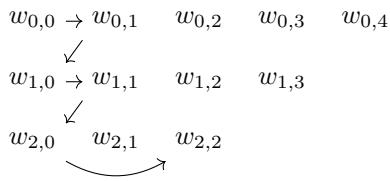


Figure 5: Critical path of digit removal for $e = 5$ and $v = 3$.

In a first attempt, we can compute each digit as $w_{i,j} \leftarrow G_{j+1}(w_{i,0})$. In some cases, however, we can do better by reusing computed elements. In particular, we can set $w_{i,k} \leftarrow w_{i,j}$ for $k < j$ without affecting correctness; however, also this is not always desirable because it can lead to a depth increase of the digit removal procedure. For example, it is never beneficial to take $w_{i,1} \leftarrow w_{i,j}$ in terms of noise growth, because $w_{i,1}$ lies on the critical path. Our implementation takes the heuristic approach of computing $w_{i,j} \leftarrow G_{j+1}(w_{i,0})$ for each value of

$j+1$ that is a power of 2, and setting $w_{i,j} \leftarrow w_{i,j+1}$ otherwise. This heuristic does not increase the multiplicative depth compared to Chen/Han digit removal.

Even and Odd Functions. The digit extraction function for $p = 2$ is an even function. Following Lemma 3.5, we can find a representation of degree $e+1$ or less that contains only even exponents. Specifically, we write the digit extraction polynomial as $G_e(X) = F(X^2)$ for some polynomial $F(X)$ of degree $\lfloor (e+1)/2 \rfloor$. Such polynomials can be evaluated more efficiently than regular ones by first computing X^2 before applying a standard baby-step/giant-step method. This requires asymptotically $\sqrt{2mn}$ non-scalar multiplications for evaluating m polynomials of degree n.

Similarly to the case above, the digit extraction function for an odd prime p is an odd function. Following Lemma 3.5, we can find a representation of degree $(e-1)(p-1)+1$ that contains only odd exponents. Specifically, we write the digit extraction polynomial as $G_e(X) = X \cdot F(X^2)$ for some polynomial $F(X)$ of degree $(e-1)(p-1)/2$. Such polynomials can be evaluated more efficiently than regular ones, using one the methods of Lee et al. [18]. Their first method omits unused powers of X in the baby-step, and can be evaluated with optimal multiplicative depth. Their second method first evaluates $F(X^2)$ using the strategy from above, and then multiplies by X. This increases the depth by at most one. Both methods require asymptotically $\sqrt{2mn}$ non-scalar multiplications for evaluating m polynomials of degree n. All experiments in Section 5 are conducted with the first variant.

Function Composition. We propose a new method to obtain the digit extraction function modulo p^e by decomposing it as $g_e = g_{e,e'} \circ g_{e'}$ for some $e' < e$. In our method, the relevant domain of $g_{e,e'}$ is therefore no longer $\mathbb{Z}$, but rather the range of $g_{e'}$. Our analysis starts from the following definitions.

Definition 4.1. Let $f \in \mathcal{F}_{p^e}$ be a function from $\mathbb{Z}_{p^e}$ to itself. If there exists a polynomial $F(X) \in \mathbb{Z}[X]$ that satisfies $F(a) = f(a) \pmod{p^e}$ for all $a \in S \subseteq \mathbb{Z}$, then f is a *polyfunction modulo* p^e *over* S and $F(X)$ is a *representation* of f.

Definition 4.2. An element $O(X) \in \mathbb{Z}[X]$ is called a *null polynomial modulo* p^e *over* $S \subseteq \mathbb{Z}$ if the function $f \in \mathcal{F}_{p^e}$ that it represents maps every element from S to zero. In other words, we have that $O(a) = 0 \pmod{p^e}$ for all $a \in S$.

The inner function $g_{e'}$ can directly be represented as a polynomial in the even or odd representation. For the outer function $g_{e,e'}$, we can use the adapted definitions from above, where we define the set

$$S = \left\{ k + i \cdot p^{e'} : -(p-1)/2 \leqslant k \leqslant (p-1)/2 \text{ and } i \in \mathbb{Z} \right\}. \tag{15}$$

This coincides with the range of $g_{e'}$. For $p = 2$, we slightly need to change the definition of S and allow $0 \leqslant k \leqslant 1$.

Since digit extraction is an idempotent operation, one possible representation of $g_{e,e'}$ is $G_e(X)$. But the domain of $g_{e,e'}$ is restricted to S, so we can find other representations by adding null polynomials that satisfy Definition 4.2. Therefore, our problem reduces to studying null polynomials over S, which we can construct as follows. Consider

$$H_j(X) = \begin{cases} (X)_j & \text{if } p = 2 \\ \left(X + \frac{p-1}{2}\right)_j & \text{if } p \text{ is an odd prime.} \end{cases} \tag{16}$$

To ease notation, we also write $H(X) = H_p(X)$. Moreover, let

$$(X)_{i,j} = \left(\prod_{k=0}^{i-1} H(X - k \cdot p^{e'}) \right) H_j(X - i \cdot p^{e'}) \tag{17}$$

for $0 \leqslant j < p$. Then we can adapt Theorem 3.1 as follows.

Theorem 4.1. *A polynomial $O(X) \in \mathbb{Z}[X]$ is a null polynomial modulo p^e over the set S from Equation* (15) *of degree at most n if and only if there exist $a_{i,j} \in \mathbb{Z}$ such that*

$$O(X) = \sum_{0 \leqslant d(i,j) \leqslant n} a_{i,j} \cdot O_{i,j}(X), \quad \textit{with} \quad O_{i,j}(X) = p^{\max(e - i \cdot e' - \nu_p(i!), 0)} \cdot (X)_{i,j}\,.$$

In this equation, the function $d(i, j) = p \cdot i + j$ denotes the degree of $O_{i,j}(X)$. It is also implicitly assumed that $0 \leqslant j < p$.

Proof. ($\Leftarrow$) Evaluating Equation (17) at any $a \in S$ gives

$$(X)_{i,j}\,|_{X=a} = \left(\prod_{k=0}^{i-1} H(a - k \cdot p^{e'}) \right) H_j(a - i \cdot p^{e'}). \tag{18}$$

From the definition of $H(X)$ in Equation (16) and the restriction to S, it follows that $H(a)$ is divisible by $p^{e'}$. In fact, exactly one factor of $H(X)$ will be divisible by $p^{e'}$ when evaluated at $X = a$. Let $X - q$ be this linear factor where $0 \leqslant q \leqslant 1$ (if $p = 2$) or $-(p-1)/2 \leqslant q \leqslant (p-1)/2$ (if p is odd), then we can set $a - q = \alpha \cdot p^{e'}$ for some α. Equation (18) is then divisible by

$$\prod_{k=0}^{i-1} \left(a - q - k \cdot p^{e'}\right) = \prod_{k=0}^{i-1} \left(\alpha \cdot p^{e'} - k \cdot p^{e'}\right) = p^{i \cdot e'} \cdot (X)_i\,|_{X=\alpha}.$$

As already pointed out in Section 2.3, the evaluation of $(X)_i$ at any integer is divisible by $p^{\nu_p(i!)}$. Hence our result is divisible by $p^{i \cdot e' + \nu_p(i!)}$, and it follows that

$O_{i,j}(X)|_{X=a}$ is divisible by $p^{\max(e,i\cdot e'+\nu_p(i!))} \geqslant p^e$. Any $\mathbb{Z}$-linear combination of these $O_{i,j}(X)$ is thus a null polynomial modulo p^e over S.

($\Rightarrow$) We prove the following assertion for $0 \leqslant m \leqslant n+1$ by applying induction on m:

$$O(X) = \sum_{m \leqslant d(i,j) \leqslant n} b_{i,j} \cdot (X)_{i,j} + \sum_{0 \leqslant d(i,j) < m} a_{i,j} \cdot O_{i,j}(X), \tag{19}$$

for some $a_{i,j}, b_{i,j} \in \mathbb{Z}$.

The base case $m = 0$ is trivial since the second sum is empty, and the first sum amounts to writing a polynomial in the basis given by $(X)_{i,j}$. It is therefore possible to find appropriate constants $b_{i,j}$ that satisfy Equation (19).

Now suppose that Equation (19) was established for some $m < n+1$, that is

$$O(X) = b_{i',j'} \cdot (X)_{i',j'} + \sum_{m < d(i,j) \leqslant n} b_{i,j} \cdot (X)_{i,j} + \sum_{0 \leqslant d(i,j) < m} a_{i,j} \cdot O_{i,j}(X),$$

with $d(i',j') = m$. Evaluating both sides at $X = a$ with $a = i' \cdot p^{e'} + j'$ (if $p = 2$) or $a = i' \cdot p^{e'} + j' - (p-1)/2$ (if p is odd) gives

$$0 = O(a) = b_{i',j'} \cdot \prod_{k=0}^{i'-1} (X)_p \,|_{X=(i'-k)\cdot p^{e'}+j'} \cdot (j')! \pmod{p^e}.$$

Taking the p-adic valuation of the right-hand side and following a similar line of reasoning as in the first part of this proof, we get

$$\nu_p\left(b_{i',j'} \cdot \prod_{k=0}^{i'-1} (X)_p \,|_{X=(i'-k)\cdot p^{e'}+j'} \cdot (j')! \right) = \nu_p(b_{i',j'}) + i' \cdot e' + \nu_p((i')!) \geqslant e.$$

The constants $b_{i',j'}$ are integers, so $\nu_p(b_{i',j'}) \geqslant \max(e - i' \cdot e' - \nu_p((i')!), 0)$. We can therefore write $b_{i',j'} = a_{i',j'} \cdot p^{\max(e-i'\cdot e'-\nu_p((i')!),0)}$ for some $a_{i',j'} \in \mathbb{Z}$, which results in

$$O(X) = \sum_{m < d(i,j) \leqslant n} b_{i,j} \cdot (X)_{i,j} + \sum_{0 \leqslant d(i,j) \leqslant m} a_{i,j} \cdot O_{i,j}(X).$$

This expression replaces m by $m+1$ in Equation (19) and thereby completes the induction. The final result follows by setting $m = n+1$ in Equation (19). □

To study the degree of null polynomials restricted to the set S, we consider an adapted variant of the Smarandache function that takes two inputs:

$$\mu_p(e, e') = \min\{i \in \mathbb{N} : e' \cdot i + \nu_p(i!) \geqslant e\}.$$

Then it is clear that

$$O_{\mu_p(e,e'),0}(X) = \prod_{k=0}^{\mu_p(e,e')-1} H(X - k \cdot p^{e'}) \tag{20}$$

is a monic null polynomial of degree $p \cdot \mu_p(e, e') \approx p \cdot \lceil e/e' \rceil$.

Now we have all ingredients available to find a better representation of $g_{e,e'}$. Starting from $G_e(X)$, we can apply Euclidean division and reduce it modulo the null polynomial of Equation (20). This results in a representation $G_{e,e'}(X)$ that has degree strictly less than $p \cdot \lceil e/e' \rceil$. This can be much smaller than the degree of $G_e(X)$, which is equal to $(e-1)(p-1)+1$.

For odd primes p, the function $g_{e,e'}$ is odd and we can directly choose $G_{e,e'}(X)$ with only odd-exponent terms. However, if $p = 2$ then $-S \neq S$, hence $g_{e,e'}$ is not defined for all inputs from $-S$. The function is therefore not even, and we cannot directly choose $G_{e,e'}(X)$ with only even-exponent terms. One possible solution is to allow $-1 \leqslant k \leqslant 1$ in Equation (15) instead of $0 \leqslant k \leqslant 1$. However, this increases the degree of the polynomial from Equation (20) by 50%, hence also the degree of the resulting polynomial representation. We did not incorporate this solution in our implementation.

Finally, we note that the set of null polynomials modulo p^e over S of degree bound n still forms a p^e-ary lattice. This lattice is given by

$$\left\{ \sum_{0 \leqslant d(i,j) \leqslant n} a_{i,j} \cdot O_{i,j}(X) \mid a_{i,j} \in \mathbb{Z} \right\} \subseteq \mathbb{P}_n,$$

where the basis vectors are $O_{i,j}(X)$.

Asymptotic Complexities. We now analyze the asymptotic depth and time complexities of our composite approach. Specifically, its depth is bounded by

$$\lceil \log_2 ((p-1) \cdot (e'-1) + 1) \rceil + \left\lceil \log_2 \left(p \cdot \left\lceil \frac{e}{e'} \right\rceil \right) \right\rceil \approx \lceil \log_2 e \rceil + 2 \cdot \lceil \log_2 p \rceil,$$

counting only non-scalar multiplications. The first term comes from $g_{e'}$ and the second one from $g_{e,e'}$. When compared to the regular $G_e(X)$, there is a depth increase of $\lceil \log_2 p \rceil$. Note that we can also apply function composition multiple times in a row, and then the depth will increase with $\lceil \log_2 p \rceil$ per stage. In terms of scalar multiplications, there is a depth increase of 1 for each stage of function composition. For the sake of noise control, our approach favors a small number of stages and a low value of p.

Performance-wise, we make a difference between scalar and non-scalar multiplications. The baby-step/giant-step technique can asymptotically evaluate

a polynomial of degree n with $2\sqrt{n}$ non-scalar and n scalar multiplications. If the polynomial is even or odd, these numbers reduce to respectively $\sqrt{2n}$ and $n/2$. As such, we have the following time complexities for the digit extraction function:

- For $p = 2$, the original method can evaluate the digit extraction polynomial asymptotically with $\sqrt{2e}$ non-scalar and $e/2$ scalar multiplications. Our composite approach reduces this to respectively $\sqrt{2e'} + 2\sqrt{2e/e'}$ and $e'/2 + 2e/e'$. The number of non-scalar multiplications is minimal if $e' = 2\sqrt{e}$, which gives $4\sqrt[4]{e}$ non-scalar and $2\sqrt{e}$ scalar multiplications. Since $p = 2$, this analysis assumes that $G_{e,e'}(X)$ has both even- and odd-exponent terms.
- For larger values of p, the original method can evaluate the digit extraction polynomial asymptotically with $\sqrt{2pe}$ non-scalar and $pe/2$ scalar multiplications. Our composite approach reduces this to respectively $\sqrt{2p}(\sqrt{e'} + \sqrt{e/e'})$ and $(p/2) \cdot (e' + e/e')$. The number of non-scalar multiplications is minimal if $e' = \sqrt{e}$, which gives $2\sqrt{2p}\sqrt[4]{e}$ non-scalar and $p\sqrt{e}$ scalar multiplications. Since $p \neq 2$, this analysis assumes that $G_{e,e'}(X)$ has only odd-exponent terms. Moreover, the degree of $G_e(X)$ is approximated as pe.

In conclusion, our method reduces the number of non-scalar multiplications from $\mathcal{O}(\sqrt{pe})$ to $\mathcal{O}(\sqrt{p}\sqrt[4]{e})$ asymptotically. The number of scalar multiplications are reduced from $\mathcal{O}(pe)$ to $\mathcal{O}(p\sqrt{e})$.

Table 3 shows the number of operations to evaluate digit extraction, comparing the Alperin-Sheriff/Peikert/Halevi/Shoup and Chen/Han method to our approach. The even and odd entries represent the standard version (without function composition). The tuples represent the indices (e, e', e'') of the digit extraction function, where e'' is the index of the innermost function and e is the index of the outermost function. All tuples are chosen to minimize the number of non-scalar multiplications. It is clear from the table that our composite method works especially well for low p and high e, where the performance benefits can be fully exploited without a significant depth increase. On the other hand, it turns out that even for large values of e (up to 256), splitting in more than 2 stages does not (much) increase performance anymore.

Lattices. Another method to find better polynomial representations is via lattice problems. Given a polynomial $F(X)$ and the null lattice, we can solve the closest vector problem to find a null polynomial $O(X)$ that lies closest to $F(X)$. The representation $F(X) - O(X)$ is then equivalent to the original one, but it has smaller coefficients. This leads to less noise growth in FHE ciphertexts.

Table 3: Non-scalar depth and operation count for the digit extraction function.

p	e	method	depth	#(non-scalar mults)	#(scalar mults)
2	64	AS/P/H/S	63	63	0
		C/H	6	16	64
		Our even	6	12	32
		(64, 16)	7	9	15
	256	AS/P/H/S	255	255	0
		C/H	8	33	256
		Our even	8	25	128
		(256, 32)	9	15	31
		(256, 67, 16)	10	13	22
3	64	AS/P/H/S	126	126	0
		C/H	7	24	127
		Our odd	7	20	64
		(64, 16)	9	16	22
		(64, 25, 8)	10	15	24
	256	AS/P/H/S	510	510	0
		C/H	9	49	511
		Our odd	9	38	256
		(256, 24)	11	23	40
		(256, 92, 8)	12	21	58

This lattice trick can also be combined with all earlier described techniques. For even or odd functions, we can start from a lattice that only contains even or odd null polynomials. These can be found via simple linear algebra on the original null lattice. For the function composition approach, we can start from the null lattice restricted to the set S as defined earlier.

Example 4.1. *The advantage of using lattices is demonstrated on the bit extraction polynomial. Our method was able to find the following representations for $p = 2$:*

- *Bit extraction modulo 2^8 can be done with $G_8(X) = 13X^8 - 12X^6$.*
- *Starting from the result modulo 2^8, bit extraction modulo 2^{25} can be done with $G_{25,8}(X) = 6X^5 - 15X^4 + 10X^3$.*

Both polynomials have remarkably small coefficients, since they are defined modulo 2^8 and 2^{25} respectively.

Multivariate Approach. We considered one more strategy to compute better polynomial representations based on multivariate equations. The idea is to write out consecutive digit extraction polynomials in a pattern that minimizes the

non-scalar multiplications. This gives a system of non-linear equations, which can be solved for the coefficients of the digit extraction polynomial. Although this strategy does not generalize to higher parameters, we were able to find bit extraction polynomials for $e \leqslant 16$ that can be evaluated with $\lceil \log_2 e \rceil$ non-scalar multiplications, which is provably minimal. These instances are listed in Table 4.

Table 4: Recursive evaluation of the bit extraction polynomial.

e	$G_e(X)$
2	X^2
4	$G_2(X)^2$
8	$112 \cdot G_2(X) + (94 \cdot G_2(X) + 121 \cdot G_4(X))^2$
16	$11136 \cdot G_4(X) + (28504 \cdot G_2(X) + 8968 \cdot G_4(X) - G_8(X)) \cdot (15364 \cdot G_4(X) + 14115 \cdot G_8(X))$

5 Implementation and Performance

We implemented our new digit removal procedure for the BGV scheme in `HElib`. For two reasons, we did not implement it for the BFV scheme: firstly, there is no software library that supports BFV bootstrapping; secondly, BGV and BFV are known to be equivalent in terms of bootstrapping, and only differ in some minor implementation details. Therefore, any performance discrepancy would reflect the underlying arithmetic operations and not our improvements.

We give experimental results for general bootstrapping in Table 5 and for thin bootstrapping in Table 6. The tables show capacity (number of bits in the noise) and execution time. The factorization of the parameter m determines the complexity of the linear maps (explained in [15]), but is not relevant for digit removal. The regular plaintext modulus is p^r, which is augmented to p^e during bootstrapping. The function composition method was not used for these tables, since its effect is thoroughly analyzed in Section 5.1. All experiments were run on a single-threaded Intel® Core™ i7-6700HQ CPU with 8 GB memory and Ubuntu 22.04.1 LTS installed.

The "improvement" in the last row of Tables 5 and 6 was computed in the amortized sense, i.e. as the ratio

$$\text{improvement} = \frac{\text{old execution time}}{\text{new execution time}} \cdot \frac{\text{new remaining capacity}}{\text{old remaining capacity}}.$$

We achieve a significant improvement for all tested parameter sets, ranging from $1.3\times$ to $2.6\times$. The speedups are higher for general bootstrapping than for thin bootstrapping. The reason is that the general version requires multiple digit

removals, whereas the thin version requires only one. In terms of noise capacity, the advantage also tends to be in our direction. This is likely a consequence of two facts: we replaced the lifting polynomial by the digit extraction polynomial of lower degree; and the coefficients of our polynomials are smaller due to the lattice trick.

5.1 Function Composition

Our implementation also includes the function composition approach, which is asymptotically cheaper for high-precision plaintext spaces (i.e. large values of r and e). We demonstrate its benefit in Table 7 for plaintext moduli up to 2^{59}. For technical reasons (not inherent to the BGV scheme), `HElib` does not support more than 59 bits of precision, so we could not test with higher values than this. Furthermore, bootstrapping is only supported for $p^e < 2^{30}$ [15], so it is impossible to run bootstrapping with the parameter sets from Table 7. We therefore show the results for digit extraction only. Finally, note that the parameters e and e' have the same meaning as in Section 4.3.

Table 7 has four values per column: the original built-in implementation; our standard method without function composition; our method with partial function composition; and our method with full function composition. The third value of each column (partial function composition) is generated by applying function composition to each row of Figure 5, except for the last one. The motivation for this is as follows. Since the bottom right element of Figure 5 lies on the critical path, it determines the multiplicative depth of digit removal. Turning this argument around, we can reduce the depth by not applying function composition in the last row. In other words, there is a depth-efficiency trade-off, where function composition favors efficiency and the standard method favors depth.

The "speedup" in the last row of Table 7 was computed as the ratio between the original approach and our three methods:

$$\text{speedup} = \frac{\text{old execution time}}{\text{new execution time}}.$$

This cost measure ignores the remaining noise capacity, because we cannot run the full bootstrapping procedure and therefore don't have this number available. Again, we achieve major speedups compared to `HElib`'s built-in digit removal, ranging from 1.6× to 2.8×. Since digit removal is the main bottleneck, bootstrapping would exhibit almost identical speedups.

Table 5: General bootstrapping in `HElib` (original/ours).

Cyclotomic index m		$127 \cdot 337$	$101 \cdot 451$	$43 \cdot 757$
Number of slots		2016	1000	2268
Params (p, r, e)		$(2, 8, 15)$	$(17, 4, 6)$	$(127, 2, 4)$
Security level (bits)		81	78	66
Number of digit removals		21	40	14
Capacity (bits)	Initial	1151	1136	1134
	Linear maps	100	147	140
	Digit extract	307/298	541/514	671/712
	Remaining	744/753	448/475	323/282
Execution time (sec)	Linear maps	134	150	290
	Digit extract	2014/743	2665/1879	1407/863
	Total	2248/877	2815/2029	1697/1153
Improvement		2.6×	1.5×	1.3×

Table 6: Thin bootstrapping in `HElib` (original/ours).

Cyclotomic index m		$127 \cdot 337$	$101 \cdot 451$	$43 \cdot 757$
Number of slots		2016	1000	2268
Params (p, r, e)		$(2, 8, 15)$	$(17, 4, 6)$	$(127, 2, 4)$
Security level (bits)		81	78	66
Capacity (bits)	Initial	1151	1136	1134
	Linear maps	137	174	164
	Digit extract	267/260	445/435	604/640
	Remaining	747/754	517/527	366/330
Execution time (sec)	Linear maps	35	32	31
	Digit extract	105/35	65/46	101/64
	Total	140/70	97/78	132/95
Improvement		2.0×	1.3×	1.3×

Table 7: High-precision digit removal in `HElib` (original/our standard method/partial function composition/full function composition).

Cyclotomic index m		42799	63973
Number of slots		2016	2592
Params (p, r, e, e')		$(2, 51, 59, 16)$	$(3, 32, 37, 6)$
Security level (bits)		80	77
Capacity (bits)	Initial	1137	1335
	Digit extract	1049/991/970/1006	1142/1047/1103/1170
Execution time (sec)		180/100/67/64	191/151/124/119
Speedup		1.8×/2.7×/2.8×	1.3×/1.5×/1.6×

6 Conclusion

Although polynomial functions over rings are commonly used in cryptography, their properties are currently not well understood. This paper contributed to the analysis of such polyfunctions, including existence, computation and equivalence of polynomial representations, among other things.

Our theory is directly applicable to FHE bootstrapping: we found sparse representations (either even or odd) for the digit extraction function, which is the bottleneck in bootstrapping; we also proposed a new method to decompose digit extraction into multiple stages, each of which can be evaluated with a polynomial of low degree. Altogether, we observed speedups of up to 2.6× for bootstrapping and up to 2.8× for digit removal, including our function composition approach. Finding the optimal way to evaluate the polynomials during bootstrapping, taking into account both noise growth and execution time, remains an interesting open problem.

Acknowledgments

This work is supported in part by the European Commission through the Horizon 2020 research and innovation program Belfort ERC Advanced Grant 101020005 and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-21-C-0034. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This work was additionally supported in part by CyberSecurity Research Flanders with reference number VR20192203, and in part by the Research Council KU Leuven grant C14/18/067. Robin Geelen is funded in part by Research Foundation – Flanders (FWO) under a PhD Fellowship fundamental research (project number 1162123N).

References

[1] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In *Annual Cryptology Conference*, pages 1–20. Springer, 2013.

[2] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485, 1967.

[3] Toshinori Araki, Assi Barak, Jun Furukawa, Marcel Keller, Yehuda Lindell, Kazuma Ohara, and Hikaru Tsuchida. Generalizing the spdz compiler for other protocols. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2018.

[4] Manjul Bhargava. P-orderings and polynomial functions on arbitrary subsets of dedekind rings. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1997(490-491):101–128, 1997.

[5] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325. Association for Computing Machinery, January 2012.

[7] L. Carlitz. Functions and polynomials (mod p^n). *Acta Arithmetica*, 9(1):67–78, 1964.

[8] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 315–337. Springer, Heidelberg, April / May 2018.

[9] Zhibo Chen. On polynomial functions from $\mathbb{Z}_n$ to $\mathbb{Z}_m$. *Discrete Mathematics*, 137(1-3):137–145, 1995.

[10] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.

[11] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

[12] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 129–148. Springer, 2011.

[13] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012*, pages 1–16, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[14] Ashwin Guha and Ambedkar Dukkipati. An algorithmic characterization of polynomial functions over $\mathbb{Z}_{p^n}$. *Algorithmica*, 71(1):201–218, 2015.

[15] Shai Halevi and Victor Shoup. Bootstrapping for helib. *Journal of Cryptology*, 34(1):1–44, 2021.

[16] Gordon Keller and Frank R Olson. Counting polynomial functions (mod p^n). *Duke Mathematical Journal*, 35(4):835–838, 1968.

[17] Aubrey J Kempner. Polynomials and their residue systems. *Transactions of the American Mathematical Society*, 22(2):240–288, 1921.

[18] Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. Optimal minimax polynomial approximation of modular reduction for bootstrapping of approximate homomorphic encryption. Cryptology ePrint Archive, Paper 2020/552, 2020. `https://eprint.iacr.org/archive/2020/552/20200803:084202`.

[19] Shujun Li. Null polynomials modulo m. *arXiv preprint math/0510217*, 2005.

[20] Michael S Paterson and Larry J Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, 1973.

[21] David Singmaster. On polynomial functions (mod m). *Journal of Number Theory*, 6(5):345–352, 1974.

[22] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

[23] Ernst Specker, Norbert Hungerbühler, and Micha Wasem. The ring of polyfunctions over $\mathbb{Z}/n\mathbb{Z}$, 2021.

Chapter 7

Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification

Publication Data

Cong, K., Geelen, R., Kang, J., and Park, J. Revisiting oblivious top-k selection with applications to secure k-nn classification. In *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part I* (2024), M. Eichlseder and S. Gambs, Eds., vol. 15516 of *Lecture Notes in Computer Science*, Springer, pp. 3–25

Contribution

I developed the Top-k selection network and the k-NN evaluation algorithm in collaboration with other co-authors.

Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification

Kelong Cong[1*], Robin Geelen[2], Jiayi Kang[2], and Jeongeun Park[3*]

[1] Zama, Paris, France
[2] COSIC, ESAT, KU Leuven, Belgium
[3] Norwegian University of Science and Technology (NTNU)

Abstract. An oblivious Top-k algorithm selects the k smallest elements from d elements while ensuring the sequence of operations and memory accesses do not depend on the input. In 1969, Alekseev proposed an oblivious Top-k algorithm with complexity $\mathcal{O}(d \log^2 k)$, which was later improved by Yao in 1980 for small $k \ll \sqrt{d}$.

In this paper, we revisit the literature on oblivious Top-k and propose another improvement of Alekseev's method that outperforms both for large $k = \Omega(\sqrt{d})$. Our construction is equivalent to applying a new truncation technique to Batcher's odd-even sorting algorithm. In addition, we propose a combined network to take advantage of both Yao's and our technique that achieves the best concrete performance, in terms of the number of comparators, for any k. To demonstrate the efficiency of our combined Top-k network, we implement a secure non-interactive k-nearest neighbors classifier using homomorphic encryption as an application. Compared with the work of Zuber and Sirdey (PoPETS 2021) where oblivious Top-k was realized with complexity $\mathcal{O}(d^2)$, our experimental results show a speedup of up to 47 times (not accounting for difference in CPU) for $d = 1000$.

1 Introduction

Outsourcing computation has been a popular solution to resolve modern conflicts between large data collection versus limited local storage and computational power. Stimulated by regulations such as the General Data Protection Regulation (GDPR), data confidentiality received growing attention in outsourced computation. Fully Homomorphic Encryption (FHE) is a powerful cryptographic technique that allows arbitrary computations over encrypted

*Work done while the author was at COSIC, ESAT, KU Leuven.

data without decrypting intermediate values. This interesting property enables secure computations that are *non-interactive* and propels FHE into a key privacy preserving technology [11, 14, 15, 16, 27, 33]. With promising speedup from hardware accelerators [4, 5, 18, 29], which can be up to three orders of magnitude faster than CPUs, FHE can soon provide feasible solutions for a wider range of real-world, privacy preserving applications.

Despite its promising potential, developing efficient FHE programs remains difficult. An important part of the inefficiency is the amplification in computation complexity when a plaintext program is converted into a program operating on the corresponding FHE ciphertexts. For example, in the homomorphic evaluation of the if-else paradigm, each conditional statement needs to be executed. By extension, when traversing a binary tree, the full tree is touched instead of a single path. In other words, the data secrecy guaranteed by FHE comes at the cost of increased computational complexity, even if the overhead of FHE is low.

Data-oblivious algorithms in FHE. Fortunately, the increase in computational complexity does not apply to *data-oblivious* programs where the sequence of operations and memory accesses do not depend on the input. Therefore, data-oblivious programs can be directly translated to their low-level FHE analogue. In this sense, describing a high-level algorithm in a data-oblivious manner is an FHE-friendly paradigm.

As an example, suppose we want to sort d encrypted elements homomorphically. Then which sorting algorithm should we use? Quicksort and heapsort turn out to be not data-oblivious, and realizing those homomorphically is impractical despite their optimal time complexity of $\mathcal{O}(d \log d)$. In contrast, Batcher's odd-even merge sort [22] is a data-oblivious algorithm with time complexity $\mathcal{O}(d \log^2 d)$, and it can be realized homomorphically with the same complexity.

Additionally, the most appropriate cost measure depends on the FHE scheme. In BGV [7], BFV [6, 17] and CKKS [12], controlling the multiplicative depth (i.e., the number of consecutive multiplications) of the sorting algorithm is crucial. On the other hand, optimizing the algorithmic complexity is more important in the TFHE [13] scheme.

Oblivious Top-k selection. Given d elements, a Top-k algorithm selects its k smallest (or largest) elements, while the output elements are not necessarily sorted. Top-k selection is an important building block for various applications, in which the k most important records in a huge information space (consisting of d records) are extracted by defining proper scoring functions and returning those with the best ranks. Widely used examples include the k-nearest

neighbors classification technique [23], recommender systems [20] and genetic algorithms [26].

This work focuses on Top-k algorithms that are data-oblivious. Since oblivious programs can be visualized as networks of low-level modules, the terms *Top-k network* and *oblivious Top-k* are used interchangeably. Section 2 introduces the basics of the network visualization.

Research into oblivious Top-k has a long history. In this work, we revisit the complexity upper bounds for the oblivious Top-k algorithm derived by Alekseev in 1969 [2], which was then improved by Yao in 1980 [36]. Alekseev proposed a procedure to select the Top-k out of $2k$ elements, which can be generalized into Top-k out of d elements with complexity $\mathcal{O}(d \log^2 k)$ [22]. This method provides better asymptotic complexity than recent FHE realizations and also outperforms those in practice. Yao, on the other hand, introduced an unbalanced recursive procedure in [36, Lemma 3.2] to improve Alekseev's for small $k \ll \sqrt{d}$.

These results, however, have not received much attention so far. Recent realizations of oblivious Top-k in secure computation either use an oblivious sorting algorithm [21, 34, 37] or call oblivious Min (or Max) k times repeatedly [8, 9, 19]. The complexity in the former method is not parameterized by k since it always contains redundant comparisons, and the latter method results in complexity $\mathcal{O}(kd)$, which grows linearly in k.

1.1 Our contributions

Firstly, we revisit Yao's recursive approach for oblivious Top-k selection. We observe that not all parameters d and k can be reduced to the recursive base case ($k = 1$ or $k = 2$) in [36, Theorem 3.1]. Therefore, we fix this by handling one more case using symmetry. This construction results in a Top-k network with complexity $\mathcal{O}(d \log k)$ for small $k \ll \sqrt{d}$.

Secondly, we also propose another improvement of Alekseev's method independent of Yao's. Specifically, we improve Alekseev's order-preserving merging procedure from $\mathcal{O}(k \log^2 k)$ to $\mathcal{O}(k \log k)$ by truncating Batcher's merge. As such, our network is essentially a truncated version of Batcher's odd-even sorting network with complexity $\mathcal{O}(d \log^2 k)$, where redundant comparisons are removed.

Since our truncated method is better for large $k = \Omega(\sqrt{d})$ and Yao's method is better for small k, our third contribution is to introduce a combined method which takes advantage of both. For concrete values of k and d, the combined network recursively calls our truncated merge method or Yao's method, depending on which one uses fewer comparators. As such, the complexity of our combined network is upper bounded by the better method of Yao's and our

truncated Top-k, and slightly improves on those methods for some parameters. If k and d are known in advance, the combined network can be preprocessed by the server.

Lastly, to demonstrate the efficiency of our combined Top-k network, we present non-interactive and secure k-Nearest Neighbors (k-NN) classification as an application. We use the TFHE homomorphic encryption scheme to handle large multiplicative depth efficiently, and our protocol is implemented in the TFHE-rs[1] library. Compared with prior work [37], where oblivious Top-k was realized with complexity $\mathcal{O}(d^2)$, our experimental results show a speedup of up to 47 times (not accounting for difference in CPU) for $d = 1000$ and $k = 3$.

1.2 Related work

Oblivious Top-k. To simplify the explanation, we denote an oblivious Top-k out of d procedure as a (k, d)-selector. In 1969, Alekseev [2] proposed a $(k, 2k)$-selector using sorting as a subprocedure. This method was generalized to arbitrary d [22], which achieves a complexity of $(\lceil d/k \rceil - 1)(2S(k) + k)$ comparators for oblivious Top-k selection. We use $S(k)$ to denote the number of comparators in k-sorting (i.e., sorting k elements).

The above complexity was improved by Yao in 1980. Specifically, in the proof of [36, Theorem 3.1], an unbalanced recursive procedure was introduced, which yields better networks for small k. This recursive procedure will be discussed in detail in Section 3.3.

Surprisingly, the constructions above have been ignored in research over the past few decades. To our knowledge, recent data-oblivious Top-k solutions can be categorised into three types:

- The first type applies a sorting algorithm directly, and then discards the $d - k$ irrelevant elements. For example, in a recent homomorphic k-NN realization, Zuber and Sirdey [37] use sorting of complexity $\mathcal{O}(d^2)$ to achieve Top-k.
- The second type repeatedly finds the Min (or Max) using the so-called tournament method [19], where inputs are compared pairwise and the "winner" proceeds to the next stage. This requires $\mathcal{O}(kd)$ comparisons in total.
- The third type is from hardware-related research [31]. This work builds a bitonic Top-k algorithm by removing the unnecessary comparisons in bitonic sorting, thereby achieving $\mathcal{O}(d \log^2 k)$. The number of comparators in bitonic sorting is always higher than Batcher's odd-even merge sort,

[1] `https://github.com/zama-ai/tfhe-rs`

but it is useful in hardware designs due to a more cache-friendly memory access pattern.

Secure k-NN classification. Chen et al. [9] proposed two secure k-NN classifiers based on a mixture of homomorphic encryption and secure multi-party computation. However, both versions use an approximate circuit for Top-k selection and therefore do not necessarily return the nearest neighbors. Additionally, their protocols are interactive, which makes them less suitable for outsourced computation in the context of cloud computing.

The most closely related work is that of Zuber and Sirdey [37], who also propose a non-interactive k-NN algorithm based on the TFHE scheme. The authors use a specialized, FHE-friendly approach to perform the Top-k selection step, known as the delta-matrix method. This technique is asymptotically worse than standard sorting algorithms. Our approach differs in this key step where we identify Top-k selection networks that involve fewer comparison operators. The delta-matrix method can be extended with a counting operation (called majority vote in [3]) to sum the results of the individual classes. However, this step is not necessary in sorting-based approaches, because they allow us to select the Top-k smallest elements directly.

Other related works are either very slow or rely on totally different security models. For example, Shaul et al. [32] implement a secure k-NN algorithm based on BGV homomorphic encryption [7] but they report an execution time of several hours for a moderately sized dataset. A completely different approach is taken by the SCONEDB model via a scalar-product-preserving encryption scheme [35]. However, this protocol computes the query result in the clear, which leaks useful information to the server. Another very recent paper proposes a lightweight k-NN solution, but it needs to distribute trust between two non-colluding servers [30].

2 Preliminaries on data-oblivious algorithms

This section introduces the network visualization of oblivious algorithms. We also give an example of Batcher's (d_1, d_2)-merging algorithm [22, Chapter 5].

A network comprises of interconnected *basic modules*. In the case of sorting and selection networks, this basic module is a comparator. Figure 1 shows the network of odd-even merge sort for $d = 4$, where inputs enter from the left and a vertical line compares two elements. The comparator swaps the inputs if the first one is greater than the second. By counting the number of vertical lines and the number of vertical lines in series, we know that the algorithm

Figure 1: The basic module and the network of odd-even merge sort for $d = 4$.

has 5 comparators and a depth of 3. Here the depth refers to the number of consecutive comparisons on the longest path from input to output.

2.1 Batcher's merging network

A crucial component of many sorting networks (and our Top-k selection network) is a *merge* procedure. Given two sorted arrays of size d_1 and d_2, then a (d_1, d_2)-merging algorithm produces a sorted array that contains the same elements.

Batcher's merging network is specified in Algorithm 1 (vector indexing is done in subscript). This algorithm is based on a recursive decomposition of the problem: first, the input arrays are split into their even- and odd-index components. Then the even- and odd-index arrays are merged separately via two recursive calls. One can prove that, after these recursive calls, the smallest element is in array $\mathbf{v}$, the second and third smallest elements are either in array $\mathbf{v}$ or $\mathbf{w}$ (yet not in the same one), and so on. As such, the result can be reconstructed by pairwise comparison of the elements of the recursive calls.

Theorem 2.1. *The $(2^i, 2^i)$-merge contains $2^i \cdot i + 1$ comparators and has a comparison depth of $i + 1$.*

3 Top-k selection networks

3.1 Revisiting Alekseev's Top-k network

Alekseev [2] proposed a merge procedure to construct a $(k, 2k)$-selector: partition and sort two length-k arrays to obtain $\mathbf{x}$ and $\mathbf{y}$, then compare and interchange

$$\mathbf{x}_0 \text{ with } \mathbf{y}_{k-1},\ \mathbf{x}_1 \text{ with } \mathbf{y}_{k-2},\ \ldots,\ \mathbf{x}_{k-1} \text{ with } \mathbf{y}_0. \tag{1}$$

This can be generalized into Top-k out of d elements by partitioning the inputs into $\lceil d/k \rceil$ length-k arrays and applying Alekseev's procedure (two k-sortings and k comparisons) $\lceil d/k \rceil - 1$ times as in the tournament procedure [22]. It solves the Top-k problem using $(\lceil d/k \rceil - 1)(2S(k) + k)$ comparators, where $S(k)$ is the number of comparators for k-sorting. Realizing $S(k)$ with practical sorting networks (e.g., Batcher's odd-even merge sort) leads to an asymptotic

Algorithm 1 Batcher's (d_1, d_2)-merge

Input: Two sorted arrays $\mathbf{x}$ (of size d_1) and $\mathbf{y}$ (of size d_2)
Output: Sorted array that contains the entries of $\mathbf{x}$ and $\mathbf{y}$

function MERGE($\mathbf{x}, \mathbf{y}$)
  if $d_1 \cdot d_2 = 0$ **then**
    return $(\mathbf{x}, \mathbf{y})$
  else if $d_1 \cdot d_2 = 1$ **then**
    return COMPARE($\mathbf{x}_0, \mathbf{y}_0$)
  else ▷ Merge even- and odd-index components
    $\mathbf{v} \leftarrow$ MERGE($(\mathbf{x}_0, \mathbf{x}_2, \ldots, \mathbf{x}_{2\lceil d_1/2 \rceil - 2}), (\mathbf{y}_0, \mathbf{y}_2, \ldots, \mathbf{y}_{2\lceil d_2/2 \rceil - 2})$)
    $\mathbf{w} \leftarrow$ MERGE($(\mathbf{x}_1, \mathbf{x}_3, \ldots, \mathbf{x}_{2\lfloor d_1/2 \rfloor - 1}), (\mathbf{y}_1, \mathbf{y}_3, \ldots, \mathbf{y}_{2\lfloor d_2/2 \rfloor - 1})$)
    $\mathbf{z} \leftarrow (\mathbf{v}_0, \mathbf{w}_0, \mathbf{v}_1, \mathbf{w}_1, \ldots)$
    for $i \leftarrow 1$ to $\lfloor (\text{size}(\mathbf{z}) - 1)/2 \rfloor$ **do**
      $(\mathbf{z}_{2i-1}, \mathbf{z}_{2i}) \leftarrow$ COMPARE($\mathbf{z}_{2i-1}, \mathbf{z}_{2i}$)
    end for
    return z
  end if
end function
function COMPARE(x, y) ▷ Comparator module
  return $(\min(x, y), \max(x, y))$
end function

complexity of $S(k) = \mathcal{O}(k \log^2 k)$, so this Top-$k$ network consists of $\mathcal{O}(d \log^2 k)$ comparators.

Reinterpretation as order-preserving merging. Let (d_1, d_2, k)-merge denote an order-preserving merge where the inputs are two sorted arrays of length d_1 and d_2, and the output is the sorted Top-k out of $d_1 + d_2$ elements. Then Alekseev's Top-k procedure can be reinterpreted into three steps:

1. sort $\lceil d/k \rceil$ length-k arrays;
2. apply $\lceil d/k \rceil - 2$ times (k, k, k)-merge in the tournament manner, where each merge consists of the procedure from (1) and a k-sorting of the output;
3. apply the procedure from (1).

In step 2, each (k, k, k)-merge has complexity $S(k) + k$. Using practical sorting networks, such as Batcher's odd-even sorting network, leads to complexity $\mathcal{O}(k \log^2 k)$ for (k, k, k)-merge.

3.2 Our truncated sorting network

Our order-preserving merging. We achieve (k, k, k)-merges differently: we observe that Batcher's (d_1, d_2)-merge in Algorithm 1 is order-preserving with an output array of length $d_1 + d_2$. Since only the Top-k smallest elements are of interest, directly running Batcher's (d_1, d_2)-merge would be excessively costly. Instead, we generalize the merging step from Algorithm 1 into Algorithm 2, which removes redundant comparisons and outputs at most k elements.

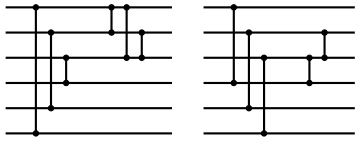
Algorithm 2 Our truncated (d_1, d_2, k)-merge

Input: Two sorted arrays $\mathbf{x}$ (of size $d_1 \leq k$) and $\mathbf{y}$ (of size $d_2 \leq k$) and $k > 0$
Output: Sorted array that contains the entries of $\mathbf{x}$ and $\mathbf{y}$, or their k smallest entries if $k < d_1 + d_2$

1: **function** MERGE($\mathbf{x}, \mathbf{y}, k$)
2: **if** $d_1 \cdot d_2 = 0$ **then**
3: $\mathbf{z} \leftarrow (\mathbf{x}, \mathbf{y})$
4: **else if** $d_1 \cdot d_2 = 1$ **then**
5: $\mathbf{z} \leftarrow$ COMPARE($\mathbf{x}_0, \mathbf{y}_0$)
6: **else** ▷ Merge even- and odd-index components
7: $\mathbf{v} \leftarrow$ MERGE($(\mathbf{x}_0, \mathbf{x}_2, \ldots, \mathbf{x}_{2\lceil d_1/2 \rceil - 2}), (\mathbf{y}_0, \mathbf{y}_2, \ldots, \mathbf{y}_{2\lceil d_2/2 \rceil - 2}), \lfloor k/2 \rfloor + 1$)
8: $\mathbf{w} \leftarrow$ MERGE($(\mathbf{x}_1, \mathbf{x}_3, \ldots, \mathbf{x}_{2\lfloor d_1/2 \rfloor - 1}), (\mathbf{y}_1, \mathbf{y}_3, \ldots, \mathbf{y}_{2\lfloor d_2/2 \rfloor - 1}), \lfloor k/2 \rfloor$)
9: $\mathbf{z} \leftarrow (\mathbf{v}_0, \mathbf{w}_0, \mathbf{v}_1, \mathbf{w}_1, \ldots)$
10: **for** $i \leftarrow 1$ to $\lfloor (\text{size}(\mathbf{z}) - 1)/2 \rfloor$ **do**
11: $(\mathbf{z}_{2i-1}, \mathbf{z}_{2i}) \leftarrow$ COMPARE($\mathbf{z}_{2i-1}, \mathbf{z}_{2i}$)
12: **end for**
13: **end if**
14: **return** TRUNCATE($\mathbf{z}, k$)
15: **end function**
16: **function** TRUNCATE($\mathbf{x}, k$) ▷ Truncate to k elements
17: $i \leftarrow \min(\text{size}(\mathbf{x}), k)$
18: **return** $(\mathbf{x}_0, \ldots, \mathbf{x}_{i-1})$
19: **end function**

Theorem 3.1. *The truncated (k, k, k)-merge contains $\mathcal{O}(k \log k)$ comparators and has a comparison depth of $\mathcal{O}(\log k)$.*

Note that our order-preserving merge procedure is not only asymptotically better than Alekseev's $\mathcal{O}(k \log^2 k)$, but is also better in practice: as Figure 2 shows, it contains fewer comparisons for a small value of $k = 3$.



(a) Alekseev's (b) Ours

Figure 2: Two constructions of a $(3, 3, 3)$-merge network.

Our truncated sorting network. Realizing step 2 in Alekseev's Top-k with our truncated (k, k, k)-merge is essentially a truncated version of Batcher's odd-even sorting algorithm, where the Top-k elements are selected in a recursive approach. As can be seen in Algorithm 3, we split the initial array into two parts, find the Top-k elements of these two parts recursively, and then call Algorithm 2 to compute the final result.

Moreover, our Algorithm 3 also improves the input partitioning in Alekseev's step 1. Specifically, we observe that the truncated network is more efficient if the chunk size is chosen as a multiple of $\mu = 2^{\lceil \log k \rceil}$. We therefore use the following heuristic: if $d > \mu$, the first chunk's size is computed as a multiple of μ that is close to $d/2$. Otherwise, the first chunk's size is equal to $\lceil d/2 \rceil$. The second chunk simply consists of the remaining elements (i.e., the ones that are not in the first chunk). As an example, the resulting network for $d = 16$ and $k = 3$ is shown in Figure 3, where each box represents a merging procedure.

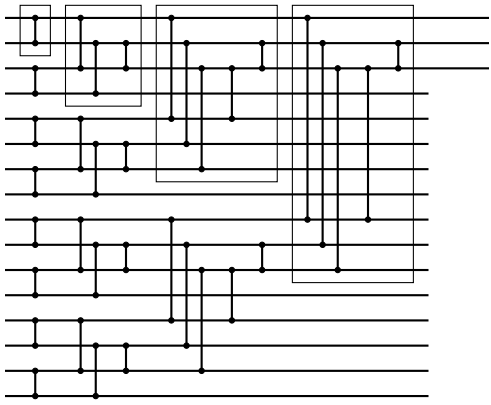


Figure 3: Our network for finding the 3 smallest values out of 16, which has 35 comparators and depth 9. Boxes visualize our truncated $(d, d, 3)$-merge for $d = 1, 2, 3, 3$ from the leftmost to the rightmost box.

Algorithm 3 Our truncated odd-even merge sort

Input: Array $\mathbf{x}$ (of size $d > 0$) and $k > 0$
Output: Sorted array that contains the entries of $\mathbf{x}$, or its k smallest entries if $k < d$

function SORT($\mathbf{x}, k$)
  if $d = 1$ **then**
    return $\mathbf{x}$
  else ▷ Sort two chunks separately and merge
    $i \leftarrow$ CHUNKSIZE(d, k)
    $\mathbf{v} \leftarrow$ SORT($(\mathbf{x}_0, \ldots, \mathbf{x}_{i-1}), k$)
    $\mathbf{w} \leftarrow$ SORT($(\mathbf{x}_i, \ldots, \mathbf{x}_{d-1}), k$)
    return MERGE($\mathbf{v}, \mathbf{w}, k$)
  end if
end function
function CHUNKSIZE(d, k) ▷ Compute size of first chunk
  $\mu \leftarrow 2^{\lceil \log k \rceil}$
  if $d \leq \mu$ **then**
    return $\lceil d/2 \rceil$
  else
    return $\mu \cdot \lceil d/(2\mu) \rceil$
  end if
end function

Theorem 3.2. *Our network for finding the k smallest elements out of d has time complexity $\mathcal{O}(d \log^2 k)$ and depth $\mathcal{O}(\log d \cdot \log k)$.*

Proof. For the ease of asymptotic analysis, we restrict the parameters d and k to powers of two. In this case, the full algorithm reduces to Batcher's odd-even sorting network until obtaining d/k sorted arrays of size k, and then performing the (k, k, k)-merge recursively as in the tournament method.

Using Theorem 3.1, the comparison depth is

$$1 + 2 + \ldots + \log k + \mathcal{O}(\log k) \cdot \log \frac{d}{k} = \mathcal{O}(\log d \cdot \log k),$$

and the total number of comparisons is

$$\sum_{i=1}^{\log k} \frac{d}{2^i}(2^{i-1}(i-1)+1) + \mathcal{O}(k \log k) \cdot \frac{d}{k} = \mathcal{O}\left(\sum_{i=1}^{\log k} \frac{d}{2} \cdot i + d \log k\right)$$
$$= \mathcal{O}(d \log^2 k).$$

□

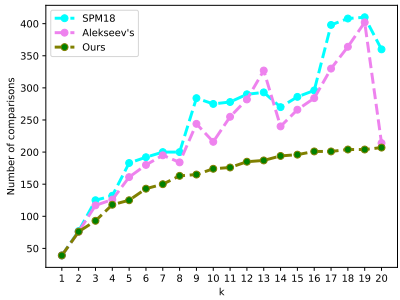


Figure 4: The number of comparisons $U(k, d)$ in selecting the Top-k elements out of $d = 40$ using the method in SPM18 [31], Alekseev's method and our truncated network. The number of comparisons is shown only for $1 \leq k \leq d/2$ as $U(k, d) = U(d - k, d)$ by symmetry.

Note that our Algorithm 3 always outputs sorted results, but the output of the Top-k problem does not need to be sorted. Therefore, two more optimizations are incorporated in the implementation: (1) if $k > d/2$, we can exchange the roles of k and $d - k$ (this will be explained in more detail in Section 3.3); (2) the last merge box can be replaced by Alekseev's merge procedure [2], where only $d_1 + d_2 - k$ comparators are used.

Comparison with related work. To the best of our knowledge, there exist three Top-k methods of complexity $\mathcal{O}(d \log^2 k)$: Alekseev's procedure (Section 3.1), a method based on bitonic sorting [31], and our method from Algorithm 3. Despite the same asymptotic complexity, our algorithm has the fewest comparators, following the explanation in Algorithm 3 and Section 1.2. An example of $d = 40$ is presented in Figure 4. Note that the monotonicity in our Top-k is a result of our input partitioning optimization.

3.3 Yao's Top-k selection network revisited

Yao [36] designed a Top-k selection network via direct recursion. Using $\lfloor d/2 \rfloor$ comparators, the input is first partitioned into two halves $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_{\lceil d/2 \rceil})$ and $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{\lfloor d/2 \rfloor})$ such that $\mathbf{x}_i < \mathbf{y}_i$ for $i = 1, \ldots, \lfloor d/2 \rfloor$. At most $\lfloor k/2 \rfloor$ elements of the desired output will be in array $\mathbf{y}$, so we can use a $(\lfloor k/2 \rfloor, \lfloor d/2 \rfloor)$-selector to find those elements. Then the output of this selector and array $\mathbf{x}$ are given to a $(k, \lceil d/2 \rceil + \lfloor k/2 \rfloor)$-selector, which produces the final result. An example construction of a $(4, 9)$-selector is given in Figure 5.

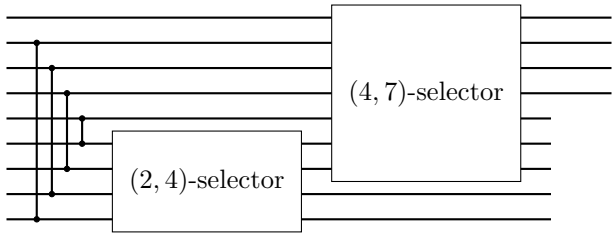


Figure 5: Recursive construction of a $(4, 9)$-selector using Yao's method.

Theorem 3.3 (Yao [36])**.** *The comparator count $U_Y(k, d)$ for Top-k using Yao's recursion satisfies*

$$U_Y(k, d) \leq d\lceil \log(k+1) \rceil + c_k (\log d)^{\lceil \log((k+1)/3) \rceil},$$

where $c_k = \mathcal{O}(k^{2+\lambda_k})$ and $\lambda_k = \mathcal{O}(\log \log k)$.

For small $k \ll \sqrt{d}$, the complexity $U_Y(k, d)$ is dominated by the first term $d\lceil \log(k+1) \rceil$. This is asymptotically lower than our complexity of $\mathcal{O}(d \log^2 k)$ (see Section 3.2). However, this is not true for $k = \Omega(\sqrt{d})$, because the second term in $U_Y(k, d)$ is asymptotically larger than $\mathcal{O}(d \log^2 k)$.

The pseudocode for Yao's method is given in Algorithm 4. Next to the recursion described above, multiple special cases are handled:

- If $k = 1$ or $k = 2$, we directly use the tournament method (pseudocode for the tournament method is omitted for brevity).
- We observe that if $k > d/2$, one can reduce the number of comparators by exchanging the roles of k and $d - k$: instead of computing the k smallest elements, we find the $d - k$ largest elements and return the remaining ones (made explicit by the set difference on line 8). The reverted functionality is called YaoSwap and returns the largest entries instead of the smallest ones.

3.4 Combined network

Since our truncated method is better for large $k = \Omega(\sqrt{d})$ and Yao's method is better for small k, we combine them into one oblivious Top-k network for improved performance. More specifically, the combined network recursively calls our truncated merge method or Yao's method, depending on which one uses fewer comparators. As Figure 6 shows, the complexity of our combined network is upper bounded by the better method of Yao's and our truncated Top-k, and it slightly improves on those methods for some parameters. This

Algorithm 4 Yao's Top-k selection network

Input: Array $\mathbf{x}$ (of size $d > 0$) and $0 < k \leq d$
Output: Array that contains the k smallest entries of $\mathbf{x}$

function YAO($\mathbf{x}, k$)
  if $k = 1$ **then**
    $\mathbf{x} \leftarrow \text{TOUR}(\mathbf{x})$
  else if $k = 2$ **then**
    $(\mathbf{x}_0, \mathbf{x}_2, \ldots, \mathbf{x}_{d-1}) \leftarrow \text{TOUR}(\mathbf{x}_0, \mathbf{x}_2, \ldots, \mathbf{x}_{d-1})$
    $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{d-1}) \leftarrow \text{TOUR}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{d-1})$
  else if $k > d/2$ **then** ▷ Exchange k with $d - k$
    return $\mathbf{x} \setminus \text{YAOSWAP}(\mathbf{x}, d - k)$
  else ▷ Denote $\mathbf{x}_{i\ldots j} = (\mathbf{x}_i, \ldots, \mathbf{x}_j)$
    base $\leftarrow (d \bmod 2) - 1$
    for $i \leftarrow 1$ to $\lfloor d/2 \rfloor$ **do**
      $(\mathbf{x}_{\text{base}+i}, \mathbf{x}_{d-i}) \leftarrow \text{COMPARE}(\mathbf{x}_{\text{base}+i}, \mathbf{x}_{d-i})$
    end for
    $\mathbf{x}_{\lceil d/2 \rceil \ldots d-1} \leftarrow \text{YAO}(\mathbf{x}_{\lceil d/2 \rceil \ldots d-1}, \lfloor k/2 \rfloor)$
    $\mathbf{x}_{0\ldots\lceil d/2 \rceil + \lfloor k/2 \rfloor - 1} \leftarrow \text{YAO}(\mathbf{x}_{0\ldots\lceil d/2 \rceil + \lfloor k/2 \rfloor - 1}, k)$
  end if
  return $\text{TRUNCATE}(\mathbf{x}, k)$
end function

improvement is only possible because the values of d and k change dynamically throughout the recursive calls of Yao's method.

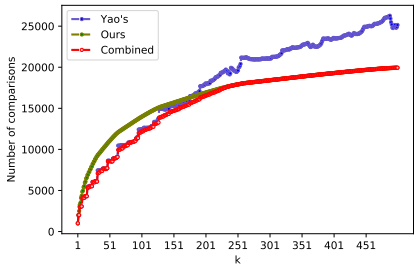


Figure 6: $U(k, d)$ in selecting the Top-k elements out of $d = 1000$ using the network with our truncated merge, the network with Yao's recursion and the combined method. The number of comparisons is shown only for $1 \leq k \leq d/2$ as $U(k, d) = U(d - k, d)$ by symmetry.

4 Our k-NN protocol instantiated with TFHE

We introduce our application of Top-k to secure k-NN, which consists of two phases: computation of the squared distances and finding the k closest vectors, together with the corresponding class labels. This is done based on the TFHE encryption scheme. Commonly used TFHE notations and symbols for the k-NN classification are summarized in Table 1. In particular, we use $\langle \mathbf{v}, \mathbf{w} \rangle$ for the dot product between two vectors $\mathbf{v}$ and $\mathbf{w}$. We also define $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, where q is a positive integer and N is a power of two.

4.1 Threat model and security for k-NN

Similarly to previous works [9,37] on secure k-NN classification, our threat model considers a semi-honest (honest-but-curious) server that follows the protocol correctly, but tries to obtain information about the client from publicly known data. To reach our security goals, the client encrypts its query before sending it to the server. The database itself is owned by the server and therefore does not need to be encrypted. After the homomorphic operation (via our protocol) with the server's data, the final output is also encrypted under the client's key. Since the data seen by the server is always encrypted, the client's input privacy is guaranteed by the IND-CPA property of TFHE. Our work does not consider model privacy, so the client might infer information about the server's database.

4.2 TFHE building blocks

TFHE ciphertexts and basic operations. The TFHE scheme [13] uses several ciphertext types based on the (ring) learning with errors problem [25, 28]. Each

Table 1: List of symbols for the TFHE scheme and the k-NN classification.

Meaning	Symbol
The LWE/RLWE dimension	n/N
The standard deviation of the noise	σ
The gadget base/size	g/ℓ
The plaintext/ciphertext modulus	t/q
The size of the database	d
The vector dimension of the database	γ
The desired number of nearest neighbors	k

ciphertext contains a noise or error term e that is added during encryption and removed during decryption. The ciphertext types are the following:

- $\mathsf{LWE}_{\mathbf{s}}(m) = (a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$, where

$$b = \sum_{i=1}^{n} -a_i \cdot s_i + \Delta \cdot m + e.$$

  The message $m \in \mathbb{Z}_t$ (with $t \ll q$) is encoded in the ciphertext under a scaling factor $\Delta = q/t$. We call t the plaintext modulus and q the ciphertext modulus. For LWE ciphertexts, the secret key is a vector $\mathbf{s} = (s_1, s_2, \ldots, s_n)$.
- $\mathsf{RLWE}_s(m) = (a, b) \in \mathcal{R}_q^2$, where $b = -a \cdot s + \Delta \cdot m + e$. Both the message $m \in \mathcal{R}_t$ and the secret key s are polynomials.

To distinguish between these types, LWE ciphertexts will be written as $\mathbf{c}$ and RLWE ciphertexts as c. Homomorphic computations are built from the following operations over the ciphertext space:

- $\mathsf{SampleExtract}(c, i) \rightarrow \mathbf{c}$: this procedure extracts one coefficient of a plaintext element encrypted as an RLWE ciphertext into an LWE ciphertext. It takes $c = \mathsf{RLWE}_s(M(X))$ and an index $0 \leq i < N$, and outputs $\mathbf{c} = \mathsf{LWE}_{\mathbf{s}}(M_i)$, where M_i is the i-th coefficient of $M(X)$. The entries of the LWE key $\mathbf{s}$ will be equal to the coefficients of the RLWE key s.
- $M(X) \cdot c \rightarrow c'$: this procedure multiplies a plaintext element by an RLWE ciphertext. Specifically, it takes $M(X) \in \mathcal{R}_t$ and $c = \mathsf{RLWE}_s(m)$, and outputs $c' = \mathsf{RLWE}_s(M(X) \cdot m)$.
- $c_1 + c_2 \rightarrow c'$: this procedure adds two RLWE ciphertexts. Specifically, it takes $c_1 = \mathsf{RLWE}_s(m_1)$ and $c_2 = \mathsf{RLWE}_s(m_2)$, and outputs $c' = \mathsf{RLWE}_s(m_1 + m_2)$. Note that this procedure can also take a ciphertext and a plaintext element instead of two ciphertexts.
- $\mathsf{KeySwitch}(\mathbf{c}_i, \mathsf{ksk}) \rightarrow c$: this procedure converts a set of LWE ciphertexts (indexed by i) to an RLWE ciphertext. The output RLWE ciphertext encrypts the same set of numbers (coefficient-wise) as the input LWE ciphertexts.
- $\mathsf{Bootstrap}(\mathbf{c}, \mathsf{bk}, f) \rightarrow \mathbf{c}'$: this procedure reduces the noise of the input LWE ciphertext, while at the same time evaluating a negacyclic function f (i.e., it needs to satisfy $f(m + t/2) = -f(m)$). If the function is unknown, we need to initialize the procedure with an encrypted *accumulator*. This is a ciphertext $\mathsf{RLWE}_s(T(X))$ that encodes the desired function, obtained via $\mathsf{KeySwitch}$. The *test polynomial* $T(X)$ depends directly on the function f.

Homomorphic computation of the squared distance. One essential building block of k-NN classification is computation of the squared distance between an encrypted target vector and a cleartext data point. We adapt the method of Zuber and Sirdey [37] to compute the squared distance between a target vector and a model vector efficiently. Their method actually computes the *difference* between two squared distances, but we need the squared distance itself to be compared in the sorting network.

We are given one target vector $c \in \mathcal{R}_q^2$ (the client's encrypted input), which is an RLWE ciphertext that encodes $\mathbf{v} \in \mathbb{Z}_t^\gamma$. And we have a model vector $\mathbf{w} \in \mathbb{Z}_t^\gamma$ stored in the database. We assume that the model vector is given in cleartext since the server owns the database in our scenario. The goal here is to compute $\|\mathbf{v} - \mathbf{w}\|_2^2 = \|\mathbf{v}\|_2^2 - 2 \cdot \langle \mathbf{v}, \mathbf{w} \rangle + \|\mathbf{w}\|_2^2$ homomorphically. To do this, the model vector is encoded in two ways:

$$M(X) = \sum_{i=0}^{\gamma-1} \mathbf{w}_{\gamma-i-1} \cdot X^i \quad \text{and} \quad M'(X) = \left(\sum_{i=0}^{\gamma-1} \mathbf{w}_i^2 \right) \cdot X^{\gamma-1}.$$

Similarly, the target vector $\mathbf{v}$ is encrypted as

$$c = \mathsf{RLWE}_s \left(\sum_{i=0}^{\gamma-1} \mathbf{v}_i \cdot X^i \right) \tag{2}$$

and

$$c' = \mathsf{RLWE}_s \left(\left(\sum_{i=0}^{\gamma-1} \mathbf{v}_i^2 \right) \cdot X^{\gamma-1} \right).$$

Then the squared distance between the encrypted target vector c and the model vector $\mathbf{w}$ can be computed as

$$c'' = c' - 2M(X) \cdot c + M'(X). \tag{3}$$

The result computed in (3) is an RLWE ciphertext that encrypts a polynomial, the $(\gamma - 1)$-th coefficient of which gives us the squared distance. Therefore, we run $\mathsf{SampleExtract}(c'', \gamma - 1)$ to get $\mathsf{LWE}_{\mathbf{s}}(\|\mathbf{v} - \mathbf{w}\|_2^2)$, which works if $\gamma \leq N$.

An optimization. It is sufficient for the k-NN application to compute the squared distances between target and model vectors up to a certain constant. In particular, since the ciphertext c' is identical for each squared distance, it can simply be removed from (3) and we obtain

$$c'' = -2M(X) \cdot c + M'(X). \tag{4}$$

This reduces the communication between client and server by 50% as now only one RLWE ciphertext is sent.

Comparison operations. Comparing two encrypted numbers can be done with programmable bootstrapping. For example, Zuber and Chakraborty [8] proposed two homomorphic comparison operators to build min and arg min functions. Apart from the minimum and its argument, our protocol also requires the maximum and its argument, so we implement a different algorithm.

We are given four ciphertexts $\mathbf{c}_0 = \mathsf{LWE}_{\mathbf{s}}(m_0)$ and $\mathbf{c}_1 = \mathsf{LWE}_{\mathbf{s}}(m_1)$, and their corresponding labels $\mathbf{c}'_0 = \mathsf{LWE}_{\mathbf{s}}(m'_0)$ and $\mathbf{c}'_1 = \mathsf{LWE}_{\mathbf{s}}(m'_1)$. We need to compute the following four results:

- An LWE encryption of $\min(m_0, m_1)$.
- An LWE encryption of $\max(m_0, m_1)$.
- An LWE encryption of m'_i with $i = \arg\min(m_0, m_1)$.
- An LWE encryption of m'_i with $i = \arg\max(m_0, m_1)$.

First, we homomorphically compute the difference of the squared distances as $\mathbf{c}' = \mathbf{c}_0 - \mathbf{c}_1 = \mathsf{LWE}_{\mathbf{s}}(m)$, where $m = m_0 - m_1$. This ciphertext encrypts a positive number if $m_1 < m_0$. The input ciphertext of bootstrapping is set to $\mathbf{c}'$, which serves as a selector. The minimum can now be computed with the function

$$f(m) = \begin{cases} m_0 & \text{if } -t/4 < m \leq 0 \\ m_1 & \text{if } 0 \leq m < t/4. \end{cases} \tag{5}$$

Here we only consider the domain $(-t/4, t/4)$ to guarantee that f is negacyclic. The test polynomial can now be constructed as

$$T(X) = \sum_{i=0}^{N/2-1} m_1 \cdot X^i - \sum_{i=N/2}^{N-1} m_0 \cdot X^i,$$

where we used $f(m) = -f(m - t/2) = -m_0$ for $t/4 < m < t/2$. Similarly, the test polynomial for arg min can be constructed by replacing m_0 and m_1 with m'_0 and m'_1 in (5). Note that these four values are actually encrypted, so both test polynomials are obtained via $\mathsf{KeySwitch}$ on $\mathbf{c}_0$, $\mathbf{c}_1$, $\mathbf{c}'_0$ and $\mathbf{c}'_1$. Finally, observe that the maximum can be computed as $\max(m_0, m_1) = m_0 + m_1 - \min(m_0, m_1)$ and similarly for arg max.

4.3 The protocol

Squared distance computation. First, the client encrypts the target vector using (2) and sends it to the server. Then, for each model vector, the server evaluates the formula in (4) and extracts the $(\gamma - 1)$-th coefficient to compute its squared distance. The result of the distance computation satisfies $\|\mathbf{v} - \mathbf{w}\|_2^2 < t/4$, such that the comparators can be built using (5). Although our protocol uses the Euclidean distance between target and model vectors, one could also replace

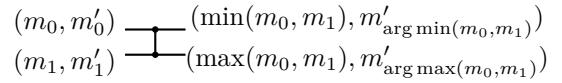


Figure 7: An augmented comparator, where $\arg\min(m_0, m_1)$ and $\arg\max(m_0, m_1)$ refer to the indices (either 0 or 1) of the minimum and maximum element.

this by essentially any distance metric that can be computed efficiently with FHE.

Precision reduction (optional). The squared distances may be computed using a large plaintext modulus (t_{dist}), but the input of programmable bootstrapping (PBS) expects a small plaintext modulus (we need $t_{\mathsf{sort}} \ll 2N$). If the two plaintext moduli are different, we need to perform a precision reduction, which can be done with one subtraction and one bootstrapping operation for every squared distance. The subtraction is necessary because we need to "recenter" the plaintext space. For example, consider plaintext moduli $t_{\mathsf{dist}} = 2 \cdot t_{\mathsf{sort}}$, and their scaling factors $2 \cdot \Delta_{\mathsf{dist}} = \Delta_{\mathsf{sort}}$. Encoded plaintexts of the form $(m_i \cdot \Delta_{\mathsf{dist}}, (m_i + 1) \cdot \Delta_{\mathsf{dist}})$ are mapped to $(m_i/2) \cdot \Delta_{\mathsf{sort}}$ since we want to reduce the precision by one bit in this example. Before bootstrapping, the center of $(m_i \cdot \Delta_{\mathsf{dist}}, (m_i + 1) \cdot \Delta_{\mathsf{dist}})$ needs to be at $(m_i/2) \cdot \Delta_{\mathsf{sort}} = m_i \cdot \Delta_{\mathsf{dist}}$. As such, we need to subtract $\Delta_{\mathsf{dist}}/2$ from the initial plaintext and then perform bootstrapping with the identity function. This method easily generalizes to the case where t_{sort} is any multiple of t_{dist}.

Precision reduction is only necessary if γ is high or if the precision of every element in the feature vector is large in comparison to t_{sort}. Section 5 shows that precision reduction is necessary for MNIST but not for the breast cancer dataset.

The Top-k selection network. To instantiate our Top-k network for privacy preserving k-NN, we need a comparator that also outputs arg min and arg max (to represent the label) next to the minimum and maximum. This comparator is visualized in Figure 7 and its instantiation is described in Section 4.2.

Using the squared distance values as the scoring function, we then apply our combined Top-k network composed of augmented comparators. The output of this phase is a set of k LWE ciphertexts that encrypt the predicted class labels, which are sent back to the client for decryption. Finally, the client computes the most common class label in the clear via majority voting. This is acceptable for most use cases as typically k is much smaller than d.

Noise growth of our protocol. Programmable bootstrapping is used to lower the noise level of its input, and computing a non-linear function at the same time. However, even though homomorphic comparisons are implemented with bootstrapping, the squared distances are never refreshed during the sorting phase. This is because the accumulator is generated by KeySwitch and is therefore a noisy ciphertext. Yet, both datasets tested in the next section have an output noise that remains at least 10 bits below the 64-bit ciphertext modulus. This is sufficient to support a plaintext precision of 10 bits without requiring extra bootstrapping operations.

5 Evaluation

5.1 Implementation and experimental setup

Our prototype implementation is written in the Rust programming language using the TFHE-rs[2] library. The source code can be found on GitHub.[3] All experiments in this section are executed on machines with Intel(R) Core(TM) i9-9900 CPU @ 3.10 GHz using the Ubuntu 20.04 operating system. Our implementation supports multi-threading in the sorting network, i.e., if two comparators are on the same level in the network, then they may be executed in parallel.

Our experiment uses (a reduced version of) two datasets: the MNIST[4] and breast cancer[5] datasets. We preprocess the MNIST dataset in two ways: (1) the images are downsized to 8×8 pixels which are feature vectors of length $\gamma = 64$; (2) elements in every feature vector are converted to ternary values. The breast cancer dataset has $\gamma = 32$ and we preprocess the feature vectors to use binary values. This kind of preprocessing is similar to [37] where the authors also convert the MNIST images to 8×8 pixels and divide values by 300.

We run our privacy preserving k-NN protocol using different values of d and k for both datasets and report the timing, accuracy and bandwidth results below. All experiments are done with the best feature vectors as the model. This is done by creating $10,000$ plaintext models at random and selecting the one that gives the highest accuracy when evaluated on all the possible test vectors. Then we average prediction/inference over 200 randomly selected test vectors.

[2] https://github.com/zama-ai/tfhe-rs

[3] https://github.com/KULeuven-COSIC/ppknn

[4] https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits

[5] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Table 2: The TFHE parameters used in our experiments. Note that when the homomorphic computation is done, the most significant bit is reserved for padding. Hence if $t = 2^i$, then the actual message uses $i - 1$ bits.

Parameter	Value
LWE dimension (n)	856
RLWE polynomial degree (N)	4096
LWE standard deviation (σ_{LWE})	2^{44}
RLWE standard deviation (σ_{RLWE})	2^2
Decomp params bootstrapping (g, ℓ)	$(2^{22}, 1)$
Decomp params LWE-to-LWE (g, ℓ)	$(2^3, 6)$
Decomp params LWE-to-RLWE (g, ℓ)	$(2^{23}, 1)$
Ciphertext modulus (q)	2^{64}
Plaintext modulus (t_{sort})	2^6
Plaintext modulus MNIST (t_{dist})	2^9
Plaintext modulus breast cancer (t_{dist})	2^6
Dataset message space MNIST	$\mathbb{Z}_3$
Dataset message space breast cancer	$\mathbb{Z}_2$

The TFHE parameters are given in Table 2. These parameters are adapted from TFHE-rs.[6] We make a distinction between the plaintext modulus for distance computation (t_{dist}) and sorting (t_{sort}). That is, if $t_{\mathsf{dist}} \neq t_{\mathsf{sort}}$, then the precision reduction step from Section 4 needs to be used. Our definition of the plaintext modulus includes the padding bit. This extra padding bit is necessary to satisfy negacyclicity when the data is encoded. For example, if the plaintext modulus is $t = 2^6$, then the message space is 5 bits since one bit is reserved for padding. The parameters from Table 2 guarantee 128 bits of security [1].

The sections below primarily report the computation time and bandwidth. Memory usage is not detailed since it is not a significant overhead in our construction. Namely, our biggest experiment ($d = 1000$, $k = \left\lfloor \sqrt{d} \right\rfloor$) used only 700 MB of memory. Bootstrapping and key switching keys dominate the memory usage.

5.2 Computation time

The computation time and accuracy probabilities for the MNIST dataset are shown in Table 3, together with the results taken (and extrapolated) from [37].

[6] `https://github.com/zama-ai/tfhe-rs/blob/release/0.1.x/tfhe/src/shortint/parameters/mod.rs`

Modulo the difference in CPU (we estimate that our CPU is at most two times faster than theirs), the wall-clock time ranges from 1.7× to over 47× faster than prior work [37] while maintaining a good level of accuracy.

The main reason for our acceleration is the performance gain in the costly Top-k selection step. In the delta-matrix method of Zuber and Sirdey [37], a $d \times d$ matrix is constructed. Each element at position (i, j) in the matrix is 0 if the target vector is closer to the i-th model vector than to the j-th vector, and 1 otherwise. As such, building the matrix itself requires $(d^2 - d)/2$ comparison operations, then additional scoring operations are needed. In comparison, our Top-k network scales linearly with d and quadratically with $\log k$.

Additionally, this experiment demonstrates the effect of precision reduction. Starting with 9 bits of precision for the distance computation, we reduce to 6 bits before the start of the sorting network. From the results, we see that this has very little effect on accuracy (note that precision reduction is not applied in the "Clear accuracy" column).

Table 3: Computation time and accuracy for the MNIST dataset. The distance computation is performed using 9 bits of precision, then it is converted to 6 bits before running the selection network. The computation times prefixed with ∼ are estimated using extrapolation. The number of parallel threads is τ.

		Duration (s)			Comparators		Accuracy	
k	d	[37]	$\tau = 4$	$\tau = 1$	[37]	Ours	Clear	FHE
3	40	30	8.7	17.5	780	93	0.81	0.79
	175	696	31.9	78.1	15225	431	0.94	0.94
	269	1524	47.4	119.5	36046	666	0.95	0.94
	457	4248	78.9	202.3	104196	1136	0.98	0.97
	1000	∼ 20837	168.0	441.1	499500	2493	0.98	0.96
5	40	∼ 33	11.6	25.5	780	125	0.74	0.73
	175	∼ 636	43.1	112.7	15225	598	0.92	0.90
	269	∼ 1505	62.7	173.0	36046	928	0.94	0.93
	457	∼ 4351	105.0	291.1	104196	1586	0.97	0.97
	1000	∼ 20859	227.5	642.3	499500	3485	0.98	0.96
$\lfloor\sqrt{d}\rfloor$	40	∼ 33	13.1	28.1	780	143	0.75	0.74
	175	∼ 639	68.4	171.8	15225	1015	0.89	0.89
	269	∼ 1516	117.7	310.4	36046	1789	0.95	0.94
	457	∼ 4402	209.0	530.2	104196	3412	0.95	0.94
	1000	∼ 21410	455.8	1252	499500	9121	0.98	0.97

Similarly, the computation time and accuracy probabilities for the breast cancer dataset are presented in Table 4. For this dataset, there is no precision reduction step (i.e., $t_{\mathsf{dist}} = t_{\mathsf{sort}}$), because γ is low, the feature vectors are somewhat sparse and we preprocess the data to have binary feature vectors. Since our plaintext modulus is only 6 bits (one bit is reserved as the padding bit and another one for the sign), the squared distance cannot exceed 4 bits. As such, we have some errors when compared to the plaintext algorithm since the result may overflow into the padding bit occasionally. Fortunately, the overflow does not happen often and our FHE accuracy closely trails the plaintext accuracy.

Table 4: Computation time and accuracy for the breast cancer dataset. No precision reduction is performed. The computation times prefixed with $\sim$ are estimated using extrapolation. The number of parallel threads is τ.

		Duration (s)			Comparators		Accuracy	
k	d	[37]	$\tau = 4$	$\tau = 1$	[37]	Ours	Clear	FHE
3	10	4	1.8	3.2	45	18	0.94	0.92
	30	~ 18	5.0	11.5	435	68	0.94	0.94
	50	~ 51	7.4	19.0	1225	118	0.94	0.94
	200	~ 830	25.5	76.0	19900	493	0.95	0.94
5	10	~ 2	2.2	4.2	45	21	0.91	0.88
	30	~ 18	7.5	16.7	435	91	0.95	0.93
	50	~ 52	11.6	28.8	1225	161	0.96	0.95
	200	~ 831	40.2	114.6	19900	685	0.96	0.96
$\lfloor\sqrt{d}\rfloor$	200	~ 836	69.9	185.7	19900	1234	0.95	0.95

5.3 Bandwidth

Both our solution and [37] require bootstrapping for the computation, so evaluation keys should be sent at the setup phase. This costs 160 MB in our case, smaller than 200 MB of [37]. We use three different evaluation keys: two key switching keys (53.5 MB) and one bootstrapping key (107 MB) which takes the dominant part of the whole key size. On the other hand, the previous work uses two different bootstrapping keys, leading to higher memory consumption. We note that the evaluation key size is not considered as online bandwidth in both works, since it is only sent once and reused for the repeated computation.

After executing our protocol, the server returns the k selected labels, which are in the form of LWE ciphertexts. Therefore, the answer size would be k times 6.7 KB. As an optimization, we can easily pack k LWE ciphertexts into an RLWE

ciphertext almost for free as long as $k \leq N$ [10]. We can also reduce the size of the answer by switching the modulus from 64 bits to 32 bits [7] and reduce the degree of the polynomials from 4096 to 1024 by key switching. The resulting answer has a size of 8 KB, which is smaller than k LWE ciphertexts for $k \geq 2$.

6 Conclusion and future directions

Top-k selection algorithms are broadly used in various applications, and secure computation highly benefits from the obliviousness of Top-k selection. We revisited the constructions by Alekseev (1969) and Yao (1980), and then proposed additional improvements for $k = \Omega(\sqrt{d})$. Our resulting combined Top-k network has complexity $\mathcal{O}(d \log^2 k)$ in general and $\mathcal{O}(d \log k)$ for small $k \ll \sqrt{d}$.

The efficiency of our combined Top-k network is demonstrated with an application: homomorphic k-NN classification. Compared with the state of the art [37], where oblivious Top-k was realized with complexity $\mathcal{O}(d^2)$, our experimental results show a speedup of up to 47 times.

Future directions. Our TFHE instantiation of k-NN quantizes values (of 8 bits or more) down to binary or ternary values, in order to work with the restricted plaintext space. This step affects the accuracy of our secure k-NN protocol. In the future, we hope to investigate techniques that would support plaintexts with large precision, for example as proposed by Liu et al. [24].

Although our combined Top-k network has the best performance compared to existing methods, it does not give the optimal asymptotic complexity $\mathcal{O}(d \log k)$ for all parameters. Further improvements would therefore be interesting. Moreover, many secure computation applications include oblivious Top-k as a building block. It would also be interesting to incorporate our solution to improve the performance of those applications.

Acknowledgments

This work was supported by CyberSecurity Research Flanders with reference number VR20192203. Additionally, this work has been supported in part by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. FA8750-19-C-0502, and by the Defence Advanced Research Projects Agency (DARPA) under contract No. HR001120S0032. In addition, this work is supported in part by the European Commission through the Horizon 2020 research and innovation

program Belfort ERC Advanced Grant 101020005. Robin Geelen is funded by Research Foundation – Flanders (FWO) under a PhD Fellowship fundamental research (project number 1162123N). The authors would also like to thank Frederik Vercauteren for giving feedback on this paper.

References

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[2] Vladimir Evgen'evich Alekseev. Sorting algorithms with minimum memory. *Cybernetics*, 5(5):642–648, 1969.

[3] Yulliwas Ameur, Rezak Aziz, Vincent Audigier, and Samia Bouzefrane. Secure and non-interactive-nn classifier using symmetric fully homomorphic encryption. In *International Conference on Privacy in Statistical Databases*, pages 142–154. Springer, 2022.

[4] Michiel Van Beirendonck, Jan-Pieter D'Anvers, and Ingrid Verbauwhede. FPT: a fixed-point accelerator for torus fully homomorphic encryption. Cryptology ePrint Archive, Report 2022/1635, 2022. `https://eprint.iacr.org/2022/1635`.

[5] Jonas Bertels, Michiel Van Beirendonck, Furkan Turan, and Ingrid Verbauwhede. Hardware acceleration of FHEW. In Maksim Jenihhin, Hana Kubátová, Nele Metens, Jaan Raik, Foisal Ahmed, and Jan Belohoubek, editors, *26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2023, Tallinn, Estonia, May 3-5, 2023*, pages 57–60. IEEE, 2023.

[6] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[8] Olive Chakraborty and Martin Zuber. Efficient and accurate homomorphic comparisons. Cryptology ePrint Archive, Report 2022/622, 2022. `https://eprint.iacr.org/2022/622`.

[9] Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya P. Razenshteyn, and M. Sadegh Riazi. SANNS: Scaling up secure approximate k-nearest neighbors search. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2111–2128. USENIX Association, August 2020.

[10] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21, Part I*, volume 12726 of *LNCS*, pages 460–479. Springer, Heidelberg, June 2021.

[11] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[13] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.

[14] Kelong Cong, Debajyoti Das, Georgio Nicolas, and Jeongeun Park. Panacea: Non-interactive and stateless oblivious RAM. Cryptology ePrint Archive, Report 2023/274, 2023. `https://eprint.iacr.org/2023/274`.

[15] Kelong Cong, Debajyoti Das, Jeongeun Park, and Hilder V. L. Pereira. SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 563–577. ACM Press, November 2022.

[16] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1135–1150. ACM Press, November 2021.

[17] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

[18] Robin Geelen, Michiel Van Beirendonck, Hilder V. L. Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios Dimou, Ingrid Verbauwhede, Frederik Vercauteren, and David W. Archer. BASALISC: Flexible asynchronous hardware accelerator for fully homomorphic encryption. Cryptology ePrint Archive, Report 2022/657, 2022. `https://eprint.iacr.org/2022/657`.

[19] Ilia Iliashenko and Vincent Zucca. Faster homomorphic comparison operations for BGV and BFV. *PoPETs*, 2021(3):246–264, July 2021.

[20] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[21] Kristján Valur Jónsson, Gunnar Kreitz, and Misbah Uddin. Secure multi-party sorting and applications. Cryptology ePrint Archive, Report 2011/122, 2011. `https://eprint.iacr.org/2011/122`.

[22] Donald E Knuth. *The art of computer programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

[23] Oliver Kramer. *K-Nearest Neighbors*, pages 13–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[24] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 130–160. Springer, Heidelberg, December 2022.

[25] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

[26] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[27] Jeongeun Park and Mehdi Tibouchi. SHECS-PIR: Somewhat homomorphic encryption-based compact and scalable private information retrieval. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 86–106. Springer, Heidelberg, September 2020.

[28] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[29] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Nathan Manohar, Nicholas Genise, Srinivas Devadas, Karim Eldefrawy, Chris Peikert, and Daniel Sánchez. Craterlake: a hardware accelerator for efficient unbounded computation on encrypted data. In Valentina Salapura, Mohamed Zahran, Fred Chong, and Lingjia Tang, editors, *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022*, pages 173–187. ACM, 2022.

[30] Sacha Servan-Schreiber, Simon Langowski, and Srinivas Devadas. Private approximate nearest neighbor search with sublinear communication. In *2022 IEEE Symposium on Security and Privacy*, pages 911–929. IEEE Computer Society Press, May 2022.

[31] Anil Shanbhag, Holger Pirk, and Samuel Madden. Efficient top-k query processing on massively parallel hardware. In Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 1557–1570. ACM, 2018.

[32] Hayim Shaul, Dan Feldman, and Daniela Rus. Secure k-ish nearest neighbors classifier. *PoPETs*, 2020(3):42–61, July 2020.

[33] Anselme Tueno, Yordan Boev, and Florian Kerschbaum. Non-interactive private decision tree evaluation. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 174–194. Springer, 2020.

[34] Guan Wang, Tongbo Luo, Michael T Goodrich, Wenliang Du, and Zutao Zhu. Bureaucratic protocols for secure two-party sorting, selection, and permuting. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 226–237, 2010.

[35] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 139–152, 2009.

[36] Andrew Chi-Chih Yao. Bounds on selection networks. *SIAM Journal on Computing*, 9(3):566–582, 1980.

[37] Martin Zuber and Renaud Sirdey. Efficient homomorphic evaluation of k-NN classifiers. *PoPETs*, 2021(2):111–129, April 2021.

Chapter 8

Faster Private Decision Tree Evaluation for Batched Input from Homomorphic Encryption

Publication Data

Cong, K., Kang, J., Nicolas, G., and Park, J. Faster private decision tree evaluation for batched input from homomorphic encryption. In *SCN 24, Part II* (Sept. 2024), C. Galdi and D. H. Phan, Eds., vol. 14974 of *LNCS*, Springer, Cham, pp. 3–23

Contribution

I designed decision tree evaluation protocols and analyzed their performance in discussion with other co-authors.

Faster Private Decision Tree Evaluation for Batched Input from Homomorphic Encryption

Kelong Cong[1*], Jiayi Kang[2], Georgio Nicolas[2], and Jeongeun Park[3*]

[1] Zama, Paris, France
[2] COSIC, ESAT, KU Leuven, Belgium
[3] Norwegian University of Science and Technology (NTNU)

Abstract. Privacy-preserving decision tree evaluation (PDTE) allows a client that holds feature vectors to perform inferences against a decision tree model on the server side without revealing feature vectors to the server. Our work focuses on the non-interactive batched setting where the client sends a batch of encrypted feature vectors and then obtains classifications, without any additional interaction. This is useful in privacy-preserving credit scoring, biometric authentication, and many more applications.

In this paper, we propose two novel non-interactive batched PDTE protocols, BPDTE_RCC and BPDTE_CW, based on two batched ciphertext-plaintext comparison algorithms, our batched range cover comparison (RCC) comparator and the constant-weight (CW) piece-wise comparator, respectively. When comparing 16-bit batched encrypted values to a single plaintext value, our comparison algorithms show a speedup of up to 72× compared to the state-of-the-art Level Up (CCS'23). Moreover, we introduced a new tree traversal method called adapted SumPath, to achieve $\mathcal{O}(1)$ complexity of the server's response, whereas Level Up has $\mathcal{O}(2^d)$ complexity for a depth-d tree and the client needs to look up classification values in a table. Overall, our PDTE protocols attain the optimal server-to-client communication complexity and are up to 17× faster than Level Up in batch size 16384.

1 Introduction

In the era of big data, machine learning (ML) has emerged as a powerful tool to connect data and extract valuable information. Many well-known companies

*Work partially done while the author was at COSIC, ESAT, KU Leuven.

such as Amazon, Microsoft and IBM are present in this market by providing machine learning as a service (MLaaS). Namely, the cloud server holds a pre-trained machine learning model and provides useful service by performing inference with clients' data.

However, clients' data may be confidential and sharing it in the clear with the server can threaten their privacy. This leads to rising interests in privacy-preserving machine learning protocols [11, 12, 23, 34]. This work focuses on Private Decision Tree Evaluation (PDTE) [1, 11, 18, 23, 28, 29], where the server holds a decision tree classification model and the client obtains the inference result without revealing the input data.

In particular, the focus of this work is on *non-interactive batched* PDTE. Non-interactive implies the client sends a query and receives the output without additional interactions with the server. This allows the client to stay offline during the evaluation process and achieve full outsourcing. Two recent works, SortingHat [11] and Level Up [23] use homomorphic encryption (HE) for non-interactive PDTE. In particular, SortingHat uses schemes such as TFHE [10] and FINAL [3], outperforming for single-query scenarios, whereas Level Up employs the levelled BFV [5, 14] scheme, which supports homomorphic evaluations in a SIMD (Single-Instruction Multiple-Data) manner.

Batched PDTE allows evaluations of the same decision tree for multiple samples in parallel. Precisely, for a fixed decision tree held by the server and a client with multiple feature vectors as inputs, batched PDTE allows the client to send and receive once, instead of sending these feature vectors over and over to get the inference result of each. This could be useful in PDTE applications, e.g., when a bank outsources a credit-scoring decision tree and needs evaluations for various applicants without revealing their profiles [9, 19, 32].

Our work focuses on the batched PDTE using BFV, and our newly-proposed protocols, BPDTE_RCC and BPDTE_CW, outperform Level Up for large batch sizes (e.g. > 2100). Since PDTE consists of ciphertext-plaintext comparisons in decision nodes and a tree traversal procedure for aggregation, these building blocks are improved in Section 3 and Section 4, respectively.

In Section 3, we propose two batched ciphertext-plaintext comparisons, our batched RCC comparator and the constant-weight piece-wise comparator, which are based on the prior RCC comparator [23] and folklore bit-wise comparator [15, 22, 23]. By fully exploiting the fact that batched encrypted values are compared to a single plaintext value, we achieve up to over $72\times$ speedup for 16-bit numbers while maintaining a low multiplicative depth.

Moreover, Level Up uses SumPath for tree traversal, where the amortized response of the server is $\mathcal{O}(2^d)$ for a decision tree of depth d and the client needs to look up classification values in a table. This further restricts the extension of

decision tree evaluations to tree ensembles. Therefore, we introduce an adapted SumPath in Section 4, where the amortized response of the server is $\mathcal{O}(1)$ at the cost of $\mathcal{O}(\log_2 d)$ multiplicative depth.

By combining the adapted SumPath with batched ciphertext-plaintext comparisons, our two batched non-interactive PDTE protocols, BPDTE_RCC and BPDTE_CW, avoid the client looking up classification values and are also up to 17× faster than Level Up in batch size 16384.

1.1 Related Work

In interactive PDTE, the client and server communicate multiple rounds and perform a secure two-party computation. Previous protocols in [2, 4, 8, 30] fall into this category, and an enlightening survey of PDTE was presented in [18]. With sufficient bandwidth, decision tree training is also feasible, as in [20, 33]. Interactive protocols, however, do not support computation outsourcing since the client needs to be online during the evaluation.

For non-interactive PDTE, SortingHat and Level Up are the respective state-of-art using non-batched FHE such as TFHE and batched data via BGV/BFV. Other prior works include [31] and [28] using additive homomorphic encryption, [21] that improves non-interactive comparisons, [1] that uses private information retrieval (PIR) in tree traversal, and Tueno et al. [29] that firstly made non-interactive PDTE practical. A concurrent work [26] evaluates binary decision trees in a ciphertext-ciphertext operation setting based on CKKS and proposes a decision tree training method. Their protocol uses the SIMD packing method to run a protocol per an input efficiently by mapping one tree model into one ciphertext, therefore, the purpose of using SIMD packing is different to ours.

2 Preliminaries

2.1 Notation

Bold symbols such as $\mathbf{a}$ denote arrays of elements. The notation $\mathbf{a}[i]$ denotes the i-th element in $\mathbf{a}$, and $\mathbf{a}[i, j]$ denotes the sub-array from the i-th element to the j-th element (both inclusive) in $\mathbf{a}$. The first element in the array has index 1. The notation $\mathbb{1}_f$ denotes the binary output of evaluating the condition f, which equals 1 if f holds and 0 otherwise.

2.2 Decision Trees

A decision tree represents a function $\mathcal{T} : X \longrightarrow \{0, \ldots, k-1\}$ which maps an n-dimensional feature vector into a classification value. The function $\mathcal{T}$ contains m *decision nodes* organized hierarchically in depth d, together with $m+1$ *leaves*, each associated with a value in $\{0, \ldots, k-1\}$. Table 1 presents a complete list of symbols used in a decision tree.

The decision tree evaluation amounts to traversing a path from the root node to a resulting leaf, whose associated classification value is returned as the output. Precisely, each decision node compares an input feature x_i to a pre-trained threshold value y_j, yielding $b \leftarrow \mathbb{1}_{x_i \geq y_j}$. If $b = 1$, the evaluation proceeds to the right child node; otherwise, it moves to the left child node. As such, the evaluation path contains at most d decision nodes and ends up in an *output leaf*, whose corresponding classification value is returned.

2.3 Levelled Homomorphic Encryption

Levelled homomorphic encryption (LHE) such as BGV [6] and BFV [5, 14] allows evaluations of bounded-depth circuits without knowing the secret key. In practice, applications with higher multiplicative depth necessitate larger LHE parameters, consequently resulting in higher communication, storage and computation costs. Hence, algorithms with reduced multiplicative depth are preferred for LHE.

For BGV/BFV, the ring $R = \mathbb{Z}[X]/\left(X^N + 1\right)$ where N is a power of 2 is widely used. With a plaintext modulus t and a ciphertext modulus $q \gg t$, the plaintext

Table 1: List of symbols for a decision tree

Symbol	Meaning
$\mathcal{T}$	Decision tree
d	Depth of decision tree
m	Number of decision nodes
$\mathbf{y} = \{y_1, \ldots, y_m\}$	Thresholds for decision nodes
X	Collection of feature vectors
n	Dimension of a feature vector
s	Bitlength of a feature
$\mathbf{x} = \{x_1, \ldots, x_n\}$	Input feature vector
k	Number of classification values
$\mathbf{v} = \{v_1, \ldots, v_{m+1}\}$	Classification values associated with leaf nodes

space is $R_t = R/tR$ and the ciphertext space is $R_q \times R_q$ where $R_q = R/qR$. For a prime t that satisfies $t \bmod 2N = 1$, the polynomial $\left(X^N + 1\right)$ splits into N linear factors modulo t. Therefore, according to the Chinese Reminder Theorem, there exists an isomorphism $R_t \cong \mathbb{F}_t^N$ between the plaintext space R_t and N copies of $\mathbb{F}_t$, with each termed a *slot* [27]. This enables encoding and encrypting messages in N slots into a single ciphertext and performing homomorphic operations over encoded values in a SIMD manner.

2.4 PDTE and Tree Traversal

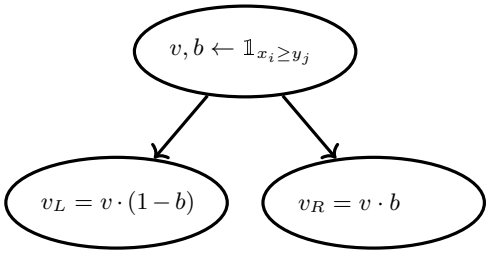
Suppose the server holds a pre-trained decision tree model $\mathcal{T}$, and a client wants to evaluate $\mathcal{T}$ on his feature vectors without disclosing them to the server or interactions during the evaluation. This necessitates a non-interactive PDTE, which could be achieved using homomorphic encryption.

In the homomorphic evaluation $\mathcal{T}$, a homomorphic comparison in a decision node gives an encrypted bit $\mathsf{Enc}(b) \leftarrow \mathsf{Enc}(\mathbb{1}_{x_i \geq y_j})$. Since the server cannot infer the value of b from $\mathsf{Enc}(b)$, determining which child node (left or right) to evaluate is infeasible unless a costly PIR procedure is incorporated [1]. Otherwise, *both* child nodes of every decision node must be evaluated, resulting in evaluations of all the m decision nodes in $\mathcal{T}$.

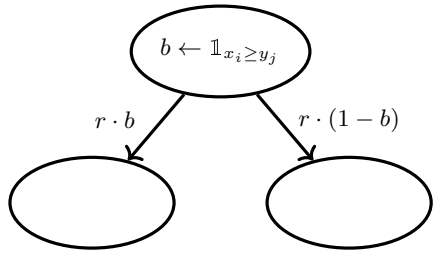
Tree traversal is a data-oblivious procedure to aggregate evaluation results of these m decision nodes. In previous works, SortingHat employs Path Conjugation for tree traversal, which is also used in [30]. On the other hand, Level Up [23] employs another SumPath method, which is also used in [18, 28, 29].

In Path Conjugation, every decision node is associated with two values: a node value v and a control bit b comparing some feature value to a threshold value. The node value is determined by the node value and the control bit of the previous decision node, as illustrated in Figure 1a. As such, the leaf node is also associated with a node value, which equals one for the desired output leaf and zero otherwise.

In SumPath, every edge is assigned an *edge cost* determined by the control bit of the previous decision node, as illustrated in Figure 1b. Since each leaf node is connected to the root in a unique path, summing up the edge costs along this path yields the *path cost* of a leaf node. As such, only the path cost of a desired output leaf equals to zero, and for all other leaves path costs are non-zero values.



(a) Path Conjugation, where v denotes the value stored in a node



(b) SumPath, where r is a random number or 1

Figure 1: Two oblivious tree traversal methods

2.5 Oblivious Binary Codes Comparison

Binary encoding for an integer $x \in [0, L-1]$ is generally classified into two categories: binary representation $BR(x)$ of length $\log_2 L$, or a constant-weight encoding $CW_{h,\ell}(x)$ of weight h and bit length ℓ. In the latter category, the bit length ℓ is determined by the relation $\binom{\ell}{h} \geq L$, which approximates to $\ell \in O(\sqrt[h]{h!L} + h)$ [22]. Notably, $CW_{1,L}(x)$ yields the one-hot encoding of x.

Constant-Weight Equality Operator Typically, the bitlength ℓ in constant-weight codes is higher than $\log_2 L$ in the binary representation. However, constant-weight codes support oblivious equality checks of a low multiplicative depth [22]. Precisely, the equality check for $\mathbf{a} = CW_{h,\ell}(a)$ and $\mathbf{b} = CW_{h,\ell}(b)$ can be achieved by evaluating

$$\begin{aligned} h' &\coloneqq \sum_{i=1}^{\ell} \mathbf{a}[i] \cdot \mathbf{b}[i] \\ \mathsf{EQ}(a,b) &= \frac{1}{h!} \prod_{i=0}^{h-1} (h' - i), \end{aligned} \tag{1}$$

where the multiplicative depth is $1 + \lceil \log_2 h \rceil$ and the number of multiplications is $\ell + h - 1$.

Range Cover Comparison (RCC) Operator This constant-weight equality operator can furthermore be combined with a range cover representation [17,25] to obtain a low-depth comparator, as proposed by Mahdavi et al. in Level Up [23].

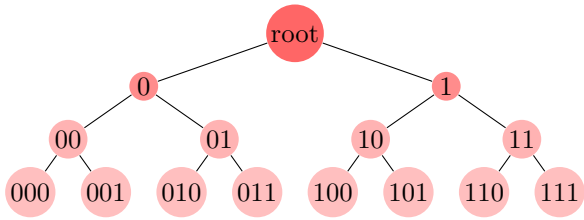


Figure 2: A binary interval tree containing $[0, 7]$. For example, the point encoding of the number 5 is $PE(5) = \{1, 10, 101\}$ and the range cover of $[1, 7]$ is $RC(1, 7) = \{1, 01, 001\}$.

Precisely, given $a, b \in [0, 2^s - 1]$, computing

$$\mathsf{GT}(a, b) = \begin{cases} 1, & \text{if } a > b \\ 0, & \text{otherwise} \end{cases}$$

is equivalent to checking whether the point a lies in the range $[b+1, 2^s - 1]$, i.e.

$$\mathsf{GT}(a, b) = \mathbb{1}_{a \in [b+1, 2^s - 1]}.$$

This leads to the following definition of an *interval tree* where points and ranges can be efficiently represented, as visualized in Figure 2.

Definition 2.1 (Adapted from [23, 25])**.** Let T be a binary interval tree whose leaf nodes contain elements in $[0, 2^s - 1]$. A *range cover* $RC(b + 1, 2^s - 1)$ contains the set of nodes in T such that (1) it contains at most one node in each level (2) its set of children at the leaf level is exactly $[b+1, 2^s - 1]$. A *point encoding* $PE(a)$ contains the set of nodes from leaf a to the root (except the root itself).

As observed in [25], if $a \notin [b + 1, 2^s - 1]$, then $RC(b + 1, 2^s - 1) \cap PE(a) = \emptyset$; otherwise, they will intersect at one and only one node. As $RC(b + 1, 2^s - 1)$ contains at most s elements (one node at each level), this comparison contains at most s equality checks of i bits for $i = 1, 2, \ldots, s$, i.e.

$$\mathsf{GT}(a, b) = \sum_{i=1}^{s} \mathsf{EQ}\big(RC(b + 1, 2^s - 1)[i], PE(a)[i]\big), \tag{2}$$

assuming $RC(b + 1, 2^s - 1)[i]$ has i digits. In Level Up [23], the s numbers in range cover are encoded using $CW_{h,\ell}(\cdot)$ where the weight h is small (such as 2 or 4), and the ℓ is the lowest number satisfying $\binom{\ell}{h} \geq 2^s$. Then their equality checks are performed using Equation (1). As such, this comparator contains $s \cdot (\ell + h - 1)$ multiplications in multiplicative depth $1 + \lceil \log_2 h \rceil$.

Folklore Bit-Wise Comparator The folklore comparator compares the binary representations of two numbers *bit-by-bit* [15, 22, 23]. Precisely, bit-wise comparisons can be achieved with degree-2 polynomials, i.e. for $a, b \in \{0, 1\}$,

$$\theta_{EQ}(a, b) = 1 - (a - b)^2$$

$$\theta_{GT}(a, b) = (1 - a) \cdot b.$$

Then using recursion, Algorithm 5 compares two numbers of bit length s using $2s - 1$ multiplications, and the lowest multiplicative depth to realize this algorithm is $(1 + \log s)$.

Algorithm 5 Folklore bit-wise comparator

Input: $\mathbf{a} = BR(a)$,$\mathbf{b} = BR(b) \in \{0, 1\}^s$
Output: $\mathsf{GT}(a, b)$
1: **function** BitwiseComp($\mathbf{a}, \mathbf{b}$)
2: **if** $s = 1$ **then**
3: **return** $\theta_{GT}(\mathbf{a}[1], \mathbf{b}[1])$
4: **else**
5: **return** $\theta_{GT}(\mathbf{a}[1], \mathbf{b}[1]) + \theta_{EQ}(\mathbf{a}[1], \mathbf{b}[1]) \cdot$ BitwiseComp($\mathbf{a}[2, s], \mathbf{b}[2, s]$)
6: **end if**
7: **end function**

3 Batched Ciphertext-Plaintext Comparisons

In the batched PDTE, a client encrypts N feature vectors $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$ and queries about the inference results for each of them using the decision tree $\mathcal{T}$ with thresholds $\mathbf{y}$. In the SIMD evaluation of a decision node, the server homomorphically compares features $\{x_i^{(1)} \in \mathbf{x}^{(1)}, x_i^{(2)} \in \mathbf{x}^{(2)}, \ldots, x_i^{(N)} \in \mathbf{x}^{(N)}\}$ to a threshold value $y_i \in \mathbf{y}$. Since threshold values are stored in the server in plaintexts, this amounts to performing a batched ciphertext-plaintext comparison.

This section proposes two methods for batched ciphertext-plaintext comparisons with improved performance.

3.1 Batched ciphertext-plaintext RCC comparator

The RCC comparator for two numbers of s bits, as described in (2), contains at most s equality checks whose operands are of i bits for $i = 1, 2 \ldots, s$. In

Level Up, these equality checks are performed using the constant-weight operator in Equation (1). This allows comparing batched encrypted values to various plaintext values.

In our batched ciphertext-plaintext comparison using RCC, batched encrypted values are compared to the same plaintext value. For this scenario, we follow the procedure above and optimize a subcomponent, the constant-weight equality operator in Equation (1). This further leads to a distinct ciphertext packing method from Level Up, which improves the amortized communication and storage.

Ciphertext-Plaintext Constant-Weight Equality Operator Given $\mathbf{a} = CW_{h,\ell}(a)$ and $\mathbf{b} = CW_{h,\ell}(b)$, the equality operator in Equation (1) is data-oblivious to both a and b, demonstrating its suitability for ciphertext-ciphertext comparisons.

In the ciphertext-plaintext scenario, the equality check only needs to be data-oblivious to a. Therefore, Equation (1) can be further simplified into

$$\mathsf{EQ}(a, b) = \prod_{\mathbf{b}[i]=1} \mathbf{a}[i], \tag{3}$$

and its homomorphic evaluation requires $(h - 1)$ ciphertext-ciphertext multiplications in depth $\lceil \log_2 h \rceil$ and zero ciphertext-plaintext multiplications.

Our Ciphertext Packing Although the ciphertext packing method in Level Up naturally supports our batched ciphertext-plaintext RCC comparator, its storage and communication cost could be further improved, as pointed out in the Future Work section of [23]. In line with this, we introduce another ciphertext packing method, as depicted in Figure 3.

Precisely, let N denote the number of SIMD slots for given BFV parameters, our method allows to pack N values for one feature $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ into BFV ciphertexts. Subsequently, these s-bit values are compared to a plaintext threshold value y.

As explained in Section 2.5, comparing two values is equivalent to checking the intersection between the point encoding of one element and the range cover of the other. In our method, point encodings of features are encrypted and packed, and the range cover of the threshold y is in plaintext.

For each feature $x^{(i)}$, its point encoding $PE\left(x^{(i)}\right)$ is a length-s vector and the component $x^{(i)}_{(j)} = PE\left(x^{(i)}\right)[j]$ contains j bits where $j = 1, \ldots, s$. Each $x^{(i)}_{(j)}$ is further encoded using constant weight h_j into $CW_{h_j,\ell_j}\left(x^{(i)}_{(j)}\right)$ of length ℓ_j.

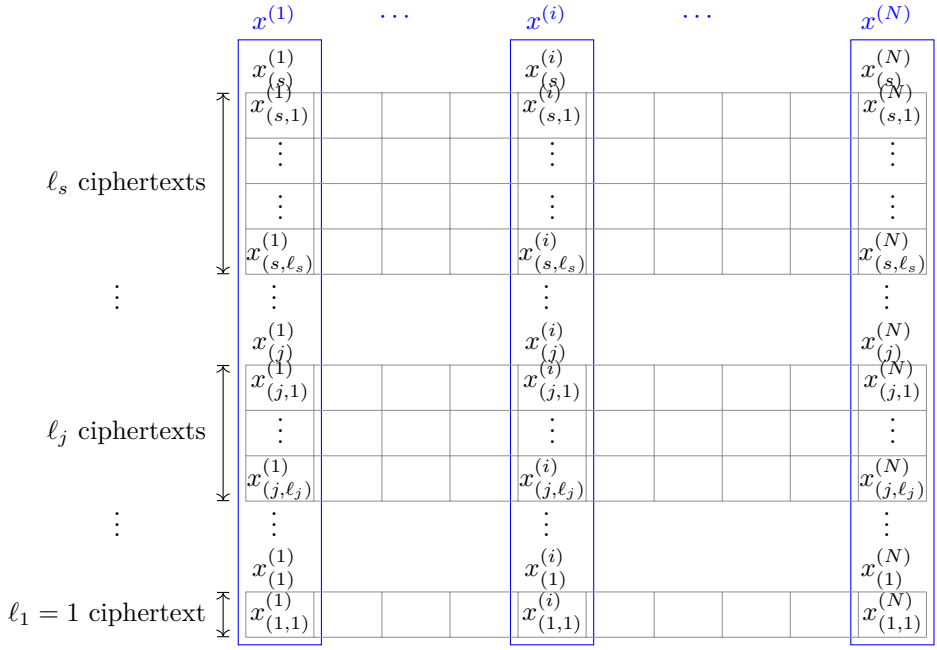


Figure 3: Our method of packing N values for one feature $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ of s bits into BFV ciphertexts, which will be compared to one plaintext threshold value y using our batched RCC comparator.

Since the bit-length of $x^{(i)}_{(j)}$ is independent of i and decreases as j decreases, the Hamming weight for encoding is also independent of i and $h_s = \max(h_j)$. The bit length ℓ_j is determined by the relation $\binom{\ell_j}{h_j} \geq 2^j$, which approximates to $\ell_j \in O(\sqrt[h_j]{h_j! 2^j} + h_j)$, as explained in Section 2.5. For simplicity, the ℓ_j bits in $CW_{h_j,\ell_j}\left(x^{(i)}_{(j)}\right)$ are denoted as $x^{(i)}_{(j,k)} = CW_{h_j,\ell_j}\left(x^{(i)}_{(j)}\right)[k]$ where $k = 1, \ldots, \ell_j$.

In practice, Hamming weight h_s are small numbers. For example, two common choices of h_s in the Level Up implementation are 2 and 4. Therefore, we choose $h_s = h_{s-1} = \cdots = h_{j'}$ for some small j', and the Hamming weight h_j steadily decreases with decreasing j until $h_1 = 1$. Therefore, the length $\ell_j \in O(\sqrt[h_j]{h_j! 2^j} + h_j)$ decreases exponentially with j. As such, the amortized storage for our ciphertext packing is

$$\frac{\ell_s + \ell_{s-1} + \ldots + \ell_1}{N} \ll \frac{s \cdot \ell_s}{N},$$

and the right-hand side (RHS) corresponds to the amortized storage for Level Up ciphertext packing.

Homomorphic Evaluation of our Batched RCC Comparator On the other hand, the range cover $RC(y\_range)$ determined by y contains maximum s numbers, each with bit precision ranging from 1 to s. Section 2.5 details y_range for the GT comparator, and for GE, LT and LE comparators, the y_range can be constructed similarly. Denote $RC(y\_range)[j]$ of j bits as y_j, which are encoded using constant weight h_j into $CW_{h_j,\ell_j}\left(y_{(j)}\right)$ with binary components $y_{(j,k)} = CW_{h_j,\ell_j}\left(y_{(j)}\right)[k]$ where $k = 1, \ldots, \ell_s$.

As such, our ciphertext-plaintext constant-weight equality operation gives

$$\mathsf{EQ}(x^{(i)}_{(j)}, y_{(j)}) = \prod_{y_{(j,k)}=1} x^{(i)}_{(j,k)}, \tag{4}$$

which contains $h_j - 1$ ciphertext-ciphertext multiplications in depth $\log_2 h_j$. Similar to Equation (2), the comparison result can be obtained from

$$\mathsf{COMP}(x^{(i)}, y) = \sum_{j=1}^{s} EQ(x^{(i)}_{(j)}, y_{(j)}) \tag{5}$$

where COMP is predetermined choice of GT,GE, LT or LE.

Overall, our batched ciphertext-plaintext RCC comparator requires

$$\frac{\sum_{j=1}^{s}(h_j - 1)}{N} < \frac{s \cdot (h_s - 1)}{N}$$

ciphertext-ciphertext multiplications at depth $\log_2 h_s$ and zero ciphertext-plaintext multiplications. The RHS corresponds to the number of ciphertext-ciphertext multiplications of the RCC comparator in Level Up, which also requires $\frac{s \cdot \ell_s}{N}$ ciphertext-plaintext multiplications.

3.2 Batched Ciphertext-Plaintext Constant-Weight Piece-Wise Comparator

Inspired by this bit-by-bit comparison in Algorithm 5, we propose a *piece-by-piece* comparator for constant-weight codes, which is only oblivious to one operand and is therefore suitable for ciphertext-plaintext comparisons.

Let $\mathbf{a} = CW_{h,\ell}(a)$ and $\mathbf{b} = CW_{h,\ell}(b)$, and suppose encryptions $\{\mathsf{Enc}(\mathbf{a}[i]), 1 < i \leq \ell\}$ and the plaintext $\mathbf{b}$ are given. The first piece in $\mathbf{a}$ is from its most significant bit (inclusive) to the position of the first one in $\mathbf{b}$ (exclusive).

If there is any number one in this first piece, then $\mathsf{GT}(a,b)=1$. This condition is checked by summing all elements in this piece to obtain a number $x \in \{0,1,\ldots,h\}$. Then evaluating the function $\theta_{GTZero}(x,h) = 1 - \frac{1}{h!}\prod_{i=1}^{h}(i-x)$ returns one if $x \in \{1,\ldots,h\}$ and zero if $x=0$.

Otherwise, if the first one in $\mathbf{a}$ has the same position as $\mathbf{b}$, we compare the code in lower digits piece-by-piece recursively. The complete algorithm is presented in Algorithm 6, and the minimum multiplicative depth to realize it is $\lceil \log_2((h+1)+(h-1+1)+\ldots+(2+1)+1)\rceil = \left\lceil \log_2\left(\frac{(h+4)(h-1)}{2}+1\right)\right\rceil$.

Algorithm 6 Constant-weight piece-wise comparator

Input: $\mathbf{a} = CW_{h,\ell}(a)$,$\mathbf{b} = CW_{h,\ell}(b) \in \{0,1\}^{\ell}$
Output: $\mathsf{GT}(a,b)$

function PIECEWISECOMP($\mathbf{a},\mathbf{b},h$)
  $\mathbf{c} \leftarrow [i \mid \mathbf{b}[i]=1]$ ▷ $\mathbf{c}$ is an ordered array of size h
  if $h=1$ **then**
    return $\sum_{i=1}^{\mathbf{c}[1]-1}\mathbf{a}[i]$
  else
    $\alpha \leftarrow \theta_{GTZero}(\sum_{i=1}^{\mathbf{c}[1]-1}\mathbf{a}[i],h)$
    return $\alpha + (1-\alpha)\cdot\mathbf{a}\big[\mathbf{c}[1]\big]\cdot$ PIECEWISECOMP($\mathbf{a}\big[\mathbf{c}[1]+1,\ell\big],\mathbf{b}\big[\mathbf{c}[1]+1,\ell\big],h-1$)
  end if
end function

The ciphertext packing strategy for the constant-weight piece-wise comparator is presented in Figure 4. Compared to the ciphertext packing for the RCC comparator in Figure 3, no point encoding is needed, hence the amortized storage $\frac{\ell}{N}$ is also lower for comparable choices of Hamming weight h_s and h.

3.3 Benchmarking Batched Ciphertext-Plaintext Comparisons

For the experiment, we assume a client sends ciphertexts corresponding to N values of s bits each, which will be compared to a plaintext value in the server. After the homomorphic evaluation, the client receives a ciphertext whose SIMD slots encode the N comparison results.

We consider four methods for such batched ciphertext-plaintext comparisons: 1) the RCC operator in [23] with the same plaintext values in all slots, 2) our batched RCC in Section 3.1, 3) the folklore bit-wise comparator with the same plaintext values in all slots and 4) our constant-weight piece-wise comparator in Section 3.2.

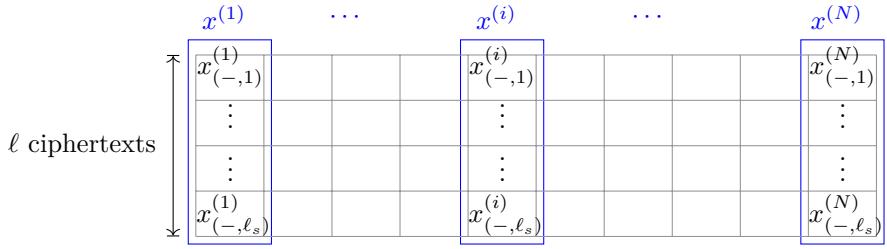


Figure 4: Our method of packing N values for one feature $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ of s bits into BFV ciphertexts, which will be compared to one plaintext threshold value y using the constant-weight piece-wise comparator. Each feature $x^{(i)}$ is encoded using constant weight h into $CW_{h,\ell}\left(x^{(i)}\right)$ of length ℓ, and its binary components $CW_{h,\ell}\left(x^{(i)}\right)[k]$ are denoted as $x^{(i)}_{(-,k)}$, with the first subscript indicating no point encoding is applied.

Table 2 presents their performance for input bitlength 8 and 16. Specifically, the performance of 1) and 3) are obtained from running the Level Up implementation[1], and 2) and 4) are implemented using Microsoft SEAL [24].

In summary, our methods 2) and 4) provide computation time ranges from 4.8× to over 72× faster than prior methods 1) and 3) while maintaining comparable communication costs and multiplicative depth.

4 Tree Traversal Methods

From homomorphic evaluations of decision nodes and tree traversal, the server obtains an encrypted value $\mathsf{Enc}(r_j)$ for each leaf j, where $1 \leq j \leq m+1$. This value r_j indicates whether the leaf j is the output leaf, and we denote the array of r_j as $\mathbf{r}$.

In Path Conjugation, the result vector $\mathbf{r}_c$ is a unit vector whose inner product with $\mathbf{v}$ yields the predicted classification. The encrypted classification value is sent to the client. However, this unit vector in Path Conjugation comes with a price: it requires an expensive RLWEtoRGSW conversion [11] procedure in TFHE. Let w denote the multiplicative depth of a ciphertext-plaintext comparison algorithm in BFV, instantiating Path Conjugation in BFV leads to a high multiplicative depth $\mathcal{O}(d \cdot w)$.

On the other hand, SumPath returns the encryption of $\mathbf{r}_s$ to the client, whose value is zero for the output leaf and non-zero otherwise. The client decrypts,

[1] `https://github.com/RasoulAM/private-decision-tree-evaluation`

Table 2: Performance of different batched ciphertext-plaintext comparators in BFV with $N = 2^{14}$ and $t = 65537$. The multiplication depth refers to the depth of ciphertext-ciphertext multiplications. Non-applicable parameters are denoted as $\perp$.

		Amortized Computational Time		Amortized Client-to-server Communication Cost		Multiplicative Depth	
		$s = 8$	$s = 16$	$s = 8$	$s = 16$	$s = 8$	$s = 16$
RCC [23]	$h = 2$	245 μs	8340 μs	45 kb	1342 kb	1	1
	$h = 4$	188 μs	1526 μs	20 kb	136 kb	2	2
	$h = 8$	$\perp$	1308 μs	$\perp$	70 kb	3	3
Our batched RCC	$h_s = 2$	19 μs	41 μs	11 kb	180 kb	1	1
	$h_s = 4$	39 μs	82 μs	8 kb	38 kb	2	2
Folklore bit-wise [23]	$\perp$	457 μs	1982 μs	1 kb	3 kb	3	4
Constant-weight piece-wise	$h = 2$	10 μs	18 μs	3 kb	52 kb	2	2
	$h = 4$	37 μs	39 μs	1 kb	5 kb	4	4

obtains the index of the output leaf, and looks up its corresponding classification value. Its instantiation in both TFHE and BFV is fast and straightforward, and its low multiplicative depth $\mathcal{O}(w)$ enables PDTE using practical BFV parameters.

However, compared to Path Conjugation, the server-to-client communication in SumPath is $\mathcal{O}(m)$ larger. Besides, integrating decision trees into a tree ensemble [7,16] is a widely used technique to improve prediction accuracy. Since SumPath requires the client to look up the classification value for every decision tree, its applicability for private evaluations of tree ensembles is strongly limited.

Then a natural question is whether there is a tree traversal method that not only achieves low multiplicative depth but also yields a unit result vector with reasonable computation costs. This leads to our adapted SumPath method.

4.1 Our Adapted SumPath Method

The edge cost computation in SumPath is visualized in Figure 1b. Our adaption of SumPath follows from this observation: when the parameter r in Figure 1b is set to be 1 for all decision nodes, the path cost of every leaf counts the number of unsatisfied conditions in the path from the root to that leaf. As such, the path cost of the desired leaf equals zero, and the path costs of all the other leaves are in $\{1, \ldots, d-1\}$.

Since the function

$$\theta_{EQZero}(x, d) = \frac{1}{(d-1)!} \prod_{i=1}^{d-1} (i - x)$$

maps zero to one and any elements in $\{1, \ldots, d-1\}$ to zero[2], evaluating $\theta_{EQZero}(\cdot, d)$ on the path cost of each leaf maps the result vector $\mathbf{r}_s$ in SumPath into the desired unit vector denoted as $\mathbf{r}_{as}$.

As such, using our adapted SumPath for tree traversal leads to multiplicative depth $\mathcal{O}(w + \log_2 d)$ for PDTE, where w is the multiplicative depth for one homomorphic comparison in BFV.

Optimization: Tree Truncation Since the server knows the classification values in leaves $\mathbf{v}$ in plaintext, the procedure above can be optimized. Precisely, in the inner product $\mathbf{r}_{as} \cdot \mathbf{v}$, the components in $\mathbf{r}_{as}$ that correspond to zero labels do not contribute. Therefore, these leaves can be *truncated* from the decision tree, obviating the need to compute their path costs and evaluations of $\theta_{EQZero}(\cdot, d)$. The visualization of the tree truncation technique is included in Appendix A.

By renaming the most abundant label to zero, at least $\frac{1}{k}$ leaves have zero classification values and can be truncated. Moreover, badly trained models may contain decision nodes whose children leaves both have zero classification values. These nodes can also be truncated without impacting the final output.

[2]This function is also used in the concurrent work [26] to integrate decision trees into random forests.

5 Batched Private Decision Tree Evaluation

5.1 Security Model

Our work considers the client/server scenario, where a cloud server holds a pre-trained decision tree model $\mathcal{T}$ and a client holds multiple input feature vectors $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$ and wants to know the inference result with $\mathcal{T}$ for each of them. The goal is to protect input privacy such that the server would not be able to learn the clients' input values. Guaranteeing model privacy is out of the scope of this paper: a client with enough resources may be able to reverse-engineer the server-side cloud model after a given number of queries. Moreover, the protocol should be non-interactive to allow full outsourcing computations to the server.

Our threat model is similar to prior works, where the server is an honest-but-curious adversary. This implies that the server always follows the protocol strictly but may attempt to deduce information from the client's inputs.

5.2 Protocol

For setup, the server performs a tree truncation to $\mathcal{T}$ to get $Trun(\mathcal{T})$ and receives the necessary keys (e.g. relinearization keys) from the client. Under the standard circular security assumption, these keys do not leak information about the client's secret key. Our batched PDTE protocol is as follows.

1. The client sends encryptions of N input feature vectors to the server.
2. For j-th decision node where $1 \leq j \leq m$, the server homomorphically compares encryptions of N feature values and the plaintext threshold y_j in an SIMD manner. Section 3 provides two methods for such comparisons. The output $\mathsf{Enc}(\mathbf{b}_j)$ is a ciphertext encoding N binary numbers in its SIMD slots.
3. The server performs adapted $\mathsf{SumPath}$ to $\{\mathsf{Enc}(\mathbf{b}_j)\}_{j=1,\ldots,m}$ in $\mathcal{T}$, whose homomorphic inner product with $\mathbf{v}$ gives a ciphertext. The SIMD slots of this ciphertext are N classification values
4. The client decrypts this ciphertext to obtain these N classification values, one for each feature vector.

Security of Batched PDTE Clients' feature vectors, comparison results of decision nodes, and classification labels are all encrypted using BFV schemes with 128-bit security parameters. Its semantic security (IND-CPA) ensures the server (honest but curious) cannot infer corresponding plaintexts, preserving the client's privacy.

Table 3: Characteristics of UCI datasets used in our evaluation, where # **Decision Nodes/Leaves (before|after)** gives the number of decision nodes/leaves in each model before and after tree truncation if that number changes.

	# **Features** n	**Depth** d	# **Decision Nodes** m	# **Leaves**
Breast	30	7	15	16|8
Heart	13	3	4	5|3
Spam	57	11	108|107	109|52
Steel	33	5	5	6|1

5.3 Implementation and Performance

We implement two versions of our batched PDTE protocol using different comparators: BPDTE_RCC using our batched RCC comparators in Section 3.1, and BPDTE_CW using constant-weight piece-wise comparators in Section 3.2. These protocols are evaluated on UCI datasets [13] and compared with the state-of-art prior works [11, 23].

Experimental Details We use the same UCI datasets as in prior works: Breast, Heart, Spam and Steel. Furthermore, we apply a tree truncation procedure to reduce server computation without influencing the output. Table 3 presents the key properties of these datasets. Our implementation uses the Microsoft SEAL library (v4.1.1) [24], which supports BFV in the SIMD manner and it is also used by Level Up. For SortingHat and Level Up, we use the implementation provided by the authors. Experiments are conducted on a desktop with an Intel Core i7-13700 CPU and 32GB of RAM using a single thread.

Results and Discussion We compare our batched PDTE protocol with prior works in terms of amortized server computation time including comparisons and tree traversals, and amortized query size, *i.e.*, the client-to-server communications. The amortized server-to-client communication is lower than $1kb$ for all protocols and therefore not listed.

Table 4: Amortized performance of different PDTE protocols with batch size 16384 and input feature bit-length $s = 11$, where SortingHat uses TFHE with $N = 2^{11}$, Level Up uses BFV with $N = 2^{13}$ and BPDTE_CW with $h = 2$ uses BFV with $N = 2^{14}$

	SortingHat ($s = 11$)			**Level Up** ($s = 11, h = 4$)			**BPDTE_CW** ($s = 11, h = 2$)		
	Comparison	Traversal	Query Size	Comparison	Traversal	Query Size	Comparison	Traversal	Query Size
Breast	7 ms	178 ms	960 kb	139 μs	117 μs	310 kb	9 μs	139 μs	90 kb
	Total: 185 ms			Total: 256 μs			Total: 148 μs		
Heart	3 ms	47 ms	416 kb	156 μs	25 μs	135 kb	3 μs	18 μs	117 kb
	Total: 50 ms			Total: 181 μs			Total: 21 μs		
Spam	69 ms	1283 ms	1824 kb	378 μs	1089 μs	589 kb	78 μs	1326 μs	513 kb
	Total: 1352 ms			Total: 1467 μs			Total: 1404 μs		
Steel	3 ms	59 ms	1056 kb	125 μs	34 μs	341 kb	4 μs	12 μs	297 kb
	Total: 62 ms			Total: 159 μs			Total: 16 μs		

Table 4 and Table 5 compare the amortized performance of different PDTE protocols with batch size 16384 for input feature bit-length $s = 11$ and $s = 16$, respectively. This corresponds to the scenario where the client sends encryptions of 16384 feature vectors $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(16384)}\}$ and wants to know the classification output for each of them. The large batch size is useful in practice, for example, when a bank outsources a credit-scoring decision tree and needs to evaluate numerous applicants securely. For completeness, we also compare PDTE protocols with different batch sizes in Appendix B. Since the maximum bit-length supported by SortingHat is 11, SortingHat is not listed in Table 5. Moreover, for $s = 11$, BPDTE_CW outperforms BPDTE_RCC in both communication and computation, hence BPDTE_RCC is not listed in Table 4.

As for BFV parameters, BPDTE_CW and BPDTE_RCC use larger parameters than Level Up to provide higher depth. Precisely, Level Up uses SumPath, where the amortized response of the server is $\mathcal{O}(m)$ and the client needs to look

up classification values in a table. On the other hand, BPDTE_CW and BPDTE_RCC use the Adapted SumPath method, where the amortized response of the server is $\mathcal{O}(1)$ at the cost of $\mathcal{O}(\log_2 d)$ multiplicative depth.

As a remark, it is possible to combine our batched ciphertext-plaintext comparators with SumPath for PDTE, which requires the same BFV parameters as in Level Up and therefore attains better communication and computational performance. However, with this $\mathcal{O}(m)$ response, the client needs to perform a table lookup to obtain classification values and the extension to tree ensembles is restricted.

In summary, with batch size 16384, SortingHat is about 10^3 slower than those supporting SIMD operations. Compared to Level Up, BPDTE_RCC and BPDTE_CW are 1.5× to 17× faster overall and have comparable query sizes. For large precision (e.g. $s = 16$), BPDTE_RCC provides slightly lower query sizes than BPDTE_CW (e.g. 0.73×) at the expense of slightly higher computation costs (e.g. $1.4 - 2\times$).

Table 5: Amortized performance of different PDTE protocols with batch size 16384 and input feature bit-length $s = 16$, where Level Up uses BFV with $N = 2^{13}$, BPDTE_RCC with $h_s = 4$ and and BPDTE_CW with $h = 2$ both use BFV with $N = 2^{14}$

	Level Up ($s = 16, h = 4$)			**BPDTE_RCC** ($s = 16, h_s = 4$)			**BPDTE_CW** ($s = 16, h = 2$)		
	Comparison	*Traversal*	*Query Size*	*Comparison*	*Traversal*	*Query Size*	*Comparison*	*Traversal*	*Query Size*
Breast	583 μs	159 μs	968 kb	75 μs	139 μs	1140 kb	17 μs	138 μs	1560 kb
	Total: 742 μs			Total: 214 μs			Total: 155 μs		
Heart	309 μs	34 μs	420 kb	20 μs	18 μs	494 kb	4 μs	18 μs	676 kb
	Total: 343 μs			Total: 38 μs			Total: 22 μs		
Spam	1857 μs	1595 μs	1839 kb	536 μs	1501 μs	2166 kb	118 μs	1489 μs	2964 kb
	Total: 3452 μs			Total: 2037 μs			Total: 1607 μs		
Steel	262 μs	46 μs	1065 kb	25 μs	12 μs	1254 kb	6 μs	12 μs	1716 kb
	Total: 308 μs			Total: 37 μs			Total: 18 μs		

6 Conclusion

In this work, we proposed two batched ciphertext-plaintext comparisons, our batched RCC comparator and the constant-weight piece-wise comparator. Compared to directly applying previous methods to this scenario, our evaluation of these comparison operators shows a speedup of up to 72× for 16-bit numbers while maintaining comparable communication costs and multiplicative depth.

These batched ciphertext-plaintext comparisons, together with our adapted SumPath tree traversal method, lead to two non-interactive PDTE protocols, BPDTE_RCC and BPDTE_CW. Compared to the prior state-of-art [23], these protocols not only avoid the client looking up classification values in a table but also demonstrate an enhanced performance of up to 17× in batch size 16384.

Acknowledgments

This work is partially supported by the Research Council KU Leuven under the grant C24/18/049, CyberSecurity Research Flanders with reference number VR20192203. The authors would also like to thank Dr. Svetla Nikova for the valuable discussions and Prof. Frederik Vercauteren for helpful feedback on this paper.

References

[1] Sofiane Azogagh, Victor Delfour, Sébastien Gambs, and Marc-Olivier Killijian. PROBONITE: private one-branch-only non-interactive decision tree evaluation. In Michael Brenner, Anamaria Costache, and Kurt Rohloff, editors, *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Los Angeles, CA, USA, 7 November 2022*, pages 23–33. ACM, 2022.

[2] Jianli Bai, Xiangfu Song, Shujie Cui, Ee-Chien Chang, and Giovanni Russello. Scalable private decision tree evaluation with sublinear communication. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22*, pages 843–857. ACM Press, May / June 2022.

[3] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 188–215. Springer, Cham, December 2022.

[4] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS 2015*. The Internet Society, February 2015.

[5] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Berlin, Heidelberg, August 2012.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[7] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[8] Justin Brickell, Donald E. Porter, Vitaly Shmatikov, and Emmett Witchel. Privacy-preserving remote diagnostics. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 498–507. ACM Press, October 2007.

[9] Ching-Chin Chern, Weng-U. Lei, Kwei-Long Huang, and Shu-Yi Chen. A decision tree classifier for credit assessment problems in big data environments. *Inf. Syst. E Bus. Manag.*, 19(1):363–386, 2021.

[10] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Berlin, Heidelberg, December 2016.

[11] Kelong Cong, Debajyoti Das, Jeongeun Park, and Hilder V. L. Pereira. SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 563–577. ACM Press, November 2022.

[12] Kelong Cong, Robin Geelen, Jiayi Kang, and Jeongeun Park. Revisiting oblivious top-k selection with applications to secure k-nn classification. Cryptology ePrint Archive, Report 2023/852, 2023.

[13] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. http: //archive.ics.uci.edu/ml.

[14] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

[15] Yidi Hao, Baodong Qin, and Yitian Sun. Privacy-preserving decision-tree evaluation with low complexity for communication. *Sensors*, 23(5):2624, 2023.

[16] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition.* Springer Series in Statistics. Springer, 2009.

[17] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

[18] Ágnes Kiss, Masoud Naderpour, Jian Liu, N. Asokan, and Thomas Schneider. SoK: Modular and efficient private decision tree evaluation. *PoPETs*, 2019(2):187–208, April 2019.

[19] Wan'an Liu, Hong Fan, and Meng Xia. Credit scoring based on tree-enhanced gradient boosting decision trees. *Expert Syst. Appl.*, 189:116034, 2022.

[20] Wen-jie Lu, Zhicong Huang, Qizhi Zhang, Yuchen Wang, and Cheng Hong. Squirrel: A scalable secure two-party computation framework for training gradient boosting decision tree. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 6435–6451. USENIX Association, 2023.

[21] Wenjie Lu, Jun-Jie Zhou, and Jun Sakuma. Non-interactive and output expressive private comparison from homomorphic encryption. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 67–74. ACM Press, April 2018.

[22] Rasoul Akhavan Mahdavi and Florian Kerschbaum. Constant-weight PIR: Single-round keyword PIR via constant-weight equality operators. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 1723–1740. USENIX Association, August 2022.

[23] Rasoul Akhavan Mahdavi, Haoyan Ni, Dimitry Linkov, and Florian Kerschbaum. Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 2945–2958. ACM Press, November 2023.

[24] Microsoft SEAL (release 4.1). `https://github.com/Microsoft/SEAL`, January 2023. Microsoft Research, Redmond, WA.

[25] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy*, pages 350–364. IEEE Computer Society Press, May 2007.

[26] Hojune Shin, Jina Choi, Dain Lee, Kyoungok Kim, and Younho Lee. Fully homomorphic training and inference on binary decision tree and random forest. Cryptology ePrint Archive, Report 2024/529, 2024.

[27] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *DCC*, 71(1):57–81, 2014.

[28] Raymond K. H. Tai, Jack P. K. Ma, Yongjun Zhao, and Sherman S. M. Chow. Privacy-preserving decision trees evaluation via linear functions. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part II*, volume 10493 of *LNCS*, pages 494–512. Springer, Cham, September 2017.

[29] Anselme Tueno, Yordan Boev, and Florian Kerschbaum. Non-interactive private decision tree evaluation. In Anoop Singhal and Jaideep Vaidya, editors, *Data and Applications Security and Privacy XXXIV - 34th Annual IFIP WG 11.3 Conference, DBSec 2020, Regensburg, Germany, June 25-26, 2020, Proceedings*, volume 12122 of *Lecture Notes in Computer Science*, pages 174–194. Springer, 2020.

[30] Anselme Tueno, Florian Kerschbaum, and Stefan Katzenbeisser. Private evaluation of decision trees using sublinear cost. *PoPETs*, 2019(1):266–286, January 2019.

[31] David J. Wu, Tony Feng, Michael Naehrig, and Kristin E. Lauter. Privately evaluating decision trees and random forests. *PoPETs*, 2016(4):335–355, October 2016.

[32] Defu Zhang, Xiyue Zhou, Stephen C. H. Leung, and Jiemin Zheng. Vertical bagging decision trees model for credit scoring. *Expert Syst. Appl.*, 37(12):7838–7843, 2010.

[33] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. Cerebro: A platform for multi-party cryptographic collaborative learning. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 2723–2740. USENIX Association, August 2021.

[34] Martin Zuber and Renaud Sirdey. Efficient homomorphic evaluation of k-NN classifiers. *PoPETs*, 2021(2):111–129, April 2021.

A Tree Truncation

For the decision tree in Figure 5, applying the tree truncation gives Figure 6.

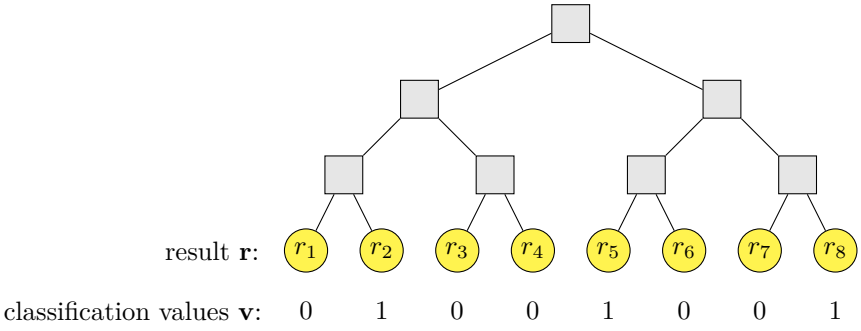


Figure 5: An example decision tree in depth $d = 3$ with $m = 7$ decision nodes, $m+1 = 8$ leaves and $k = 2$ classification values. In its PDTE, the server obtains an encrypted value $\mathsf{Enc}(r_j)$ for each leaf j, where $1 \leq j \leq 8$

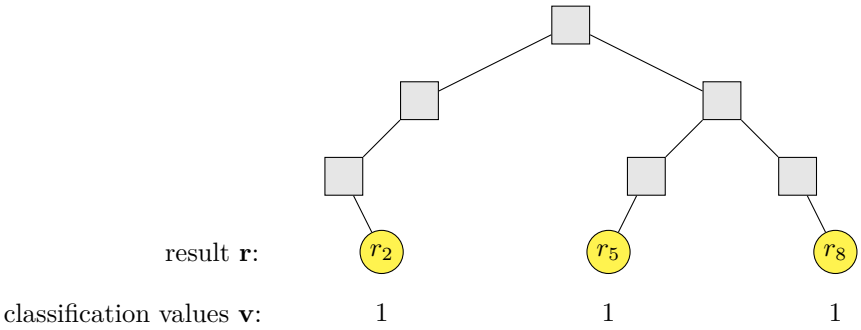


Figure 6: The truncated decision tree in Figure 5, where the tree contains 6 decision nodes instead of 7 and the result vector contains 3 elements instead of 8.

B Performance comparison in different batch sizes

In batched PDTE with batch size a, the client sends encryptions of a feature vectors $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(a)}\}$ and wants to know the classification output for each of them. This appendix discusses PDTE running times for a fixed decision tree $\mathcal{T}$ but different a.

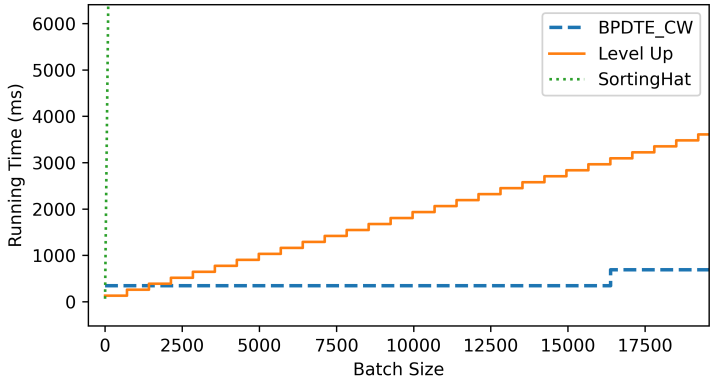


Figure 7: Computation time (comparison+tree traversal) for the **Heart** model of different PDTE protocols with input feature length $s = 11$ and different batch sizes in x-axis.

For SortingHat with $N = 2^{11}$, Level Up with $N = 2^{13}$ and $h = 4$, BPDTE_CW with $N = 2^{14}$ and $h = 2$ (*i.e.* PDTEs in Table 4), Figure 7 compares their running times for the **Heart** model with 11-bit feature precision. Since SortingHat does not support SIMD packing, the total running time scales linearly with a, assuming their FHE parameters are fixed. In Level Up, components of 712 features are packed in one ciphertext in their implementation, hence the total running time is a step function with step 712. In BPDTE_CW, components of 2^{14} features are packed in one ciphertext, hence the total running time is a step function with step 2^{14}.

As shown in Figure 7, for PDTEs of the **Heart** model, SortingHat is the fastest for batch sizes from 1 to ~ 10, Level Up is the fastest for batch sizes from ~ 10 to ~ 2100, and BPDTE_CW is the fastest for batch sizes larger than ~ 2100. PDTEs of other models attain similar behaviour, but intersection points for the optimal PDTE will differ.

Chapter 9

SQUiD: ultra-secure storage and analysis of genetic data for the advancement of precision medicine

Publication Data

Blindenbach, J., Kang, J., Hong, S., Karam, C., Lehner, T., and Gürsoy, G. SQUiD: ultra-secure storage and analysis of genetic data for the advancement of precision medicine. *Genome Biology 25*, 314 (2024)

Contribution

I am a co-first author of this work together with J. Blindenbach and S. Hong. I designed the SQUiD protocol in discussion with other co-authors.

SQUiD: ultra-secure storage and analysis of genetic data for the advancement of precision medicine

Jacob Blindenbach[1,2,3,5], Jiayi Kang[4,5], Seungwan Hong[2,3,5,6], Caline Karam[3], Thomas Lehner[3], and Gamze Gürsoy[2,3,1,6]

[1] Department of Computer Science, Columbia University
[2] Department of Biomedical Informatics, Columbia University
[3] New York Genome Center
[4] COSIC, KU Leuven
[5] These authors contributed equally
[6] Corresponding authors

Abstract. Cloud computing allows storing the ever-growing genotype-phenotype datasets crucial for precision medicine. Due to the sensitive nature of this data and varied laws and regulations, additional security measures are needed to ensure data privacy. We develop SQUiD, a **s**ecure **qu**eryable **d**atabase for storing and analyzing genotype-phenotype data. SQUiD allows storage and secure querying of data in a low-security, low-cost public cloud using homomorphic encryption in a multi-client setting. We demonstrate SQUiD's practical usability and scalability using synthetic and UK Biobank data.

1 Background

Precision medicine aims to tailor medical care to the characteristics of an individual's unique genetic makeup, lifestyle, and environment. This approach has garnered considerable attention worldwide due to its potential to enhance patient outcomes and mitigate healthcare expenses [36]. But, several significant obstacles impede the realization of the full potential of precision medicine. One such challenge is the need for extensive and diverse patient genotype-phenotype datasets in order to advance the diagnosis and treatment of future patients [76]. However, this need for large amounts of data is often in conflict with the need to protect patient privacy [6]. This challenge is further complicated by the heterogeneous regulatory landscape governing privacy protection, with varying definitions and practices across different jurisdictions (*e.g.*, General Data Protection Regulation [GDPR] in Europe *vs.* frameworks in USA) [30, 74].

Furthermore, individual hospitals and institutions maintain their own policies due to the prevalence of health data breaches and privacy attacks.

Genomic data plays a pivotal role in precision medicine research, enabling the customization of medical care based on specific genetic variants, biomarkers, and inherited traits. Thus, there is a surge in data generation, which has challenged the ability of local servers to accommodate the rapid growth of data size and increased computational requirements [73]. Therefore, there is a pressing need and significant push towards cloud computing. However, this exacerbates the concerns about the privacy and prohibitions on use of personal data due to local, global, and/or institutional privacy policies. For example, with the introduction of the GDPR in Europe, the storage of genomic and related data in the cloud has become more stringent with the requirement of appropriate security measures in place such as encryption. Starting from 2023, many states in the US (California, Connecticut, Colorado, Utah, and Virginia) are entering a new GDPR-like privacy era that will have similar requirements about storing genomic and related data in the cloud [4]. Yet, the current state of privacy preservation through laws and institutional policies is fraught with instability and unpredictability, which poses significant challenges to the research community. If the data is kept in the encrypted form in cloud servers, then researchers, who are approved for access, need to download large quantities of data locally and decrypt them to perform analysis, which defeats the purpose of outsourcing the storage to the cloud. This situation creates additional hurdles for scientists, especially when attempting to combine multiple data sets to gain statistical power. Furthermore, it creates significant delays in research and requires large amounts of resources, which impedes the democratization of data access. As a result, advances in medicine will significantly be impacted if new privacy-preserving frameworks that comply with laws and policies are not developed and implemented.

Homomorphic encryption (HE) is one of the cryptographic tools that enables direct computations of functions on encrypted data in the public cloud. Homomorphic encryption has emerged as a useful approach to keep the data secure at rest, at transit, and during analysis. But, this approach also has thus far presented severe bottlenecks in its applicability, scalability, and performance [2, 55]. However, recent advances in algorithm designs and computing power have enabled an increase in the use of homomorphic encryption in genomics. For example, it has been shown that privacy-enhancing genome-wide association studies (GWAS) can be possible [13, 47, 68, 77]. It has also been shown that secure genotype imputation is feasible and scalable using homomorphic encryption [23,39,78]. Homomorphic encryption was also used for genomic variant querying [22], regression analysis for rare disease variants [75], and inference using genetic variants in machine learning applications [66]. These methods have added tremendous algorithmic advances to the field and paved the

way for more practical privacy-preserving analysis of genomes. However, their use in genotype-phenotype database settings has been limited. This is primarily attributed to two factors. Firstly, homomorphic encryption relies on public key cryptography, which is designed for client-server scenarios where the client owns the dataset and delegates computation to the cloud. However, in the context of genotype-phenotype databases, the data owner encrypts the data while multiple researchers access and analyze it. Secondly, the computational burden associated with homomorphic encryption makes it infeasible for applications involving large sample sizes. Both the storage size of encrypted data (*i.e.*, ciphertexts) and the computation times for homomorphic encryption are several orders of magnitude greater than those for the original plaintexts [67].

Here, we developed **S**ecure **Qu**ery Protocols for Genotype-Phenotype **D**atabases (**SQU**i**D**), a scalable framework designed to store and query genotype-phenotype databases in an ultra-secure cloud-based setting using homomorphic encryption. In our approach, we incorporate several key components: a ciphertext packing storage method to minimize the required storage space for encrypted data, a set of optimizations we developed to reduce query processing time, and an innovative cryptographic primitive (public key-switching) to enable homomorphic encryption for multiple users. We demonstrate that SQUiD is capable of efficiently executing various types of queries on large scale genotype-phenotype datasets, all the while maintaining the encryption of the data in the cloud. Specifically, it can perform tasks such as counting the number of patients in a filtered cohort, computing the Minor Allele Frequency (MAF) of genetic variants in a cohort, calculating Polygenic Risk Scores (PRS) for patients, and generating a cohort of genetically similar patients in remarkably short timeframes. Our findings highlight the potential of SQUiD as a valuable tool for secure, timely, and efficient analysis and interpretation of genetic and phenotypic data. At a time when data breaches are becoming increasingly common in healthcare settings, where data is a commodity, SQUiD provides a key resource to safeguarding patient privacy and enabling data providers to adhere to evolving laws and regulations, while ensuring the democratization of data.

2 Results

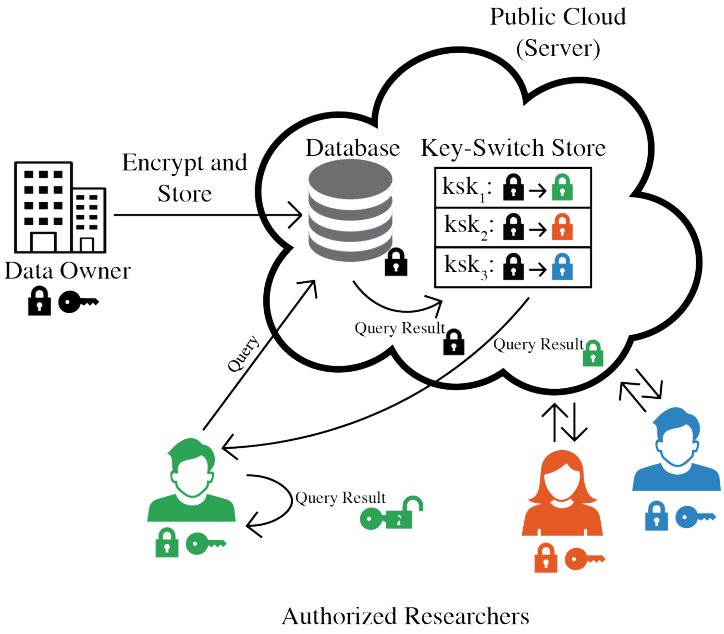


Figure 1: An architecture overview of SQUiD. Initially, the data owner uploads their encrypted genotype and phenotype data to the public cloud. Within the cloud, only authorized researchers are permitted to securely query the data. Authorization is granted through possession of a key-switching key, which is stored in the key-switching store. When a researcher initiates a query on the databases, the database responds by encrypting the query result under the data owner's public key. Subsequently, the key-switching store transforms this encrypted result to be under the querier's key. The encrypted result is then sent back to the querier, who can decrypt it using their own secret key.

Our conceptual framework allows ultra-secure interactions with encrypted genotype-phenotype databases

Our conceptual framework is focused on solving real-world security challenges encountered in the storage and querying large-scale genotype-phenotype datasets. These challenges involve safeguarding the confidential information contained within such data from third party cloud providers and outside adversaries. Our framework is based on a scenario that involves three parties: the data owner, the researcher(s), and the public cloud. The data owner, who in many

cases could be an organization such as the National Institutes of Health (NIH), possesses a vast amount of genotype-phenotype data that can be used for various analyses. Due to the large size of this data and the limited computing power and resources, the data owner encrypts the data and stores it in the public cloud. Their role is limited to authenticating clients who have permission to access the encrypted data, and they do not participate in the computation phase. The client, typically a researcher, seeks to perform computations on the data to obtain results. However, due to the large size of the data and the limitations of their computing power, it is not feasible for them to download, decrypt, and analyze the data locally. The client, therefore, interacts with the encrypted data deposited to the cloud. An overview of each party's role in the architecture of SQUiD is visualized in Fig. 1.

The cloud server does not have knowledge of the information contained in the data because all data stored in the public cloud is encrypted. Computations are performed directly on this encrypted data *without decryption* using homomorphic encryption. In homomorphic encryption [18,19,31], data, referred to as plaintext, is encrypted into ciphertexts. Addition and multiplication can be performed on ciphertexts such that two ciphertexts can be added or multiplied to produce a new ciphertext, which can be decrypted to the sum or product of the corresponding plaintexts. In our scenario, the complex functions behind our queries are sequences of homomorphic additions and multiplications which the cloud server performs on the encrypted genotype-phenotype data. The outputs of these functions will remain encrypted and are only decrypted after the outputs are sent back to the client. Operations on encrypted data are possible because plaintexts and ciphertexts are expressed as polynomials in HE. The algebraic structures of these polynomials are exploited to enable the computation on encrypted data (see Supplementary Information for more details). Importantly, retrieving the plaintext polynomial from the ciphertext polynomials without the secret key is extremely difficult. This difficulty is equivalent to the difficulty of solving the Ring Learning with Errors (RLWE) problem, which is known to be computationally hard under suitable parameters [56].

Using homomorphic encryption with appropriate parameters ensures that the sensitive information is protected while computations are being performed and the output is provided to the client in the encrypted form. Fig. 2 describes the four key components of our framework: encrypted data storage, access authorization, query capabilities, and the API for user-friendly interactions with the database. Unfortunately, this scenario cannot be realized with traditional homomorphic encryption, which is based on a two-party (data owner and public cloud) system. In a traditional two-party system, the data owner encrypts the data with their public key and decrypts the results with their private key. Thus, the researcher cannot query and decrypt the results since they do not (and should not) have access to the data owner's private key. To overcome

this challenge, we adopted the established concept of the Proxy Re-encryption system [61] to develop a theoretical realization and practical implementation of it within the framework of homomorphic encryption. This adaptation, which we refer to as the public key-switching technique [12, 14, 45], enables secure multi-client queries on encrypted data without the need for exchanging secret keys, specifically addressing the needs of our application in the biomedical domain.

The public key-switching operation serves as a cryptographic primitive facilitating the conversion of ciphertexts encrypted under one secret key to ciphertexts encrypted under a second secret key, without the need to decrypt the ciphertexts to plaintexts or possess access to the second secret key. Precisely, in key-switching, the original ciphertext needs to be homomorphically decrypted within the ciphertext space of the second secret key, which requires a key-switching key ksk, *i.e.*, encryption of the first secret key under the second secret key. Since the entire key-switching procedure together with ksk occurs within an encrypted space, the underlying messages remain secure without knowledge of either the first or the second secret key. Moreover, in public key-switching, ksk is generated by encrypting the first secret key with a public key, which does not require knowledge of the second secret key. In our scenario, this means the data owner can use their secret key for this operation without needing to access any of the clients' secret keys. This capability holds immense value in establishing secure interactions with an encrypted database. For example, in SQUiD, the encrypted database can compute a researcher's query under the encryption of the owner's key, convert the computed result from an encryption under the owner's key to under the public key of the researcher, and send this encrypted result to the researcher. Importantly, this conversion takes place without the need to decrypt the query result or disclose any information about it to the cloud. The researcher can effectuate this conversion by solely providing their public key, thereby circumventing any security risks associated with sharing their secret key.

When granting access to a new researcher, both the data owner and the researcher collaborate to create a public key-switching key, which is subsequently added to the key-switching store in the public cloud (Fig. 2B). The key-switching store offers two significant advantages. Firstly, it allows the data owner to remain offline during any query, as the researcher exclusively interacts with the public cloud where the pre-calculated and stored public key-switching keys reside. Secondly, the data owner retains control over data access by managing the inclusion or exclusion of researchers' public key-switching keys within the store, thus ensuring the ability to govern data access. We show that performance overhead from generating a key-switching key and key-switching a ciphertext under the encryption of one key to another to be less than a second (Additional file 1: Fig. 9). The public key-switching key of each authorized

researcher is stored in a dedicated key-switching store that dynamically expands to accommodate the number of authorized researchers. We show that the extra storage required for the key-switching store is minimal, around 55 MB per researcher (see Additional file 1: Fig. 10 and Additional file 1: Supplementary Material for an explanation why it is larger than a regular key storage).

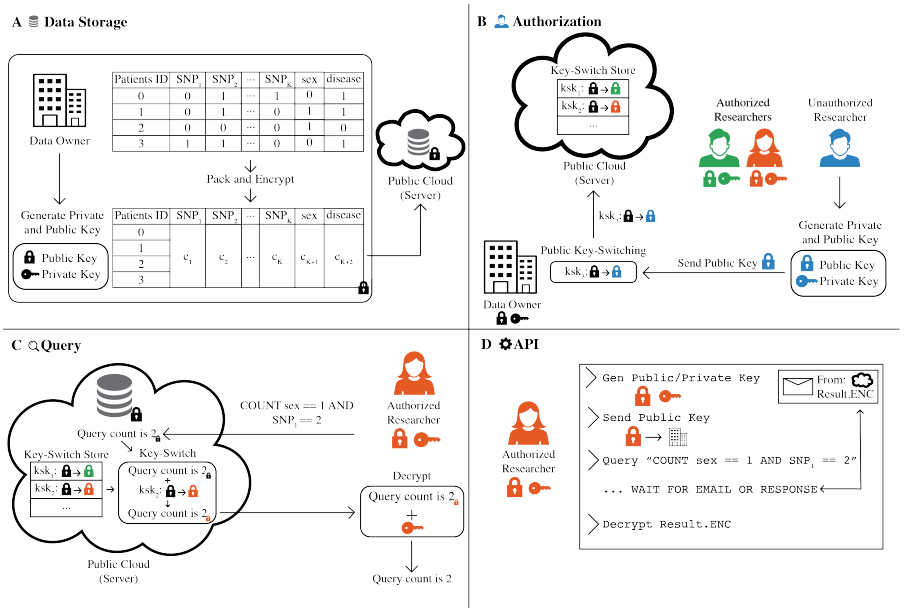


Figure 2: **(A) Data Storage.** The owner vertically packs their data to reduce storage costs, then encrypts their data with a public key, and then uploads the data to the public cloud. **(B) Authorization.** The onboarding process for a new researcher starts with the creation of their public and private key. The researcher sends their public key to the data owner for authorization. The owner authorizes the researcher by creating a key-switching key to switch the encryption of data to the researcher's key, and uploading this key to the key-switching key store in the public cloud. The data owner can revoke a researcher's access by removing the key-switching key from the store. **(C) Query.** An authorized researcher can submit one of four queries to the public cloud, which performs the necessary operations homomorphically on encrypted data under the data owner's key. The result is then re-encrypted under the researcher's public key and sent back for decryption. **(D) API.** We created a command-line API for researchers to use SQUiD easily. It generates a public and private key for the researcher, sends the public key to the data owner for authorization, sends queries to the server, and decrypts any encrypted results received via email or through a returned file.

Vertical packing allows efficient storage of encrypted large genotype-phenotype databases

We designed SQUiD to handle sensitive genotype-phenotype data from a large number of patients. SQUiD is specifically tailored to ingest data that has already undergone quality control and is stratified for population structure correction. Here we represent the data as a table with columns for basic attributes like age, sex, gender, etc; genotypes for single nucleotide polymorphisms (SNPs); and the phenotype or disease status of the patients, and rows for each patient in the database (Fig. 2A). While each entry in the table needs to be an integer for the homomorphic encryption libraries we used, continuous phenotypes can be discretized into integers via scaling (see Supplementary Information for more details). The encryption of this data introduces additional storage requirements compared to its original unencrypted form. Consequently, the storage expenses associated with storing large genotype-phenotype databases in their encrypted state can be substantial. In order to optimize storage within the SQUiD framework, we adopt a vertical packing approach for our data organization (see Methods). This method involves storing the genotypes of multiple patients for a single SNP within a single ciphertext. Vertical packing in homomorphic encryption is a method where multiple pieces of data are combined into a single, larger unit before encryption. This allows multiple calculations to be performed simultaneously on all the packed data within one operation, rather than processing each piece of data individually in a Single Instruction, Multiple Data (SIMD) fashion (see Methods for details) [69]. By vertically packing our data (Fig. 2A), we effectively reduce the number of ciphertexts necessary to accommodate a substantial volume of data, thereby minimizing the associated storage costs. Such packing still enables homomorphic updates (addition of new patients/SNPs/attributes) to the encrypted database without the need for decryption (see Methods for details).

We benchmarked the storage requirements of SQUiD on four different types of SNPs: ClinVar SNPs, Illumina Human1M-Duo v3.0 DNA Analysis BeadChip SNPs, Whole Exome Sequencing (WES) SNPs, and Whole Genome Sequencing (WGS) SNPs. These SNPs can be stored either at a per-chromosome level or genome-wide in SQUiD. Clinvar contains approximately 70,000 SNPs and Illumina BeadChip arrays contain approximately 1,072,820 SNPs. We estimated that around 8.2 million and 84 million SNPs would be observed in exomes and whole genomes at a population level, respectively, by using the data from 1000 Genomes Project [5]. We have benchmarked the packed storage of SQUiD against an unpacked homomorphic encryption storage, a storage encrypted with the industry standard AES-128-CBC, and a plaintext storage that stores SNPs as single bytes for the various SNP sets (Fig. 3). We found that the storage cost for SQUiD is 49,960x better than the unpacked homomorphic storage cost (Fig.

3). This efficiency is achieved because a single packed ciphertext in SQUiD can store data for up to 49,960 patients, whereas an unpacked ciphertext can only store data for one. Furthermore, vertical packing reduces the time for encryption by 49,960 fold compared to unpacked homomorphic encryption as fewer ciphertexts need to be encrypted (Additional file 1: Fig. 11). Vertical packing also improves query performance (Additional file 1: Fig. 12), because HE operations on ciphertexts compute these operations pairwise (*i.e.*, SIMD-like) on the vertically packed data. The query performance of the unpacked solution for the count, MAF, PRS, and similarity queries quickly becomes impractical and is outperformed by the packed solution in databases with as few as 10 patients for the count and MAF queries, and just 1 patient for the PRS and similarity queries (Additional file 1: Fig. 12).

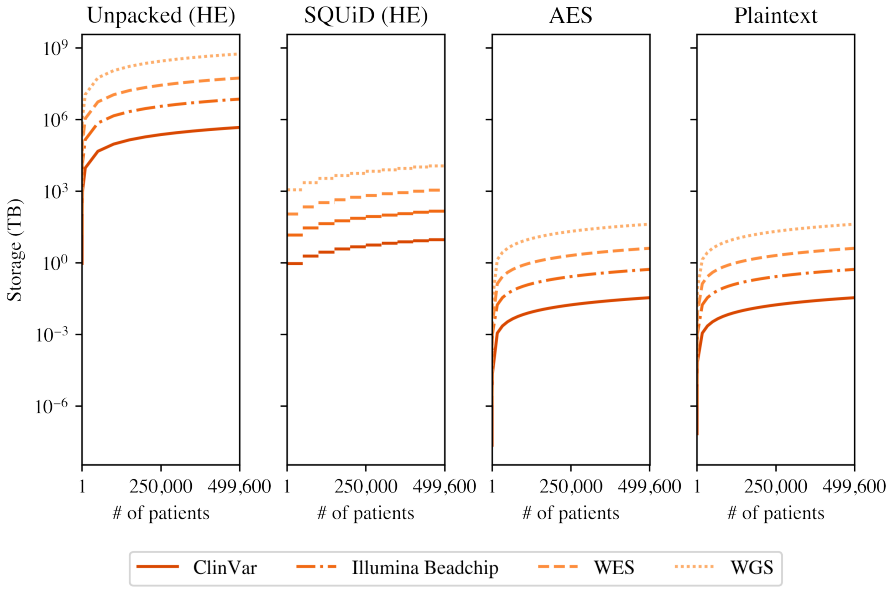
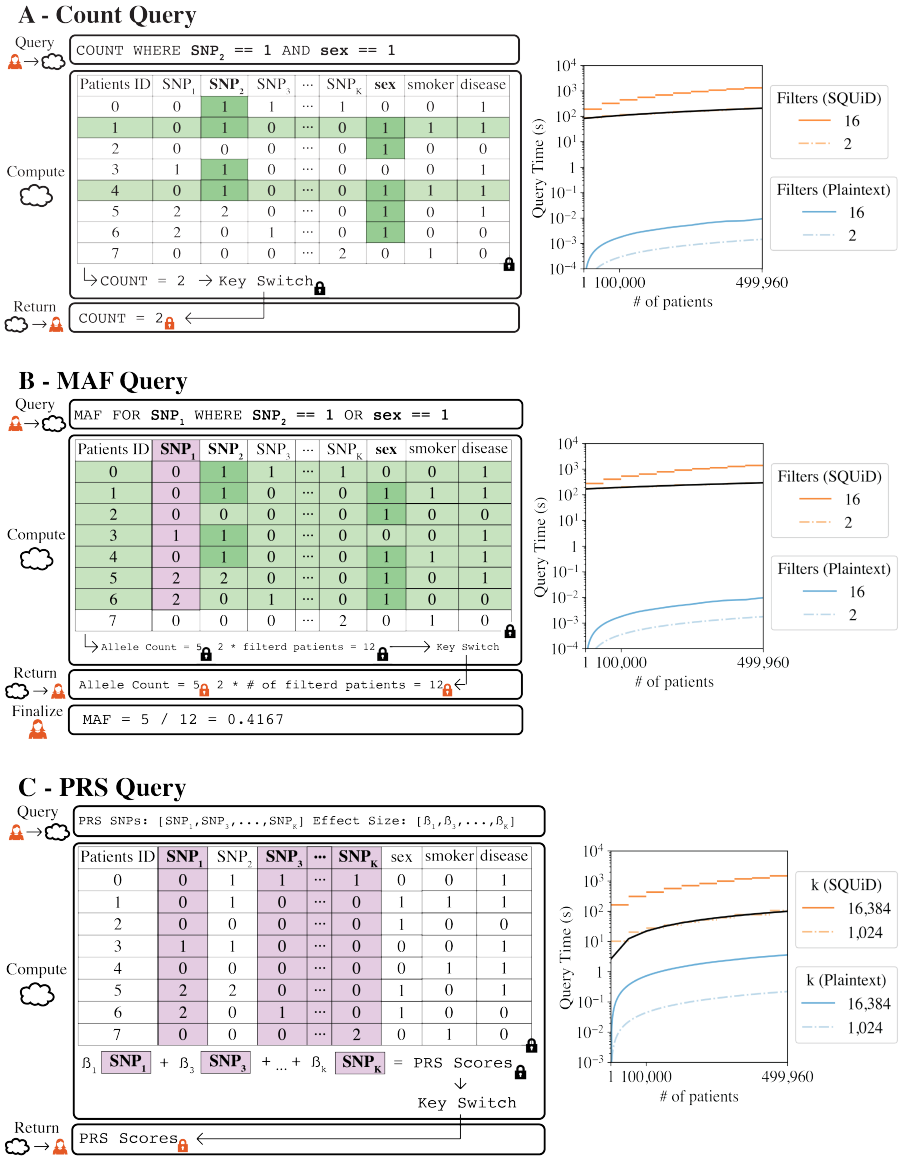


Figure 3: Plots showing the storage space required to store the ClinVar, Illumina Beadchip, WES, and WGS SNP genotypes with different schemes. The number of SNPs for WES and WGS is approximated using the 1000 Genomes Project.

Enabling secure, scalable, and fast analysis of genotypes and phenotypes

We have devised four encrypted query functionalities within the SQUiD framework. This modular design allows for seamless implementation of

additional functionalities to accommodate diverse analysis requirements. Our queries include count, MAF, PRS and similarity. Fig. 2C depicts how querying works under the public key-switching framework.



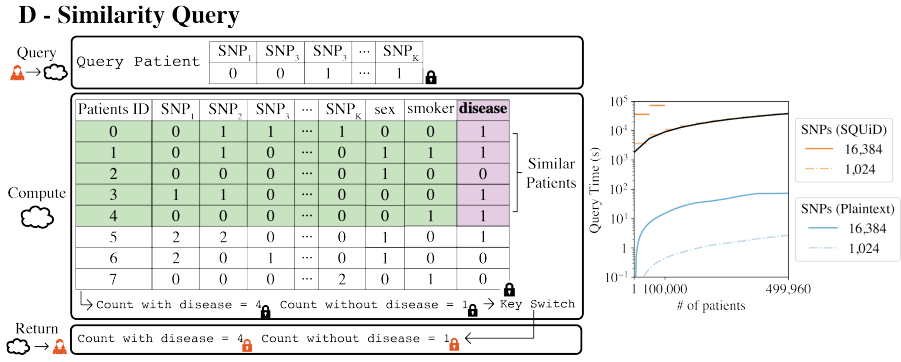


Figure 4: **(A, B, C, D)** For each query, the plots on the right show the query time by varying the number of filters for the count and MAF query, by varying the number of SNPs and effect sizes (k) for the PRS query, and by varying the number of SNPs for the similarity query. The query time for SQUiD and the query time of a plaintext solution are shown for comparison. The plaintext solution works on a database encrypted with AES. For each plaintext query, the necessary components for the query are decrypted and then computed on. **(A) Count Query.** The count query returns the number of patients that pass a given filter in the query (patients who pass the filter are highlighted in green, with darker green cells indicating passing a condition). A black line of best fit for a count query with 2 filters is given as the equation time (s) = 0.00025(# of patients)+82.71. Due to the strict linear scaling, the performance of our query can easily be interpolated by this line of best fit. **(B) MAF Query.** The MAF query creates a filtered cohort of patients (patients who pass the filter are highlighted in green, with darker green cells indicating passing a condition) and computes the MAF of a target SNP for that cohort (purple SNPs). A black line of best fit for a MAF query with 2 filters is given as time (s) = 0.00025(# of patients) + 170. **(C) PRS Query.** The PRS query returns the PRS score of all patients for a predetermined PRS SNP set and their effect sizes. A black line of best fit for a prs query with 1,024 effect SNPs is given as time (s) = 0.00019(# of patients) + 2.6. **(D) Similarity Query.** The similarity query returns the number of patients with and without a disease from a cohort of patients similar to a target patient (patients highlighted in green). The target patient's genome is encrypted with the owner's public key when it is sent to the public cloud. A black line of best fit for a similarity query with 1,024 SNPs is given as time (s) = 0.073(# of patients) + 1800.

Count queries within the SQUiD framework ascertain the number of patients satisfying specific filters or equality checks. For example, a count query could count the number of patients with type-2 diabetes (T2D), whose SNP on gene TCF7L2 has a heterozygous alternative allele. MAF queries are employed to compute the MAF for a given target SNP within a filtered patient cohort. For instance, SQUiD can compute two MAF queries: one for a target SNP on the

TCF7L2 gene within a cohort of patients with T2D, and another within a cohort of patients without T2D to study correlations between SNPs on the TCF7L2 gene and T2D. We can further add many different filters to build the cohort such as constraining it to patients with homozygous SNPs on a gene of interest. PRS queries involve the calculation and return of the PRS for all patients given a list of GWAS SNPs and their coefficients. PRS queries require only the coefficients and SNPs to be supplied post training such as those found on the PGS catalog [50]. Finally, similarity queries take a target patient's encrypted genotype as input, build a cohort of genetically similar patients in the database through a scoring function like the squared L_2-norm, and output the number of similar patients with and without a particular disease of interest (see Methods and Additional file 1: Supplementary Material). To evaluate the performance of each query, we conducted benchmarking against a plaintext implementation. The plaintext implementation keeps the genotype-phenotype data encrypted at rest (as mandated by the policies) and decrypts the necessary components of the data to compute the query in plaintext, while SQUiD keeps the data encrypted both at rest and during computation, enabling much stronger security as the data no longer has visibility to the computing party. This plaintext implementation models the current data access guidelines set by initiatives such as the dbGaP and UK Biobank where researchers download encrypted data, decrypt the data locally, and then analyze the data in plaintext [3].

We implemented SQUiD using the HE library, HElib [42], and benchmarked SQUiD on an *n2-standard-64* Google Cloud instance with an Intel Xeon Gold 6268 processor running at 2.8 GHz and 256 GB of memory (see Additional file 1: Supplementary Material for more details on the experimental setup and HE parameters). On a dataset with 499,600 patients, SQUiD can perform a count query with 2 filters in 4 minutes (0.004 seconds in plaintext), a MAF query with 2 filters in 5 minutes (0.004 seconds in plaintext), a PRS query with 1,024 effect SNPs in 2 minutes (0.59 seconds in plaintext), and a similarity query with 1,024 SNPs in 10 hours (2.7 seconds in plaintext) (Fig. 4).

We investigated the overhead of the HE library used in SQUiD and the overhead of the algorithms developed in SQUiD by comparing the query times against a solution that is a direct translation of SQUID algorithms without using the HE library (denoted as "SQUiD without HE"). SQUiD without HE computes the plaintext versions of the SQUiD queries by converting the HE library functions to their plaintext counterparts and then using these functions in the same way that SQUiD does. We found that the SQUiD without HE solution has a 20x overhead for a count and MAF queries with 2 filters, 2.5x overhead for a PRS query with 1,024 effect SNPs (k), and a 180x overhead for a similarity query with 1,024 SNPs compared to the plaintext solution (Additional file 1: Fig. 13).

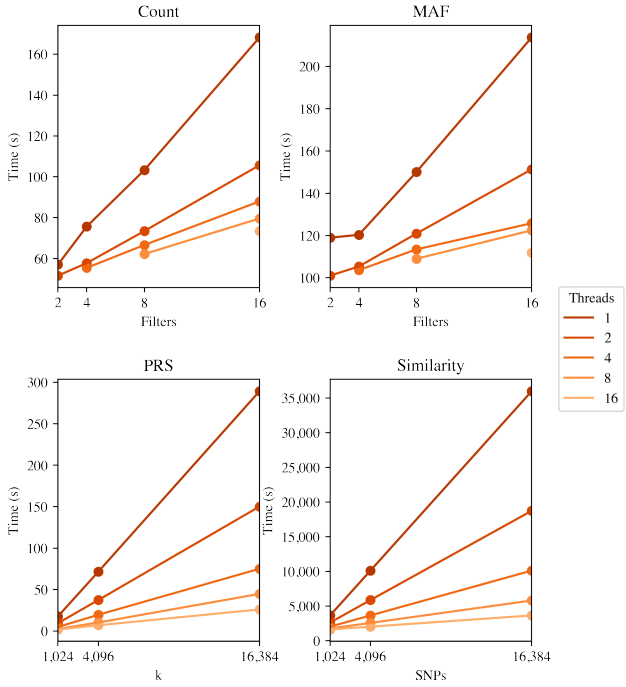


Figure 5: Plots of count, MAF, PRS, and similarity query time by the number filters, effect SNPs (k), and SNPs varying the number of threads. We benchmarked the time for each query on a database with 49,960 patients using 2, 4, 8, and 16 filters for the count and MAF queries, and 1,024, 4,096, and 16,384 SNPs for the PRS and similarity queries.

We also show that our queries are highly parallelizable because of their linear structure. Each query involves computing a filter, a linear combination, or an L_2 similarity for a set of SNPs across patients. Since these operations are performed for each patient, SQUiD achieves parallelism by chunking the database rows and processing these chunks concurrently. Our benchmarking in a multi-threaded environment shows that query performance scales linearly with the number of cores used (Fig. 5). On large databases with millions of patients, this scaling ensures reasonable query performance. To demonstrate this, we also ran each query with 50 threads on a database with 9,992,000 patients and found that a count query with 2 filters took 3 minutes, a MAF query with 2 filters took 4 minutes, a PRS query with 1,024 effect SNPs took 5 minutes, and a similarity query with 1,024 SNPs took 4 hours (Additional file 1: Fig. 14).

Our data show that all the functionalities implemented in SQUiD exhibit linear scaling relative to the size of their inputs. Specifically, the count and MAF queries scale linearly with the number of filters with a slope of 0.62, the PRS query scales linearly with the number of SNPs with a slope of 0.001, and the similarity query scales linearly with the number of SNPs given for the target patient with a slope of 0.068 (Fig. 4). Our slopes consistently indicate a slow growth in runtime. Notably, the runtime of all protocols is proportional to the number of patients in the database and independent of the total number of SNPs in the database (Additional file 1: Fig. 15). A plaintext implementation of our protocols would also scale linearly with the number of patients in the database and the number of filters and SNPs involved in the query. Thus, SQUiD achieves optimal linear scaling as expected from a plaintext implementation, which signifies its ability to efficiently adapt to larger datasets in the future. Furthermore, with the expected decrease in the price of cloud computing in the future, the steady runtime observed for all queries ensures that increasing the size of the databases beyond the limits benchmarked in this study will yield steady performance outcomes, enabling real-world applications of SQUiD with biobank-scale data. We also show that the SQUiD's communication cost for all queries except PRS query is constant regardless of the number of patients in the database while communication cost increases with the number of patients for all query types in plaintext (Additional file 1: Fig. 16 and 17). Overall the communication is minimal. Comparable to an instagram post which has a maximum size of 4.3 MB (1,080 by 1,350 pixels) [1], most of our queries use less than 50 MB on databases with 100,000 patients.

We also developed an API and a command line interface (CLI) to facilitate interaction with SQUiD, thereby enhancing its usability for researchers (Additional file 1: Fig. 18). The API and CLI enable researchers to execute various queries and perform essential functions through simple commands. For instance, researchers can generate private and public keys required for encryption and authorization, send the public key to the data owner, execute all desired queries (See Additional file 1: Table 4 for query parameters), and decrypt the returned query results. The API simplifies the deployment process for researchers who are not experts in privacy and security when utilizing SQUiD.

SQUiD can reproduce known genotype-phenotype relationships in UK Biobank

We studied the relationship between patients with T2D and a control group in the UK Biobank dataset to assess the accuracy of the MAF and count queries in SQUiD. Firstly, we calculated the MAFs for the top five SNPs with the largest difference between T2D patients and the control group patients (Fig.

6A). We compared the MAFs computed by SQUiD with the MAFs computed in plaintext to show there is no difference between them. Secondly, for these same five SNPs, we computed a chi-square statistic by using the allele counts for the control and case group (T2D in our case) [13]. We used the count query in SQUiD to get the allele counts and then computed the chi-square statistic in plaintext. The chi-square scores obtained from SQUiD queries are identical to the plaintext computation results (Fig. 6B). Note that SQUiD does not directly execute GWAS, it has the capability to generate cohorts with specific attributes. We have shown that it can create accurate cohorts that will result in accurate GWAS demonstrated by the GWAS for T2D (Fig. 6).

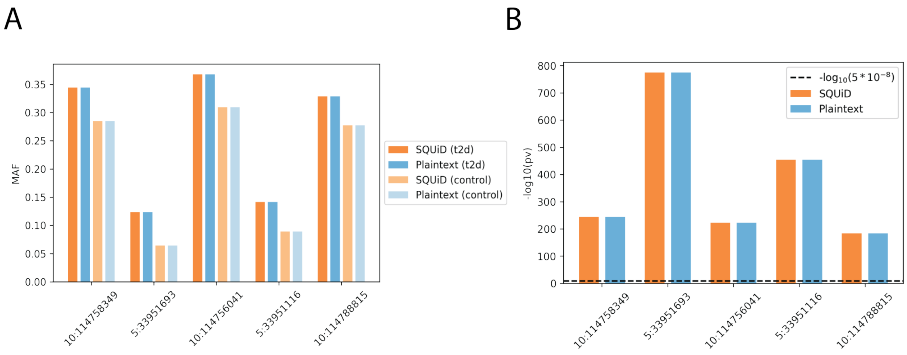


Figure 6: **(A)** Histograms of the MAFs of SNPs exhibiting the most substantial difference between control and T2D patient groups. The MAFs were calculated with SQUiD (orange), and in plaintext (blue). **(B)** Plot of -log(p-value) for the SNPs in (A).

We further evaluated the accuracy of SQUiD by replicating the sparse PRS calculations for standing height and T2D performed in the UK Biobank PRS study [72] using both plaintext calculations and the SQUiD PRS query. The standing height and T2D PRS use 51,209 and 183,830 SNPs, respectively. They are the traits with the most number of SNPs involved in PRS calculations in the UK Biobank. We performed these calculations for 20,000 randomly selected patients in the UK Biobank. Our analysis revealed no observable difference in the PRS distribution and scores between plaintext and SQUiD queries (Fig. 7). Notably, the sole discrepancy between the calculations arose from a marginal loss in precision. To accommodate the requirements of using integers in SNP effect sizes in SQUiD PRS queries, the effect sizes were multiplied by 1,000 and converted to integers. However, the resulting precision loss was minimal (Fig. 7B,C).

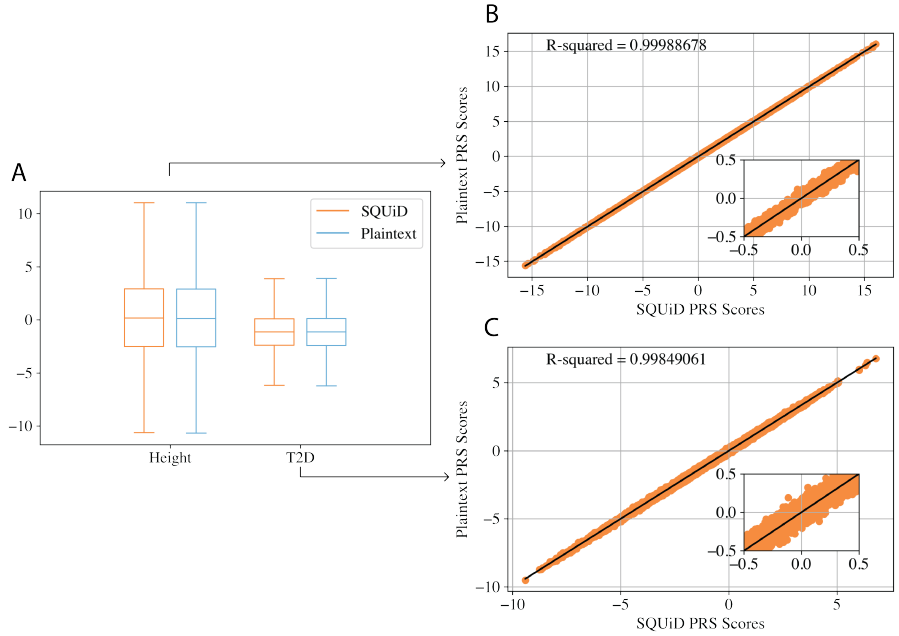


Figure 7: **(A)** Boxplots of the PRS score distributions of UK Biobank patients for standing height and type 2 diabetes (T2D) calculated with SQUiD (orange) vs plaintext (blue). **(B)** A scatter plot of the height PRS calculated by SQUiD vs. plaintext, where each point represents a patient. The black line is a line of best fit with an R^2 of 0.9999. **(C)** The same plot as (B) for T2D with an R^2 value of 0.9985.

3 Discussion

We introduce SQUiD, a novel, secure, and user-friendly queryable genotype-phenotype database implemented using homomorphic encryption. We envision SQUiD as a valuable tool for data owners, including hospitals, non-profit academic research institutions, and government health agencies, offering them a secure means to store genotype-phenotype data in the cloud while enabling authorized researchers to securely analyze this data. We propose that our system has the potential to replace existing genotype-phenotype databases, delivering enhanced security measures without compromising functionality. By employing homomorphic encryption, SQUiD offers a robust, scalable, and practical solution to mitigate privacy risks associated with sensitive genetic and phenotypic data. We demonstrate this by showing that SQUiD can scale with increasing numbers of patients and SNPs in a genotype-phenotype database, by performing a simple GWAS study on UKBB data, as well as by replicating PRS calculations in

UKBB [72]. All our query protocols (count query, MAF query, PRS query, and similarity query) and encryption protocols (setup of the database) were run on single-threads unless otherwise indicated.

SQUiD leverages homomorphic encryption, which, to date, presents three key challenges. Firstly, traditional homomorphic encryption was designed for a two-party setting involving a server and a client. Secondly, it is known to incur a high storage cost. Lastly, analysis with homomorphic encryption tends to be slow. To overcome the first challenge, we adopted the established concept of the Proxy Re-encryption system [61] and adapted it to develop a theoretical and practical implementation within the framework of homomorphic encryption. This adaptation, which we refer to as the public key-switching technique, enables secure multi-client queries on encrypted data, specifically addressing the needs of our application in the biomedical domain.

Furthermore, we demonstrate a significant improvement in storage efficiency through the application of a well-known vertical packing storage method, achieving a storage enhancement of 49,960 times compared to a naive homomorphic encryption solution. While storing SNP genotypes using homomorphic encryption increases storage costs relative to state-of-the-art encryption methods like AES, this approach is indispensable as homomorphic encryption allows execution of functions on encrypted data. While our queries demonstrate slower performance compared to plaintext solutions, we consider the trade-off between security and performance to be within acceptable limits. Additionally, implementing multi-threading significantly enhances performance. This performance overhead is unlikely to significantly impact the usability and utility of the framework for researchers. This is because the alternative is to download a large database and analyze the data locally, which is a much more time-consuming and resource-intensive process. Therefore, we believe that our framework offers an optimal balance of security and performance.

Although encrypted database systems do exist, to the best of our knowledge, none of them offer the same level of security guarantees and functionality as SQUiD. A developed secure database framework named CryptDB [62] offers efficient secure data storage and query performance. However, it does not offer the functionalities provided by SQUiD for two main reasons. Firstly, this framework is unable to compute the same set of queries as SQUiD. For instance, CryptDB lacks the ability to add *and* multiply encrypted database items, a necessary requirement for computing the linear combinations in PRS queries. Secondly, and more critically, CryptDB exhibits significant information leakage during equality checks used in the filtering process in count and MAF queries. Specifically, CryptDB exposes the count of unique items within the columns used for the equality checks. For genotype-phenotype databases that store SNPs with just three possible genotypes with known allele frequencies, CryptDB would

expose the patients with the same genotypes for each SNP. This information could be combined with the known and well-studied population frequencies of each SNP to devise a simple attack that reconstructs the genotype values for each patient in the database, resulting in a complete breach of security. On the other hand, SQUiD offers a solution for issues of data protection, data privacy and stigma for researchers, funders, clinicians and patients. Furthermore, while databases that keep data encrypted at rest with AES can answer the same queries as SQUiD, they cannot perform these queries as securely as SQUiD does. For any query, these databases must first decrypt the relevant data to compute the query, exposing the data to potential attacks. In contrast, SQUiD can perform all queries without the need for decryption, significantly improving security over existing systems.

Privacy-preserving MAF calculations using homomorphic encryption were proposed before [48]. Notably, SQUiD's MAF query differs from this approach as it computes the MAF within a filtered patient cohort, where the filtering is done via protocols developed in this work. For a detailed mathematical exposition of these distinctions, refer to Additional file 1: Supplementary Material.

We compared our patient similarity queries to existing private patient similarity queries (SPQ). Many existing SPQ protocols such as Wang et al. [75] privately compute patient similarity under the secure multiparty computation security assumptions, assuming non-colluding parties. Since SQUiD employs homomorphic encryption, no assumptions about collusion between parties are necessary. Additionally, our query process involves a single round of communication, with the querying researcher sending a query to the cloud and receiving a prompt response. In contrast, the protocol outlined in [75] necessitates an interactive protocol with multiple rounds.

We also empirically compared SQUiD to [65] due to the similar security settings. The latter proposes a partial homomorphic encryption algorithm that supports only ciphertext addition and scalar multiplication operations for computing patient similarity using a squared L_2-norm. We implemented the euclidean distance (equivalent to the squared L_2-norm protocol) from [65] to the best of our understanding for comparison purposes. Additional file 1: Fig. 19 shows that SQUiD can compute the squared L_2-norm faster for larger datasets with an approximately 2x speed up for datasets with 49,960 patients.

We envision three use cases for this framework: 1- Funding agencies such as NIH can employ this framework to disseminate insights derived from the data currently available through the NIMH Data Archive (NDA) or Database of Genotypes and Phenotypes (dbGAP). 2- Multi-site consortia can employ this framework to disseminate data to their members while keeping the data secure in cloud storage. 3- Learning health systems can employ this framework to disseminate data to their researchers while keeping the data secure in cloud

storage. Our secure framework is designed to enable users to form specific patient cohorts based on desired characteristics. Within this system, users can also determine the distribution of PRS for a particular disease across various patient populations. For example, one can explore the PRS distribution for schizophrenia among patients diagnosed with bipolar disorder. Additionally, the framework allows for the analysis of disease outcomes in patients who share genetic similarities with a specific patient of interest, facilitating more personalized and targeted approaches to healthcare and research.

We designed SQUiD with ease of use in mind for both researchers and data owners. Data owners are only required to provide a VCF file for the genotypes and a CSV file for the phenotype data. SQUiD then handles the packing, encryption, and uploading of this data to a public cloud platform. The SQUiD codebase includes a cloud-deployable API framework, allowing researchers to query the data through API calls seamlessly.

SQUiD's design is scalable to support multiple data owners. In a multi-owner system, each data owner independently prepares, encrypts, and uploads their data to the public cloud. Each data owner's information is stored in a separate encrypted database along with a corresponding key store. When a researcher wishes to query the data, they send their public key to each data owner. The data owners then generate a public key-switching key using their secret key and store this in their key-switching store. The researcher's query is processed in the public cloud, where it is evaluated across the encrypted databases. The results are key-switched using the respective data owner's key-switching store, ensuring that the final query results are encrypted under the researcher's public key. These results are aggregated and sent back to the researcher, who can then decrypt them to obtain the final output.

This multi-owner implementation introduces additional storage, as the public cloud must store multiple key-switching stores for each data owner. It also needs to maintain evaluation keys (relinearization keys, rotation keys, and bootstrapping keys) for each data owner. Despite this, the process ensures the privacy of each data owner is maintained, and the researcher can securely access aggregated results without compromising individual data security. There are also approaches that do not require multiple evaluation keys for each owner [58], however, they are not tailored to the specific needs of genotype-phenotype data.

SQUiD can handle missing data. For count and MAF queries, SQUiD defaults to excluding patients with missing values from the cohort being analyzed. For the PRS and similarity queries, any column with missing SNPs will be excluded from the PRS and similarity query calculation.

4 Conclusions

SQUiD presents an innovative and impactful solution for a world grappling with escalating concerns surrounding security and privacy of genetic and clinical data. By circumventing the challenges posed by the ever-changing, heterogeneous landscape of data protection laws, SQUiD offers a robust framework to safeguard sensitive information. Moreover, we firmly believe that SQUiD has the potential to enhance patient trust by ensuring the security and controlled utilization of their data for specific research purposes, thus has the potential to increase participation in genetic research. Lastly, although this study focused on genotype-phenotype analyses for proof of principle, SQUiD's modular design allows for the integration of other discrete data modalities and analytic approaches, as the need arises. This adaptability will be critical at a time when precision medicine research is rapidly expanding to encompass more complex molecular and clinical datasets.

5 Methods

Security and threat models

Our security assumption is based on the current data-sharing policies within many public and private entities. That is, the data owner and authorized researchers are mutually trusted. Thus, authorized researchers are allowed to query the genotype-phenotype data that do not threaten the confidentiality of patients according to the data use agreements. The inherent data leakage from query results and potential inference attacks from authorized researchers are therefore not considered.

Meanwhile, genotypes and phenotypes as well as a subset of the queries are protected from the public cloud and attackers. More precisely, we consider the following three threat models for database management [10, 38]:

- Snapshot attackers that obtain a snapshot of the database
- Persistent passive attackers that compromise the cloud server to obtain not only the database but also queries and all server's operations
- Active attackers that fully compromise the server to deviate from pre-designed protocols for queries

In our SQUiD construction, snapshot attackers receive ciphertexts of the Brakerski-Gentry-Vaikuntanathan (BGV) homomorphic encryption scheme. We use the security level estimator from HElib [9, 42] to choose BGV parameters that provide a 128-bit security level against known attacks. Consequently, the

security towards snapshot attackers inherits from BGV's IND-CPA security, *i.e.*, the ciphertexts are almost indistinguishable from random characters.

For persistent passive attackers, there are many ways that querying encrypted databases can result in private information leakage [32, 44, 54, 59]. Most prominent ones include leakage through (1) *access pattern*, which determines if certain records are consistently accessed; and (2) *search pattern*, which indicates if and when an encrypted query is repeated. Many cryptosystems, including property-preserving encryption (PPE) [7,53] and searchable encryption (SE) [29, 71], fail to protect against these types of information leaks. This is primarily due to their inherent functionality, which inadvertently discloses properties of datasets, thereby compromising privacy. However, homomorphic encryption schemes such as BGV provide a solution that does not leak access and search patterns [46]. Using HE to encrypt databases propels algorithms that have to touch all the relevant records in the dataset for a single query. For example, to find out whether an encrypted input is in the encrypted database, the input needs to be compared with every single encrypted value in the database homomorphically. This prevents access pattern leakages since the access pattern remains uniform for all queries. In addition, search pattern leakages are prevented due to the IND-CPA security under carefully selected parameters, since encrypted queries are indistinguishable from one another, regardless of their contents [46].

It is worth mentioning that persistent passive attackers do not learn additional information about the database from knowledge of the server's computation patterns. Precisely, when an authorized researcher sends a query f, the server performs a series of operations on the encrypted database $\mathbf{Enc}(m)$ to obtain $\mathbf{Enc}(f(m))$. The function f is in plaintext for Count, MAF and PRS queries and contains ciphertexts for similarity queries. In all these cases, the computation pattern for the server is predefined and contains operations such as homomorphic additions, multiplications, and key switching. As such, inference attacks from persistent passive attackers are also prevented, as only computational patterns of different functions are revealed but not any computation result $f(m)$.

While the problem of defending against active attackers is challenging and still unsolved [37,38], our SQUiD construction provides reasonable mitigation towards active attackers. Namely, active attackers can deviate from pre-determined operations in SQUiD and therefore send wrong computation results to authorized researchers, but they can not learn information about the database.

Homomorphic Encryption

Encryption is a procedure that maps the *plaintext* data into its *ciphertext*, such that the plaintext can not be deduced from the ciphertext without knowing the

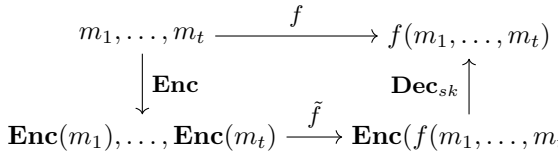


Figure 8: The homomorphic evaluation of a function f on ciphertexts

secret key. Homomorphic encryption is a class of encryption schemes with an additional property: computations can be performed over ciphertexts without knowing the secret key.

Figure 8 visualizes this property in a commutative diagram, which enables secure computation outsourcing.

To compute a function f on plaintexts $m_1, \ldots, m_t$ without revealing them, plaintexts are encrypted and corresponding ciphertexts $\mathbf{Enc}(m_1), \ldots, \mathbf{Enc}(m_t)$ are sent to the public cloud. Then, a function $\tilde{f}$, which corresponds to the HE-friendly version of the desired function f, is evaluated among the ciphertexts homomorphically. As a result, a ciphertext of $f(m_1, \ldots, m_t)$ is derived, which contains the evaluation result equivalent to that of a plain evaluation. Therefore, the decryption of the final ciphertext outputs the desired evaluation result.

HE ciphertexts contain a *noise* component, whose value grows with homomorphic operations. This is controlled by pre-fixed HE parameters, which is also used to set a noise budget. If the number of operations in an algorithm is too large such that the noise consumption exceeds the budget, then the result can no longer be decrypted correctly. To avoid this, a *bootstrapping* operation is introduced to refresh the ciphertexts, enabling the *fully* homomorphic encryption (FHE) schemes that support evaluations of *arbitrary* circuits for different operations including multiplications and additions (*i.e.,* arbitrary f) [34]. Detailed realizations of homomorphic operations are included in the supplementary material.

Brakerski-Gentry-Vaikuntanathan scheme (BGV)

The BGV scheme is an FHE scheme that relies on the hardness of the Ring Learning-with-error (RLWE) problem [57]. Its basic building blocks are homomorphic addition ADD and multiplication MULT. Since any computable function can be realized with additions and multiplications, the homomorphic evaluation of any computable f can be realized with ADDs and MULTs. Bootstrapping in BGV is a very costly operation [25, 41]. It is, therefore, common to use BGV in the *levelled* manner, *i.e.,* to choose the HE noise

parameter with large noise capacity such that computations can be performed without bootstrapping. Our study uses the levelled version of BGV.

BGV allows efficient computations in the amortized sense. It supports Single Instruction Multiple Data (SIMD) operations, which allows multiple values to be packed into one BGV ciphertext, enabling computations over a single ciphertext to be performed on all packed values in an efficient manner [70]. Details of the SIMD packing are included in the supplementary material.

Public key-switching

In general, HE binary operations only support input ciphertexts that are encrypted under the same key. Therefore, in the scenario of multiple users each holding their own keys, there is a natural need to convert a ciphertext encrypted under one key to another ciphertext that encrypts the same message under a different key. A naive approach is to decrypt and re-encrypt with a different key, but this exposes the original secret key and the message to the party that performs this procedure. To prevent such leakages, the above procedure can be done *homomorphically* such that the evaluation party can not access the message in the clear. Such a technique is called *key switching*. Mathematically, when converting the key system from $(\mathsf{pk}, \mathsf{sk})$ to $(\mathsf{pk}^*, \mathsf{sk}^*)$, the evaluation party does not need to know sk, but a key-switching key $\mathsf{ksk}_{(\mathsf{sk}\to\mathsf{sk}^*)}$ which leaks no information about secret keys.

Our scenario exploits the key-switching key $\mathsf{ksk}_{(\mathsf{sk}\to\mathsf{sk}^*)}$. While the traditional key-switching key generation uses both sk and sk^*, only sk and pk^* are needed in our design, hence it is called public key-switching. This design preserves the confidentiality of sk^* as it does not need to be shared to compute the key-switching key. In our scenario, the secret key of the authorized researchers does not need to be sent to the data owner to generate the key-switching key. Please see supplementary material for the mathematical details of the realization of public key-switching with BGV and how we control the increasing noise.

Database construction with vertical packing

The dataset in SQUiD is represented as a matrix $M = \{m_{(i,j)} | 1 \leq i \leq r, 1 \leq j \leq k\}$, where r is the number of patients, k is the number of attributes (features), and the value in position (i, j) corresponds to the j-th feature of the i-th patient (*e.g.,* the genotype of j-th SNP of i-th patient). We use the term *vertical* or *horizontal* for the direction in the matrix, which corresponds to an attribute for all patients or all attributes for a single patient, respectively.

As we explained earlier, BGV supports packing multiple messages into one ciphertext. SQUiD packs elements *vertically*: let ℓ denote the packing capacity in a ciphertext, then the r elements in the jth column are encrypted into $\lceil r/\ell \rceil$ ciphertexts

$$\mathsf{ct}_{(s,j)} = \mathbf{Enc}\left(\{m_{(\ell \cdot s+1,j)}, m_{(\ell \cdot s+2,j)}, \cdots, m_{(\ell \cdot (s+1),j)}\}\right)$$

where $1 \leq s \leq \lceil r/\ell \rceil$ and $m_{i,j}$ is considered as 0 for $i > r$. Overall, entire dataset is encrypted into $C = \{\mathsf{ct}_{(s,j)} \mid 1 \leq s \leq \lceil r/\ell \rceil, 1 \leq j \leq k\}$.

The update, insert, and delete operations on a vertically packed encrypted database vary slightly from their typical implementations.

- *Update:* To update a single value m' at index i, j, a new encryption of $\mathsf{ct}_{(s,j)}$ where $s = \lceil i/\ell \rceil$ needs to be uploaded where

$$\mathsf{ct}_{(s,j)} = \mathbf{Enc}\left(\{m_{(\ell \cdot s+1,j)}, m_{(\ell \cdot s+2,j)}, \cdots, m', \cdots, m_{(\ell \cdot (s+1),j)}\}\right)$$

- *Insert:* To insert a new row at $r+1$, if ciphertexts are not fully packed (i.e., $\ell \nmid r$), then the last row of packed ciphertexts contains zeros at row index $r+1$, which are updated. Otherwise, the following k fresh ciphertexts are added, forming the last row of C.

$$\left\{\mathsf{ct}_{(s+1,j)} = \mathbf{Enc}\left(\{m_{(\ell \cdot r+1,j)}, 0, \cdots, 0\}\right),\ 1 \leq j \leq k\right\}$$

- *Delete:* To delete an entry at index i, j, a plaintext, which encodes zero at the $i \bmod \ell$-th slot and one elsewhere is multiplied with $\mathsf{ct}_{(\lceil i/\ell \rceil,j)}$.

Note that update and insert operations both upload new ciphertexts with low noise, but the delete operation increases the noise with a plaintext-ciphertext multiplication. To bound the noise growth, we set a number α for the maximum times of consecutive delete operations. On the $(\alpha+1)$-th time to delete an entry, an update should be performed instead, after which α deletes are again allowed. For SQUID with our experimental parameters, the value α is taken to be 5.

Functionalities

In this section, we describe the supported functionalities of SQUiD and the evaluation procedures using homomorphic encryption.

Count queries. The first category of queries is to *count* the number of patients *whose* attributes satisfy certain conjunctive (AND) and/or disjunctive (OR) relations. Its evaluation contains two stages, filtering and vertical aggregation.

v \ u	0	1	2
-1	0	0	0
0	1	0	0
1	0	1	0
2	0	0	1

Table 1: The truth table of $\mathsf{EQtest}(u, v)$ for SNPs. We assume that the query value u is not missing ($u \neq -1$).

Filtering. Suppose the researcher specifies $\tau > 1$ selection criteria (either in plaintext or ciphertext) and their relation (AND and/or OR). The filtering stage outputs a *predicate vector* $\mathbf{p}$ composed of r encrypted binary numbers. If the element $\mathbf{p}[i]$ decrypts to 1, then the patient i is in this pre-defined cohort.

First, we explain how to homomorphically check a single selection criterion, which amounts to performing a homomorphic equality test between the given value in a query and a value in the matrix. The key idea is to find a polynomial representation, which can be evaluated as a sequence of homomorphic additions and multiplications.

$$\text{Function } \mathsf{EQTest} \iff \text{Polynomial} \iff \text{Sequence of } \mathsf{ADD}\text{s and } \mathsf{MULT}\text{s}$$

Without loss of generality, we consider the inputs of EQTest as genotype values in $\{-1, 0, 1, 2\}$ where -1 indicates a missing SNP, and denote them as u and v. As shown in Table 1, this function determines a unique truth table.

We derive the polynomial representation of $\mathsf{EQTest}(u, v)$ as follows. Let v be an encrypted matrix value, and u be the query value, which can be either in the clear or encrypted depending on the researcher. If u is provided in the clear, then we can interpolate the u-th column of the truth table 1 into a degree-3 polynomial F_u with input variable v. If u is also encrypted, then we precompute a polynomial F of degree 5 that maps 0 to 1 and $\{\pm 1, \pm 2, 3\}$ to 0, whose input variable is $u - v \in \{0, \pm 1, \pm 2, 3\}$. Note that we do not consider the case where both u and v are missing because it is assumed that the query value would never look for missing SNPs.

Second, we explain how to homomorphically combine the results of multiple equality checks using AND and OR. Let $\{(u_k, v_k)\}_{k=1}^{\tau}$ be the set of (encrypted or unencrypted) queries and (encrypted) matrix values, then for each patient i we compute the expression homomorphically as Equation (1), where d and b are constants in Table 2. The evaluation decrypts to 1 if the data of the patient

Query type	b	d
Conjunction	0	0
Disjunction	1	1

Table 2: Constants in circuit (1) [49]

i matches the selecting criteria, and 0 otherwise.

$$x_i = d + \prod_{k=1}^{\tau} [b + \mathsf{EQTest}(u_k, v_k)] \tag{1}$$

Vertical Aggregation. Suppose each ciphertext provides ℓ SIMD slots, then the predicate vector for r patients is batched into $\lceil r/\ell \rceil$ ciphertexts. The procedure of summing over these batched messages is a *vertical* aggregation.

Our design fully exploits the advantages of parallel computing. Namely, we perform $\mathcal{O}(r/\ell)$ homomorphic additions with additive depth $\mathcal{O}(\log(r/\ell))$ to obtain one ciphertext, whose ℓ slots are then aggregated with $\mathcal{O}(\log \ell)$ homomorphic rotations and additions. To support larger databases sizes, not all ℓ slots might be aggregated as the aggregated value would overflow the ring in the BGV scheme. In these cases, it is up to the client to aggregate the remaining slots. Please see supplementary material for details of homomorphic addition and rotations with BGV.

PRS queries. The second category of queries is to obtain *polygenic risk scores* of all the patients.

Definition 5.1. The polygenic risk score (PRS) of a patient is a linear combination of values of attributes in a subset S. For given coefficients (*i.e.,*effect sizes) $\{\beta_j\}_{j \in S}$, the PRS for patient i is $f_i = \sum_{j \in S} \beta_j \cdot m_{(i,j)}$, where $m_{(i,j)}$ is the genotype of the j-th SNP for i-th patient.

The PRS for each patient can be calculated with homomorphic multiplication and additions. Please see supplementary material for details of homomorphic addition and multiplications with BGV.

Horizontal aggregation. PRS queries aggregate information horizontally. We use parallel computing to minimize the execution time, and as can be seen from the Results section, answering PRS queries is relatively fast.

MAF queries. The third category of queries is to calculate the *minor allele frequency* for a target SNP of a filtered cohort of patients.

Definition 5.2. Minor allele frequency (MAF) is the frequency at which the minor allele occurs in a given population or a cohort. Let $\mathbf{p}$ be the predicate vector for r patients, where $\mathbf{p}[i]$ indicates whether the patient i is in the cohort. Then, for the dataset $M = \{m_{(i,j)}\}$, the MAF for SNP j with $\mathbf{p}$ is

$$\mathsf{AF}(\mathbf{p}, j) = \left(\sum_{i=1}^{r} m_{(i,j)} \cdot \mathbf{p}[i]\right) / \left(2\sum_{i=1}^{r} \mathbf{p}[i]\right),$$

$$\mathsf{MAF}(\mathbf{p}, j) = \min(\mathsf{AF}(\mathbf{p}, j), 1 - \mathsf{AF}(\mathbf{p}, j)).$$

As the homomorphic division and minimum comparisons are currently expensive operations, the cloud instead computes the numerator and denominator homomorphically and then returns the results to the clients for decryption, division, and the minumum operation.

Similarity queries. The fourth category of queries determines whether a specific individual (denoted as d) is genetically similar to patients *with* a certain disease or those *without*. There are two similarity metrics for researchers to choose from.

Definition 5.3. Suppose the database stores k attributes and the last attribute is the disease.

1. The L_2-distance similarity score $S_{L_2}(i, d)$ is defined as

$$S_{L_2}(i, d) = \sum_{j=1}^{k-1} (m_{(i,j)} - d_j)^2.$$

2. The Jaccard similarity score $S_{\mathsf{Jcd}}(i, d)$ is defined as

$$S_{\mathsf{Jcd}}(i, d) = \sum_{j=1}^{k-1} \mathsf{EQTest}(m_{(i,j)}, d_j),$$

where $\mathsf{EQTest}(\cdot, \cdot)$ equals to 1 if two inputs are equal and 0 otherwise.

In other words, the similarity score $S_{(\cdot)}(i, d)$ horizontally aggregates the result of the squared difference or EQTest.

As a result of this query, the researcher will receive two encrypted values r_1, r_2 from the cloud. The value r_1 is the number of patients that are genetically

similar to the target d with this disease, r_2 is the number of patients that are genetically similar to the target d and *do not* have the disease.

These two values are homomorphically computed as follows.

1. Similar to the filtering method in Section 5, the cloud computes a predicate $\mathbf{p}$ for patients with this disease.
2. To count similar patients, the cloud computes the similarity score $S_{(\cdot)}(i, d)$ between the target d and patient i. Then the cloud homomorphically checks if $S_{(\cdot)}(i, d)$ is greater than the pre-determined threshold t, which is done by evaluating the interpolation polynomial of degree $\text{Range}(S_{(\cdot)}(i, d)) - 1$. In our implementation, we use the Paterson-Stockmeyer method [60], a well-established technique [24, 27, 28, 43], to evaluate polynomials efficiently. As such, we get a predicate $\mathbf{p}_s$.
3. Multiplying the two predicates $\mathbf{p}$ and $\mathbf{p}_s$ component-wise realizes the `AND` relation and leads to another vector, whose vertical aggregation gives r_1.
4. Multiplying the inverse predicate $\neg\mathbf{p}$ and predicate $\mathbf{p}_s$ component-wise realizes the `AND` relation and leads to another vector, whose vertical aggregation gives r_2.

Availability of data and materials

The code for SQUiD, including the SQUiD API and CLI, is available under the GNU General Public License at `https://github.com/G2Lab/SQUiD/` [15] and `10.5281/zenodo.14176458` [16]. This repository contains all benchmarking code necessary to reproduce the results presented in this paper. Instructions for running the code are provided in the Benchmarking section of the README. UK Biobank data was downloaded from `https://www.ukbiobank.ac.uk/` [21] under application number 100316, and simulated data can be generated using `https://github.com/G2Lab/SQUiD/`. Example data is also included in the repository. The variants and their effect sizes used in the PRS calculations were download from the PGS Catalog at `https://www.pgscatalog.org/` [51, 52].

Funding

This project was funded by NIH grants R00HG0110909 and R35GM147004 to GG, funding from Warren Alpert Foundation to GG and TL, a National Science Foundation GRFP fellowship to JB, and funding from by CyberSecurity Research Flanders (reference number VR20192203) to JK.

References

[1] Help center. https://help.instagram.com/1631821640426723. Accessed: 2023-7-18.

[2] Introduction. https://homomorphicencryption.org/introduction/. Accessed: 2023-4-3.

[3] UK biobank data access guide, April 2023.

[4] U.S. data privacy laws to enter new era in 2023. *Reuters*, January 2023.

[5] 1000 Genomes Project Consortium, Adam Auton, Lisa D Brooks, Richard M Durbin, Erik P Garrison, Hyun Min Kang, Jan O Korbel, Jonathan L Marchini, Shane McCarthy, Gil A McVean, and Gonçalo R Abecasis. A global reference for human genetic variation. *Nature*, 526(7571):68–74, October 2015.

[6] Julián N Acosta, Guido J Falcone, Pranav Rajpurkar, and Eric J Topol. Multimodal biomedical AI. *Nat. Med.*, September 2022.

[7] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, 2004.

[8] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[9] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[10] Pedro Geraldo M. R. Alves and Diego F. Aranha. A framework for searching encrypted databases. *J. Internet Serv. Appl.*, 9(1):1:1–1:18, 2018.

[11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011.

[12] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, February 2006.

[13] Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. *Proc. Natl. Acad. Sci. U. S. A.*, 117(21):11608–11613, May 2020.

[14] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology — EUROCRYPT'98*, pages 127–144. Springer Berlin Heidelberg, 1998.

[15] Jacob Blindenbach, Jiayi Kang, Seungwan Hong, Caline Karam, Thomas Lehner, and Gamze Gürsoy. SQUiD. `https://github.com/g2lab/squid`, 2024.

[16] Jacob Blindenbach, Jiayi Kang, Seungwan Hong, Caline Karam, Thomas Lehner, and Gamze Gürsoy. SQUiD. https://doi.org/10.5281/zenodo.14176458, 2024.

[17] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[18] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[19] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[20] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 505–524. Springer, 2011.

[21] Clare Bycroft, Colin Freeman, Desislava Petkova, Gavin Band, Lloyd T. Elliott, Kevin Sharp, Allan Motyer, Damjan Vukcevic, Olivier Delaneau, Jared O'Connell, Adrian Cortes, Samantha Welsh, Alan Young, Mark

Effingham, Gil McVean, Stephen Leslie, Naomi Allen, Peter Donnelly, and Jonathan Marchini. The uk biobank resource with deep phenotyping and genomic data. *Nature*, 562(7726):203–209, oct 2018.

[22] Gizem S Çetin, Hao Chen, Kim Laine, Kristin Lauter, Peter Rindal, and Yuhou Xia. Private queries on encrypted genomic data. *BMC Med. Genomics*, 10(Suppl 2):45, July 2017.

[23] Fook Mun Chan, Ahmad Qaisar Ahmad Al Badawi, Jun Jie Sim, Benjamin Hong Meng Tan, Foo Chuan Sheng, and Khin Mi Mi Aung. Genotype imputation with homomorphic encryption. In *Proceedings of the 6th International Conference on Biomedical Signal and Image Processing*, ICBIP '21, pages 9–13, New York, NY, USA, November 2021. Association for Computing Machinery.

[24] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 34–54. Springer, 2019.

[25] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved fhe bootstrapping. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I*, pages 315–337. Springer, 2018.

[26] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[27] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 221–256. Springer, 2020.

[28] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1135–1150. ACM, 2021.

[29] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88, 2006.

[30] European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council. `https://data.europa.eu/eli/reg/2016/679/oj`, May 2016.

[31] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 144, 2012.

[32] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K Cunningham. Sok: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 172–191. IEEE, 2017.

[33] Robin Geelen, Michiel Van Beirendonck, Hilder V. L. Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios D. Dimou, Ingrid Verbauwhede, Frederik Vercauteren, and David W. Archer. BASALISC: programmable hardware accelerator for BGV fully homomorphic encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):32–57, 2023.

[34] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[35] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 850–867. Springer, 2012.

[36] Geoffrey S Ginsburg and Kathryn A Phillips. Precision medicine: From science to value. *Health Aff.*, 37(5):694–701, May 2018.

[37] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built on top of encrypted

data. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1353–1364. ACM, 2016.

[38] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. Why your encrypted database is not secure. In Alexandra Fedorova, Andrew Warfield, Ivan Beschastnikh, and Rachit Agarwal, editors, *Proceedings of the 16th Workshop on Hot Topics in Operating Systems, HotOS 2017, Whistler, BC, Canada, May 8-10, 2017*, pages 162–168. ACM, 2017.

[39] Gamze Gürsoy, Eduardo Chielle, Charlotte M Brannon, Michail Maniatakos, and Mark Gerstein. Privacy-preserving genotype imputation with fully homomorphic encryption. *Cell Syst*, 13(2):173–182.e3, February 2022.

[40] Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481, 2020. `https://eprint.iacr.org/2020/1481`.

[41] Shai Halevi and Victor Shoup. Bootstrapping for helib. *Journal of Cryptology*, 34(1):7, 2021.

[42] IBM. Helib: An implementation of homomorphic encryption (2.0.0). `https://github.com/homenc/HElib`, January 2021.

[43] Ilia Iliashenko and Vincent Zucca. Faster homomorphic comparison operations for bgv and bfv. *Proceedings on Privacy Enhancing Technologies*, 2021(3):246–264, 2021.

[44] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *Ndss*, volume 20, page 12. Citeseer, 2012.

[45] Anca Ivan and Yevgeniy Dodis. Proxy cryptography revisited. `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=626ecbdfdf0f92ef306865cc28503350d2591008`. Accessed: 2023-7-28.

[46] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. Sok: Cryptanalysis of encrypted search with leaker–a framework for leakage attack evaluation on real-world data. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 90–108. IEEE, 2022.

[47] Duhyeong Kim, Yongha Son, Dongwoo Kim, Andrey Kim, Seungwan Hong, and Jung Hee Cheon. Privacy-preserving approximate GWAS computation based on homomorphic encryption. *BMC Med. Genomics*, 13(Suppl 7):77, July 2020.

[48] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(5):S3, Dec 2015.

[49] Myungsun Kim, Hyung Tae Lee, San Ling, Shu Qin Ren, Benjamin Hong Meng Tan, and Huaxiong Wang. Better security for queries on encrypted databases. Cryptology ePrint Archive, Paper 2016/470, 2016. `https://eprint.iacr.org/2016/470`.

[50] Samuel A Lambert, Laurent Gil, Simon Jupp, Scott C Ritchie, Yu Xu, Annalisa Buniello, Aoife McMahon, Gad Abraham, Michael Chapman, Helen Parkinson, John Danesh, Jacqueline A L MacArthur, and Michael Inouye. The polygenic score catalog as an open database for reproducibility and systematic evaluation. *Nat. Genet.*, 53(4):420–425, April 2021.

[51] Samuel A. Lambert, Laurent Gil, Simon Jupp, Scott C. Ritchie, Yu Xu, Annalisa Buniello, Aoife McMahon, Gad Abraham, Michael Chapman, Helen Parkinson, John Danesh, Jacqueline A. L. MacArthur, and Michael Inouye. The polygenic score catalog as an open database for reproducibility and systematic evaluation. *Nature Genetics*, 53:420–425, 2021.

[52] Samuel A. Lambert, Benjamin Wingfield, Joel T. Gibson, Laurent Gil, Santhi Ramachandran, Florent Yvon, Shirin Saverimuttu, Emily Tinsley, Elizabeth Lewis, Scott C. Ritchie, Jingqin Wu, Rodrigo Canovas, Aoife McMahon, Laura W. Harris, Helen Parkinson, and Michael Inouye. Enhancing the polygenic score catalog with tools for score calculation and ancestry normalization. *Nature Genetics*, 2024.

[53] Kevin Lewi and David J Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1167–1178, 2016.

[54] Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu-an Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, 265:176–188, 2014.

[55] Jibang Liu, Yung-Hsiang Lu, and Cheng-Kok Koh. Performance analysis of arithmetic operations in homomorphic encryption. 2010.

[56] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[57] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013.

[58] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proceedings on Privacy Enhancing Technologies*, (4):291–311, 2021.

[59] Simon Oya and Florian Kerschbaum. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption. In *USENIX Security Symposium*, pages 127–142, 2021.

[60] Mike Paterson and Larry Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, 2:60–66, 03 1973.

[61] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, 20(4):1–31, November 2017.

[62] Raluca Ada Popa, Catherine M S Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, October 2011. Association for Computing Machinery.

[63] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[64] Oded Regev. The learning with errors problem (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 191–204. IEEE Computer Society, 2010.

[65] Ahmed Salem, Pascal Berrang, Mathias Humbert, and Michael Backes. Privacy-preserving similar patient queries for combined biomedical data. *Proc. Priv. Enhancing Technol.*, 2019(1):47–67, January 2019.

[66] Esha Sarkar, Eduardo Chielle, Gamze Gürsoy, Oleg Mazonka, Mark Gerstein, and Michail Maniatakos. Fast and scalable private genotype

imputation using machine learning and partially homomorphic encryption. *IEEE Access*, 9:93097–93110, June 2021.

[67] Vasily Sidorov, Ethan Yi Fan Wei, and Wee Keong Ng. Comprehensive performance analysis of homomorphic cryptosystems for practical data processing. February 2022.

[68] Jun Jie Sim, Fook Mun Chan, Shibin Chen, Benjamin Hong Meng Tan, and Khin Mi Mi Aung. Achieving GWAS with homomorphic encryption. *BMC Med. Genomics*, 13(Suppl 7):90, July 2020.

[69] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. 71(1):57–81.

[70] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71:57–81, 2014.

[71] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000*, pages 44–55. IEEE, 2000.

[72] Yosuke Tanigawa, Junyang Qian, Guhan Venkataraman, Johanne Marie Justesen, Ruilin Li, Robert Tibshirani, Trevor Hastie, and Manuel A Rivas. Significant sparse polygenic risk scores across 813 traits in UK biobank. *PLoS Genet.*, 18(3):e1010105, March 2022.

[73] Tomoya Tanjo, Yosuke Kawai, Katsushi Tokunaga, Osamu Ogasawara, and Masao Nagasaki. Practical guide for managing large-scale human genome data in research. *J. Hum. Genet.*, 66(1):39–52, January 2021.

[74] U.S. Department of Health and Human Services. Health insurance portability and accountability act. U.S. Government Printing Office, 1996.

[75] Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. HEALER: homomorphic computation of ExAct logistic rEgRession for secure rare disease variants analysis in GWAS. *Bioinformatics*, 32(2):211–218, January 2016.

[76] Robyn Ward and Geoffrey S Ginsburg. Local and global challenges in the clinical implementation of precision medicine, 2017.

[77] Meng Yang, Chuwen Zhang, Xiaoji Wang, Xingmin Liu, Shisen Li, Jianye Huang, Zhimin Feng, Xiaohui Sun, Fang Chen, Shuang Yang, Ming Ni, Lin Li, Yanan Cao, and Feng Mu. TrustGWAS: A full-process workflow for encrypted GWAS using multi-key homomorphic encryption and pseudorandom number perturbation. *Cell Syst*, 13(9):752–767.e6, September 2022.

[78] Junwei Zhou, Botian Lei, Huile Lang, Emmanouil Panaousis, Kaitai Liang, and Jianwen Xiang. Secure genotype imputation using homomorphic encryption. *Journal of Information Security and Applications*, 72:103386, February 2023.

A Supplementary Material, Tables and Figures

Security of homomorphic encryption and RLWE

Most widely-used homomorphic encryption schemes [18, 19, 26, 31], including the BGV scheme we employed, rely on the hardness of a mathematical problem called the Ring Learning with Errors (RLWE) problem, *i.e.,* the ring version of Learning with Errors (LWE) problem. The hardness of the LWE problem has served as a common assumption due to its quantum reduction to a hard lattice problem proposed by Regev [63] and its robustness against various attacks [11, 17]. Later, Lyubashevsky *et al.* [57] introduced the RLWE problem and demonstrated its reduction to the worst-case lattice problems over ideal lattices. Compared to LWE-based cryptographic schemes, RLWE-based schemes are more efficient, especially for power-of-2 cyclotomic rings [64].

To delve deeper into the RLWE problem, we introduce the following notation: the $\mathfrak{m}$-th cyclotomic polynomial $\Phi_{\mathfrak{m}}(X)$ of degree $\mathfrak{n}$, an integer-coefficient polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\Phi_{\mathfrak{m}}(X)$, a modulus Q and the quotient ring $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$, an error distribution χ, and $\leftarrow$ for a uniform sampling.

Given a secret $s \in \mathcal{R}_Q$, the hardness of RLWE guarantees that the following two distributions in $\mathcal{R}_Q \times \mathcal{R}_Q$ are (computationally) indistinguishable under appropriately chosen parameters:

$$\mathcal{D}_s = \{(a, b) : a \leftarrow \mathcal{R}_Q, e \leftarrow \chi, b = as + e\}, \mathcal{D}_{\text{rand}} = \{(a, b) : a, b \leftarrow \mathcal{R}_Q\}.$$

As such, the secret s is masked using a pair of elements, which are indistinguishable from two uniform elements. To achieve the desired λ-bit security in RLWE-based HE schemes, parameters $(\mathfrak{m}, Q)$ need to be chosen such that the best-known LWE attacks require at least 2^λ operations. Concrete parameters are generally estimated following [9], which are also standardized in the white paper [8].

Experimental setup and HE parameters

We benchmarked SQUiD on an *n2-standard-64* Google Cloud instance with an Intel Xeon Gold 6268 @ 2.8 GHz processor and 256 GB of memory. For all our benchmarking, we employed parameters with 128-bits of security according to the LWE-estimator from HElib [9, 42]. See our list of parameters in Table 3.

Parameter	SQUiD
$\mathfrak{m}$	99,928
$\mathfrak{t}$	99,929
$\mathfrak{n}$	49,960
log Q	967
λ	128
Slot size (ℓ)	49,960
Max # of patients	2 billion
Minimum server memory (GB)	60

Table 3: Table showing the HE parameters for SQUiD where $\mathfrak{m}$ is the dimension of the cyclotomic ring, t is the plaintext modulus, $\mathfrak{n}$ is the polynomial degree, $\log Q$ is the bit length of the ciphertext modulus, λ is the security parameter, slot size is the maximum number of patients that can be stored in a single ciphertext for a single attribute, Max # of patients is the maximum number of patients that can be stored in the database, and Minimum server memory gives a recommendation for the memory requirement of the server to run SQUiD.

API parameters for queries

Count: Our simplest query counts the number of patients who pass a filter. This filter can be either conjunctive (And) or disjunctive (Or) and consist of only equality causes. For example in Fig. 4A, we have the query "COUNT WHERE $\text{SNP}_2 == 1$ AND sex = 1" which will count the number of female patients where the second SNP is 1. We compute this query by first computing a predicate for each patient if they pass the filter. Since homomorphic encryption only supports addition and multiplication, we compute the filter consisting of AND gates, OR gates, and equality function using polynomial approximations. The domain of the inputs to all these functions is limited to (0,1,2) for SNPs and (0,1) for all other variables in our database. Thus, the degree of these polynomials is small which keeps SQUiD performant.

MAF: Our minor allele frequency (MAF) query computes the MAF of a target SNP on a filtered cohort of patients filtered using a similar filtering technique from the count query. Our MAF queries return two values, the allele count and the number of filtered patients times two. The researcher has to finalize the MAF computation by dividing the returned allele count by the returned number of filtered patients times two. We move the division operation to the research because division is an expensive operation to perform homomorphically.

PRS: Our polygenic risk score (PRS) query computes a PRS given a set of SNPs and a set of effect sizes. A PRS score is calculated with a linear combination of the SNP and the effect sizes which we can naturally compute

Query	Parameters	Description
Count	Filter: array of pairs of strings and ints Conjunctive: boolean	Filter is [("rs7903146",2),("T2D",1)] Conjunctive is true Returns the number of patients in the database that have both alleles of the rs7903146 SNP and T2D.
MAF	Filter: array of pairs of strings and ints Conjunctive: boolean Target SNP: string	Filter is [("T2D",1)] Conjunctive is true Target SNP is "rs7903146" Returns the MAF for the rs7903146 SNP of patients in the database that have T2D.
PRS	PRS: array of pairs of strings and ints	Parameters are [("rs12562034", 4), ("rs11240779", 10), ("rs4970383", -4)] Returns the PRS score for each patient by summing up the multiplication of the rs12562034 SNP by 4, the rs11240779 SNP by 10, and the rs4970383 SNP by -4.
Similarity	Target patient: ciphertext Threshold: int Disease column: string	Target patient is MO8v3J7kJt (Encrypted) Threshold is 10 Disease column is T2D Returns the number of patients with and without T2D who are similar to the target patient by a threshold of 10 using a L_2 similarity score.

Table 4: List of parameters for the count, MAF, PRS, and similarity queries with examples.

using homomorphic encryption. The effect sizes, which are often represented as floating point numbers, are scaled to integers within SQUiD as SQUiD only supports integer inputs. The resulting scores are scaled down accordingly.

Similarity: The similarity query counts the number of patients with and without a disease from a cohort consisting of patients similar to the target query patient. This target patient is encrypted with the data owner's public key before it is sent to the cloud to protect the target patient data. To compute the similarity query, we first compute a similarity cohort by scoring the similarity of the target patient with every patient in the database. We score patients based on their squared Euclidean distance (squared L_2 distance) from the target patient. If these scores are beneath a predetermined threshold, the patient

is considered similar to the target patient and added to the similarity cohort. Next, we count the number of patients with and without a disease in this cohort.

The BGV scheme

Brakerski-Gentry-Vaikuntanathan (BGV) [19] is a well-known lattice-based homomorphic encryption scheme that allows for computations over encrypted data. Its lattice-based structure further provides reasonable quantum resistance. Realizing computations as sequences of additions and multiplications, BGV provides a large computation capacity with reasonable parameter choices. Furthermore, it supports computations in a SIMD manner, which significantly enhances the amortized performance.

Below we describe the basic procedures in BGV, the realization of public key-switching and its noise control. Mathematical and cryptographic notations are listed in Table 5.

Mathematical Basics:	
t	plaintext modulus
Q	ciphertext modulus
$\mathbb{Z}$	set of integers
$\mathbb{Z}_t$	the ring of integers modulo t, *i.e.*, $\mathbb{Z}_t = \mathbb{Z}/t\mathbb{Z}$
$\mathbb{Z}_Q$	the ring of integers modulo Q, *i.e.*, $\mathbb{Z}_Q = \mathbb{Z}/Q\mathbb{Z}$
X	variable of polynomial
R	the cyclotomic ring $\mathbb{Z}[X]/\left(\Phi_{\mathfrak{m}}(X)\right)$ where $\Phi_{\mathfrak{m}}(X)$ is the $\mathfrak{m}$-th cyclotomic polynomial. The degree of $\Phi_{\mathfrak{m}}(X)$ is $\mathfrak{n} = \phi(\mathfrak{m})$, where $\phi(\cdot)$ denotes the Euler totient function.
R_t	$\mathbb{Z}_t[X]/\Phi_{\mathfrak{m}}(X)$
R_Q	$\mathbb{Z}_Q[X]/\Phi_{\mathfrak{m}}(X)$
HE Parameters:	
χ_{ter}	uniform ternary distribution with coefficients from $\{-1, 0, 1\}$
χ_{err}	error distribution of scheme, typically a discrete Gaussian distribution unless specified otherwise

Table 5: Notations of BGV homomorphic encryption scheme

Key generation, encryption and decryption. Sample the secret key sk from χ_{ter}, and the public key is computed as follows

$$\mathsf{pk} = \left([a \cdot \mathsf{sk} + te]_Q, -a\right) \in R_Q^2,$$

where $a \leftarrow u_Q$ and $e \leftarrow \chi_{\mathsf{err}}$. Anyone can see the public key, but this leaks no information on the secret key sk, as guaranteed by the hardness of the ring learning-with-errors (RLWE) problem.

With the public key, anyone can generate a ciphertext for a plaintext m by computing

$$\mathsf{ct} = \mathbf{Enc}_{\mathsf{pk}}(m) = \left([[m]_t + u \cdot \mathsf{pk}_0 + te_0]_Q \, , [u \cdot \mathsf{pk}_1 + te_1]_Q \right),$$

where $u \leftarrow \chi_{\mathsf{ter}}$ and $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$. Due to the randomness in encryption, ciphertexts of the same plaintext m are always not identical, reflecting the IND-CPA security which avoids the search pattern leakage.

A message m and its ciphertext $(\mathsf{ct}_0, \mathsf{ct}_1)$ are related via

$$\mathsf{ct}_0 + \mathsf{ct}_1 \cdot \mathsf{sk} = [m]_t + tv \pmod{Q},$$

where $v = u \cdot e + e_1 \cdot \mathsf{sk} + e_0$ is the noise term.

Only those knowing the secret key sk will be able to decrypt. Decrypting a ciphertext $ct = (ct_0, ct_1)$ is to calculate

$$[\mathsf{ct}_0 + \mathsf{ct}_1 \cdot \mathsf{sk} \pmod{Q}]_t \, ,$$

and it decrypts to the correct message m if the noise term v is lower than $\lfloor Q/t \rfloor$.

Public key-switching. The key-switching technique is used to switch the secret key of a given ciphertext without decryption. Given a ciphertext ct that encrypts a message m under secret key sk, the goal is to obtain a new ciphertext that encrypts the same message m under another secret key sk^*.

To compute this, the algorithm requires an additional ciphertext that encrypts the secret key sk under sk^*, which is called the key-switching key $\mathsf{ksk}_{(\mathsf{sk} \to \mathsf{sk}^*)}$. In the BGV scheme, this key is derived from sk^*, but we use an alternative derivation with pk^*, the corresponding public key of sk^*, as follows:

$$\mathsf{ksk} = \left([\mathsf{sk} + u^* \cdot \mathsf{pk}_0^* + te_0^*]_Q \, , [u^* \cdot \mathsf{pk}_1^* + te_1^*]_Q \right) \in R_Q^2,$$

where $\mathsf{pk}^* = (\mathsf{pk}_0^*, \mathsf{pk}_1^*)$, $u^* \leftarrow \chi_{\mathsf{key}}$ and $e_0^*, e_1^* \leftarrow \chi_{\mathsf{err}}$. In other words, ksk is just $\mathbf{Enc}_{\mathsf{pk}^*}(\mathsf{sk})$, i.e. the encryption of sk under pk^*.

With ksk and $(\mathsf{ct}_0, \mathsf{ct}_1)$ which decrypt to m under sk, we can construct $(\mathsf{ct}_0^*, \mathsf{ct}_1^*)$ which also decrypt m but under sk^*. Precisely, we construct

$$(\mathsf{ct}_0^*, \mathsf{ct}_1^*) = \left([\mathsf{ct}_0 + \mathsf{ct}_1 \cdot \mathsf{ksk}_0]_Q \, , [\mathsf{ct}_1 \cdot \mathsf{ksk}_1]_Q \right).$$

Therefore,

$$\begin{aligned}\mathsf{ct}_0^* + \mathsf{ct}_1^* \cdot \mathsf{sk}^* &= \mathsf{ct}_0 + \mathsf{ct}_1 \cdot \mathbf{Enc}_{\mathsf{pk}^*}(\mathsf{sk})_0 + ct_1 \cdot \mathbf{Enc}_{\mathsf{pk}^*}(\mathsf{sk})_1 \cdot \mathsf{sk}^* \\ &= \mathsf{ct}_0 + \mathsf{ct}_1 \cdot (\mathbf{Enc}_{\mathsf{pk}^*}(\mathsf{sk})_0 + \cdot \mathbf{Enc}_{\mathsf{pk}^*}(\mathsf{sk})_1 \cdot \mathsf{sk}^*) \\ &= \mathsf{ct}_0 + \mathsf{ct}_1 \cdot (\mathsf{sk} + tv^*) \\ &= m + t(v + \mathsf{ct}_1 \cdot v^*) \mod Q,\end{aligned}$$

where $v^* = u^* \cdot e^* + e_1^* \cdot \mathsf{sk}^* + e_0^*$, verifying $(\mathsf{ct}_0^*, \mathsf{ct}_1^*)$ which also decrypts m but under sk^*.

Reducing noise during public key-switching. The public key-switching above has noise component $ct_1 \cdot v^*$, whose size can be further reduced by two methods: coefficient digit decomposition [20], and a temporary enlargement of ciphertext modulus [35]. We combine the two methods in an approach similar to the Section 4.3 of [40], where the ciphertext modulus $Q = \prod_{j=1}^{l} D_j$ is decomposed into l coprime and odd digits, and an expansion factor P, which is odd and coprime to Q, is introduced.

As such, the key-switching key $\mathsf{ksk}_{(\mathsf{sk} \to \mathsf{sk}^*)}$ is no longer a ciphertext with two components, but a matrix of dimension $2 \times l$. For $1 \leq j \leq l$, the j-th column of this matrix is $\mathbf{Enc}_{\mathsf{pk}^*}(P\check{D}_j \, \mathsf{sk})$ with respect to an enlarged ciphertext modulus PQ, where $\check{D}_j$ stands for the product $D_1 \cdots D_{j-1}$.

Storage costs of ciphertexts and public key-switching keys. The size of a ciphertext is dependent on the HE parameters used. With the parameters used in SQUiD, a ciphertext is a pair of polynomials of degree $\mathfrak{n} = 49,960$ with ciphertext modulus $Q \approx 967$ bits. The theoretical minimum storage would be

$$\begin{aligned}12 \text{ MB} \approx\ & 2 \text{ polynomials per ciphertext} \\ & \times 49{,}961 \text{ coefficients per polynomial} \\ & \times 967 \text{ bits per coefficient.}\end{aligned}$$

In practice, we measured the storage of a single ciphertext to be 13 MB.

With the coefficient digit decomposition and modulus enlargement techniques, the *ksk* is a $2 \times l$ matrix of polynomials with enlarged ciphertext modulus PQ. In practice, we set $l = 3$ and $PQ \approx 1,291$ bits. Thus, the theoretical minimum storage of a *ksk* would be

$$\begin{aligned} 48 \text{ MB} \approx 2 \times 3 &\text{ polynomials per } ksk \\ &\times 49{,}961 \text{ coefficients per polynomial} \\ &\times 1{,}291 \text{ bits per coefficient.} \end{aligned}$$

In practice, we measured it to be 55 MB.

Homomorphic operations. Adding two ciphertexts $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ and $\mathsf{ct}' = (\mathsf{ct}'_0, \mathsf{ct}'_1)$ that encrypt m and m' with a same key sk respectively gives

$$(\mathsf{ct}_0 + \mathsf{ct}'_0) + (\mathsf{ct}_1 + \mathsf{ct}'_1) \cdot \mathsf{sk} = [m + m']_t + t(v + v' + u) \pmod{Q},$$

so the ciphertext $([\mathsf{ct}_0 + \mathsf{ct}'_0]_Q, [\mathsf{ct}_1 + \mathsf{ct}'_1]_Q)$ is an encryption of $m + m' \pmod{t}$ under almost additive noise.
Multiplication of two ciphertexts is more complex, which involves taking the tensor product of two ciphertexts as polynomial vectors to obtain $(\mathsf{ct}_0\mathsf{ct}'_0, \mathsf{ct}_0\mathsf{ct}'_1 + \mathsf{ct}_1\mathsf{ct}'_0, \mathsf{ct}_1\mathsf{ct}'_1)$. One can check that

$$\mathsf{ct}_0\mathsf{ct}'_0 + (\mathsf{ct}_0\mathsf{ct}'_1 + \mathsf{ct}_1\mathsf{ct}'_0) \cdot \mathsf{sk} + \mathsf{ct}_1\mathsf{ct}'_1 \cdot \mathsf{sk}^2 = m \cdot m' + tv_0 \pmod{Q},$$

where $v_0 = m \cdot v' + m' \cdot v + v \cdot v'$.

Here, an additional step is needed for the term with sk^2. If we consider $(0, \mathsf{ct}_1\mathsf{ct}'_1)$ as a ciphertext with secret key sk^2, then performing the key switching procedure with the key $\mathsf{ksk}_{\mathsf{sk}^2 \to \mathsf{sk}}$ gives a ciphertext $(\mathsf{ct}''_0, \mathsf{ct}''_1)$. The noise control in this step is similar to the previous section. The final output of the homomorphic multiplication is $(\mathsf{ct}_0\mathsf{ct}'_0 + \mathsf{ct}''_0, \mathsf{ct}_0\mathsf{ct}'_1 + \mathsf{ct}_1\mathsf{ct}'_0 + \mathsf{ct}''_1)$.

SIMD batching. The BGV scheme supports homomorphic operations on multiple plaintext slots simultaneously. This follows from the fact that the cyclotomic polynomial $\Phi_{\mathfrak{m}}(X)$ of degree $\mathfrak{n}$ splits modulo t into ℓ irreducible factors of same degree $\mathfrak{n}/\ell$, *i.e.*, $\Phi_{\mathfrak{m}}(X) = \prod_{i=1}^{\ell} F_i(X)$. Leveraging the Chinese Remainder Theorem (CRT), the following ring isomorphism

$$R_t \cong \prod_{i=1}^{\ell} \mathbb{Z}_t[X] / \big(F_i(X)\big)$$

is established, which enables the encoding of ℓ messages $\{z_1 \ldots, z_\ell\} \in \prod_{i=1}^{\ell} \mathbb{Z}_t[X] / \big(F_i(X)\big)$ into a single plaintext in R_t. Typically, each of the ℓ messages is called a *slot*, and altogether they are regarded as a length-l vector. Since computations over a ciphertext are performed on all packed values, BGV provides efficient computations in an amortized manner.

Homomorphic rotation. BGV supports an additional operation called *rotation*, which permutes plaintext slots circularly. Specifically, let ct be an encryption of a plaintext vector $\mathbf{z} = (z_1, z_2, \ldots, z_\ell)$. Performing a (right) rotation on ct by v results in a new ciphertext $\mathsf{ct}_{\mathsf{rot}}$ encrypting the plaintext vector $\mathbf{z}_{\mathsf{rot}} = (z_{v+1}, z_{v+2}, \ldots, z_\ell, z_1, \ldots, z_v)$ under the same secret key.

The homomorphic rotation consists of two key components: automorphism and key switching. We refer interested readers to Section 3 of [40] for the general hypercube structures of rotations in BGV.

Polynomial evaluation using the Paterson-Stockmeyer method

Polynomial evaluations require numerous binary operations including additions, scalar multiplications (where one operand is a constant), and non-scalar multiplications. The Paterson-Stockmeyer method [60] uses fewer non-scalar multiplications, such that a degree-d polynomial is evaluated using $\mathcal{O}(\sqrt{d})$ non-scalar multiplications.

In the homomorphic evaluation of polynomials, non-scalar multiplications are translated to ciphertext-ciphertext multiplications, whose evaluation costs are much more expensive than the other two. According to the benchmark in [33], it is around 160× and 15× the cost of homomorphic evaluations of additions and scalar multiplications, respectively. Therefore, the Paterson-Stockmeyer method has been widely used for polynomial evaluations in homomorphic encryption [24, 27, 28, 43].

Below we follow [28] to sketch the evaluation of a degree-d polynomial $f(x) = \sum_{i=0}^{d} a_i x^i$ using the Paterson-Stockmeyer method. Assume there exist integers L and H such that $d = LH - 1$ and $L \approx \sqrt{2(B+1)}$. Then the polynomial can be rewritten into

$$f(x) = \sum_{i=0}^{H-1} \Big(\sum_{j=0}^{L-1} a_{iL+j} \cdot x^j\Big) \cdot x^{iL}.$$

Therefore, the "low powers" $\{x, x^2, \ldots, x^{L-1}\}$ can be computed with $L-2$ non-scalar multiplications, whose linear combinations give the inner sum. The "high powers" $\{x^L, x^{2L}, \ldots, x^{(H-1)L}\}$ are then computed with $H-1$ non-scalar multiplications, whose subsequent products with the inner sum require another $H-1$ non-scalar multiplications. In total, the procedure requires

$$L - 2 + 2(H-1) = L + 2H - 4$$

non-scalar multiplications, which is minimal and achieves $\mathcal{O}(\sqrt{B})$ when $L \approx \sqrt{2(B+1)}$.

Comparison of SQUiD's MAF calculation with existing methods

In the MAF protocol by Kim and Lauter [48], the MAF for SNP j is computed as follows:

$$\mathsf{MAF}(j) = \frac{\min(m_j, 2r - m_j)}{2r} \quad \text{with} \quad m_j = \sum_{i=1}^{r} m_{i,j},$$

where r represents the number of patients in the database, and $m_{i,j}$ denotes the genotype of patient i at SNP j.

In SQUiD, a cohort is initially created before the MAF computation. The membership of the cohort is defined by a predicate vector p, which is 1 for patients in the cohort and 0 otherwise. The key distinction in the MAF computation between SQUiD and [48] lies in the need for multiplication of each $m_{i,j}$ term within the summation by the corresponding predicate, ensuring the inclusion of only those patients who are part of the cohort. Additionally, the total number of patients r in the denominator is a constant in [48], but in SQUID it varies for different cohort sizes and needs to be computed as the sum of predicates. All other parts of the MAF computation are the same in both SQUiD and [48]. That is, a SQUiD MAF query without a filter has the same computation and result as [48]. Furthermore, in both SQUiD and [48], the division and minimum operations are executed in plaintext. In summary, the total runtime and accuracy of the SQUiD MAF calculation and that of [48] are expected to be exactly the same.

Continuous phenotype values

SQUiD supports continuous phenotype values such as weight, blood pressure, heart rate, etc. However, as SQUiD exclusively processes integer data inputs, these values need to be discretized into integers by scaling the values. This discretization occurs during the data encryption by the data owner before transmitting it to the cloud. Once in the cloud, SQUiD effectively filters these now integer values for counting and MAF queries using range filters. Range filters are set by an upper and lower bound and are computed using the same comparisons thresholds from the similarity query.

We benchmarked the range query performance in Additional file 1: Fig. 20. We found that it took approximately 51 minutes to compute a count query with a range filter on 49,960 patients and 58 minutes to compute a MAF query with the same parameters.

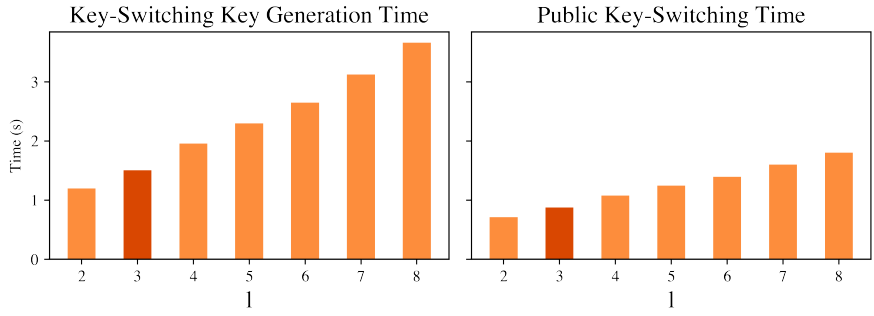


Figure 9: The runtime for key-switching key generation and key-switching by the number of digits used in the decomposition (l). Digit decomposition is a technique to reduce the error growth in ciphertexts. The more we decompose the ciphertext and key-switching key, the more secure our system becomes and the less error growth the ciphertext experiences [40]. We set $l = 3$, as is commonly chosen. It takes 1.5 seconds to generate a key-switching key when the digits are decomposed into three parts.

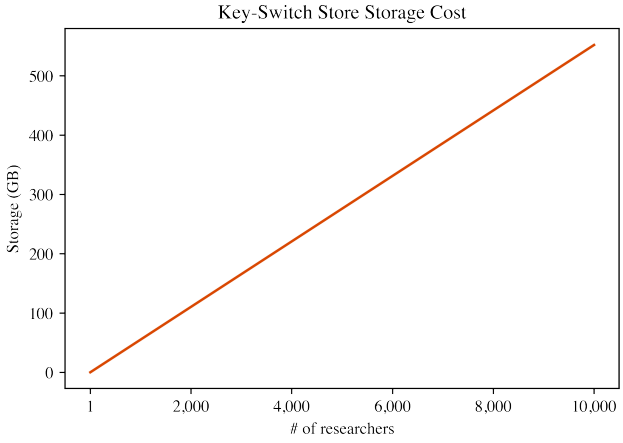


Figure 10: The additional storage space for the public key-switching key store (with $l = 3$) by the number of authorized researchers in the store. Since each key-switching key is a single ciphertext, the storage required remains minimal at approximately 55 gigabytes (GB) for 1,000 researchers.

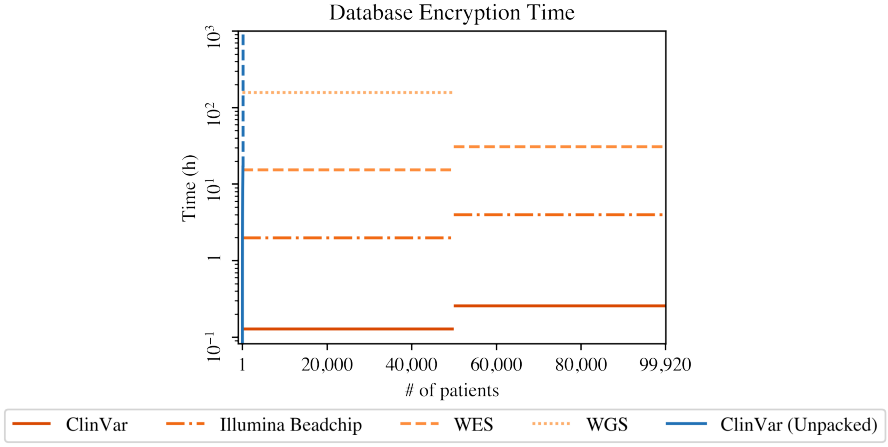


Figure 11: The time to encrypt databases with various SNP sets by the number of patients in the database. We measured the setup time by encrypting the various databases using 60 threads running simultaneously. We compared the setup of SQUiD for the ClinVar SNP set to the setup time of an unpacked HE solution (blue) for the ClinVar SNP set. The ClinVar (Unpacked) line was extrapolated for databases that took more than a day to generate.

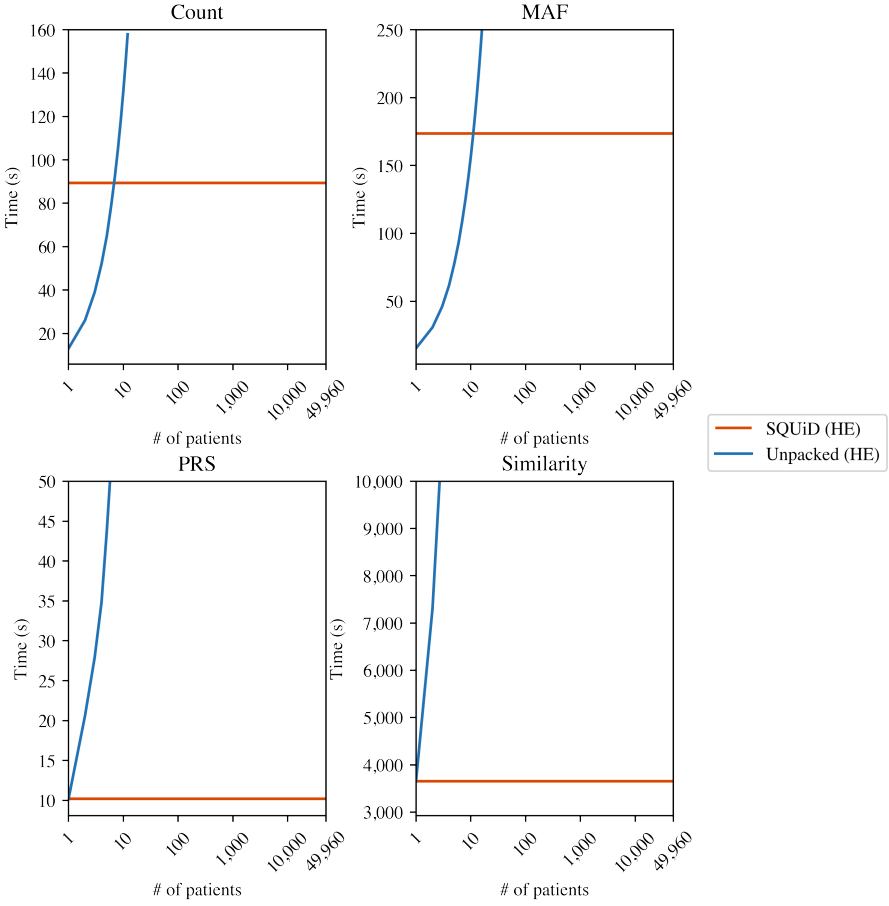


Figure 12: The unpacked and packed query performance for the count, MAF, PRS, and similarity queries on databases with varying numbers of patients (ranging from 1 patient to 49,960 patients with 1,024 SNPs). The similarity and PRS queries were tested with 1,024 effect sizes and SNPs, respectively, and the count and MAF queries were tested with 2 filters.

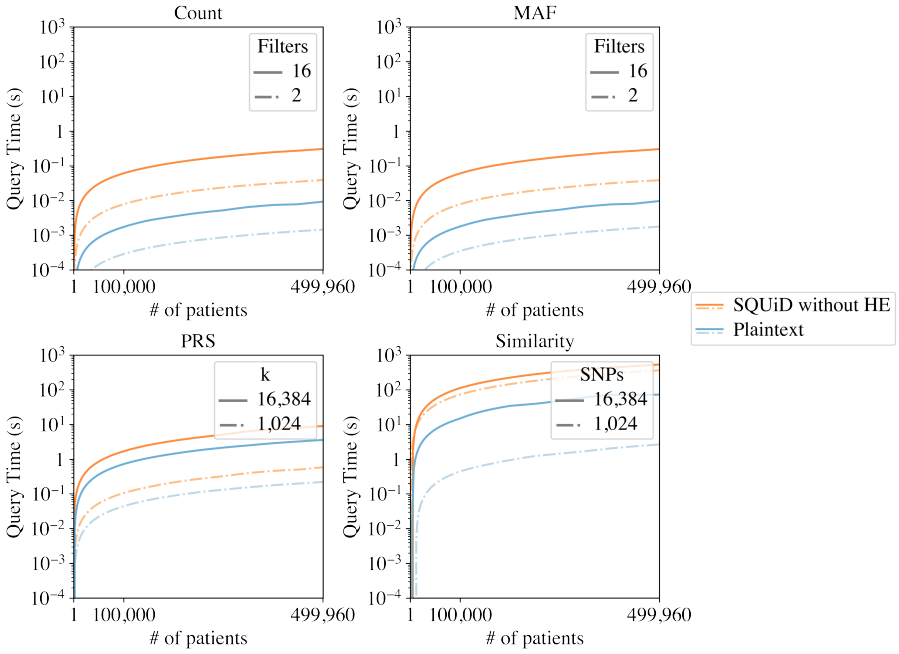


Figure 13: The query time for SQUiD without HE and plaintext. SQUiD without HE implements the queries from SQUiD by translating HE library functionalities to their plaintext counterparts and then using these functionalities in the same way as SQUiD. We ran the count and MAF queries with 2 filters, the PRS query with 1,024 effect SNPs (k), and the similarity query with 1,024 SNPs.

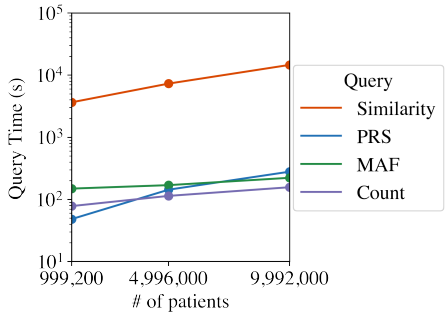


Figure 14: The time to perform count, MAF, PRS and similarity queries on databases with an increasing number of patients. The count and MAF queries were run with 2 filters. The PRS query had 1,024 effect SNPs, and the similarity query had 1,024 SNPs. Each query was executed in parallel with 50 threads.

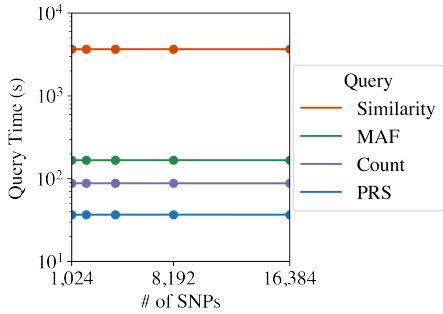


Figure 15: The time to perform a count query with 2 filters, MAF query with 2 filters, PRS with 1,024 effect SNPs, and a similarity queries with 1,024 SNPs on databases with increasing number of SNPs.

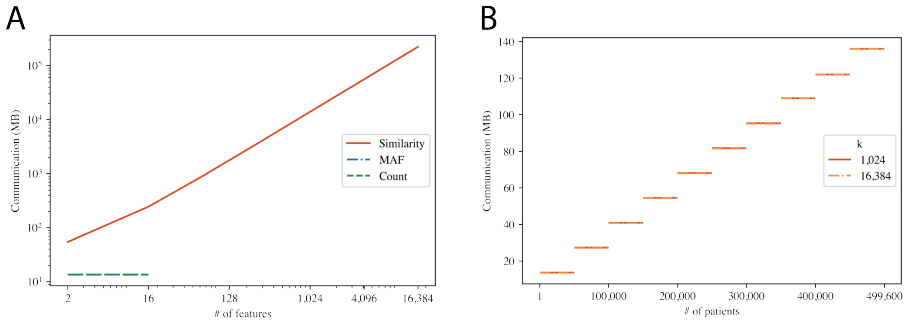


Figure 16: The communication cost for all queries by the number of features in the query or the number of patients in the database. We benchmark our communication cost by measuring the end-to-end communication of a single query. Each query needs one communication round thus the total communication is the cost of receiving a query from a client and sending back the computation result. We separated the communication performance into two plots because the scaling for our queries depend on different factors. (A) We show the communication costs of the count, MAF, and similarity queries by the number of features (filters for the count and MAF query, and SNPs for the similarity query). (B) We show the communication of the PRS query by the number of patients in the database.

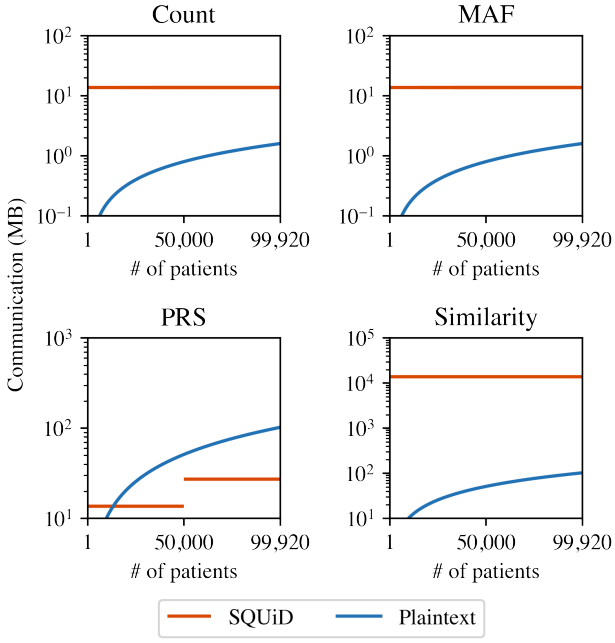


Figure 17: The communication cost for all queries and plaintext solutions by the number of patients in the database. The plaintext database keeps the data encrypted at rest, packing 16 one byte SNPs into a single encrypted 128-bit AES block. When the database receives a query, it sends the encrypted data necessary to compute the query. Thus, the client has to decrypt and compute the query themselves. We benchmarked for all four types of queries with 16 filters for the count and MAF queries, 1,024 effect SNPs for the PRS query, and the 1,024 SNPs for the similarity query. Compared to SQUiD, the plaintext communication always scales linearly with the number of patients while this is only true for SQUiD for PRS queries (since a PRS score for each patient needs to be returned).

Figure 18: Screenshot of SQUiD command line interface (CLI). **(A)** Welcome help message about showing core SQUiD functionalities. **(B)** setConfig call sets the address, port, and API key used to communicate with the server for all successive calls. **(C)** getContext call that pulls the context from the server to ensure the encryption schemes are synced locally and on the server. **(D)** genKeys call creates a public and private key for the user. **(E)** authorize call sends the public key to the server for authorization. The server generates a key-switching key which will be used to re-encrypt all query results sent back to the user to be encrypted under the user's public key. **(F)** count query call which counts the number of patients in the database with a first SNP that has value 1. **(G)** failed query call that shows what parameters should be supplied for a correct query call. **(H)** MAF query call that has the correct parameters.

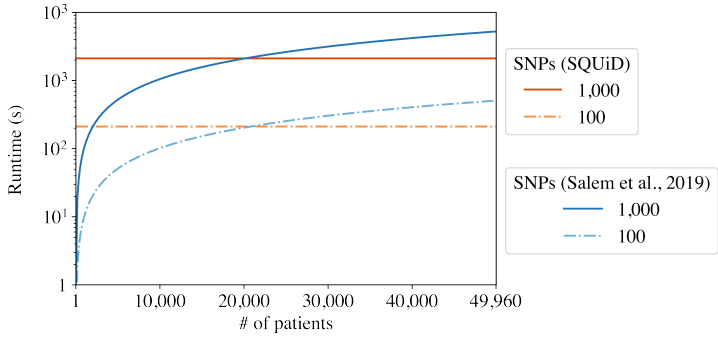


Figure 19: The runtime of the L_2 similarity score computation for SQUiD and [65] (Salem et al., 2019) for 100 SNPs and 1,000 SNPs. For score computation with fewer than 20,000 patients, [65] exhibits faster performance for both 100 and 1,000 SNPs. However, as the patient dataset scales up, SQUiD consistently outperforms [65], demonstrating its superior efficiency in handling larger datasets.

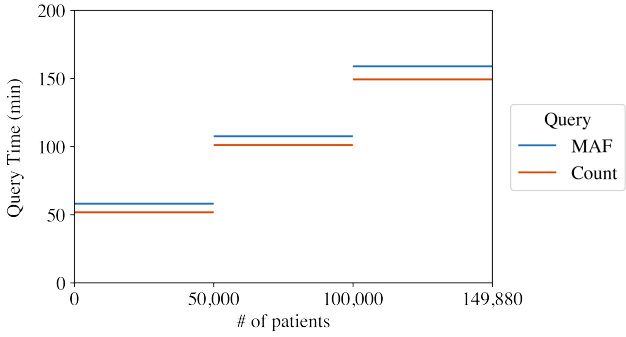


Figure 20: The query time for the count and MAF query with a range filter by the number of patients in the database. Each query only had one range filter.

Chapter 10

Blind zkSNARKs for Private Proof Delegation and Verifiable Computation over Encrypted Data

Publication Data

Gama, M., Beni, E. H., Kang, J., Spiessens, J., and Vercauteren, F. Blind zkSNARKs for private proof delegation and verifiable computation over encrypted data. *IACR Communications in Cryptology 2*, 3 (2025)

Contribution

The high-level description of the protocol was conceived in collaboration with other authors. I mainly contributed to the FHE analysis and the proof of decryption.

Blind zkSNARKs for Private Proof Delegation and Verifiable Computation over Encrypted Data

Mariana Gama[1], Emad Heydari Beni[1,2], Jiayi Kang[1], Jannik Spiessens[1], and Frederik Vercauteren[1]

[1] COSIC, KU Leuven, Belgium
[2] Nokia Bell Labs, Antwerp, Belgium

Abstract. In this paper, we show for the first time it is practical to privately delegate proof generation of zkSNARKs to a single server for computations of up to 2^{20} R1CS constraints. We achieve this by computing zkSNARK proof generation over homomorphic ciphertexts, an approach we call blind zkSNARKs. We formalize the concept of blind proofs, analyze their cryptographic properties and show that the resulting blind zkSNARKs remain sound when compiled using BCS compilation. Our work follows the framework proposed by Garg et al. (Crypto'24) and improves the instantiation presented by Aranha et al. (Asiacrypt'24), which implements only the FRI subprotocol. By delegating proof generation, we are able to reduce client computation time from 10 minutes to mere seconds, while server computation time remains limited to 20 minutes. We also propose a practical construction for vCOED supporting constraint sizes four orders of magnitude larger than the current state-of-the-art verifiable FHE-based approaches. These results are achieved by optimizing Fractal for the GBFV homomorphic encryption scheme, including a novel method for making homomorphic NTT evaluation packing-friendly by computing it in two dimensions. Furthermore, we make the proofs publicly verifiable by appending a zero-knowledge Proof of Decryption (PoD). We propose a new construction for PoDs optimized for low proof generation time, exploiting modulus and ring switching in GBFV and using the Schwartz-Zippel lemma for proof batching; these techniques might be of independent interest. Finally, we implement the latter protocol in C and report on execution time and proof sizes.

1 Introduction

Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) are undeniably one of the most promising advanced cryptographic protocols developed in the last decade. Simply put, a zkSNARK allows one to generate a proof $\pi \leftarrow \mathsf{Prove}_F(u, y)$ that some output y is the result of $F(u)$. The proof π can be verified in sublinear cost compared to the computation F itself, a property known as *succinctness*. Additionally, the proof π does not leak any private data required for (or generated by) computing F, a property known as *zero-knowledge*. Because of these unique properties, zkSNARK proofs are particularly useful in two main categories of real-world applications.

The first category is Privacy-Enhancing Technologies (PETs), where zkSNARKs and their subcomponents allow one to produce an efficiently verifiable statement of any complexity while retaining privacy. Examples of applications are anonymous credential systems [77] (for proving possession of some private credentials) and private transaction systems [10] (for proving that a private ledger has been correctly updated). Note that in these settings, it is the user who performs the costly proving operation, limiting the complexity of the statement to be proven. Solutions for this problem were proposed in [35, 48, 75], where multi-party computation was used to build a zero-knowledge proof delegation (zkDel) scheme. Even though this approach is more efficient than single-server delegation, it requires selecting a group of external servers who are trusted not to collude and is, therefore, harder to bootstrap in practice.

Secondly, zkSNARKs are also used to add verifiability to outsourced computations (e.g., in Cloud Computing scenarios [30, 44]). The user outsources a computation to the server and then verifies a zkSNARK proof to check whether the returned output is the result of the requested computation. Since the prover cannot compute the proof without knowing the witness, the user has to reveal their inputs and outputs to the server. This can be avoided by using verifiable Computation Over Encrypted Data (vCOED). Possible constructions for vCOED were first described by Gennaro et al. [51] and have received more academic interest in the last years following the emergence of zkSNARKs. These schemes focus on proving the correct execution of homomorphic computations using proof systems and are better known as verifiable Fully Homomorphic Encryption (vFHE). Although recent works [6] have achieved leaps in performance, they struggle with arithmetizing the maintenance operations in homomorphic encryption (HE) schemes.

In [49], Garg et al. proposed a technique for constructing both zkDel and vCOED schemes by applying HE schemes to proof systems (as opposed to applying proof systems to HE schemes, as in vFHE). Generally, an HE

scheme $\mathcal{E}$ allows generating a ciphertext $\mathtt{ct}[u] \leftarrow \mathsf{Enc}(u)$ and computing $\mathtt{ct}[y] \leftarrow \mathsf{Hom}_F(\mathtt{ct}[u])$, where $\mathtt{ct}[y]$ is an encryption of $y = F(u)$; in other words, it allows homomorphically computing on ciphertexts. The technique presented in [49] can be described as replacing the prover computation by $\mathtt{ct}[\pi] \leftarrow \mathsf{Hom}_{\mathsf{Prove}_F}(\mathtt{ct}[u], \mathtt{ct}[y])$, i.e., homomorphically computing the proving computation. In the zkDel setting, this allows the prover to delegate the costly proof generation to a single untrusted server. In the vCOED setting, it promises better performance than vFHE since it proves the operations in the plaintext space instead of the ciphertext space, avoiding the need to arithmetize HE schemes. In [5], Aranha et al. consider a similar approach and present a concrete instantiation of this technique for the vCOED setting. Our paper further extends this line of research; we refer to Section 3 for a thorough comparison with previous works. In particular, we answer the following fundamental questions:

1. What security and privacy guarantees do these constructions provide?
2. Can these constructions achieve practical performance?
3. Can these constructions be efficiently publicly verified?

To answer the first question, we construct a theoretical framework for a new primitive we call blind proofs by defining a blind variant of holographic Interactive Oracle Proofs (hIOPs) and zkSNARKs. We also define the completeness, zero-knowledge and soundness properties and prove that they hold for blind hIOPs. We then show that the corresponding properties also hold for blind zkSNARKs, which can be obtained from blind hIOPs by applying the usual BCS compilation [12].

To answer the second question, we provide the first complete method for computing blind zkSNARKs, which remains practical even for large computations. We optimize the Fractal zkSNARK [37] so that it can be efficiently evaluated using the recent GBFV homomorphic encryption scheme [50] and design specialized homomorphic circuits using two-dimensional NTTs.

Finally, we address the last question by showing that public verifiability can be achieved through Proofs of Decryption (PoDs). We propose a state-of-the-art construction incorporating two techniques from homomorphic encryption, modulus switching and ring switching, and propose a novel method for batching PoDs. We implement[1] our PoD in C and show that proving decryption of several thousands of ciphertexts can be done in a matter of seconds and results in a small 12KB proof.

[1] `https://github.com/KULeuven-COSIC/blind_zkSNARKs`

2 Technical overview

As discussed above, instead of computing Prove_F, the prover computes $\mathsf{Hom}_{\mathsf{Prove}_F}$ to obtain a proof for valid computation of F *without seeing the values that* F *was computed on*. The blindly generated proof should then be decrypted before it can be verified. We coin these schemes *blind proofs* due to their similarity to blind signatures [29], which allow one to sign a message without revealing its content.

Consider, for instance, an interactive proof system that proves the claim $\vec{a} \odot \vec{b} = \vec{c}$ for some vectors of field elements that were previously committed to. Define polynomials f_a, f_b, f_c that interpolate these vectors over some evaluation domain H. Now, notice that a polynomial Z_H that vanishes on H, i.e. $Z_H(a) = 0 \Leftrightarrow a \in H$, will divide the polynomial $f_a \cdot f_b - f_c$ if and only if the claim holds. In other words, the original claim has been reduced to the claim that $Q \coloneqq (f_a \cdot f_b - f_c)/Z_H$ is a polynomial of bounded degree. More generally, this technique can be used to reduce claims about generic computations to low-degree tests of rational functions of polynomials, also known as rational constraints [37].

Moving this proof system to the blind setting requires the prover to perform these proving steps homomorphically, starting from the HE encryptions of $\vec{a}, \vec{b}$ and $\vec{c}$. Akin to all HE applications, achieving practical efficiency requires tailoring the proving computation to homomorphic evaluation. In particular, it includes avoiding nonlinear computations since they require costly multiplications between ciphertexts that can be orders of magnitude more expensive than scalar multiplications depending on the specific HE scheme.

Hence, we select the Fractal proof system [37] since it reduces – using a limited number of non-linear computations – the claim of a generic computation (in the form of an R1CS constraint system) to rational constraints. The remaining proving steps, namely the low-degree tests, can be performed using the FRI [9] proof system, which is completely linear. In broad terms, FRI proves that the evaluation $Q\big|_L$ at some domain $|L| > |H|$ is sufficiently close to a Reed-Solomon (RS) code, i.e., it corresponds to a polynomial of bounded degree. Notice that if the commitments to $\vec{a}, \vec{b}, \vec{c}$ are in the form of vector commitments to $f_a\big|_L, f_b\big|_L, f_c\big|_L$, then the RS code of $Q(X)$ does not need to be explicitly computed, as each of its elements can be efficiently reconstructed from openings to those commitments (e.g. computing $Q(\ell_i)$ from $f_a(\ell_i), f_b(\ell_i)$ and $f_c(\ell_i)$). The fact that Fractal and FRI are especially suitable for the blind setting due to their linearity was first noticed by Garg et al [49].

However, homomorphic operations accumulate noise in the ciphertexts, and HE parameters have to be large enough to accommodate this noise growth. Therefore, besides avoiding non-linear operations, one should also limit the

total depth of the homomorphic circuit. Take, for example, the computation of the "domain extension" from $\vec{a} = f_a\big|_H$ to $f_a\big|_L$, which is also completely linear. One usually computes the coefficients of f_a using an inverse NTT followed by an NTT over the larger domain to compute $f_a\big|_L$. Although NTTs are usually performed using a base-2 Fast Fourier Transform (FFT) to minimize the total number of operations, Aranha et al. [5] have suggested computing them in a larger base b such that the circuit depth is $\log_b(N)$ for a vector of size N. Together with several other optimizations, they were able to provide a practical construction for computing blind FRI for polynomials up to degree 2^{15}.

As a first step towards making blind zkSNARKs practical, we provide new algorithms that are even better tailored to HE evaluations. Thereby, we are able to compute blind FRI for polynomials up to degree 2^{20} in a practical estimated runtime. Crucially, we construct algorithms that can better exploit the Single Instruction Multiple Data (SIMD) capabilities that many HE schemes natively support. This allows us to encrypt a vector $\vec{a}$ of size N into N/P ciphertexts encrypting P plaintext elements each. HE operations performed on these packed ciphertexts will then correspond to element-wise operations on the packing vectors of size P. Therefore, our algorithms are not only fairly linear and correspond to low-depth circuits, they also accommodate some level of parallelism to exploit the SIMD property.

We now revisit the example of blindly computing an NTT. It is well-known that the base-2 FFT algorithm supports local element-wise operations if and only if it simultaneously performs a bit-reversal permutation. In the blind setting, such permutation would significantly increase the cost of subsequent homomorphic operations. Reversing the permutation before the next FFT invocation would also be costly since it requires swapping elements that are packed in different ciphertexts. In this paper, we describe a different approach that orders the N/P ciphertexts in two dimensions – one orthogonal to the SIMD packing and one parallel to the packed vectors – and then computes a two dimensional NTT (2D-NTT) [40]. This allows us to select an optimal homomorphic circuit that balances operation count, circuit depth and parallelism. Specifically, the NTT performed parallel to the packing is performed as a matrix-vector multiplication that uses the baby-step giant-step algorithm to minimize costly HE operations. The second NTT can be computed as a large base butterfly FFT since the permutation it causes will be orthogonal to the ciphertext packing. In Section 6.1, we show that the permutation caused by the 2D-NTT does not significantly increase the HE operation count.

Another challenge that arises when blindly computing proof systems is that they generally rely on large field sizes ($\log|\mathbb{F}| \approx 128$ to 256) for soundness and one would therefore require an HE scheme that supports large plaintext spaces. HE schemes such as BFV [23, 46]/BGV [24] have the plaintext space

$\frac{\mathbb{Z}[X]}{(X^n+1,p)} \cong \mathbb{Z}_p^n$ and therefore support a large packing size n, but have the disadvantage that noise growth scales linearly with p. CLPX [31], on the other hand, uses the plaintext space $\frac{\mathbb{Z}[X]}{(X^n+1,X-b)} \cong \mathbb{Z}_p$ of size $p = b^n + 1$ and therefore supports no packing. However, this allows encrypting plaintexts using lower norm polynomial representatives, thereby significantly decreasing the noise growth. We avoid this dichotomy by using a recent HE scheme named GBFV [50] that uses the generalized plaintext space $\frac{\mathbb{Z}[X]}{(\Phi_m(X),t(X))}$, where $\Phi_m(X)$ is the m-th cyclotomic polynomial and $t(X)$ is the plaintext modulus. We select parameters Φ_m and t such that the plaintext space becomes isomorphic to $\mathbb{F}_{p^2}^{\ell}$, where $p = 2^{64} - 2^{32} + 1$ is the Goldilocks prime and ℓ is the SIMD capacity. The field extension $\mathbb{F}_{p^2}$ is a popular choice in zkSNARK implementations because of its efficient arithmetic [17]. We notice that this extension is also especially suitable to the blind zkSNARK setting since it causes small noise growth. In GBFV, the SIMD packing capacity $\ell \in \{96, 192, 384, 768, 1536, 3072\}$ is smaller than the lattice dimension $\deg(\Phi_m)$. However, for our homomorphic circuit, this does not lower performance since we find the lowest operation count for $\ell = 384$.

Thus far, we have addressed our techniques for the homomorphic evaluation of Fractal's hIOP proof system. These techniques are covered in more detail in Section 6, where we provide an explicit construction for the Blind hIOP (BhIOP) compiled from the Fractal hIOP. Ben-Sasson et al. [12] have shown that using BCS compilation one can compile an IOP into a zkSNARK in the Random Oracle Model (ROM). However, naively applying this compilation in the blind setting would require homomorphically computing commitments and hash functions. Fortunately, we prove that it is possible to simply apply BCS compilation to BhIOPs, as previously claimed by Garg et al. and by Aranha et al. We build upon the existing theoretical framework by Holmgren [60] and Block et al. [16] to prove that any BhIOP compiled from a round-by-round knowledge sound hIOP will be round-by-round *plaintext* knowledge sound (rbr-pks), a property we define for BhIOPs. We also extend the property of *non-adaptivity*, defined by Nassar et al. [73] for IOPs, to the blind setting, and show that BCS compilation can only be applied to rbr-pks non-adaptive BhIOPs, resulting in Designated-Verifier Blind zkSNARKs (DV-BzkSNARK). To summarize, it is sufficient for the blind prover to commit to the HE ciphertext representations of the hIOP prover messages instead if the verifier is the holder of the secret key that can decrypt them.

Note that DV-BzkSNARKs are applicable to the vCOED setting. The client encrypts the inputs to some computation F using an HE scheme and sends them to the server. The server computes F homomorphically and sends the encrypted outputs and accompanying DV-BzkSNARK proof back to the client,

who can then decrypt them and verify the proof. In the zkDel setting, the client is a prover that privately outsources the proving computation by sending HE ciphertexts of their private inputs (or the entire computation trace) to the server. However, the returned proof is designated-verifier and can thus only be verified by the client prover. We solve this by extending our compiler such that the secret key holder can construct a Publicly Verifiable BzkSNARK (PV-BzkSNARK) from a DV-BzkSNARK by appending decryptions along with a zero-knowledge Proof of Decryption (PoD), as suggested in [49]. By doing so, any party can verify a normal zkSNARK proof on plaintexts and subsequently check that they correspond to the ciphertexts committed to in BCS compilation – without learning the secret key that could be used to decrypt the private prover inputs.

The viability of PV-BzkSNARKs for zkDel depends on the computation cost that is required from the client prover to generate the PoDs. Therefore, we construct a prover-efficient PoD scheme specifically for this setting. In particular, we want to prove that the inherent noise in some ciphertext (c_0, c_1) is bounded by $\|v_{inh}\|_\infty < \mathcal{B}_q$, where $v_{inh} \coloneqq c_0 + c_1 \cdot \mathtt{sk} - \lfloor (q/t) \cdot m \rceil$ for some message m, secret key $\mathtt{sk}$ and ciphertext modulus q. We build upon the lattice-based sigma-protocol LNP22 by Lyubashevsky et al. [68]; here, we discuss a few difficulties regarding compatability with our setting. Firstly, because LNP22 uses the Jonhsonn-Lindenstrauss lemma, it can only prove 2-norm bounds with some relaxation factor $\psi^{(L2)}$; because we intend to prove infinity norms, this factor is increased to $\sqrt{n}\psi^{(L2)}$. As a consequence we can only construct a valid proof that $\|v_{inh}\|_\infty < \mathcal{B}_q$ if $\|v_{inh}\|_\infty < \mathcal{B}_q/(\sqrt{n}\psi^{(L2)})$. We observe that in the context of HE ciphertexts this can be interpreted as a $\log(\sqrt{n}\psi^{(L2)})$-bit noise loss. Secondly, to accommodate our parameter selection with a non-power-of-two cyclotomic, we have to prove the above linear relation in vectorized form instead of proving it over the smaller power-of-two rings that are used in LNP22. Furthermore, LNP22 uses a modulus $q'' = 5 \bmod 8$ that is not optimal for efficient HE computations; we address this by modulus switching down to modulus q'' before committing to the ciphertexts in BCS compilation. Reducing the modulus is possible since in both the vCOED and zkDel settings, this modulus should only accommodate limited subsequent noise growth. It also has the added benefit of decreasing the proof size by approximately a factor of $\log(q/q'')$.

However, the LNP22-based PoDs described above would require a proving runtime of multiple hours because they have $\mathcal{O}(rn^2)$ proving complexity, and FRI-based BzkSNARKs for constraint systems of size 2^{20} would require proving $r \geq 2000$ decryptions. We can decrease proving time by using the following observation: the previous modulus switching operation permits switching to a smaller lattice dimension while retaining the same level of RLWE-based security.

Therefore, we provide the prover with the required key material to ringswitch [54] down to a smaller dimension n', which accelerates the client prover computation and simultaneously decreases the proof size by a factor of n/n'. Finally, using a new technique for batching HE PoDs, we can further decrease the client prover runtime from minutes to seconds. We exploit the additive homomorphism of the HE ciphertexts $[\mathtt{ct}_i]_{i\in[r]}$ to compute a random linear combination $\mathtt{ct}^* := \sum \alpha_i \mathtt{ct}_i$ that corresponds to the message $m^* := \sum \alpha_i m_i$. Using the Schwartz-Zippel lemma, we show it is sufficient to prove that $\mathtt{ct}^*$ decrypts to m^*. Thereby, we decrease the asymptotic prover runtime from $\mathcal{O}(rn^2)$ to $\mathcal{O}(n^2 + rn \log n)$ by computing more of the total proving computation as ring operations. On the other hand, since soundness relies on the correctness of the HE scheme for this linear combination, we require extra noise depth from the HE scheme. We can again increase efficiency by modulus switching to a modulus q'' in the PoD after performing these homomorphic operations, since this enables a subsequent ringswitch down to an even smaller lattice dimension n''.

3 Related Works

The two most relevant works are Garg et al. [49] and Aranha et al. [5]. In [49], the authors first establish FRI on hidden values leveraging linearly-homomorphic encapsulation (LHEncap), a concept more general than HE. They use it to construct a polynomial commitment scheme on hidden values, which is then combined with a polynomial IOP-based SNARK to enable verifiable private delegation of computation (corresponding to vCOED in our paper) and private outsourcing zkSNARKs to a single server (corresponding to zkDel in our paper). They also provide a construction for weighted threshold signatures without trusted setup.

In [5], the authors introduce a general transformation for IOPs to work over HE, denoted as HE-IOP (where the adaptation to holographic IOPs corresponds to the BhIOP in our paper), in the interest of building vCOED. They focus on making FRI over HE (HE-FRI) practical by proposing several optimizations, including a shallow fold algorithm that reduces the Fold operation depth from n to 1 at the cost of increasing complexity from $\mathcal{O}(2^n)$ to $\mathcal{O}(n \cdot 2^n)$ for input size 2^n. This shallow fold algorithm is also utilized in our instantiation of blind zkSNARKs with Fractal. Furthermore, the authors provide an open-source implementation. Based on this implementation, for a batch of 4096 polynomials of degree up to 2^{15}, the prover needs 207.8 seconds to run HE-FRI (including the Reed-Solomon code encoding) using 32 threads.

Below we present detailed comparisons with these two prior works.

Comparison with Garg et al. [49] Our work constructs vCOED and zkDel based on the observation by Garg et al. that Fractal and FRI can efficiently be computed on hidden values due to their linearity. Their work provided the first theoretical framework for vCOED and zkDel and discusses efficiency in asymptotic terms. We extend their work by providing a concrete compiler for constructing vCOED and zkDel, defining the required properties of the hIOP/HE scheme and proving that the resulting constructions achieve the completeness, soundness and zero-knowledge properties. Notably, their framework compiles Polynomial IOPs (PIOPs) to blind zkSNARKs by using polynomial commitment schemes on hidden values, which they construct using FRI on hidden values. Our compiler constructs blind zkSNARKs starting from Reed-Solomon hIOPs (RS-hIOPs) by using FRI directly, avoiding the need of a polynomial commitment scheme on hidden values.

We further extended their work by providing concrete algorithms for blind Fractal and a proof of decryption scheme. We also select specific parameters for Fractal, FRI and the HE scheme which allows us to gauge concrete efficiency and identify practical challenges. For example, as pointed out in [5], the assumption of *decryptable* LHEncap in [49] overlooked the challenge that the evaluation correctness of HE schemes needs to support the homomorphic operations in the blind proofs, posing a significant obstacle for noise management in practice. Our work is the first to demonstrate the feasibility of such assumptions in the context of Fractal, as shown in Table 1. In fact, our algorithm not only ensures the resulting noise remains below the decryption bound but also leaves a sufficient noise gap to enable circuit privacy (via noise flooding) in vCOED and accommodate the relaxation factor in the PoD of the zkDel scheme.

Comparison with Aranha et al. [5] Aranha et al. [5] demonstrated the feasibility of FRI in the blind setting (for polynomials of degree 2^{15}) through implementation. We extend their work by also demonstrating the practicality of blind FRI (for polynomials of degree 2^{20}) through microbenchmarking[2], i.e. deriving runtime estimates from operation counts. Furthermore, we also demonstrate the practicality of the Fractal zkSNARK in the blind setting, of which blind FRI is a subroutine. Fractal contains subprocedures that are more computationally intensive than FRI, such as computing Az, Bz, Cz in the R1CS constraint system and performing the lincheck protocol. Our work also provides improved algorithms for computing FRI in the blind setting. We list technical differences below.

- The security requirement of FRI necessitates the use of a large field $\mathbb{F}_{p^d}$ with a size of at least 128 bits. In [5], the arithmetic of $\mathbb{F}_{p^d}$ is emulated

[2]The GBFV scheme, which serves as the HE scheme used in our construction, has not yet been implemented for non-power-of-two cyclotomics and large plaintext spaces.

through $\mathbb{F}_p$, where d ciphertexts are needed to encrypt a single value in $\mathbb{F}_{p^d}$, resulting in memory and efficiency overhead. In contrast, we leverage the recent GBFV [50] homomorphic encryption scheme, which natively supports the arithmetic of $\mathbb{F}_{p^d}$ while offering flexibility in balancing packing capacity P and noise growth. As shown in Sections 6 and 8, we select parameter sets that offer a sufficient number of SIMD slots to achieve a practical performance as well as sufficient noise capacity for computing blind FRI after blindly computing the Fractal hIOP.

- The NTT algorithms in [5] are used for Reed-Solomon encoding in FRI, and their implementation performs the NTT for a batch of 4096 polynomials that are encoded in the 4096 SIMD slots of BGV/BFV ciphertexts. However, if one has to perform the NTT for only $k \ll 4096$ polynomials at once, their approach will only use k slots while leaving the rest unused, resulting in higher memory usage and reduced efficiency. Note that in Fractal, one will have to perform the NTT for at most $k = 4$ polynomials at once. Therefore, we take an alternative approach that allows using all slots in a ciphertext, even when processing a single polynomial ($k = 1$). We refer to it as 2D-NTT [40], which is also known as the Map-Reduce NTT in [78] and NTT with tensorial decomposition in [71]. While the 2D-NTT has been previously explored in the context of horizontal scalability for distributed zero-knowledge proof system [80] and hardware acceleration [1, 14], its compatibility with HE packing has not been studied in prior research, to the best of our knowledge.
- To verify the consistency of the messages sent by the prover, the verifier (in vCOED) needs to decrypt to obtain a subset of messages from ciphertexts. To reduce the cost of decrypting a subset, [5] employs a repacking procedure. This involves first converting slot-encoded ciphertexts into the coefficient encoding, then extracting the relevant coefficients as LWE samples, which are subsequently repacked using a key-switching algorithm [38, 39]. In our work, we use different FHE techniques to reduce the decryption cost, namely reducing the lattice dimension from ring switching [54] and reducing the ciphertext modulus from modulus switching. These techniques do not require transformation to coefficient encoding, and more importantly, their combination with the slot-wise Schwartz-Zippel lemma significantly reduces cost of PoD in the zkDel setting.

Finally, our formalization of blind proofs presents theoretical improvements over [5]. This includes highlighting the necessity of non-adaptivity in BCS compilation (although to the best of our knowledge, all known hIOPs possess this property) and deriving a tighter RBR soundness error bound, where we conclusively prove that the correctness error of the HE scheme does not help dishonest provers.

Related works regarding proof of decryption. Verifiable decryption was first introduced in [26] together with verifiable encryption, and lattice-based constructions have been proposed for the BGV homomorphic encryption scheme [24] in [4, 28, 56, 61, 66] and for the Kyber key encapsulation scheme [20] in [68, 69].

The work [66] addresses the special case where the plaintext modulus is 2, and the works [4, 28, 56, 61] discuss *distributed* verifiable decryptions, where the secret key is shared among multiple parties. While [28] also employs LNP22, their construction (as well as the construction in [61]) requires committing to both the NTT and the coefficient representations of the secret key (an element in the ring $\mathcal{R}_{m,q}$ where m is a power-of-two) and uses the NTT representation to facilitate polynomial multiplications in this ring. In contrast, we commit only to the coefficient representation and perform polynomial multiplication via matrix-vector multiplication using the rotation matrix of $\mathcal{R}_{m,q}$. This not only reduces the commitment size but also removes the restriction of m being a power-of-two cyclotomic order, making it applicable to our parameters in Section 8, where $m = 7 \cdot 3 \cdot 2^{11}$.

Other methods for vCOED vFHE [6, 47, 64, 79] is an alternative approach to achieve vCOED by applying proof systems to HE schemes. While vFHE ensures security against key-recovery attacks, the state-of-the-art vFHE constructions can only verifiably outsource computations involving hundreds of constraints in a runtime similar to our vCOED approach with 2^{20} constraints; thus, our work achieves an improvement of four orders of magnitude compared to vFHE-based approaches. The effect of key-recovery attacks can be mitigated by avoiding reusing the same HE secret key. As such, at most one bit of information about the plaintext/secret key is leaked (only when the verifier signals to the prover whether verification passes), as carefully analyzed in [5].

4 Preliminaries

4.1 Commitment scheme

We define a commitment scheme following [4, 45].

Definition 4.1 (Commitment scheme)**.** A commitment scheme $\mathcal{CT} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Open})$ includes the following probabilistic polynomial-time algorithms:

- $\mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$: for a given security parameter λ, it returns public parameters pp, which define a message space $\mathcal{S}_M$, a randomness space $\mathcal{S}_R$ and a commitment space $\mathcal{S}_C$.

- $\mathcal{CT}.\mathsf{Com}_{\mathsf{pp}}(m, r)$: for a given message $m \in \mathcal{S}_M$ and some randomness $r \leftarrow \mathcal{S}_R$, it returns a commitment $\mathtt{C}_m$.
- $\mathcal{CT}.\mathsf{Open}_{\mathsf{pp}}(m, r, \mathtt{C})$: for a given tuple $(m, r, \mathtt{C}) \in \mathcal{S}_M \times \mathcal{S}_R \times \mathcal{S}_C$, it returns either acc or rej.

Binding. A commitment scheme $\mathcal{CT}$ is computationally binding if for any $\mathsf{pp} \leftarrow \mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$ and probabilistic polynomial-time adversary $\mathcal{A}$, it holds that

$$\Pr\left[\begin{array}{c|c} m_0 \neq m_1 & \begin{array}{c}(\mathtt{C}, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathcal{CT}.\mathsf{Open}_{\mathsf{pp}}(m_0, r_0, \mathtt{C}) = \mathsf{acc} \\ \mathcal{CT}.\mathsf{Open}_{\mathsf{pp}}(m_1, r_1, \mathtt{C}) = \mathsf{acc}\end{array}\end{array}\right] = \mathsf{negl}(\lambda).$$

Hiding. A commitment scheme $\mathcal{CT}$ is computationally hiding if for any $\mathsf{pp} \leftarrow \mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$ and probabilistic polynomial-time adversary $\mathcal{A}$ it holds that

$$\left|\Pr\left[\ \mathcal{A}(\mathtt{C}) = 1 \ \middle|\ \mathtt{C} \leftarrow \mathcal{CT}.\mathsf{Com}_{\mathsf{pp}}(m_0)\ \right] - \Pr\left[\ \mathcal{A}(\mathtt{C}) = 1 \ \middle|\ \mathtt{C} \leftarrow \mathcal{CT}.\mathsf{Com}_{\mathsf{pp}}(m_1)\ \right]\right| \leq \mathsf{negl}(\lambda).$$

We also define a verification oracle $\mathcal{O}^{\mathcal{CT}}(\mathtt{C}_m, m)$ which returns acc when there exists an $r \in \mathcal{S}_R$ such that $\mathcal{CT}.\mathsf{Open}(m, r, \mathtt{C}_m) = \mathsf{acc}$ and rej otherwise.

4.2 Homomorphic Encryption (HE)

We define a secret-key HE scheme with plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$ following [25, 32, 52, 57].

Definition 4.2 (Homomorphic Encryption)**.** An HE scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ includes the following polynomial-time algorithms:

- $\mathcal{E}.\mathsf{KeyGen}(1^\lambda)$: given the security parameter λ, it returns a secret key $\mathtt{sk}$ and a public evaluation key $\mathtt{evk}$.
- $\mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}(\{m_i\}_{i\in[r]})$: given the secret key $\mathtt{sk}$ and plaintexts $\{m_i\}_{i\in[r]} \in \mathcal{P}^r$, it returns ciphertexts $\{\mathtt{ct}_i\}_{i\in[r]} \in \mathcal{C}^r$, which can also be denoted as $\{\mathtt{ct}[m_i]\}_{i\in[r]}$.
- $\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\{\mathtt{ct}_i\}_{i\in[r]})$: given the secret key $\mathtt{sk}$ and ciphertexts $\{\mathtt{ct}_i\}_{i\in[r]} \in \mathcal{C}^r$, it returns plaintexts $\{m_i\}_{i\in[r]} \in \mathcal{P}^r$.
- $\mathcal{E}.\mathsf{Eval}_{\mathtt{evk}}(f, \{\mathtt{ct}_i\}_{i\in[\ell]})$: given the public evaluation key $\mathtt{evk}$, a function $f : \mathcal{P}^\ell \to \mathcal{P}^r$ and a set of ciphertexts $\{\mathtt{ct}_i\}_{i\in[\ell]} \in \mathcal{C}^\ell$, it returns ciphertexts $\{\mathtt{ct}'_i\}_{i\in[r]} \in \mathcal{C}^r$.

CPA security. An HE scheme $\mathcal{E}$ is IND-CPA secure if for any $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, two plaintexts $m_0, m_1 \in \mathcal{P}$ and probabilistic polynomial-time adversary $\mathcal{A}$, it holds that

$$\Big|\Pr\left[\ \mathcal{A}(\mathtt{evk}, \mathtt{ct}) = 1 \ \middle|\ \mathtt{ct} \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}(m_1)\ \right] - \Pr\left[\ \mathcal{A}(\mathtt{evk}, \mathtt{ct}) = 1 \ \middle|\ \mathtt{ct} \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}(m_0)\ \right]\Big| \leq \mathsf{negl}(\lambda).$$

Correctness. An HE scheme $\mathcal{E}$ is correct for functions in $\mathcal{F}$ if for any $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, function $f \in \mathcal{F}$, plaintexts $\{m_i\}_{i\in[\ell]}$ and their encryptions $\{\mathtt{ct}_i\} \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}(\{m_i\})$, the relation

$$\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathcal{E}.\,\mathsf{Eval}_{\mathtt{evk}}(f, \mathtt{ct}_1, \ldots, \mathtt{ct}_\ell)\right) = f(m_1, \ldots, m_\ell),$$

holds with probability no lower than $1 - \mathsf{negl}(\lambda)$.

Circuit Privacy. An HE scheme $\mathcal{E}$ satisfies computational circuit privacy for functions in $\mathcal{F}$ if there exists a probabilistic polynomial-time simulator Sim such that for any security parameter λ, HE keys $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, set of plaintexts $\{m_i\}$, function $f \in \mathcal{F}$ and probabilistic polynomial-time distinguisher D, it holds that

$$\Big|\Pr\left[\ \mathsf{D}(\mathtt{ct}, \mathtt{sk}) = 1 \ \middle|\ \mathtt{ct} \leftarrow \mathsf{Sim}(1^\lambda, \mathtt{sk}, f(\{m_i\}))\ \right] - \Pr\left[\ \mathsf{D}(\mathtt{ct}, \mathtt{sk}) = 1 \ \middle|\ \begin{matrix}\{\mathtt{ct}_i\} \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}(\{m_i\}) \\ \mathtt{ct} \leftarrow \mathcal{E}.\,\mathsf{Eval}_{\mathtt{evk}}(f, \{\mathtt{ct}_i\})\end{matrix}\ \right]\Big| \leq \mathsf{negl}(\lambda).$$

This property can be achieved using a technique called noise-flooding [18,52,63].

Remark 4.1. *For use in Subsection 5.2, we define $\mathcal{E}.\mathsf{len}$ to be the minimal length in bits of a decryptable ciphertext. In the context of Subsection 5.3, the ciphertext should additionally still allow for an efficient proof of decryption.*

4.3 Relations and languages

A relation $\mathbf{R}$ on some plaintext space $\mathcal{P}$ is a subset of $(\mathbb{x}, \mathbb{w}) \in \mathcal{P}^* \times \mathcal{P}^*$. For some relation $\mathbf{R}$ we define a language $\mathcal{L}_{\mathbf{R}} = \{\mathbb{x} \mid \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathbf{R}\}$. Similarly we define an indexed relation that is a subset of $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{P}^* \times \mathcal{P}^* \times \mathcal{P}^*$ which in turn defines a relation

$$\mathbf{R}_{\mathbb{i}} = \big\{(\mathbb{x};\ \mathbb{w}) : (\mathbb{i};\ \mathbb{x};\ \mathbb{w}) \in \mathbf{R}\big\} = \big\{(x;\ w) : Az \circ Bz = Cz \text{ for } z = (x, w)\big\},$$

and the second equality shows an example where the index $\mathbb{i}$ consists of a Rank-1 Constraint Satisfiability (R1CS) circuit defined by the matrices A, B and C.

4.4 Zero-knowledge Succinct Non-interactive ARgument of Knowledge (zkSNARK)

We will define pre-processing zkSNARKs in the Random Oracle Model (ROM) following Chiesa et al. [37]. Let us denote with $\mathcal{U}(\lambda)$ the uniform distribution over all functions $\mathcal{P}^* \to \{0,1\}^\lambda$. A function $\rho \leftarrow \mathcal{U}(\lambda)$ is referred to as a Random Oracle (RO). We denote an algorithm A having oracle access to some object x as $A^{[\![x]\!]}$.

Definition 4.3 (preprocessing zkSNARK in the ROM)**.** A preprocessing zkSNARK $\Pi = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ is a non-interactive proof system for some indexed relation $\mathbf{R}$ that includes the following polynomial-time algorithm:

- $\Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathtt{i})$: for a given index $\mathtt{i}$, using access to the RO ρ, it returns the index keys $(\mathtt{ipk}, \mathtt{ivk})$.

and the following probabilistic polynomial-time algorithms:

- $\Pi.\mathsf{P}^{[\![\rho]\!]}(\mathtt{ipk}, \mathtt{x}, \mathtt{w})$: for a given index prover key $\mathtt{ipk}$, statement $\mathtt{x}$ and witness $\mathtt{w}$, using access to the RO ρ, it returns a proof π.
- $\Pi.\mathsf{V}^{[\![\rho]\!]}(\mathtt{ivk}, \mathtt{x}, \pi)$: for a given index verifier key $\mathtt{ivk}$, statement $\mathtt{x}$ and proof π, using oracle access to the RO ρ, it returns either acc or rej.

A zkSNARK should satisfy the following properties.

Completeness. For any $(\mathtt{i}, \mathtt{x}, \mathtt{w}) \in \mathbf{R}$ and $\rho \leftarrow \mathcal{U}(\lambda)$ it holds that

$$\Pr\left[\ \Pi.\mathsf{V}^{[\![\rho]\!]}(\mathtt{ivk}, \mathtt{x}, \pi) \neq \mathsf{acc} \ \middle|\ \pi \leftarrow \Pi.\mathsf{P}^{[\![\rho]\!]}(\mathtt{ipk}, \mathtt{x}, \mathtt{w})\ \right] \leq \delta$$

where δ is the completeness error and $(\mathtt{ipk}, \mathtt{ivk}) = \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathtt{i})$

Zero-knowledge. For any $(\mathtt{i}, \mathtt{x}, \mathtt{w}) \in \mathbf{R}$ and $\rho \leftarrow \mathcal{U}(\lambda)$, if there exists a probabilistic polynomial-time simulator Sim such that for any unbounded distinguisher D it holds that

$$\Big|\Pr\left[\ \mathsf{D}^{[\![\rho[\mu]]\!]}(\pi) = 1 \ \middle|\ (\mu, \pi) \leftarrow \mathsf{Sim}^{[\![\rho]\!]}(\mathtt{i}, \mathtt{x})\ \right] - \Pr\left[\ \mathsf{D}^{[\![\rho[\mu]]\!]}(\pi) = 1 \ \middle|\ \pi \leftarrow \Pi.\mathsf{P}^{[\![\rho]\!]}(\mathtt{ipk}, \mathtt{x}, \mathtt{w})\ \right]\Big| \leq z$$

where $(\mathtt{ipk}, \mathtt{ivk}) = \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathtt{i})$ and $\rho[\mu]$ equals $\mu(x)$ if μ is defined on x and otherwise equals $\rho(x)$, then Π has z-statistical zero-knowledge. If D is probabilistic polynomial-time then Π has z-computational zero-knowledge.

Soundness. For any index $\mathtt{i}$, statement $\mathbb{x} \notin \mathcal{L}_{\mathbf{R}_\mathtt{i}}$, RO $\rho \leftarrow \mathcal{U}(\lambda)$, index keys $(\mathtt{ipk}, \mathtt{ivk}) = \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathtt{i})$ and prover P^* it holds that

$$\Pr\left[\ \Pi.\mathsf{V}^{[\![\rho]\!]}(\mathtt{ivk}, \mathbb{x}, \pi) = \mathsf{acc} \ \middle|\ \pi \leftarrow \mathsf{P}^{*[\![\rho]\!]}\ \right] \leq \varepsilon$$

where ε is the soundness error.

Knowledge soundness. For any index $\mathtt{i}$, statement $\mathbb{x}$, index keys $(\mathtt{ipk}, \mathtt{ivk}) \leftarrow \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathtt{i})$ and prover P^* there exists a polynomial-time extractor Ext such that

$$\Pr\left[\ (\mathbb{x}, \mathbb{w}) \in \mathbf{R}_\mathtt{i} \ \middle|\ \mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(\mathtt{i}, \mathbb{x})\ \right]$$
$$\geq \Pr\left[\ \Pi.\mathsf{V}^{[\![\rho]\!]}(\mathtt{ivk}, \mathbb{x}, \pi) = \mathsf{acc} \ \middle|\ \pi \leftarrow \mathsf{P}^{*[\![\rho]\!]}\ \right] - \varepsilon_\mathsf{k}$$

where ε_k is the knowledge error and $\mathsf{Ext}^{\mathsf{P}^*}$ may interact with P^* by rewinding it in a black-box manner.

Remark 4.2. *Some authors [15, 37] define stronger adaptive versions of these properties. For example in knowledge soundness they have the prover P^* choose the index $\mathtt{i}$ and statement $\mathbb{x}$. Although it is possible to define all these and the following properties adaptively, for ease of notation, we will refrain.*

Remark 4.3. *Note that this definition of zkSNARKs has no restriction of proof length or verifier cost and is therefore not necessarily succinct. However, instead of using a different name such as Non-interactive Random Oracle Proof (NIROP) [12], we follow Chiesa et al. [36] and use the popularized term zkSNARK.*

4.5 Interactive Oracle Proofs (IOPs)

First introduced by Ben-Sasson et al. [12], an Interactive Oracle Proof (IOP) is a form of interactive proof where the prover sends $\mu + 1$ messages in the form of oracles $[\![m_i]\!]$ to proof strings $m_i \in \mathcal{P}^*$ and the verifier responds with some challenges $c_i \in \mathcal{C}h_i \subseteq \mathcal{P}^*$. They can be seen as a μ-round generalization of Probabilistically Checkable Proofs (PCPs). For $i \in [\mu]$, we define the concatenation $m_1 \| c_1 \| \ldots \| m_i \| c_i$ as an i-round partial transcript and $m_1 \| c_1 \| \ldots \| m_\mu \| c_\mu \| m_{\mu+1}$ as a full transcript. An holographic IOP is an extension where an encoding of some index $\mathtt{i}$ is generated in a preprocessing step for oracle access to the verifier [37].

Definition 4.4 (holographic IOP (hIOP))**.** An hIOP $\Pi = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ is a μ-round interactive proof system for some indexed relation $\mathbf{R}$ that includes the following polynomial-time algorithm:

- $\Pi.\mathsf{Ind}(\mathtt{i})$: for a given index $\mathtt{i}$, it returns the encoding of the index $\mathsf{e}[\mathtt{i}]$.

and the following probabilistic polynomial-time algorithms:

- $\Pi.\mathsf{P}(\mathsf{e}[\mathtt{i}], \mathbb{x}, \mathbb{w})$: for a given index encoding $\mathsf{e}[\mathtt{i}]$, statement $\mathbb{x}$ and witness $\mathbb{w}$ for the relation $\mathbf{R}_{\mathtt{i}}$, it returns a round message

$$m_i \leftarrow \Pi.\mathsf{P}_i(\mathsf{e}[\mathtt{i}], \mathbb{x}, \mathbb{w}, \mathtt{tr})$$

  in round $i \in [\mu+1]$ where $\mathtt{tr}$ is the current $(i-1)$-round partial transcript.
- $\Pi.\mathsf{V}^{[\![\mathsf{e}[\mathtt{i}]]\!],[\![\mathtt{tr}]\!]}(\mathbb{x})$: for a given statement $\mathbb{x}$, using oracle access to the encoded index $\mathsf{e}[\mathtt{i}]$ and the current partial transcript $\mathtt{tr} = m_1 \| \ldots \| c_{i-1} \| m_i$ to obtain queries $\{\mathsf{e}[\mathtt{i}]_i\}, \{\mathtt{tr}_i\}$, it returns a round challenge

$$c_i \leftarrow \Pi.\mathsf{V}_i(\{\mathsf{e}[\mathtt{i}]_i\}, \mathbb{x}, \{\mathtt{tr}_i\})$$

  when $i \in [\mu]$ and returns either acc or rej when $i = \mu + 1$.

We also define a function qr that maps a query index to its corresponding message index. The hIOP is public-coin if all messages sent by the verifier are random elements of some subset of the plaintext space independent of the current partial transcript. Without loss of generality, we can assume that a public-coin verifier performs all queries after receiving the final prover's message. An hIOP should satisfy the following properties.

Completeness. For any $(\mathtt{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$ it holds that

$$\Pr\left[\Pi.\left\langle \mathsf{P}(\mathsf{e}[\mathtt{i}], \mathbb{x}, \mathbb{w}), \mathsf{V}^{[\![\mathsf{e}[\mathtt{i}]]\!]}(\mathbb{x})\right\rangle \neq \mathsf{acc}\right] \leq \delta$$

where δ is the completeness error, $\mathsf{e}[\mathtt{i}] = \Pi.\mathsf{Ind}(\mathtt{i})$ and the bracket notation $\langle A, B\rangle$ represents the output of $B^{[\![\mathtt{tr}]\!]}$ where $\mathtt{tr}$ is a full transcript resulting from interaction with A.

Honest-verifier zero-knowledge. For any $(\mathtt{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$, if there exists a probabilistic polynomial-time simulator Sim such that for any unbounded distinguisher D it holds that

$$\begin{aligned}\Big|\Pr\left[\ \mathsf{D}(\mathtt{i}, \pi) = 1 \ \middle|\ \pi \leftarrow \mathsf{Sim}(\mathtt{i}, \mathbb{x})\ \right] \\ - \Pr\left[\ \mathsf{D}(\mathtt{i}, \pi) = 1 \ \middle|\ \pi \leftarrow \mathsf{View}\left\langle \Pi.\mathsf{P}(\mathsf{e}[\mathtt{i}], \mathbb{x}, \mathbb{w}), \Pi.\mathsf{V}^{[\![\mathsf{e}[\mathtt{i}]]\!]}(\mathbb{x})\right\rangle\ \right]\Big| \leq z\end{aligned}$$

where $\mathsf{e}[\mathtt{i}] = \Pi.\mathsf{Ind}(\mathtt{i})$ and $\mathsf{View}\langle\cdot\rangle$ is a random variable that contains all the query responses the verifier receives during the protocol along with the verifier's randomness, then Π has z-statistical zero-knowledge. If D is probabilistic polynomial-time then Π has z-computational zero-knowledge.

Soundness. For any index $\mathbb{i}$, statement $\mathbb{x} \notin \mathcal{L}_{\mathbf{R}_\mathbb{i}}$ and unbounded prover P^* it holds that

$$\Pr\left[\left\langle \mathsf{P}^*(\mathsf{e}[\mathbb{i}], \mathbb{x}), \Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]}(\mathbb{x})\right\rangle = \mathsf{acc}\right] \leq \varepsilon$$

where ε is the soundness error and $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$.

Knowledge soundness. For any index $\mathbb{i}$, statement $\mathbb{x}$ and unbounded prover P^* there exists a polynomial-time extractor Ext such that

$$\Pr\left[(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_\mathbb{i} \mid \mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(\mathbb{i}, \mathbb{x})\right] \geq \Pr\left[\left\langle \mathsf{P}^*, \Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]}(\mathbb{x})\right\rangle = \mathsf{acc}\right] - \varepsilon_{\mathsf{k}}$$

where ε_{k} is the knowledge error and $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$. Note that $\mathsf{Ext}^{\mathsf{P}^*}$ may interact with P^* by rewinding it in a black-box manner.

The soundness and knowledge soundness properties ensure the security of the hIOP scheme. Respectively, they guarantee (except with some small error) that a prover interacting with a verifier cannot result in acc for a statement that has no valid witness or for which the valid witness is not known. Note that knowledge soundness thus implies normal soundness. However, since hIOPs are compiled into non-interactive proofs [12], their security is best described round-by-round [27, 36]. Following Holmgren [60] and Block et al. [16], we will define round-by-round soundness using a doomed set and round-by-round knowledge soundness using a knowledge doomed set.

Definition 4.5 (Doomed set)**.** Given a public-coin holographic hIOP Π that proves an indexed relation $\mathbf{R}$, a doomed set $\mathcal{D}^\Pi$ for index $\mathbb{i}$ and error ε is a set that satisfies the following properties:

1. For any statement $\mathbb{x}$, if $\mathbb{x} \notin \mathcal{L}_{\mathbf{R}_\mathbb{i}}$ then $(\mathbb{x}, \emptyset) \in \mathcal{D}^\Pi$.
2. For any $(\mathbb{x}, \mathtt{tr}) \in \mathcal{D}^\Pi$ where $\mathtt{tr}$ is a $(i-1)$-round partial transcript for $i \in [\mu + 1]$ and any next prover message m_i, it holds that

$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}, \mathtt{tr} \| m_i \| c_i) \notin \mathcal{D}^\Pi] \leq \varepsilon.$$

3. For any full transcript $\mathtt{tr}$, if $(\mathbb{x}, \mathtt{tr}) \in \mathcal{D}^\Pi$ then $\Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!], [\![\mathtt{tr}]\!]}(\mathbb{x}) = \mathsf{rej}$.

Definition 4.6 (Knowledge doomed set)**.** Given a public-coin holographic hIOP Π that proves an indexed relation $\mathbf{R}$, a knowledge doomed set $\mathcal{D}^\Pi_{\mathsf{k}}$ for index $\mathbb{i}$ and error ε_{k} is a set for which there exists a polynomial-time extractor Ext such that the following properties are satisfied.

1. For any statement $\mathbb{x}$, it holds that $(\mathbb{x}, \emptyset) \in \mathcal{D}^\Pi_{\mathsf{k}}$.
2. For any $(\mathbb{x}, \mathtt{tr}) \in \mathcal{D}^\Pi_{\mathsf{k}}$ where $\mathtt{tr}$ is a $(i-1)$-round (partial) transcript $\mathtt{tr}$ for $i \in [\mu + 1]$ and next prover message m_i, it holds that if

$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}, \mathtt{tr} \| m_i \| c_i) \notin \mathcal{D}^\Pi_{\mathsf{k}}] > \varepsilon_{\mathsf{k}}.$$

then $\mathbb{w} \leftarrow \mathsf{Ext}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathtt{tr} \| m_i)$ such that $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_{\mathbb{i}}$.
3. For any full transcript $\mathtt{tr}$, if $(\mathbb{x}, \mathtt{tr}) \in \mathcal{D}_{\mathsf{k}}^{\Pi}$ then $\Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!],[\![\mathtt{tr}]\!]}(\mathbb{x}) = \mathsf{rej}$.

Remark 4.4. *In property 2 in Definition 4.5 and 4.6 we slightly abuse notation by having $c_{\mu+1} \leftarrow \mathcal{C}h_{\mu+1}$ denote the public-coin verifier's additional randomness used in the final verification check.*

Definition 4.7 (Round-by-round (knowledge) soundness)**.** An hIOP Π for the indexed relation $\mathbf{R}$ is round-by-round sound with error $\boldsymbol{\varepsilon}$ or round-by-round knowledge sound with error $\boldsymbol{\varepsilon}_{\mathsf{k}}$ if for every index $\mathbb{i}$ there exists a doomed set $\mathcal{D}^{\Pi}$ for error $\boldsymbol{\varepsilon}$ or knowledge doomed set $\mathcal{D}_{\mathsf{k}}^{\Pi}$ for error $\boldsymbol{\varepsilon}_{\mathsf{k}}$ respectively.

Along with the definition of IOPs, Ben-Sasson et al. [12] additionally introduced BCS compilation which compiles an IOP into a zkSNARK in the ROM. Later it was extended to hIOPs [37], round-by-round soundness notions [27], the quantum ROM [36] and recently proven unconditionally UC-secure in the ROM [34]. Many of the zkSNARKs that are deployed in practice are constructed using this compilation. We defer a high-level description of this compilaiton to Section 5.2 and describe its properties in Theorem 4.1. We define two complexity measures for hIOPs. The proof length $p = \sum_{i=1}^{\mu+1} |m_i|$ is the sum of the lengths of all prover messages and the query complexity q is the number of queries performed by the verifier.

Theorem 4.1 (BCS compiler [12,36])**.** *Any hIOP Π for indexed relation $\mathbf{R}$ with completeness error δ, proof length p, query complexity q, round-by-round soundness error $\boldsymbol{\varepsilon}$, round-by-round knowledge soundness error $\boldsymbol{\varepsilon}_{\mathsf{k}}$ and z-statistical honest-verifier zero-knowledge can be compiled into a zkSNARK $\mathbb{\Pi}$ in the ROM with RO query bound Q, security parameter λ and:*

- *Completeness error δ,*
- *Proof length p' upper bound by $\lambda(\mu + 1 + \sum_{j=1}^{q}(3 + \lceil \log_2 |m_{\mathsf{qr}(j)}| \rceil))$,*
- *Soundness error $\boldsymbol{\varepsilon}'$ where $\boldsymbol{\varepsilon}' = Q\boldsymbol{\varepsilon} + 3(Q^2+1)2^{-\lambda}$,*
- *Knowledge soundness error $\boldsymbol{\varepsilon}'_{\mathsf{k}}$ where $\boldsymbol{\varepsilon}'_{\mathsf{k}} = Q\boldsymbol{\varepsilon}_{\mathsf{k}} + 3(Q^2+1)2^{-\lambda}$,*
- *z'-Statistical honest-verifier zero-knowledge where $z' = z + p2^{-\lambda/4+2}$,*

where m_i is $\Pi.\mathsf{P}$'s ith message and $|\cdot|$ denotes length in λ bits rounded up. Both soundness and knowledge soundness error are $\Theta(Q\boldsymbol{\varepsilon})$ and $\Theta(Q\boldsymbol{\varepsilon}_{\mathsf{k}})$ respectively when considering quantum adversaries that perform no more than $Q - \mathcal{O}(q \log p)$ RO queries.

Remark 4.5. *Technically, all these error values, proof lengths, etc. can be functions of both the statement and the index but let us disregard that here since it has no influence on what follows.*

5 Blind Proofs

In this section, we introduce a new type of proof system called blind proofs, where one proves that some encrypted statement is in the language of a blind relation $\mathcal{E}[\mathbf{R}]$ with respect to some commitment $\mathtt{C}_{\mathtt{sk}}$ to a secret key $\mathtt{sk}$. This blind relation represents ciphertexts of statement-witness pairs such that the underlying plaintexts are in the relation $\mathbf{R}$. In other words, blind proofs allow the prover to generate a proof using $(\mathtt{ct}[x], \mathtt{ct}[w])$ – without knowledge of the plaintext (x, w) – that proves plaintext knowledge of (x, w) such that $x \in \mathcal{L}_{\mathbf{R}}$ for the holder of the secret key $\mathtt{sk}$ committed to in $\mathtt{C}_{\mathtt{sk}}$. We start by defining a blind relation.

Definition 5.1 (Blind relation)**.** For a given HE scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ and commitment scheme $\mathcal{CT}$ with security parameter λ, and indexed relation $\mathbf{R}$ we define the indexed blind relation

$$\mathcal{E}[\mathbf{R}] = \left\{ \begin{array}{c} (\mathbb{i}; \mathbb{x}; \mathbb{w}) = (\mathbb{i};\, \mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x];\, \mathtt{ct}[w]) : \\ \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{ct}[w])) = (x, w) \in \mathbf{R}_{\mathbb{i}} \wedge \\ \mathtt{sk} \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^{\lambda}) \ \wedge \ \mathcal{O}^{\mathcal{CT}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{sk}) = \mathsf{acc} \end{array} \right\}$$

which defines the blind relation $\mathcal{E}[\mathbf{R}_{\mathbb{i}}] = \{(\mathbb{x};\, \mathbb{w}) : (\mathbb{i};\, \mathbb{x};\, \mathbb{w}) \in \mathcal{E}[\mathbf{R}]\}$.

Theorem 5.1. *Any probabilistic polynomial-time adversary has only negligible advantage in distinguishing the underlying $(x, w) \in \mathbf{R}_{\mathbb{i}}$ given the corresponding $(\mathbb{x}, \mathbb{w}) \in \mathcal{E}[\mathbf{R}_{\mathbb{i}}]$ if $\mathcal{E}$ is an IND-CPA secure HE scheme and* Com *is computationally hiding.*

Proof. This follows from a standard hybrid argument. Let us define an adversary $\mathcal{A}$ in the game of distinguishing elements of a blind relation, which we denote as game G_0. Now let us define game G_1 where the LR oracle responds with a randomly sampled $\mathtt{C}_{\mathtt{sk}}$ instead of a commitment to the used secret key. The advantage of $\mathcal{A}$ in G_1 should be negligible because $\mathcal{E}$ is IND-CPA secure and the difference between G_0 and G_1 should be negligible because Com is hiding. Therefore we can conclude that $\mathcal{A}$ has negligible advantage in game G_0. □

5.1 Blind hIOP (BhIOP)

We define a blind version of the hIOP proof system introduced in Section 4.5.

Definition 5.2 (Blind hIOP (BhIOP))**.** For a given HE scheme $\mathcal{E}$, commitment scheme $\mathcal{CT}$ and hIOP Π for indexed relation $\mathbf{R}$, a blind hIOP $\mathcal{E}[\Pi] = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ includes the following probabilistic polynomial-time algorithms:

- $\mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathtt{i})$: for a given index $\mathtt{i}$ and security parameter λ, it returns the encoding $\mathsf{e}[\mathtt{i}] = \Pi.\mathsf{Ind}(\mathtt{i})$, the keys $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and the commitment $\mathsf{C}_{\mathtt{sk}} \leftarrow \mathcal{CT}.\mathsf{Com}(\mathtt{sk})$.
- $\mathcal{E}[\Pi].\mathsf{P}_{\mathtt{evk}}(\mathsf{e}[\mathtt{i}], \mathbb{x}, \mathbb{w})$: for a given statement $\mathbb{x} = (\mathsf{C}_{\mathtt{sk}}, \mathtt{ct}[x])$ and witness $\mathbb{w} = \mathtt{ct}[w]$ of the blind relation $\mathcal{E}[\mathbf{R}_{\mathtt{i}}]$, and evaluation key $\mathtt{evk}$, it returns a round message

$$\mathcal{E}.\mathsf{Eval}_{\mathtt{evk}}(\Pi.\mathsf{P}_i, (\mathsf{e}[\mathtt{i}], \mathtt{ct}[x], \mathtt{ct}[w], \mathtt{tr}'))$$

  in round $i \in [\mu+1]$ where $\mathtt{tr}'$ is the current $(i-1)$-round partial transcript.
- $\mathcal{E}[\Pi].\mathsf{V}_{\mathtt{sk}}^{[\![\mathsf{e}[\mathtt{i}]]\!][\![\mathtt{tr}']\!]\mathcal{O}^{\mathcal{CT}}}(\mathbb{x})$: for a given statement $\mathbb{x} = (\mathsf{C}_{\mathtt{sk}}, \mathtt{ct}[x])$ and secret key $\mathtt{sk}$, using oracle access to $\mathsf{e}[\mathtt{i}]$ and the current (partial) transcript $\mathtt{tr}'$ to obtain queries $\{\mathsf{e}[\mathtt{i}]_i\}, \{\mathtt{tr}'_i\}$, it returns

$$\Pi.\mathsf{V}_i(\{\mathsf{e}[\mathtt{i}]_i\}, \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}[x], \{\mathtt{tr}'_i\}))$$

  in round $i \in [\mu+1]$ if $\mathcal{O}^{\mathcal{CT}}(\mathsf{C}_{\mathtt{sk}}, \mathtt{sk})$ returns acc, otherwise it returns rej.

  Remark 5.1. *The $\mathcal{O}^{\mathcal{CT}}$ oracle of a blind hIOP verifier is instantiated in Theorem 5.8, similar to how the oracles of an hIOP verifier are instantiated in the compilation of Theorem 4.1.*

A blind hIOP can be public-coin similar to an hIOP. It should satisfy the completeness and soundness properties as defined in Definition 4.4. Additionally, it should satisfy the following properties.

Plaintext knowledge soundness. For any index $\mathtt{i}$, statement $\mathbb{x}$, setup $(\mathsf{e}[\mathtt{i}], \mathtt{sk}, \mathtt{evk}, \mathsf{C}_{\mathtt{sk}}) \leftarrow \mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathtt{i})$ and unbounded prover $\mathsf{P}^*_{\mathtt{evk}}$ there exists a polynomial-time extractor $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ with access to a decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ such that

$$\Pr\left[\ (x,w) \in \mathbf{R}_{\mathtt{i}} \ \middle|\ (x,w) \leftarrow \mathsf{Ext}^{\mathsf{P}^*, \mathcal{O}_{\mathsf{Dec}}}(\mathtt{i}, \mathbb{x})\ \right] \geq \Pr\left[\langle \mathsf{P}^*_{\mathtt{evk}}, \mathcal{E}[\Pi].\mathsf{V}_{\mathtt{sk}}^{[\![\mathsf{e}[\mathtt{i}]]\!]\mathcal{O}^{\mathcal{CT}}}(\mathbb{x})\rangle = \mathsf{acc}\right] - \varepsilon_{\mathsf{k}}$$

where ε_{k} is the knowledge error. Note that $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ may interact with P^* by rewinding it in a black-box manner. Similar to the plaintext scenario it is possible to define a round-by-round variant (see Definition 5.3).

Honest-verifier zero-knowledge. For any security parameter λ, $(\mathtt{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{E}[\mathbf{R}]$ and $(\mathsf{e}[\mathtt{i}], \mathtt{sk}, \mathtt{evk}, \mathsf{C}_{\mathtt{sk}}) \leftarrow \mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathtt{i})$ such that $\mathbb{x} = (\mathsf{C}_{\mathtt{sk}}, \mathtt{ct}[x])$, if there exists a probabilistic polynomial-time simulator Sim such that for any

unbounded distinguisher D it holds that

$$\left|\Pr\left[\ \mathsf{D}(\mathtt{i},\pi,\mathtt{sk})=1\ \middle|\ \pi\leftarrow\mathsf{Sim}(1^\lambda,\mathtt{i},\mathbb{x},\mathtt{sk})\ \right]-\right.$$
$$\left.\Pr\left[\ \mathsf{D}(\mathtt{i},\pi,\mathtt{sk})=1\ \middle|\ \pi\leftarrow\mathsf{View}\left\langle\mathcal{E}[\Pi].\mathsf{P}_{\mathtt{evk}}(\mathsf{e}[\mathtt{i}],\mathbb{x},\mathbb{w}),\mathcal{E}[\Pi].\mathsf{V}_{\mathtt{sk}}^{[\![\mathsf{e}[\mathtt{i}]]\!]^{\mathcal{O}^{\mathcal{CT}}}}(\mathbb{x})\right\rangle\ \right]\right|\leq z$$

then $\mathcal{E}[\Pi]$ has z-statistical zero-knowledge. If D is probabilistic polynomial-time then $\mathcal{E}[\Pi]$ has z-computational zero-knowledge.

Definition 5.2 describes how blind hIOPs can be constructed from hIOP and HE schemes. From this contruction, one can show that the resulting blind hIOP satisfies the necessary properties.

Theorem 5.2. *For security parameter λ, an HE scheme $\mathcal{E}$ and a δ-complete hIOP scheme Π for indexed relation $\mathbf{R}$, the blind hIOP $\mathcal{E}[\Pi]$ is complete with completeness error $\delta+\mathsf{negl}(\lambda)$ for indexed blind relation $\mathcal{E}[\mathbf{R}]$ if $\mathcal{E}$ is correct for the homomorphic circuit $\mathcal{E}[\Pi].\mathsf{P}$.*

Proof. For any $(\mathtt{i},\mathbb{x},\mathbb{w})\in\mathcal{E}[\mathbf{R}]$ and $(\mathsf{e}[\mathtt{i}],\mathtt{sk},\mathtt{evk},\mathtt{C}_{\mathtt{sk}})\leftarrow\mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda,\mathtt{i})$ such that $\mathbb{x}=(\mathtt{ct}[x],\mathtt{C}_{\mathtt{sk}})$, it holds that $(x,w)\in\mathbf{R}_{\mathtt{i}}$ for $(x,w)=\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}[x],\mathbb{w})$. Let E be the event that

$$\mathcal{E}[\Pi].\langle\mathsf{P}_{\mathtt{evk}}(\mathsf{e}[\mathtt{i}],\mathbb{w}),\mathsf{V}^{[\![\mathsf{e}[\mathtt{i}]]\!]^{\mathcal{O}^{\mathcal{CT}}}}\rangle(\mathbb{x})\neq\mathsf{acc}$$

and E' the event that

$$\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathcal{E}.\,\mathsf{Eval}_{\mathtt{evk}}(\Pi.\mathsf{P},(\mathsf{e}[\mathtt{i}],\mathbb{x},\mathbb{w})))=\Pi.\mathsf{P}(\mathsf{e}[\mathtt{i}],x,w)$$

over the randomness in both prover and verifier. Then we can show that

$$\Pr[E]=\Pr[E\,|\,E']\Pr[E']+\Pr[E\,|\,\neg E']\Pr[\neg E']$$
$$\leq\Pr[E\,|\,E']+\Pr[\neg E']\leq\delta+\mathsf{negl}(\lambda)$$

since $\Pr[E\,|\,E']$ represents the completeness error in the corresponding Π and $\Pr[\neg E']$ is determined by the correctness of the HE scheme. □

Notice that zero-knowledge has been defined differently from the hIOP case. Informally, an hIOP is zero-knowledge if some simulator Sim can simulate everything the verifier sees without knowledge of the witness. This is formalized by stating no distinguisher algorithm D has an advantage in distinguishing the simulation from a valid prover output. Therefore, since for blind hIOPs the verifier has knowledge of the secret key $\mathtt{sk}$, this will also be given as an input to D. We show that blind hIOPs can retain zero-knowledge by using circuit private HE schemes.

Theorem 5.3. *For an HE scheme $\mathcal{E}$ with security parameter λ and a z-computational honest-verifier zero-knowledge hIOP scheme Π for indexed relation R, the blind hIOP $\mathcal{E}[\Pi]$ is $z + \mathsf{negl}(\lambda)$-computational honest-verifier zero-knowledge if $\mathcal{E}$ is circuit-private for the homomorphic circuit $\mathcal{E}[\Pi].\mathsf{P}$.*

Proof. The simulator for the $\mathcal{E}[\Pi]$ scheme can be constructed by combining the simulator for the Π scheme and the simulator for circuit privacy in the $\mathcal{E}$ scheme. More concretely, $\mathcal{E}[\Pi].\mathsf{Sim}$ uses $\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}$ to compute the statement for $\mathbf{R}_{\mathbb{i}}$. Then, it uses $\Pi.\mathsf{Sim}$ to sample some queries $\{q_i\}$ and lastly uses $\mathcal{E}.\mathsf{Sim}$ to generate the ciphertexts $\{\mathtt{ct}[q_i]\}$. The theorem follows from a standard hybrid argument relying on the circuit privacy of $\mathcal{E}$ and the honest-verifier zero-knowledge property of the Π scheme. □

We additionally define honest-verifier zero-knowledge *in the decryption oracle setting*. A blind hIOP with this property satisfies honest-verifier zero-knowledge with a distinguisher D that has access to $\mathcal{O}_{\mathsf{Dec}}$ instead of $\mathtt{sk}$. Clearly, this property is a more relaxed form of zero-knowledge since D can not see the noise in ciphertexts. This setting will however be sufficient when the blind hIOP verifier's access to $\mathtt{sk}$ is also replaced by access to $\mathcal{O}_{\mathsf{Dec}}$. Such verifier corresponds to the zkDel setting as described in Section 1 and will be used later in Theorem 5.8. We show that zero-knowledge in this setting is achieved trivially since the HE ciphertexts are hiding.

Theorem 5.4. *For an IND-CPA secure HE scheme $\mathcal{E}$ for security parameter λ and a z-computational honest-verifier zero-knowledge hIOP scheme Π for indexed relation R, the blind hIOP $\mathcal{E}[\Pi]$ is $z + \mathsf{negl}(\lambda)$-computational honest-verifier zero-knowledge* in the decryption oracle setting*.*

Proof. The simulator $\mathcal{E}[\Pi].\mathsf{Sim}$ is constructed similar to the simulator in Theorem 5.3 except that it uses $\mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}$ to encrypt the queries $\{q_i\}$.

□

One can derive the (plaintext knowledge) soundness of a blind hIOP from the (knowledge) soundness of the underlying hIOP and the correctness of the HE scheme. We discuss only the round-by-round variants since these are relevant for the BCS compilation.

Theorem 5.5. *The blind hIOP $\mathcal{E}[\Pi]$ is round-by-round sound for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ with error ε if the hIOP Π is round-by-round sound for the indexed relation $\mathbf{R}$ with error ε.*

Proof. By the definition of round-by-round soundness, it is sufficient to show the existence of a doomed set $D' = \mathcal{D}^{\mathcal{E}[\Pi]}$ given the existence of the doomed

set $D = \mathcal{D}^{\Pi}$. We construct a doomed set D' as follows: it contains all possible HE ciphertexts that decrypt to some element in D under the secret key $\mathtt{sk}$ corresponding to $\mathtt{C}_{\mathtt{sk}}$.

$$D' = \left\{ \begin{array}{c} (\mathbb{x}', \mathtt{tr}') = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x] \| \mathtt{ct}[m_1] \| c_1 \| \ldots \| \mathtt{ct}[m_n]) : \\ 0 \leq n \leq \mu + 1 \wedge \mathcal{O}^{\mathcal{CT}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{sk}) = \mathsf{acc} \\ \wedge \exists x, m_1, \ldots, m_n, \mathtt{sk} : (x, m_1, c_1, \ldots, m_n) \in D \\ \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{ct}[m_1], \ldots, \mathtt{ct}[m_n])) = (x, m_1, \ldots, m_n) \end{array} \right\}$$

We prove that this set satisfies all properties of a doomed set for any index $\mathbb{i}$.

1. If $\mathbb{x}' = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x]) \notin \mathcal{L}_{\mathcal{E}[\mathbf{R}_{\mathbb{i}}]}$, then $x = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}[x]) \notin \mathcal{L}_{\mathbf{R}_{\mathbb{i}}}$ where $\mathtt{sk}$ is the opening of $\mathtt{C}_{\mathtt{sk}}$. This means that $(x, \emptyset) \in D$ and therefore $(\mathbb{x}', \emptyset) \in D'$.
2. For any $(\mathbb{x}', \mathtt{tr}') = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], \mathtt{tr}') \in D'$ where $\mathtt{tr}'$ is a $(i-1)$-round partial transcript for $i \in [\mu + 1]$, the corresponding plaintext transcript $(\mathbb{x}, \mathtt{tr}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{tr}')) \in D$ where $\mathtt{sk}$ corresponds to the commitment $\mathtt{C}_{\mathtt{sk}}$. Thus, for any next blind hIOP prover message $\mathtt{ct}_i$ and its decryption $m_i = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}_i)$, it holds that
$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}, \mathtt{tr} \| m_i \| c_i) \notin D] = \Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i \| c_i) \notin D'] \leq \varepsilon.$$
3. For any full transcript $\mathtt{tr}'$, if $(\mathbb{x}', \mathtt{tr}') = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x] \| \mathtt{tr}') \in D'$ then, by definition of D', the decryption $(\mathbb{x}, \mathtt{tr}) \leftarrow \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{tr}')) \in D$. Therefore, by definition of $\mathcal{E}[\Pi].\mathsf{V}$ and the assumption that D is a doomed set, it is clear that $\mathcal{E}[\Pi].\mathsf{V}^{[\![\mathtt{e}[\mathbb{i}]]\!], [\![\mathtt{tr}']\!]}(\mathbb{x}) = \mathsf{rej}$ where $\mathtt{e}[\mathbb{i}] = \mathcal{E}[\Pi].\mathsf{Ind}(\mathbb{i})$.

□

By definition 4.7, a public-coin hIOP $\Pi = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ for an indexed relation $\mathbf{R}$ is round-by-round knowledge sound with error ε_{k} if for every index $\mathbb{i}$ there exists a knowledge doomed set $\mathcal{D}^{\Pi}_{\mathsf{k}}$ for error ε_{k} that uses some polynomial-time extractor Ext. In the case of a blind hIOP for a blind relation $\mathcal{E}[\mathbf{R}_{\mathbb{i}}]$ we define round-by-round knowledge soundness slightly different since it should be able to extract the witness of the corresponding relation $\mathbf{R}_{\mathbb{i}}$.

Definition 5.3 (Round-by-round plaintext knowledge soundness)**.** A blind hIOP $\mathcal{E}[\Pi]$ for an indexed blind relation $\mathcal{E}[\mathbf{R}]$ is round-by-round plaintext knowledge sound with error $\varepsilon_{\mathsf{pk}}$ if for every index $\mathbb{i}$ there exists a knowledge doomed set $\mathcal{D}^{\mathcal{E}[\Pi]}_{\mathsf{k}}$ for error $\varepsilon_{\mathsf{pk}}$ that uses an extractor $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ with access to a decryption oracle $\mathcal{O}_{\mathsf{Dec}}$.

Theorem 5.6. *The blind hIOP $\mathcal{E}[\Pi]$ is round-by-round plaintext knowledge sound for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ with error ε_{k} if the hIOP Π is round-by-round knowledge sound for the indexed relation $\mathbf{R}$ with error ε_{k}.*

Proof. Similar to Theorem 5.5, we show that there exists a knowledge doomed set $D' = \mathcal{D}_{\mathsf{k}}^{\mathcal{E}[\Pi]}$ given the existence of the knowledge doomed set $D = \mathcal{D}_{\mathsf{k}}^{\Pi}$. Again we define a set D' to contain all HE ciphertext that decrypt to some transcript in D under the secret key $\mathtt{sk}$ corresponding to $\mathtt{C}_{\mathtt{sk}}$. It is clear that D' satisfies the first and third property of a knowledge doomed set. The second property states that for any $(i-1)$-round partial transcript $(\mathbb{x}', \mathtt{tr}') \in D'$ where $i \in [\mu + 1]$, and any next prover message $\mathtt{ct}_i$, it should hold that if

$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i \| c_i) \notin D'] > \varepsilon_{\mathsf{k}}$$

then $\mathsf{Ext}_{\mathtt{sk}}(\mathtt{e}[\mathtt{i}], \mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i)$ outputs a valid witness for $\mathbb{x}$. Similarly as in Theorem 5.5, if we define $m_i = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}_i)$ then $(\mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i \| c_i) \notin D'$ implies $(\mathbb{x}, \mathtt{tr} \| m_i \| c_i) \notin D$ for any c_i and $(\mathbb{x}, \mathtt{tr}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{tr}'))$. Therefore, we can construct the extractor $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ as first requesting $m_i = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}_i)$ from $\mathcal{O}_{\mathsf{Dec}}$ and subsequently running the extractor $\mathsf{Ext}(\mathtt{e}[\mathtt{i}], \mathbb{x}, \mathtt{tr} \| m_i)$, the output will be a valid witness for $\mathbb{x}$ since $(\mathbb{x}, \mathtt{tr}) \in D$.

□

5.2 Designated-Verifier Blind zkSNARK (DV-BzkSNARK)

From Theorem 4.1, it is clear that any public-coin hIOP Π for some indexed relation $\mathbf{R}$ can be compiled into a zkSNARK for $\mathbf{R}$ in the ROM using BCS compilation. In practice, the RO is instantiated with some suitable hash function. The compiler functions by committing to every oracle message sent by the prover using a Merkle Tree MT and instead sending the commitment root $\mathtt{C}$. Then, when the verifier queries some oracle message, the prover responds to the query by including the authentication path $\mathtt{ap}$ from the corresponding root to the queried value in the message. Lastly, since Π is public-coin, one can make the proof non-interactive using a Fiat-Shamir-like transform FS to simulate the verifier's challenges and generate a final RO output τ.

Let us now describe *public-coin* hIOP verification as follows. The verifier $\Pi.\mathsf{V}$ receives the statement x and in each round i receives a message m_i, contributing to the current partial transcript $\mathtt{tr}_i$, and responds with a challenge $c_i \leftarrow \mathcal{C}h_i$. After receiving the final prover message, the verifier queries the oracle $[\![\mathtt{tr}]\!]$ to construct a list of queries $\{q_i\}$ (same for the oracle $[\![\mathtt{e}[\mathtt{i}]]\!]$, but we dismiss holography for now for ease of notation). Lastly, the verifier returns $\mathtt{acc}$ if and only if some equality $f(x, \{q_i\}, \tau) = 0$ holds where τ represents some randomness. In Figure 1, we illustrate the behaviour of BCS compilation from Theorem 4.1 using this notation. It describes the zkSNARK verifier resulting from this compilation. Instead of performing queries, the verifier receives query responses

and checks their authentication paths and whether they where sampled using the RO.

$\underline{\Pi\!\!\Pi.\mathsf{V}(x, \pi = [\{q_i, \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau])}$

1 : **foreach** j :

2 : $\quad \mathsf{MT.Open}(q_j, \mathtt{ap}_j, \mathtt{C}_{\mathsf{qr}(j)}) \stackrel{?}{=} \mathsf{acc}$

3 : $\mathsf{FS}(x, \{\mathtt{C}_i\}) \stackrel{?}{=} \tau$

4 : $f(x, \{q_i\}, \tau) \stackrel{?}{=} 0$

Figure 1: Verifier of the compiled zkSNARK resulting from Theorem 4.1.

It should be clear that this compilation can likewise be applied to the *public-coin* BhIOP from Definition 5.2. Since the verifier $\mathcal{E}[\Pi].\mathsf{V}$ is public-coin, the only difference to the hIOP verifier will be after receiving the final prover message. By querying the oracle $[\![\mathtt{tr}]\!]$, the verifier $\mathcal{E}[\Pi].\mathsf{V}$ receives ciphertexts $\{\mathtt{ct}[q_i]\}$ that are decrypted to $\{q_i\}$ using $\mathtt{sk}$. Similarly, the verifier decrypts the statement $\mathtt{ct}[x]$ to x and then computes $b \leftarrow \Pi.\mathsf{V}_{\mu+1}(x, \{q_i\}, \tau)$, which we have previously denoted as checking whether some equality $f(x, \{q_i\}, \tau) = 0$ holds, for some randomness τ. Lastly, the verifier returns b if the commitment $\mathtt{C}_{\mathtt{sk}}$ is a valid commitment to $\mathtt{sk}$. We describe this compilation in Theorem 5.7 and the resulting verifier in Figure 2. Note that we define a subroutine $\mathsf{PartialVer}$ that performs verification without verifying the correspondence between the queries $\{\mathtt{ct}[q_i]\}$ and their plaintexts $\{q_i\}$.

The fact that $\mathcal{E}[\Pi].\mathsf{V}$ requires knowledge of the secret key $\mathtt{sk}$ has two major consequences for BCS compilation. Most notably, the resulting zkSNARK verifier $\mathcal{E}[\mathsf{dv}\Pi\!\!\Pi].\mathsf{V}$ inherits the same requirement; thus, the compiler outputs a *designated-verifier* blind zkSNARK. Secondly, the public-coin requirement for the verifier $\Pi.\mathsf{V}$ is no longer sufficient. Strictly, it ensures that the hIOP verifier is simulatable by the zkSNARK prover in BCS compilation. It can simulate the random challenges using the Fiat-Shamir tranform and simulate the queries by providing Merkle Tree openings. To ensure that the blind hIOP verifier $\mathcal{E}[\Pi].\mathsf{V}$ is simulatable by the blind zkSNARK prover, we must additionally require that it performs no queries where the query location depends on previously queried values (since those are hidden from the prover). This holds for queries to both the $[\![\mathtt{e}[\mathtt{i}]]\!]$ and the $[\![\mathtt{tr}]\!]$ oracles. Nassar et al. [73] have previously coined hIOPs with such verifiers *non-adaptive*. To our knowledge, such hIOP has never been described and so this requirement forms no restriction.

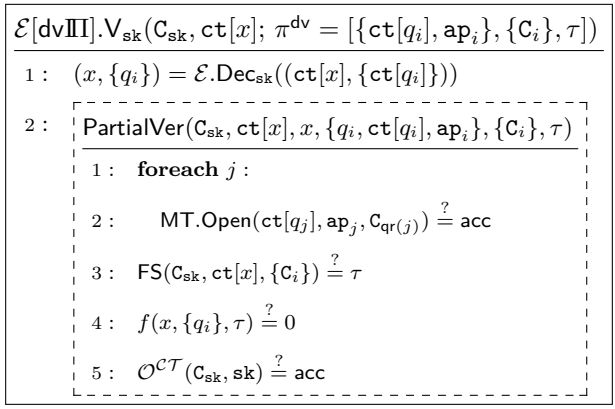


Figure 2: Verifier of the designated-verifier blind zkSNARK resulting from Theorem 5.7.

Theorem 5.7. *Let $\mathcal{E}[\Pi]$ be a non-adaptive public-coin blind hIOP for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ where Π has completeness error δ, proof length p, query complexity q, round-by-round soundness error ε, round-by-round knowledge soundness error ε_{k} and z-statistical honest-verifier zero-knowledge, and the HE scheme $\mathcal{E}$ is correct for the homomorphic circuit in $\mathcal{E}[\Pi].\mathsf{P}$ with security parameter λ. Such blind hIOP scheme can be compiled into a designated-verifier zero-knowledge non-interactive argument of plaintext knowledge for $\mathcal{E}[\mathbf{R}]$, which we will coin a designated-verifier blind zkSNARK $\mathcal{E}[\mathsf{dvIII}]$ in the ROM. Then, against Q-query adversaries, $\mathcal{E}[\mathsf{dvIII}]$ has:*

- *Completeness error $\delta + \mathsf{negl}(\lambda)$,*
- *Proof length p' upper bounded by*

$$p' = \lambda(\mu + 1 + \sum_{j=1}^{q}(2 + \lceil \log_2 |m_{\mathsf{qr}(j)}|/\mathcal{E}.\mathsf{len}\rceil)) + q\mathcal{E}.\mathsf{len},$$

- *Soundness error ε' where $\varepsilon' = Q\varepsilon + 3(Q^2+1)2^{-\lambda}$,*
- *Knowledge soundness error $\varepsilon'_{\mathsf{k}}$ where $\varepsilon'_{\mathsf{k}} = Q\varepsilon_{\mathsf{k}} + 3(Q^2+1)2^{-\lambda}$,*
- *z'-Statistical honest-verifier zero-knowledge where $z' = z + \mathsf{negl}(\lambda) + p2^{-\lambda/4+2}$.*

where m_i is $\mathcal{E}[\Pi].\mathsf{P}$'s ith prover message and $|\cdot|$ denotes length in λ bits rounded up. Both soundness and knowledge soundness error are $\Theta(Q\varepsilon)$ and $\Theta(Q\varepsilon_{\mathsf{k}})$ respectively when considering quantum adversaries that perform no more than $Q - \mathcal{O}(q \log p)$ RO queries.

Proof. The proof follows trivially from Theorem 4.1 and the discussion above. The non-adaptivity ensures that prover $\mathcal{E}[\mathsf{dv}\Pi].\mathsf{P}$ can partly simulate the verifier $\mathcal{E}[\Pi].\mathsf{V}$ to compute the query locations (it can obviously not simulate the equality checks). The proof length is similar, except for the expansion from HE encryption of the queried values.

□

5.3 Publicly-Verifiable Blind zkSNARK (PV-BzkSNARK)

We have shown that using an HE scheme, it is possible to construct a blind zkSNARK in the designated-verifier setting. This primitive already has applications as a vCOED scheme; in this setting, the client would encrypt the statement and the server would compute the encrypted witness, which can then be used to compute the proof. Now, we will show how to compile this designated-verifier blind zkSNARK into a publicly-verifiable blind zkSNARK; this also expands the application of this construction to zkSNARK delegation (the zkDel setting). In this setting, the client computes the witness and then sends the *plaintext* statement and encrypted witness to the server, who then computes the proof. In both scenarios, the client computes a (batched) Proof of Decryption (PoD) to make the proof publicly verifiable.

Below we provide a formal definition.

Definition 5.4 (Proof of Decryption)**.** For a given HE scheme $\mathcal{E}$ with security parameter λ and a commitment scheme $\mathcal{CT}$, a Proof of Decryption scheme $\mathsf{PoD}[\mathcal{E}] = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ includes the following probabilistic polynomial-time algorithms:

- $\mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$: for a given security parameter λ, it returns some public parameters $\mathtt{pp}$ which are implicit inputs to the following functions.
- $\mathsf{PoD}[\mathcal{E}].\mathsf{P}_{\mathtt{sk}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{ct})$: for a given secret key commitment $\mathtt{C}_{\mathtt{sk}}$, ciphertext $\mathtt{ct}$ and secret key $\mathtt{sk}$, it returns a proof of decryption π^{PoD} and plaintext m.
- $\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathtt{C}_{\mathtt{sk}}}(\pi^{\mathsf{PoD}}, \mathtt{ct}, m)$: for a given proof of decryption π^{PoD}, ciphertext $\mathtt{ct}$, plaintext message m and secret key commitment $\mathtt{C}_{\mathtt{sk}}$, it returns either acc or rej.

It should satisfy the following properties.

Completeness. For any public parameters $\mathtt{pp} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$, HE keys $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and ciphertexts $\{\mathtt{ct}[m_i]\}$ such that $\{m_i\} = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\{\mathtt{ct}[m_i]\})$, it holds that

$$\Pr\left[\begin{array}{c}\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathtt{C}_{\mathtt{sk}}}(\pi^{\mathsf{PoD}}, \{\mathtt{ct}[m_i], m_i\}) \\ \neq \\ \mathsf{acc}\end{array} \;\middle|\; \begin{array}{c}\mathtt{C}_{\mathtt{sk}} \leftarrow \mathsf{Com}(\mathtt{sk}) \\ \pi^{\mathsf{PoD}} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{P}_{\mathtt{sk}}(\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}[m_i]\})\end{array}\right]$$

is less than or equal to some completeness error δ.

Knowledge soundness. For any public parameters $\mathtt{pp} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$, HE keys $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and PPT prover P^*, there exists a PPT extractor $\mathsf{Ext}^{\mathsf{P}^*}$ and knowledge error ε_{k} such that

$$\Pr\left[\begin{array}{c} \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}^*}(\{\mathtt{ct}_i\}) \neq \{m_i\} \\ \wedge \\ \mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathtt{C}_{\mathtt{sk}}}(\pi^{\mathsf{PoD}}, \{\mathtt{ct}_i, m_i\}) = \mathsf{acc} \end{array} \middle| \begin{array}{c} (\pi^{\mathsf{PoD}}, \mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}_i, m_i\}) \leftarrow \mathsf{P}^* \\ \mathtt{sk}^* \leftarrow \mathsf{Ext}^{\mathsf{P}^*} \\ \mathcal{O}^{\mathcal{CT}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{sk}^*) = \mathsf{acc} \end{array}\right] \leq \varepsilon_{\mathsf{k}}.$$

Zero-knowledge. For any public parameters $\mathtt{pp} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$, HE keys $(\mathtt{sk}, \mathtt{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and ciphertexts $\{\mathtt{ct}[m_i]\}$, if there exists a probabilistic polynomial-time simulator Sim such that for any probabilistic polynomial-time distinguisher D it holds that

$$\left|\Pr\left[\ \mathsf{D}(\pi^{\mathsf{PoD}}, \mathtt{C}_{\mathtt{sk}}) = 1 \ \middle|\ (\pi^{\mathsf{PoD}}, \mathtt{C}_{\mathtt{sk}}) \leftarrow \mathsf{Sim}(1^\lambda)\ \right] - \Pr\left[\ \mathsf{D}(\pi^{\mathsf{PoD}}, \mathtt{C}_{\mathtt{sk}}) = 1 \ \middle|\ \begin{array}{c} \mathtt{C}_{\mathtt{sk}} \leftarrow \mathsf{Com}(\mathtt{sk}) \\ \pi^{\mathsf{PoD}} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{P}_{\mathtt{sk}}(\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}[m_i]\}) \end{array}\ \right]\right| \leq z$$

then $\mathsf{PoD}[\mathcal{E}]$ has z-computational zero-knowledge.

In the following, we discuss how a proof of decryption for the HE scheme $\mathcal{E}$ allows us to compile a designated-verifier blind zkSNARK $\mathcal{E}[\mathsf{dv}\Pi]$ into a publicly verifiable zkSNARK $\mathcal{E}[\mathsf{pv}\Pi]$. Any party that can verify a proof π^{dv} is able to construct a proof π^{pv} by appending the plaintext queries $\{q_i\}$ along with a PoD proving they are decryptions of the queries $\{\mathtt{ct}[q_i]\}$ that were committed to in $\{\mathtt{C}_i\}$, using the secret key committed to in $\mathtt{C}_{\mathtt{sk}}$. This results in the public verifier described in Figure 3.

$\mathcal{E}[\mathsf{pv}\Pi].\mathsf{V}(x;\ \pi^{\mathsf{pv}} = [\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], \pi^{\mathsf{PoD}}, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau])$

1 : $\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathtt{C}_{\mathtt{sk}}}(\pi^{\mathsf{PoD}}, (\mathtt{ct}[x], \{\mathtt{ct}[q_i]\}), (x, \{q_i\})) \stackrel{?}{=} \mathsf{acc}$

2 : $\mathsf{PartialVer}(\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], x, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau) \stackrel{?}{=} \mathsf{acc}$

Figure 3: Verifier of the publicly-verifiable zkSNARK resulting from Theorem 5.8.

In Figure 4, we describe the construction of π^{pv}, and in Theorem 5.8, we prove that it describes a publicly-verifiable blind zkSNARK. If the PV-BzkSNARK is used in the zkDel setting, one can replace $\mathtt{ct}[x]$ by x in the blind relation

and thus $\mathtt{ct}[x]$ requires no PoD and is not included in π^{pv}. Note that it is not necessary for the delegator to send the entire encrypted (extended) witness $\mathtt{ct}[w]$; they can simply send an encryption of the private inputs, from which $\mathtt{ct}[w]$ can be computed homomorphically. This would demand less encryption cost from the verifier and would not increase HE parameters. In the vCOED setting, one depends on Theorem 5.1 to hide x from the blind prover; therefore, $\mathtt{ct}[x]$ should be included in the proof.

Theorem 5.8. *For security parameter λ, the protocol in Figure 4 is a publicly-verifiable zkSNARK for the relation $\mathbf{R}$ in the ROM against Q-query adversaries with completeness error $\delta^{\Pi} + \delta^{\mathsf{PoD}} + \mathsf{negl}(\lambda)$ and knowledge soundness error $\varepsilon_{\mathsf{k}}^{\mathsf{PoD}} + Q\varepsilon_{\mathsf{k}}^{\Pi} + 3(Q^2+1)2^{-\lambda}$ that is $z^{\Pi} + \mathsf{negl}(\lambda) + p2^{-\lambda/4+2} + z^{\mathsf{PoD}}$-computational zero-knowledge if $\mathcal{E}$ is an IND-CPA secure and correct HE scheme,* PoD *is a zero-knowledge proof of decryption with completeness error δ^{PoD} and knowledge soundness error $\varepsilon_{\mathsf{k}}^{\mathsf{PoD}}$ that is z^{PoD}-computational zero-knowlege, and Π is an hIOP scheme with completeness error δ^{Π} and knowledge soundness $\varepsilon_{\mathsf{k}}^{\Pi}$ that is z^{Π}-computational zero-knowledge with proof length p.*

Proof. Completeness follows immediately from the completeness of the PoD scheme along with Theorem 5.7. Similarly, the zero-knowledge property follows from a hybrid argument using the zero-knowledge property of the PoD and Theorem 5.7. We discuss knowledge soundness in more detail.

Let us denote P^* as a prover that outputs a proof π^{pv} for the index $\mathtt{i}$ and statement x such that the verifier $\mathcal{E}[\mathsf{pv}\Pi].\mathsf{V}$ (see Figure 3) accepts with probability p. To prove knowledge soundness, we will show that there exists a polynomial-time extractor Ext that outputs w with probability greater than $p - \varepsilon_{\mathsf{k}}^{\mathsf{PoD}} - \varepsilon_{\mathsf{k}}^{\mathcal{E}[\mathsf{dv}\Pi]}$, when given access to P^*. Firstly, when $\mathcal{E}[\mathsf{pv}\Pi].\mathsf{V}$ accepts, the first line in Figure 3 states that $\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathtt{C}_{\mathtt{sk}}}$ also accepts. Therefore, by the knowledge soundness of PoD, the prover P^* can be used to extract a secret key $\mathtt{sk} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Ext}$ such that

$$(x, \{q_i\}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \{\mathtt{ct}[q_i]\}))$$

and $\mathtt{sk}$ is the secret key committed to in $\mathtt{C}_{\mathtt{sk}}$. Secondly, from Theorem 5.7 we know that the designated-verifier zkSNARK $\mathcal{E}[\mathsf{dv}\Pi]$ is plaintext knowledge sound. In other words, any prover that can produce a proof π^{dv} such that the verifier in Figure 2 satisfies, can be used to extract a witness $\mathbb{w} \leftarrow \mathcal{E}[\mathsf{dv}\Pi].\mathsf{Ext}_{\mathtt{sk}}$ such that $(\mathtt{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$. Note that by assumption, P^* generates proofs that satisfy the $\mathsf{PartialVer}$ subroutine in $\mathcal{E}[\mathsf{dv}\Pi].\mathsf{V}_{\mathtt{sk}}$. The knowledge soundness of the PoD discussed before ensures that also the first line in Figure 2 is satisfied. Therefore, our prover P^* can be used by the extractor $\mathcal{E}[\mathsf{dv}\Pi].\mathsf{Ext}_{\mathtt{sk}}$ where $\mathtt{sk}$ is the secret key extracted previously using $\mathsf{PoD}[\mathcal{E}].\mathsf{Ext}$. □

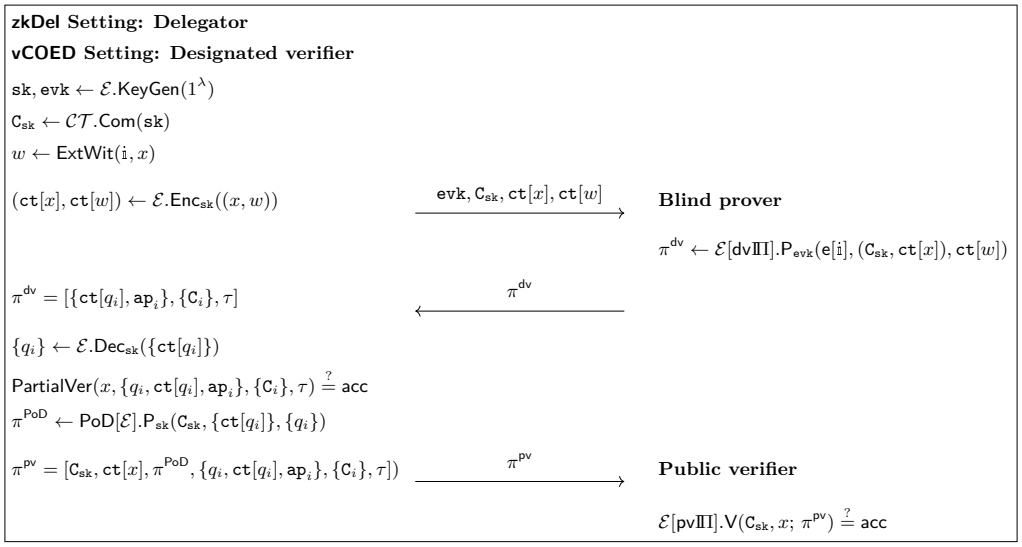


Figure 4: Compilation of a DV-BzkSNARK into a PV-BzkSNARK.

Remark 5.2. *Note that the execution of* PartialVer *by the client is only required in a setting where the blind prover is incentivized to be dishonest. In such setting, it is also required to not reuse the same HE secret key. Note that most HE schemes are vulnerable to key-recovery attacks when the client leaks to the server whether the ciphertexts properly decrypt. In the zkDel setting, the server would learn this when it has access to the publicly-verifiable proof.*

Remark 5.3. *In our PoD construction, the commitment $\mathtt{C}_{sk}$ is included in the π^{PoD}. Naturally, the proof size of π^{pv} (see Theorem 5.7) is still larger than a normal proof for the zkSNARK Π. This could be mediated by sending π^{pv} to the delegatee to delegate the computation of a recursion step.*

6 Instantiation of blind zkSNARKs

In this section, we describe algorithms for computing blind zkSNARKs efficiently. Concretely, for some specific Π, we optimize the computation of $\mathcal{E}[\Pi].\mathsf{P}$ such that the blind zkSNARK resulting from Theorem 5.7 has efficient proof generation. The input to $\mathcal{E}[\Pi].\mathsf{P}$ is the encrypted trace $\mathtt{ct}[z] = (x, \mathtt{ct}[w])$, where x and w are vectors in some finite field $\mathbb{F}$, and computing $\mathcal{E}[\Pi].\mathsf{P}$ consists of homomorphically evaluating $\Pi.\mathsf{P}$ using $\mathcal{E}.\mathsf{Eval}$.

Most HE schemes naturally support Single Instruction Multiple Data (SIMD) operations since the plaintext space can be interpreted as the vector space $\mathcal{P} = \mathbb{F}^P$ for some finite field $\mathbb{F}$. Operating on plaintexts `pt` and/or ciphertexts `ct` corresponds to pointwise operations on elements in $\mathcal{P}$, such as pointwise addition and multiplication. Using automorphisms, it is also possible to compute arbitrary linear operations on an encrypted vector, i.e. a matrix-vector multiply.

The inherent noise growth in a ciphertext depends on the type of operation: additions (both `pt-ct` as well as `ct-ct`) cause additive noise growth, whereas multiplication incurs a fixed multiplicative factor depending on the parameters of the scheme. A `pt-ct` multiplication incurs a smaller noise growth than a `ct-ct` multiplication and is also much faster to compute. As an example, for the parameter set used in our implementation, the noise growth would be on average 6.2 bits for a `pt-ct` and 10.3 bits for a `ct-ct`, and a `pt-ct` multiply is 75× faster than a `ct-ct` multiply. An automorphism causes minimal noise growth as it does not change the norm in the canonical embedding; in our implementation, it takes 1/4 of the time of a `ct-ct` multiplication. As will become clear in this section, designing efficient homomorphic circuits is largely a trade-off between the number of operations performed and the amount of noise they require.

Similar to Garg et al. [49], we suggest using the Fractal hIOP Π_{F} [37]; this choice lets us optimize the homomorphic computation of the corresponding blind hIOP in two ways. Firstly, many of the operations in Fractal can be performed element-wise on vectors, which allows us to significantly lower the number of homomorphic operations using SIMD as described above. Secondly, Fractal's linearity implies that many of the homomorphic operations will consist of the cheaper `pt-ct` operations. In Section 8, we select an HE scheme that allows us to exploit both these characteristics. In Section 6.1 below, we introduce some techniques for homomorphically computing Number Theoretic Transforms (NTTs) and arbitrary linear operations. In Section 6.2, we describe the Fractal scheme and discuss an efficient algorithm for computing it blindly, namely the algorithm $\mathcal{E}[\Pi_{\mathsf{F}}].\mathsf{P}$ from Definition 5.2.

6.1 Building blocks

Packing-friendly 2D-NTT. Consider some finite field $\mathbb{F}$ and an element $\xi_N \in \mathbb{F}$ of order N known as the primitive N-th root of unity. The (inverse) NTT transformation of a vector of polynomial evaluations $\mathbf{a} = [a_0, a_1, \ldots, a_{N-1}] \in \mathbb{F}^N$ is denoted as $\hat{\mathbf{a}} = [\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{N-1}] \in \mathbb{F}^N$ where

$$\hat{a}_k = \frac{1}{N} \sum_{n=0}^{N-1} \xi_N^{-n \cdot k} a_n \quad \forall k \in [0, N-1]. \tag{1}$$

Observe that an inverse NTT differs from the NTT only by using ξ_N^{-1} instead of ξ_N and scaling by $1/N$ at the end. Let us now assume the NTT size N can be written as $N = N_1 \cdot N_2$. As such, we can rewrite Equation (1) by iterating over $n = N_2 \cdot n_1 + n_2$ and $k = N_1 \cdot k_2 + k_1$ using $k_i, n_i \in [0, N_i - 1] = [N_i]_0$ for $i = 1, 2$

$$\begin{aligned}\hat{a}_{N_1 \cdot k_2 + k_1} &= \frac{1}{N} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \xi_N^{-(N_2 \cdot n_1 + n_2)(N_1 \cdot k_2 + k_1)} a_{N_2 \cdot n_1 + n_2} \\ &= \frac{1}{N} \sum_{n_2=0}^{N_2-1} \xi_{N_2}^{-n_2 \cdot k_2} \left(\xi_N^{-n_2 \cdot k_1} \sum_{n_1=0}^{N_1-1} \xi_{N_1}^{-n_1 \cdot k_1} a_{N_2 \cdot n_1 + n_2} \right) \quad \forall k_i \in [N_i]_0\end{aligned}$$

where $\xi_{N_1} = \xi_N^{N_2}$ and $\xi_{N_2} = \xi_N^{N_1}$ are primitive N_1-th and N_2-th roots of unity, respectively. Let us define $f_{n_2,k_1} = \sum_{n_1=0}^{N_1-1} \xi_{N_1}^{-n_1 \cdot k_1} a_{N_1 \cdot n_2 + n_1}$. Now, notice that a size N NTT can be composed into three steps:

1. for $n_2 \in [N_2]_0$ perform a size-N_1 NTT to compute $[f_{n_2,k_1}]_{k_1 \in [N_1]_0}$,
2. multiply by twiddle factors to get $f'_{n_2,k_1} = \xi_N^{-n_2 \cdot k_1} \cdot f_{n_2,k_1}$,
3. for $k_1 \in [N_1]_0$ perform a size-N_2 NTT to compute $[\hat{a}_{N_1 \cdot k_2 + k_1}]_{k_2 \in [N_2]_0}$.

While a variant of the 2D-NTT algorithm has been used to achieve distributed FFT in a distributed zero-knowledge proof system [80], our work demonstrates its effectiveness in a different context, namely for homomorphically performing NTTs on packed HE ciphertexts. Note that [5] also considered exploiting HE packing but only for performing P homomorphic NTTs at once, requiring packing and unpacking operations. They do not perform homomorphic NTTs for a degree N polynomial packed in N/P ciphertexts since it would require expensive HE operations to swap slots between ciphertexts (think of the bit-reversal that the base 2 butterfly algorithm causes). These can only be avoided by using one slot per N ciphertexts, resulting in higher memory usage and reduced efficiency.

Our approach packs the size N input into $N/P = N_1(N_2/P)$ ciphertexts and avoids swapping slots between ciphertexts by exploiting the structure of the 2D-NTT. More precisely, we pack N_2 elements $(a_{N_2 \cdot n_1}, a_{N_2 \cdot n_1 + 1}, \ldots, a_{N_2 \cdot n_1 + N_2 - 1})$ into N_2/P ciphertexts for $n_1 \in [N_1]_0$. Then, the homomorphic NTT evaluation can be performed as follows:

1. homomorphically evaluate N_2/P size-N_1 NTTs using the butterfly algorithm in some base b on packed ciphertexts,
2. perform N/P `pt-ct` multiplications to multiply with twiddle factors,
3. homomorphically evaluate N_1 size-N_2 NTTs as matrix-vector multiplications with vectors of size N_2 packed in N_2/P ciphertexts.

An example is provided in Fig. 5 for $N_2 = P$. Note that the output of the 2D-NTT is packed in column-major order, i.e. N_2 elements $(\hat{a}_{k_1}, \hat{a}_{N_1+k_1}, \ldots, \hat{a}_{N_1\cdot(N_2-1)+k_1})$ are packed in one ciphertext. Although this ordering may seem problematic, notice that it can be reversed by a subsequent NTT operation that is computed as

$$\begin{aligned} a_{N_2\cdot n_1+n_2} &= \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} \xi_N^{(N_2\cdot n_1+n_2)(N_1\cdot k_2+k_1)}\hat{a}_{N_1\cdot k_2+k_1} \\ &= \sum_{k_1=0}^{N_1-1} \xi_{N_1}^{n_1\cdot k_1}\left(\xi_N^{n_2\cdot k_1}\sum_{k_2=0}^{N_2-1}\xi_{N_2}^{n_2\cdot k_2}\hat{a}_{N_1\cdot k_2+k_1}\right) \quad \forall n_i \in [N_i]_0 \end{aligned}$$

by doing the following steps:

1. homomorphically evaluate N_1 size-N_2 NTTs as matrix-vector multiplications with vectors of size N_2 packed in N_2/P ciphertexts,
2. perform N/P `pt-ct` multiplications to multiply with twiddle factors,
3. homomorphically evaluate N_2/P size-N_1 NTTs using the butterfly algorithm in some base b on packed ciphertexts.

The algorithms in Section 6.2 require us to compute

$$[a(x)_{x\in D_2}] = \mathrm{NTT}\left(f\left(\mathrm{iNTT}([a(x)]_{x\in D_1})\right)\right) \tag{2}$$

for some evaluation domains D_1, D_2 and some function f that must be computed on ciphertext vectors in some column-major packing. It will always be the case that $|D_2| > |D_1|$ so $[\hat{a}_i]_{i\in[N]_0}$ has to be appended with zeros. Due to the column-major packing, the width $w \coloneqq N_2/P$ has to grow with $|D_2|/|D_1|$ before performing the second 2D-NTT. Importantly, this makes the matrix-vector multiplications of the second 2D-NTT less costly since they can take these zeros into account. Lastly, we remark that these techniques can also be applied when D_1, D_2 are cosets of multiplicative subgroups.

Matrix-vector multiplication. Let $\mathbf{A} \in \mathbb{F}^{nP\times nP}$ denote a square matrix and $\mathbf{v} \in \mathbb{F}^{nP}$ denote a column vector over a finite field $\mathbb{F}$. For HE schemes that pack P finite field elements per ciphertext, the multiplication of the plaintext matrix $\mathbf{A}$ with encryptions of $\mathbf{v}$ can be visualized as

$$\left(\begin{array}{c|c|c} \mathbf{A_{0,0}} & \cdots & \mathbf{A_{0,n-1}} \\ \hline \vdots & \ddots & \vdots \\ \hline \mathbf{A_{n-1,0}} & \cdots & \mathbf{A_{n-1,n-1}} \end{array}\right) \cdot \left(\begin{array}{c} \mathtt{ct}[\mathbf{v_0}] \\ \hline \vdots \\ \hline \mathtt{ct}[\mathbf{v_{n-1}}] \end{array}\right) \tag{3}$$

where $\mathbf{v_i} = [v_{i\cdot P}, v_{i\cdot P+1}, \ldots, v_{i\cdot P+P-1}] \in \mathbb{F}^P$ and $\mathbf{A_{i,j}} \in \mathbb{F}^{P\times P}$ denotes the (i,j)-th block matrix of $\mathbf{A}$. For the base case of $n = 1$, the computation of (3) can be performed using the following two methods

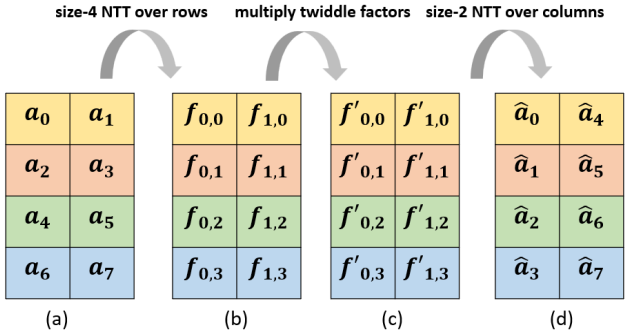


Figure 5: Illustration of inverse 2D-NTT when $N_1 = 4, N_2 = 2$. Elements within the same row (represented by the same color) are packed in one HE ciphertext.

- Full matrix in the diagonal (FD) method [58], which requires $P - 1$ automorphisms, P `pt-ct` multiplications, and $P - 1$ additions. In terms of noise depth, it incurs the equivalent of 1 `pt-ct` operation, 1 automorphism and $\lceil \log_2 P \rceil$ additions.
- Baby-Step/Giant-Step diagonal (BS/GS) method [59], which requires $2\sqrt{P} - 2$ automorphisms, P `pt-ct` multiplications, and $P - 1$ additions. In terms of noise depth, it incurs the equivalent of 1 `pt-ct` operation, 2 automorphisms and $2\lceil \log_2 \sqrt{P} \rceil$ additions.

The performance of both methods could be further improved with the *hoisting* technique [21, 59], which speeds up FD around 6x and BS/GS around 1.4x for implementations that store ciphertexts in the NTT form (e.g. the HElib library). With hoisting, whether FD or BG/GS gives better performance depends on parameter settings, in particular the dimension P.

For $n > 1$, we notice the automorphisms of $\texttt{ct}[\mathbf{v_i}]$ in FD can be re-used for multiplications with block matrices in the same block-column, i.e. $\mathbf{A_{0,i}}, \mathbf{A_{1,i}}, \ldots, \mathbf{A_{n-1,i}}$. As such, the computation of (3) requires $n(P - 1)$ automorphisms, $n^2 P$ `pt-ct` multiplications, and $n(nP - 1)$ additions. Notice that for BSGS, only the baby step automorphisms can be re-used and so FD can be more efficient for large n. Furthermore, if $\mathbf{A}$ is a sparse matrix with h non-zero elements per row, the expected number of operations can be computed by using the following lemma.

Lemma 6.1 (Adapted from [76])**.** *Given n elements grouped into m equal partitions, the expected number of partitions hit when sampling k distinct*

elements ($k \leq n - n/m$) *is given by*

$$\mathsf{PartsHit}(n, m, k) = m \cdot \left[1 - \binom{n - \frac{n}{m}}{k} \Big/ \binom{n}{k}\right].$$

Then, the homomorphic evaluation of (3) requires $n \cdot \mathsf{PartsHit}(nP^2, nP, hP)$ `pt-ct` multiplications, $n \cdot \mathsf{PartsHit}(nP(P-1), P-1, h(P-1))$ automorphisms, and $n \cdot (\mathsf{PartsHit}(nP^2, nP, hP) - 1)$ additions.

For our purposes, it will always be more efficient to perform the sparse matrix-vector multiplications with a relatively large n using FD. On the other hand, the matrix-vector products performed in 2D-NTTs always use BSGS (not considering hoisting).

6.2 Computing Fractal blindly

Fractal is a transparent, post-quantum, preprocessing zkSNARK that proves the Rank-1 Constraint Satisfiability (R1CS) $Az \circ Bz = Cz$ of $z = (x, w)$, where x is the statement, w is the (extended) witness, and A, B, C are sparse matrices that represent the computation to be proven. Its construction starts from a type of IOP named Reed Solomon encoded-holographic IOP (RS-hIOP) that is compiled into an hIOP. In a RS-hIOP, an indexer provides RS codes in an offline phase, the prover's messages are RS codes and the verifier outputs a set of rational constraints on these RS codes. A rational constraint on some RS codes checks that some rational function of the underlying polynomials has a limited degree. For proofs of invalid statements, at least one of these rational constraints will not hold.

Denote by $(f_z, f_{Az}, f_{Bz}, f_{Cz})$ the polynomials that interpolate the vectors (z, Az, Bz, Cz) over some cyclic subgroup H of $\mathbb{F}$; then, the prover's first messages will be the RS codes $(\overrightarrow{f_z}, \overrightarrow{f_{Az}}, \overrightarrow{f_{Bz}}, \overrightarrow{f_{Cz}})$ over some domain $L = \{\ell_i\}_{i \in [|L|]}$. In particular, the RS codes correspond to evaluations of these polynomials on some evaluation domain L and are used to prove three statements that together imply the satisfiability of the R1CS constraint system:

(1) $f_{Az}\big|_H \circ f_{Bz}\big|_H - f_{Cz}\big|_H = 0$
(2) $f_z\big|_I = f_x\big|_I = x$ for some subset I of H
(3) $f_{Mz}\big|_H = M \cdot f_z\big|_H$ for $M \in \{A, B, C\}$.

The previous three statements are proven using some rational constraints on $\overrightarrow{f_z}, \overrightarrow{f_{Az}}, \overrightarrow{f_{Bz}}, \overrightarrow{f_{Cz}}$ and other RS codes derived from z. In our setting, however, the blind prover will have as input the HE encrypted (extended) witness $\mathtt{ct}[w]$, leading to the (partially) encrypted trace $\mathtt{ct}[z] = (x, \mathtt{ct}[w])$. Therefore, all

RS codes derived from the trace, as well as all subsequent prover messages required for proving rational constraints, will similarly be encrypted and from now on referred to as Encrypted RS (ERS) codes. Computing them efficiently is a trade-off between minimizing the homomorphic depth and minimizing the execution time (determined by the number of required homomorphic operations and the parameters of the HE system).

6.3 Proving statement (1)

Starting from the encrypted trace $\texttt{ct}[z] = (x, \texttt{ct}[w])$, the prover computes the ERS codes $\texttt{ct}[\overrightarrow{f_{Mz}}]$ for $M \in \{A, B, C\}$. Computing the underlying $\texttt{ct}[Mz] = \texttt{ct}[f_{Mz}\big|_H]$ requires a sparse matrix-vector product of size $|H|$ over ciphertexts. These ciphertext vectors are evaluations on domain H of some polynomial and computing the corresponding ERS codes amounts to evaluating them on some domain L. This is referred to as domain extensions and they are usually implemented using an inverse NTT transform to compute $f(x)$ from $f\big|_H$, followed by an NTT transform to compute $f\big|_L = \vec{f}$. This would result in only $\mathcal{O}(|L|\log|L|)$ operations but require a depth of $2\log|L|$ `pt-ct` multiplications. Instead one could significantly reduce the `pt-ct` depth by computing the extension from $\{f^{(i)} = f(h^{i-1})\}_{i\in[|H|]}$ where $H = \{h^{i-1}\}_{i\in[n]}$ to domain L by using the barycentric form

$$f(\ell_i) = \sum_{j\in[n]} f^{(i)}\lambda_j^H(\ell_i) = Z_H(\ell_i)\sum_{j\in[n]} \frac{f^{(i)}}{Z'_H(h^{j-1})(\ell_i - h^{j-1})} = \frac{\ell_i^n - 1}{n}\sum_{j\in[n]} f^{(i)}\frac{h^{j-1}}{\ell_i - h^{j-1}}$$

where $i \in [|L|]$. However, even when using the method previously described for homomorphic matrix-vector multiplication, the large number of operations would lead to an unrealistic execution time. A naive hybrid algorithm would perform the first layers of the NTT as matrix-vector products and the remaining layers using the traditional butterfly algorithm (possibly in some other base b). This seems like a trade-off between homomorphic depth and execution time but it has one major problem. The butterfly algorithm would require us to move around elements between slots in different ciphertexts which would be very costly. Therefore we homomorphically evaluate the NTT in 2 dimension as described in Section 6.1. Only the dimension orthogonal to the ciphertext packing uses the butterfly algorithm in some base b and thus only causes permutations over complete ciphertexts which can be reverted by simply permuting rows of the matrix in Fig. 5. Notice that we can still trade off noise depth for execution time by adjusting the base b and the number of ciphertexts per row w.

Now that one has all the relevant ERS codes, one can prove statement (1) by proving the rational constraint $\deg(s) \leq |H| - 2$ for

$$s(X) = \frac{f_{Az}(X)f_{Bz}(X) - f_{Cz}(X)}{Z_H(X)}.$$

Notice that the numerator of s will vanish on H if and only if statement (1) holds. Note that the prover does not have to send $\mathtt{ct}[\overrightarrow{s}] = \mathtt{ct}[s\big|_L]$ since any of its elements can be efficiently computed at verification time. This is because the verifier is provided with the required $\mathtt{ct}[\overrightarrow{f}_{Mz}]$ and can compute the vector $Z_H\big|_L$ efficiently.

6.4 Proving statements (2) and (3)

Starting from the encrypted trace $\mathtt{ct}[z] = (x, \mathtt{ct}[w])$, the prover computes the ERS code $\mathtt{ct}[\overrightarrow{f_w}]$ that corresponds to a polynomial f_w of degree $|H| - |I| - 1$ such that

$$\forall a \in H \setminus I : f_w(a) = \frac{\mathtt{ct}[w]_{\mathsf{Ind}(a)} - f_x(a)}{Z_I(a)}$$

where $\mathsf{Ind} : H \setminus I \to [|H \setminus I|]$ indexes $H \setminus I$, the polynomial f_x interpolates the statement x over I and Z_I is the vanishing polynomial over I. This can be computed using the domain extension described above. From $\mathtt{ct}[\overrightarrow{f_w}]$ the prover (and in verification the verifier) can derive the RS code $\mathtt{ct}[\overrightarrow{f_z}]$ such that $f_z\big|_H = f_w\big|_H \circ Z_I\big|_H + f_x\big|_H = (x, w)$. Computing the ciphertext vector that f_w interpolates requires one element-wise `pt-ct` multiplication and addition.

Remark 6.1. *Blind zkSNARKs can also support private server input w_s such that $z = (x, w_c, w_s)$. Recent work [81] has shown that Blind zkSNARKs can be extended to support knowledge soundness for the private server inputs if the server commits to them in plaintext. We note that this can also be achieved by the Fractal RS-hIOP by partitioning $H = H_c \cup H_s \cup I$ and defining f_{w_c}, f_{w_s} such that $f_z = f_{w_c} \cdot Z_{I \cup H_s} + f_{w_s} \cdot Z_{I \cup H_c} + f_x$. The server commits to $\overrightarrow{f_{w_s}}$ in plaintext and opens an additional plaintext value per opening to f_z.*

In order to prove statements (2) and (3), i.e. that $f_{Mz}\big|_H = M \cdot f_z\big|_H$ for $f_z(X) = f_w(X)Z_I(X) + f_x(X)$, the Fractal protocol performs a "holographic lincheck" [37]. We only discuss a subprotocol of the holographic lincheck, named the "polynomial sumcheck", since only this particular part would require homomorphic computations in the blind setting. It is a univariate sumcheck protocol that is used to prove $\sum_{b \in H} f_{\mathsf{sc}}(b) = 0$ for

$$f_{\mathsf{sc}}(X) = \alpha(X)f_z(X) + \sum_{M \in \{A,B,C\}} \beta_M(X)f_{Mz}(X) = 0 \tag{4}$$

where α and β_M are polynomials such that f_{sc} is of degree $2|H|-2$. In the sumcheck protocol the prover sends an RS code of the degree $|H|-2$ polynomial $Xg = f_{\mathsf{sc}} \mod Z_H$, and the verifier checks the rational constraint $\deg(h) \leq |H|-2$ where $h(X) = \frac{f_{\mathsf{sc}}(X) - Xg(X)}{Z_H(X)}$. Notice that there is no need to send ERS codes for $f_{\mathsf{sc}}(X)$ and $Xg(X)$ since the verifier can efficiently derive any of their elements from the ERS codes $\mathtt{ct}[\overrightarrow{f_z}], \mathtt{ct}[\overrightarrow{f_{Az}}], \mathtt{ct}[\overrightarrow{f_{Bz}}], \mathtt{ct}[\overrightarrow{f_{Cz}}]$ and $\mathtt{ct}[\overrightarrow{g}]$.

Let us now discuss the computation of the ERS code $\mathtt{ct}[\overrightarrow{g}]$ starting from the previously derived ciphertexts. Similar to domain extension one could minimize the homomorphic depth by expressing this computation as one matrix-vector product as follows. First, notice that $g(\ell_i) = \sum_{j\in[|H|]} r_j \ell_i^{j-2}$ where r_j are the coefficients of $r = f_{\mathsf{sc}} \mod Z_H$. Now, w.l.o.g., assume that $\deg(f)+1 = k|H| = kn$ where H is the cyclic subgroup of $\mathbb{F}$ of size n. Then, we have that $Z_H(X) = X^n - 1$ and therefore $r_j = \sum_{s=0}^{k-1} \mathsf{Coeff}(f_{\mathsf{sc}})_{sn+j}$. Lastly, we compute these coefficients $\mathsf{Coeff}(f_{\mathsf{sc}})_i = \sum_{j\in[|L|]} \Lambda_{ij} f(\ell_j)$ where Λ_{ij} are the coefficients of the Lagrange polynomials such that $\lambda_j^L(X) = \sum_{i\in[|L|]} \Lambda_{ij} X^{i-1}$. Therefore, $\overrightarrow{g}$ and similarly $\mathtt{ct}[\overrightarrow{g}]$ could be computed as

$$g(\ell_i) = \sum_{t\in[|L|]} f_{\mathsf{sc}}(l_t) \sum_{j\in[|L|]} \ell_i^{j-2} \sum_{s=0}^{k-1} \Lambda_{sn+j,t} \qquad \text{for } i \in [|L|]$$

which would require $|L|^2$ $\mathtt{pt\text{-}ct}$ multiplications and $|L|\log|L|$ additions. As was the case for domain extension in the barycentric from, we will have to lower the number of required operations in exchange for a larger homomorphic depth. We propose the following algorithm.

Computing the ERS code of g

1 : $\{\alpha\big|_L, \beta_A\big|_L, \beta_B\big|_L, \beta_C\big|_L\} = \mathsf{NTT}(\alpha, \beta_A, \beta_B, \beta_C)$

2 : $\mathtt{ct}[\overrightarrow{f_{\mathsf{sc}}}] = \alpha\big|_L \circ \mathtt{ct}[\overrightarrow{f_z}] + \sum_{M\in\{A,B,C\}} \beta_M\big|_L \circ \mathtt{ct}[\overrightarrow{f_{Mz}}]$

3 : $\mathtt{ct}[\mathsf{Coeff}(f_{\mathsf{sc}})] = \mathsf{iNTT}(\mathtt{ct}[\overrightarrow{f_{\mathsf{sc}}}])$

4 : **foreach** $i \in [n]$:

5 : $\quad \mathtt{ct}[\mathsf{Coeff}(Xg)_i] = \sum_{s=0}^{k-1} \mathtt{ct}[\mathsf{Coeff}(f_{\mathsf{sc}})_{sn+i}]$

6 : $\mathtt{ct}[\overrightarrow{Xg}] = \mathsf{NTT}(\mathtt{ct}[\mathsf{Coeff}(Xg)])$

7 : $\mathtt{ct}[\overrightarrow{g}] = \mathtt{ct}[\overrightarrow{Xg}] \circ \left[l_1^{-1}\ l_2^{-1}\ \ldots\ l_{|L|}^{-1}\right]$

Again we minimize the homomorphic depth of the ciphertext space NTTs on lines 3 and 4 as described before. Notice that we can reuse the domain

evaluations of f_z and f_{Mz} since we will always have $|L| \geq \deg(f_{\mathsf{sc}})$. Remark that the computation on line 5 will have to be performed on ciphertexts in the column-major packing order. We note that this computation still only requires homomorphic additions if the width of the packing used in the iNTT of line 3 is divisible by $\deg(f_{\mathsf{sc}})/|H|$.

6.5 Proving rational constraints

We have so far shown how to blindly compute the Fractal RS-hIOP while Section 5 and specifically Theorem 5.7 only apply to hIOPs. The computations involved in the compilation from the Fractal RS-hIOP to the hIOP also require homomorphic operations when this hIOP is computed blindly. This compilation utilizes the FRI IOP [9] for checking the rational constraints. We perform this protocol batched, as first described by the authors of Aurora [11]. In batched FRI, one checks whether the rational constraint $f_{\mathsf{FRI}}(X) = \sum_i (\alpha_i + \beta_i X^{d-d_i}) f_i(X)$ has degree $d = \max_i\{d_i\}$ where α_i, β_i are some random challenges provided by the verifier instead of checking whether each rational constraint f_i is of degree d_i. Computing the ERS code for the batched rational constraint $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}}}]$ requires one `pt-ct` multiplication.

For the Fractal RS-hIOP, the set of polynomials $\{f_i\}$ will be equal to the following set of rational constraints

$$\left\{ \frac{f_{Az} f_{Bz} - f_{Cz}}{Z_H}, g, \frac{f_{\mathsf{sc}} - Xg}{Z_H}, f_z, f_{Az}, f_{Bz}, f_{Cz}, f_{\mathtt{pt}} \right\}$$

where $f_{\mathtt{pt}}$ is a polynomial that interpolates plaintext values and has therefore not been discussed. A linear combination of these polynomials can be rewritten as a linear combination over the set $\{\mathfrak{f}_i\}$ equal to

$$\left\{ f_{Az} f_{Bz}, g, f_{\mathsf{sc}}, Xg, f_z, f_{Az}, f_{Bz}, f_{Cz}, f_{\mathtt{pt}} \right\}.$$

By this we mean that $\mathtt{ct}[\vec{s}]$ can be computed using element-wise homomorphic operations on $\mathtt{ct}[\vec{\mathfrak{f}_i}]$. Every $\mathtt{ct}[\vec{\mathfrak{f}_i}]$ has been previously computed except $\mathtt{ct}\overrightarrow{f_{Az} f_{Bz}}$ which requires depth of one `ct-ct` multiplication.

In the FRI IOP, the prover interacts with the verifier in approximately $\log d$ rounds. Let us assume that the evaluation domain L is a multiplicative coset of some cyclic subgroup such that $L = \{g\omega^i\}_{i \in [2^k]}$ for $k = \log_2(|L|)$ and g a field element. In round $j \in [\log_2 d]$, the prover sends the folded evaluation $\{f_{\mathsf{FRI}/2^j}((g\omega^i)^{2^j})\}_{i \in [2^{k-j}]}$ of the degree $d/2^j$ polynomial $f_{\mathsf{FRI}/2^j}$ where

$$f_{\mathsf{FRI}/2^j}((g\omega^i)^{2^j}) = \frac{1 + \alpha_j (g\omega^{-i})^{2^j}}{2} f_{\mathsf{FRI}/2^{j-1}}((g\omega^i)^{2^{j-1}})$$

$$+ \frac{1 - \alpha_j (g\omega^{-i})^{2^j}}{2} f_{\mathsf{FRI}/2^{j-1}}((g\omega^{2^{k-1}+i})^{2^{j-1}})$$

and α_j is the j-th round verifier challenge. After the last round, the prover sends the remaining $|L|/d$ evaluations to the verifier, who checks that they are colinear. Now in the blind setting, we propose to stop FRI at round $k - p$ (where 2^p elements can be encrypted in a single ciphertext), since this would only amount to sending one ciphertext, the minimal number we can send. It is clear that the computation of $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}/2^j}}]$ from $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}/2^{j-1}}}]$ would require 2 element-wise `pt-ct` multiplications on vectors of size 2^{k-j}. As was also noticed by [5], we can trade off the number of operation for homomorphic depth by composing multiple rounds of FRI. In our case we fully compose all FRI rounds.

7 Proof of Decryption

The proof of decryption (PoD) is a key component to build a publicly-verifiable blind zkSNARK, as explained in Section 5.3. In this section, we construct PoDs from our vectorized description of the LNP22 proof system, which is described in detail in Appendix C.

All RLWE-based HE schemes such as BFV [23, 46] and the Generalized-BFV (GBFV) scheme [50], but also BGV [24] and CKKS [33], fit in a general framework: the secret key $\mathtt{sk} \in \chi_{key}$ is an element of small norm in $\mathcal{R}_{m,q}$ and a ciphertext $(c_0, c_1) \in \mathcal{C} = \mathcal{R}_{m,q}^2$ encrypts a message $m \in \mathcal{P} = \mathcal{R}_m / \mathcal{I}$ for some ideal $\mathcal{I} \subset \mathcal{R}_m$. For invariant schemes such as GBFV, we have that $\mathcal{I} = (t)$ and the decryption equation is given by

$$c_0 + c_1 \cdot \mathtt{sk} = \lfloor \Delta \cdot m \rceil + v_{inh} \in \mathcal{R}_{q,m} \tag{5}$$

where $\Delta = q/t \in \mathcal{K}_m$ is a scaling factor and v_{inh} is called the inherent noise, i.e. the polynomial with the lowest infinity norm such that the above equation holds. Furthermore, the ciphertext will decrypt correctly as long as the modulus $q \gg B_t := \|t\|_\infty^{\text{can}}$ and $\|v_{inh}\|_\infty < \mathcal{B}_q := \frac{q}{2 \cdot EF_m \cdot h_t \cdot \|t\|_\infty} - \frac{1}{2}$, where h_t is the number of non-zero terms in $t(X)$ and the bound is proven in Appendix B.2. For other schemes such as BGV and CKKS, a slight variation of the above equation describes valid decryption; in particular, in all cases, valid decryption is given by a relation over the ring $\mathcal{R}_{q,m}$, which is linear in the secret key `sk` and with the requirement that $\|v_{inh}\|_\infty < \mathcal{B}_q$ for some bound $\mathcal{B}_q$ depending on the parameters of the scheme.

7.1 Relations for the proof of decryption

Let $\mathtt{C_{sk}}$ denote a commitment to a secret key $\mathtt{sk} \in \chi_{key}$. For $1 \leq i \leq r$, let $\mathtt{ct}^{(i)} = (c_0^{(i)}, c_1^{(i)}) \in \mathcal{R}_{m,q}^2$ denote a ciphertext that decrypts to $m^{(i)}$ under the secret key $\mathtt{sk}$. Since valid decryption requires the norm of the inherent noise $v_{inh}^{(i)}$ in each ciphertext $\mathtt{ct}^{(i)}$ to be bounded by $\mathcal{B}_q$, we can derive the relation:

$$\mathbf{R_1} = \left\{ \left(\begin{array}{l} x = (\mathtt{C_{sk}}, \{\mathtt{ct}^{(i)}, m^{(i)}\}_{i\in[r]}) \\ w = (\mathtt{sk}) \end{array} \right) \;\middle|\; \begin{array}{l} \mathcal{O}^{\mathcal{CT}}(\mathtt{C_{sk}}, \mathtt{sk}) = \mathsf{acc} \\ \wedge \forall i \in [r] : \left\| v_{inh}^{(i)} \right\|_\infty < \mathcal{B}_q \text{ where} \\ v_{inh}^{(i)} \coloneqq c_0^{(i)} + c_1^{(i)} \cdot \mathtt{sk} - \lfloor \Delta \cdot m^{(i)} \rceil \end{array} \right\} \quad (6)$$

Any statement-witness pair in $\mathbf{R_1}$ gives r valid plaintext-ciphertext pairs in RLWE-based HE with respect to the secret key committed to in $\mathtt{C_{sk}}$.

In our work, $\mathtt{C_{sk}}$ is instantiated using the ABDLOP commitment scheme. For messages committed under ABDLOP, the LNP22 proof system (described in detail in Appendix C) allows proving various relations over the commitment ring $\mathsf{R}_{q''}$. This includes Approximate Norm bound proofs (ANP) of linear relations in the commitment ring $\mathsf{R}_{q''}$, as detailed in Appendix C.4. While it may seem promising to apply ANP directly to prove the boundness of inherent noises in ciphertexts, the commitment ring $\mathsf{R}_{q''}$ in the LNP22 proof system differs from the HE ciphertext ring $\mathcal{R}_{m,q}$ in two aspects.

- Firstly, the LNP22 commitment ring is defined by a power-of-two cyclotomic polynomial, typically of degree $d = 64, 128$. In the above HE schemes, the ring $\mathcal{R}_m$ is defined modulo the m-th cyclotomic polynomial where m is much larger than 128 and also not necessarily a power of two.
- Secondly, in LNP22, the modulus $q'' = \prod q_i''$ is chosen such that the cyclotomic polynomial $X^d + 1$ has two irreducible factors modulo q_i''. So even in the case where Φ_m would be a power-of-two cyclotomic polynomial, the ciphertext modulus in HE is chosen such that Φ_m fully splits modulo each prime factor of the ciphertext modulus.

To accommodate the first incompatibility, we first represent elements and relations in the ciphertext ring $\mathcal{R}_{m,q}$ as relations on vectors over $\mathbb{Z}_q$ using the coefficient embedding. Thus, the relations for the inherent noises are given by

$$\vec{v}_{inh}^{(i)} = \mathsf{Rot}_{m,q}(c_1^{(i)}) \cdot \vec{\mathtt{sk}} + \vec{c}_0^{(i)} - \overrightarrow{\lfloor \Delta \cdot m^{(i)} \rceil} \in \mathbb{Z}_q^n, \ \forall i \in [r]. \quad (7)$$

In order to prove the boundedness of $\vec{v}_{inh}^{(i)}$, we describe a vectorized version of ANP in Appendix C.5, which is referred to as vec-ANP.

As for the second incompatibility, a natural solution to accommodate different moduli is to include overflows, as in [68, Section 6.3]. Concretely, for a sufficiently

large modulus q'', there exist bounded overflows $\{\vec{\ell}^{(i)},\ \forall i \in [r]\}$ satisfying

$$\vec{v}_{inh}^{(i)} = \mathsf{Rot}_{m,q}(c_1^{(i)}) \cdot \vec{\mathtt{sk}} + \vec{c}_0^{(i)} - \overrightarrow{\lfloor \Delta \cdot m^{(i)} \rceil} + q\,\vec{\ell}^{(i)} \in \mathbb{Z}_{q''}^n,\ \forall i \in [r]. \quad (8)$$

Since inherent noises and overflows are not independent linear combinations of $\vec{\mathtt{sk}}$, proving their bounds would require us to commit to at least one of the two. This not only increases the commitment size, but also requires a higher modulus $q'' > q$ than HE ciphertexts.

To avoid this blow-up, we use a well known technique from HE, namely modulus switching, which allows transforming a valid ciphertext modulo q into a valid ciphertext modulo q'', where q'' is taken to be lower than q in our protocols. Let $\mathtt{ct}[m] = (c_0, c_1) \in \mathcal{R}_{m,q}^2$ denote a ciphertext with ciphertext modulus q and inherent noise v_{inh}. Switching the ciphertext modulus to q'' amounts to computing

$$\mathtt{ct}' = \left(\lfloor \frac{q''}{q} c_0 \rceil, \lfloor \frac{q''}{q} c_1 \rceil \right) \in \mathcal{R}_{m,q''}^2 .$$

In Appendix B.3, we derive the noise bound in $\mathtt{ct}'$ as $\|v'_{inh}\|_\infty \leq \frac{q''}{q}\|v_{inh}\|_\infty + \mathcal{B}_{ms}$, where $\mathcal{B}_{ms}$ is a constant depending on the secret key distribution. As long as we have $\|v'_{inh}\|_\infty \leq \mathcal{B}_{q''}$, the ciphertext $\mathtt{ct}'$ will be valid and thus satisfies the same equation as (7), but with q'' instead of q.

7.2 Relaxed proof of decryption

For modulus switched ciphertexts $\{\mathtt{ct}'^{(i)} \in \mathsf{R}_{m,q''}^2, i \in [r]\}$, the proof of decryption amounts to proving the relation

$$\mathbf{R_2} = \left\{ \left(\begin{array}{l} x = (\mathtt{C_{sk}}, \{\mathtt{ct}'^{(i)}, m^{(i)}\}_{i\in[r]}) \\ w = (\mathtt{sk}) \end{array} \right) \;\middle|\; \begin{array}{l} \mathcal{O}^{\mathcal{CT}}(\mathtt{C_{sk}}, \hat{\mathtt{sk}}) = \mathsf{acc} \\ \wedge \forall i \in [r] : \left\| \vec{v}_{inh}^{(i)} \right\|_\infty < \mathcal{B}_{q''} \text{ where} \\ \vec{v}_{inh}^{(i)} \coloneqq \mathsf{Rot}_{m,q''}(c_1'^{(i)}) \cdot \vec{\mathtt{sk}} + \vec{c}_0'^{(i)} - \overrightarrow{\lfloor \Delta \cdot m^{(i)} \rceil} \end{array} \right\},$$

where $\hat{\mathtt{sk}}$ denotes an embedding of $\mathtt{sk}$ into the message space of the commitment scheme $\mathcal{CT}$. In the instantiation of the ABDLOP scheme, for an element $v \in \mathcal{R}_m$, we define its embedding $\hat{\mathbf{v}} \in \mathsf{R}_{q''}^{\hat{n}}$ as $\hat{n} = \lceil \frac{n}{d} \rceil$ elements in the commitment ring $\mathsf{R}_{q''}$, such that the coefficient vector of $\hat{\mathbf{v}}$ equals $\vec{v}$ modulo q''.

In this section, we describe a protocol $\mathsf{PoD}(\mathtt{C_{sk}}, \{\mathtt{ct}'^{(i)}, m^{(i)}\}_{i\in[r]})$ using vec-ANP, which is complete for ciphertexts whose inherent noise satisfy $\left\| \vec{v}_{inh}^{(i)} \right\|_\infty < B_{\mathsf{PoD}}$, where

$$B_{\mathsf{PoD}} \coloneqq \min \left\{ \frac{\mathcal{B}_{q''}}{\psi^{(L2)}\sqrt{r \cdot n}}, \frac{q''}{41(r \cdot n)^{3/2}\psi^{(L2)}} \right\}$$

and the factor $\psi^{(L2)}$ is defined in Appendix C.3. In other words, our protocol is a *relaxed* proof of decryption with a relaxation factor

$$\Phi_r := \mathcal{B}_{q''}/B_{\mathsf{PoD}} \approx \psi^{(L2)}\sqrt{r \cdot n} \cdot \max\left\{1, \frac{41r \cdot n}{2\delta_m \|t\|_\infty}\right\}.$$

The protocol. To begin with, the prover commits to the secret key $\mathtt{sk}$ using the Ajtai part of the ABDLOP commitment scheme, i.e. $\mathtt{C}_{\mathtt{sk}} = \mathbf{A_1} \cdot \hat{\mathtt{sk}} + \mathbf{A_2} \cdot \mathbf{s_2}$ where $\mathbf{s_2} \in \mathsf{R}_{q''}^{m_2}$ is a small randomness satisfying $\|\mathbf{s_2}\|_\infty \leq \nu$.

To generate a proof of decryption for r ciphertext-plaintext pairs whose inherent noises are bounded by B_{PoD}, the prover applies the vec-ANP protocol with inputs

$$\Pi_{\text{vec-ANP}}\left((\mathbf{s_1} = \hat{\mathtt{sk}}, \mathbf{m} = \emptyset, \mathbf{s_2}), (\mathbf{W}, \mathbf{w}, B_w = B_{\mathsf{PoD}})\right),$$

where

$$\mathbf{W} = \begin{bmatrix} \mathsf{Rot}_{m,q''}(c_1^{(1)}) \\ \vdots \\ \mathsf{Rot}_{m,q''}(c_1^{(r)}) \end{bmatrix} \in \mathbb{Z}_{q''}^{r \cdot n \times n},\ \mathbf{w} = \begin{bmatrix} \vec{c}_0^{(1)} - \overrightarrow{\lfloor \Delta \cdot m^{(1)} \rceil} \\ \vdots \\ \vec{c}_0^{(r)} - \overrightarrow{\lfloor \Delta \cdot m^{(r)} \rceil} \end{bmatrix} \in \mathbb{Z}_{q''}^{r \cdot n}.$$

Denote the vector $\mathbf{W} \cdot \vec{\mathtt{sk}} + \mathbf{w} = \begin{bmatrix} \vec{v}_{inh}^{(1)} \cdots \vec{v}_{inh}^{(r)} \end{bmatrix}^\top$ as $\vec{\mathbf{u}}$; then, the above vec-ANP protocol convinces the verifier that the prover knows $\hat{\mathtt{sk}}$ such that $\mathcal{O}^{\mathcal{CT}}\left(\mathtt{C}_{\mathtt{sk}}, \hat{\mathtt{sk}}\right) = \mathsf{acc}$ and $\|\vec{\mathbf{u}}\|_\infty \leq B_{\mathsf{PoD}} \cdot \psi^\infty \leq \mathcal{B}_{q''}$. This guarantees the validity of each ciphertext-plaintext pair with respect to the secret key committed in $\mathtt{C}_{\mathtt{sk}}$.

Asymptotic Analysis. With ABDLOP parameters ensuring sufficient hardness of MSIS (for binding) and MLWE (for hiding), the protocol $\mathsf{PoD}(\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}'^{(i)}, m^{(i)}\}_{i\in[r]})$ achieves a constant amortized proof size (including commitment size, without applying the Huffman coding optimization [68]) with respect to the number of ciphertext-plaintext pairs r.

The computation cost is dominated by a subprotocol $\Pi_{eval}^{(2)}(\cdot)$, where both the prover and the verifer need to compute the function H_j, as detailed in Section C.5. This results in $\mathcal{O}(rn^2)$ computation costs. In Section 7.3, we describe a protocol that achieves computation cost $\mathcal{O}(n^2 + rn\log n)$.

7.3 Reducing the computation costs

In Figure 6, we describe another batched proof of decryption protocol that has a reduced computation cost compared to the protocol from Section 7.2.

Instead of having the linear relation in the vec-ANP proof grow with the number of ciphertexts r, we prove the decryption of a random linear combination of ciphertexts. The soundness of the protocol is based on the Schwartz-Zippel Lemma. Since the computations on the r ciphertexts are moved to the ring space, the computations are more efficient. In particular, we reduce the cost from $\mathcal{O}(rn^2)$ to $\mathcal{O}(n^2 + rn\log n)$. This comes with the change of relaxation factor from $\Phi_r = \mathcal{O}\left((rn)^{\frac{3}{2}}\right)$ to $\Phi^{\mathrm{SZ}} = \mathcal{O}\left(rn^{\frac{5}{2}}\right)$. Using the Fiat-Shamir transform, this protocol can be compiled into a non-interactive proof in the ROM.

Lemma 7.1. *Let $\mathcal{P} = \mathbb{F}^P$ denote the plaintext space of an HE scheme with P slots. If $r/|\mathbb{F}| = \mathsf{negl}(\lambda)$, then the protocol in Figure 6 is a proof of decryption for r ciphertexts $\{\mathtt{ct}'[m^{(i)}], m^{(i)}\}_{i\in[r]}$ with negligible soundness error and relaxation factor $\Phi^{SZ} := \Phi_1 \cdot N_{\mathtt{ptct}} \cdot 2^{\lceil \log r \rceil}$ where $N_{\mathtt{ptct}} = \mathcal{O}(n)$ is the noise increase bound for 1 $\mathtt{pt\text{-}ct}$ multiplication.*

Proof. We start by discussing soundness. Let us define a function $f : \mathcal{P}^r \to \mathcal{P} : \{m^{(j)}\}_{i\in[r]} \mapsto \sum_{i\in[r]} \alpha_i m^{(i)}$ for some set of challenges $\{\alpha_i\}_{i\in[r]}$ such that each α_i encodes P elements $\{\alpha_{ij}\}_{j\in[P]}$. The proof of decryption that is verified at the end implies that

$$f\left(\{m^{(i)}\}_{i\in[r]}\right) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathcal{E}.\,\mathsf{Eval}\left(f, \{\mathtt{ct}'[m^{(i)}]\}_{i\in[r]}\right)\right)$$

except with negligible probability. Under the assumption that $\mathcal{E}$ is still correct for f on those ciphertexts, this implies that

$$f\left(\{m^{(i)}\}_{i\in[r]}\right) = f\left(\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\{\mathtt{ct}'[m^{(i)}]\}_{i\in[r]}\right)\right)$$

$$\Rightarrow \quad \sum_{i\in[r]} \left(m^{(i)} - \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathtt{ct}'[m^{(i)}]\right)\right)\alpha_i = 0$$

$$\Rightarrow \forall j \in [P] : \sum_{i\in[r]} \left(m_j^{(i)} - \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathtt{ct}'[m^{(i)}]\right)_j\right)\alpha_{ij} = 0$$

where the subscript j denotes the j-th slot of a plaintext encoding. Now if the values α_{ij} were randomly sampled from $\mathbb{F}$, by the Schwartz-Zippel lemma we can conclude that for each $j \in [P]$ it holds that

$$\forall i \in [r] : m_j^{(i)} = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathtt{ct}'[m^{(i)}]\right)_j$$

except with probability $r/|\mathbb{F}|$. The relaxation factor Φ^{SZ} comes from the relaxation factor required for a PoD on one ciphertext multiplied by the noise factors added by homomorphically computing f. This ensures that the correctness assumption above holds. □

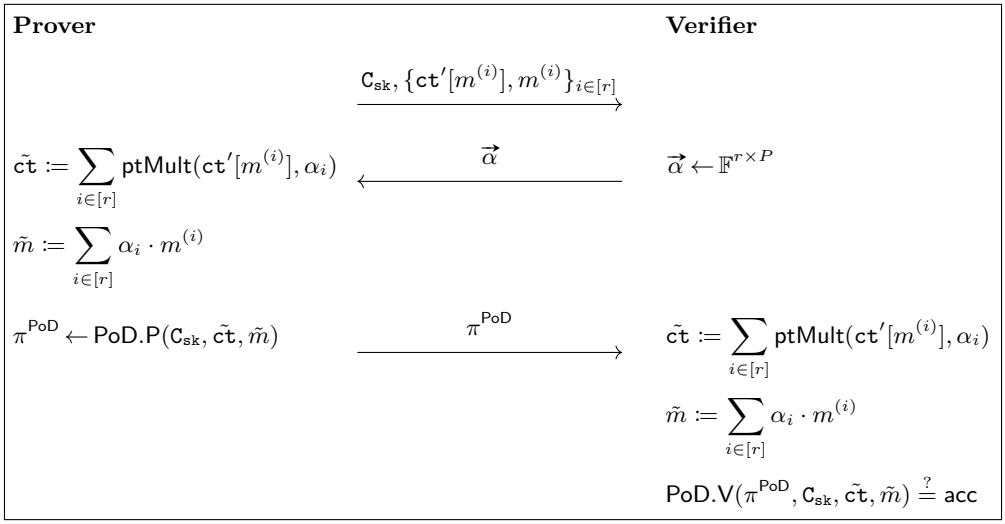


Figure 6: A PoD protocol for $\{\mathtt{ct}'[m^{(i)}], m^{(i)}\}_{i\in[r]}$ with reduced computation costs.

7.4 Proving decryptions of a subset

As discussed in Section 6, efficient instantiations of blind zkSNARKs rely on the SIMD capabilities of the used HE scheme. Therefore, when opening the commitment to a ciphertext and proving its decryption in order to reveal a queried value, we are instead revealing an entire batch of elements in the zkSNARK field. To ensure that the zero-knowledge property of the zkSNARK scheme extends to the blind zkSNARK scheme, we must avoid revealing more queried values than intended, by only revealing the queried values. We give two methods to do so.

Masking. The first method consists of simply masking the ciphertext using a plaintext-ciphertext multiplication, where the masking plaintext M encodes a 1 in the slots we want to reveal and a 0 in all other slots. Instead of giving a PoD $(\pi^{\mathsf{PoD}}, \mathtt{ct}, m)$ that some ciphertext $\mathtt{ct} = (c_0, c_1) \in \mathcal{R}_{q''}^2$ decrypts to a plaintext message $m \in \mathcal{P}$, we can simply replace the ciphertext by $\mathtt{ct}^* = \mathsf{ptMult}(\mathtt{ct}, M)$ and give a PoD $(\pi^{\mathsf{PoD}}, \mathtt{ct}^*, m^*)$ where $m^* = M \cdot m$.

Ring switching. The GBFV parameter sets we use in the blind zkSNARK are particular instances of a family of parameter sets. To illustrate this, consider

the case where we want to encrypt elements in $\mathbb{F}_{p^2}$ with p the Golidlocks prime; then, the family consists of the following: the plaintext space is given by $\mathbb{Z}[x]/(\Phi_m(x), t(x))$ with $m = 7 \cdot 3 \cdot 2^j$ and $t(x) = x^k - b$, with $k = 7 \cdot 2^{i+j-6}$ and $b = 2^{2^i}$ for some integers $0 \leq i \leq 5$ and $6 \leq j \leq 16$. The blind zkSNARK is executed using the set $j = 11$ and $i = 0$, resulting in a plaintext space corresponding to a vector of 96 elements in $\mathbb{F}_{p^2}$. As explained above, we are only interested in a subset of these elements, which opens up the possibility to construct a valid ciphertext for a smaller parameter set in the same family encrypting only the subset of values we want to reveal. For this we can use a technique called ring switching [54] to map a valid ciphertext for $j = 11$ to the smaller ring defined by $j = 8$, which is the smallest dimension that ensures 100-bit security for the modulus q'' in the relaxed PoD. The resulting protocol can be found in Section 8.2. The ring switch decreases the ciphertext size by a factor of 8, which speeds up the PoD by a factor of 64. A similar approach can be taken for the other parameter sets.

8 Implementation

In this section, we demonstrate the practicality of using our protocols in the vCOED setting and the zkDel setting. As discussed before, we select Fractal as the hIOP scheme and GBFV as the HE scheme. In what follows, we will focus on 100-bit security since this is the security level targeted by FRI-based zkSNARK implementations [17].

8.1 GBFV parameter sets

For the Fractal computation, we instantiate GBFV over the ring $\mathcal{R}_m$ for $m = 7 \cdot 3 \cdot 2^{11}$, resulting in the lattice dimension $n := \varphi(m) = 12288$. The ciphertext ring is $\mathcal{R}_{m,q}$ with ciphertext modulus $q \leq 382$ bits, and the plaintext polynomial is $t(X) = X^{7 \cdot 2^7} - 2^4$. The plaintext space is a vector space of dimension 384 over $\mathbb{F}_{p^2}$, where $p = 2^{64} - 2^{32} + 1$ is the Goldilocks prime, widely used in zkSNARK implementations because of its efficient arithmetic.

Ciphertexts committed to in the BCS compilation are in a smaller ring $\mathcal{R}_{m',q'}$ where $m' = 7 \cdot 3 \cdot 2^9$ (dimension $n' := \varphi(m') = 3072$) and a 96 bit q'. For the PoD, we switch down further to an even smaller ring $\mathcal{R}_{m'',q''}$ where $m'' = 7 \cdot 3 \cdot 2^8$ (with dimension $n'' := \varphi(m'') = 1536$) and a 48 bit q''. All these HE parameters provide at least 100-bit security according to the lattice estimator by Albrecht et al. [3].

8.2 Computing Fractal blindly

We prove that the homomorphic circuit outlined in Section 6 is feasible in practice for computing blind proofs of computations with $\mathbf{C} = 2^{20}$ R1CS constraints. This is achieved by selecting parameters for Fractal and GBFV and then demonstrating the following facts:

- they are secure instantiations of an hIOP and HE scheme respectively,
- the HE scheme will remain correct for that homomorphic circuit,
- the number of required homomorphic computations is feasible.

Fractal is calculated in a field of size $\log_2 \mathbb{F} = 128$, thus the Fractal RS-hIOP will remain sound for circuit sizes up to approximately 2^{28} constraints. The maximal degree on which we will have to perform the FRI IOP will be approximately $\mathbf{C} = 2^{20}$. Therefore, from the recent paper by Block et al. [17], we can derive that FRI will remain secure for this field size when choosing rate $\rho = 1/2$ (so $|L| = \mathbf{C}/\rho = 2^{21}$) and performing $\ell = 101$ repetitions of the query phase. In Appendix D, we argue this implies opening to 3728 $\mathbb{F}_{p^2}$ elements on average.

Let us first discuss the practicality of encrypting and sending a circuit trace of size $\mathbf{C}$, as shown in Figure 4. We propose encrypting the trace into normal BFV ciphertexts and then unpacking them into GBFV ciphertexts on the server side. As described in [50], when instantiating BFV using the same cyclotomic polynomial Φ_m, this can be achieved using one automorphism and one `pt-ct` operation per resulting GBFV ciphertext. For the parameter set with $n = 12288$, the packing size for BFV is 6144 and for GBFV is 384. To facilitate the NTT computation in proving Fractal, we only use vectors of power-of-two sizes; hence we only use 4096 slots in BFV and 256 slots in GBFV. Therefore, the client needs to encrypt $\lceil 2^{20}/4096 \rceil = 256$ ciphertexts, which takes approximately 3.3s. The resulting communication size would be approximately 313MB. However, notice that these are actually upper bounds since one would likely not encrypt and send the entire trace but only the private inputs to the computation. Then, the server could compute the other trace values homomorphically, which might require bootstrapping.

The first operation performed by the server will be unpacking into GBFV ciphertexts. Next, the server computes Fractal as described in Section 6. For the inverse NTT required in domain extension, we choose base $b = 8$ and width $w = 1$. For all other NTTs we have chosen base $b = 8$ and width $w = 2$. Regarding the FRI computation, we compose all rounds into one to maximally reduce noise depth, as in [5].

During the FRI procedure, the prover outputs GBFV ciphertexts of dimension $n = 12288$, where $P = 256$ slots are used in each ciphertext. Instead of committing to these ciphertexts in the BCS compilation, we perform rings switching to reduce the lattice dimension, resulting in GBFV ciphertexts of

dimension $n' = 3072$, where $P' = 64$ slots are used in each ciphertext. As such, opening to 3728 $\mathbb{F}_{p^2}$ elements for the FRI query phase corresponds to opening 2514 dimension-3072 ciphertexts instead of opening 2105 dimension-12288 ciphertexts, as explained in Appendix D.

The number of operations and the noise consumption in each step are presented in Table 1. We also provide the script used to compute this table.[3] Note that we only report on the noise required in the "critical path". For example, in the third row, the reported noise is that of the ciphertexts $\mathtt{ct}[\overrightarrow{f_{Mz}}]$. Also, we reduce the required noise depth by combining subsequent `pt-ct` operations. For example, all consecutive `pt-ct` multiplications $\alpha_n(\ldots(\alpha_1 \cdot \mathtt{ct}))$ can be computed using one `pt-ct` multiplication $(\alpha_1 \cdots \alpha_n) \cdot \mathtt{ct}$.

Computation	Noise (bits)	$C_{\mathtt{add}}$	$C_{\mathtt{ptct}}$	$C_{\mathtt{aut}}$	$C_{\mathtt{ctct}}$
Unpacking	9	0	4096	4096	0
Computing $\mathtt{ct}[Mz]$	14	9421459	9433747	2978354	0
Computing $\mathtt{ct}[\overrightarrow{f_z}]/\mathtt{ct}[\overrightarrow{f_{Mz}}]$	64	4636672	4653056	491520	0
Computing $\mathtt{ct}[\overrightarrow{g}]$	138	6762496	6782976	552960	0
Computing $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}}}]$	0	65536	65536	0	8192
Computing FRI	16	98305	106496	0	0
Ringswitching	9	0	196604	589812	0

Table 1: Operation count and noise estimates for computing blind Fractal. See implementation for more details.

Estimated performance for vCOED. In the vCOED setting, we choose $q = 375$ bit as the starting ciphertext modulus, and apply dynamic scaling (i.e. modulus switching to lower ciphertext moduli) when computing blind Fractal. The output ciphertexts of blind Fractal have dimension $n' = 3072$ and a 96 bit modulus q'. To decrease the proof size and remove ringswitching noise, we modswitch to a 60 bit modulus q''. The resulting ciphertexts have 9 bits of noise. Given that the GBFV decryption bound $\mathcal{B}_q \approx 51$ bits, this leaves a $51 - 9 = 42$ bit noise gap, which is sufficient to ensure circuit privacy. Circuit privacy is essential for blind hIOP to maintain zero-knowledge in vCOED. Applying noise flooding [13]

[3] `https://github.com/KULeuven-COSIC/blind_zkSNARKs/blob/main/blindFractal/estimates.m`

(i.e. adding encryptions of zeros with large noise) to ciphertexts with more than 40-bit noise gap gives sufficient security for circuit privacy [22, 42].

These noise-flooded ciphertexts are committed to in BCS compilation. The FRI-query phase requires opening on average 2514 GBFV ciphertexts (see Appendix D) of dimension $n' = 3072$ and 60-bit ciphertext modulus. This results in a 116MB proof size. To estimate the execution time for generating a blind Fractal proof, we use the operation counts in Table 1. Since GBFV is currently not implemented for non-power-of-two cyclotomics, we cannot get exact runtimes for the homomorphic operations of $n = 12288$. Therefore, we use the runtimes for the same operations in a power-of-two lattice dimension of larger size, namely 2^{14}, as presented in Table 3. Based on this, we estimate that computing blind Fractal completely sequentually for a circuit size of 2^{20} takes 9.26 hours. With a 32x speedup from parallelization, the expected runtime reduces to 17.4 min.

Remark 8.1. *For scenarios where the server has no secret inputs, circuit privacy is not necessary. Therefore, before the* BCS *compilation, ciphertexts can be switched down to dimension 768 with 24-bit modulus, utilizing 16 slots per ciphertext. Consequently, opening to 3728 $\mathbb{F}_{p^2}$ elements for the FRI query phase corresponds to revealing 2920 such ciphertexts, yielding a proof size of 13.5MB. Additionally, if the two points required for each linearity check are repacked into the same ciphertext as in [5], then the proof size can be further reduced to 7.2MB at the cost of additional repacking by the prover.*

Estimated performance for zkDel. In the zkDel setting, we choose $q = 382$ bit as the starting ciphertext modulus, and also apply dynamic scaling when computing blind Fractal. As such, the resulting ciphertext has 35.7-bit noise within a 96 bit ciphertext modulus. These ciphertexts are committed to in BCS compilation. Note that in the zkDel setting, circuit privacy is required from the HE scheme. Using the same estimation strategy as in vCOED, the computation of blind Fractal takes 10.9 hours sequentially and 20.5 minutes assuming a 32 parallelization factor.

The FRI-query phase requires opening 2514 GBFV ciphertexts of dimension $n' = 3072$ and 96-bit ciphertext modulus. This results in a 186MB server-to-client communication. Furthermore, the client prover needs to generate a PoD for 2514 masked ciphertexts. Note that since masking is deterministic, we can perform the linear combinations from Figure 6 along with the masking. This results in a ciphertext with 52.9 bits of noise.

To make the PoD protocol less costly, the resulting ciphertext is ringswitched further to dimension $n'' = 1536$ in the PoD. Before this ringswitching, we need to perform an homomorphic trace operation to aggregate messages of the

64 relevant slots in dimension $n' = 3072$ into the 32 remaining slots, and a modulus switching to gaurantee the 100-bit security. Therefore, the protocol from Figure 6 was eventually performed for 32 slots. Before ringswitching to n'', we modswitch to a ciphertext modulus q'' of 48 bits, where q'' is compatible with the requirements of the LNP22 proof system; this results in a noise term of about 5 bits. Finally, we ringswitch down further to $n'' = 1536$, which increases the noise to 14.1 bits. Since the resulting noise is below the threshold $B_{PoD} = 16.9$ bits in Appendix D.1, the final ciphertext-plaintext pair can be proven from our instantiation of the PoD protocol.

Discussion. Due to the lack of GBFV implementation for non power-of-2 cyclotomics, we were unable to provide an implementation for the blind Fractal computation. Instead, we count the number of operations and estimate the runtime using lattice dimension 2^{14} which is 4/3 times larger than our actual dimension 12288. This conservative approach leaves a margin for the fact that HE operations are slightly slower in non-power-of-two cyclotomic ciphertext space. Furthermore, we also provide runtime estimates with 32x speedup from parallelization. We think it is a reasonable assumption, since all operations in our algorithm are highly parallelizable and the server is expected to have powerful computational resources. Furthermore, the peak memory usage during the blind Fractal computation remains smaller than 100GB.

Comparison with HELIOPOLIS. It is not trivial to compare our estimation to [5] since they only implement blind FRI and choose $\log |\mathbb{F}| \approx 256$ bit field size. Thus, we make a new estimation[4] for only computing FRI using the parameters $m = 2^{14}$ and $t(X) = X^{2^8} - 332$ which results in a HE plaintext space of $\mathbb{F}_p^P$ for $P = 256$ and p approximately 268 bits. Using the same method as before, we estimate a sequential runtime of 71s and a parallelized runtime of 2.2s assuming a 32x speedup. In comparison, the implementation of [5] requires 207s runtime on a 32-thread machine. While achieving a 32x speedup on a a 32-thread machine may not be realistic (despite our method being highly parallelizable), our estimated runtime still demonstrates a significant improvement for blind FRI. Furthermore, we also compare to the operation count required for their NTT algorithm for polynomials of degree 2^{18}, as described in Figure 7 in Appendix E. Our blind 2D-NTT only requires 584704 operations in $n = 2^{13}$ while their blind NTT requires at least 10^7 operations in $n = 2^{12}$. Even though their approach can batch blind evaluation of FRI on many polynomials, we note that in practice FRI is always batched using a random linear combination (as described in Section 6.5) to decrease the proof size.

[4] `https://github.com/KULeuven-COSIC/blind_zkSNARKs/blob/main/blindFractal/largefield/estimates_FRIonly.m`

8.3 Proof of Decryption

We implemented the proof-of-decryption protocol presented in Section 7 using the C programming language[5]. We leveraged basic primitives used in Lazer [72], a library for lattice-based zero-knowledge proofs, and thoroughly extended it to construct our proof of decryption for GBFV ciphertexts.

Below, we present execution times for the $\Pi_{\text{vec-ANP}}$ protocol with increasing number of input ciphertexts. We conclude that, in practice, following the technique from Figure 6 is always beneficial performance-wise, since the client needs to apply $\Pi_{\text{vec-ANP}}$ to a single ciphertext at the comparably negligible expense of executing additional HE operations. Lazer parameters for the complete proof of decryption using the protocol from Figure 6 with $r = 2514$ ciphertexts and noise bound $B_{PoD} = 16.9$ bits are presented in Table 5 (Appendix D.1). The proof size for this parameter set is 12KB and can be computed as described in [68, Section 6.1].

r	$\Pi_{\text{vec-ANP}}$ w/out $\Pi^{(2)}_{eval}$	$\Pi^{(2)}_{eval}$		Total runtime	
		single thread	8 threads	single thread	8 threads
1	0.04	1.15	0.45	1.19	0.49
8	0.14	6.92	1.26	7.06	1.40
64	0.93	53.01	8.09	53.94	9.02
512	7.28	424.17	64.30	431.45	71.58
1024	14.68	846.59	126.89	861.27	141.57
2048	29.40	1688.15	253.55	1717.55	282.95
4096	58.81	3407.10	516.12	3465.91	574.93

Table 2: Runtimes in seconds for the PoD instantiated with the parameters in Table 5 and increasing number of pt-ct pairs (r). Using the optimized method given in Figure 6 we can always reduce to the first row in practice.

Experiments. In Table 2, we present the runtimes in seconds for our $\Pi_{\text{vec-ANP}}$ proof of decryption protocol. computed on a machine with an Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz, 8 cores and 377GB RAM. We provide separate numbers for the following two subprotocols:

- The $\Pi_{\text{vec-ANP}}$ protocol up to the execution of $\Pi^{(2)}_{eval}$, including the initial commitment to the FHE secret key and the computation of $\vec{\mathbf{z}} = bR\vec{\mathbf{u}} + \vec{\mathbf{y}}$.

[5] https://github.com/KULeuven-COSIC/blind_zkSNARKs/blob/main/proof-of-decryption/README.md

- The $\Pi_{eval}^{(2)}$ protocol for proving that $\vec{\mathbf{z}}$ was computed correctly, including the computation of the quadratic functions H_j.

Constructing the H_j functions involves a relatively large matrix multiplication (for computing $\vec{R}_j \cdot \mathbf{W}$) which represents around 96% of the total runtime. Therefore, we tested two variations of the PoD: (1) a single-threaded version that would be used by a proof delegator with low-end device, and (2) a multi-threaded matrix multiplication using OpenMP leveraging 8 cores for when a more powerful machine is available. We note that the referred matrix multiplication involves only public information, meaning that it would be possible to delegate it to the server computing the zkSNARK. However, the single thread execution is already significantly faster than locally computing the zkSNARK proof itself.

As we discuss in Section 7.3, executing the PoD using the optimized method presented in Figure 6 results in reduced computational costs for the client. In fact, not using this method and instead directly applying the $\Pi_{\text{vec-ANP}}$ to all the ciphertexts resulting from the Fractal query phase would imply executing the protocol with on average $r = 2514$, as detailed in Appendix D. Conversely, with the protocol in Figure 6, we need to prove correct decryption of a single ciphertext, taking the client less than 2 seconds. Our current implementation consumes 15MB RAM for $r = 1$, but with better memory management this number could be further optimised.

Acknowledgments

We thank Robin Geelen for helping us use the GBFV implementation and thank Vadim Lyubashevsky and Patrick Steuer for helping us use the Lazer library. We also thank Xinxuan Zhang for pointing out a mistake in the first version. This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT - No. 101020788) and by CyberSecurity Research Flanders with reference number VR20192203.

References

[1] Aikata Aikata, Ahmet Can Mert, Sunmin Kwon, Maxim Deryabin, and Sujoy Sinha Roy. REED: Chiplet-based accelerator for fully homomorphic encryption. Cryptology ePrint Archive, Report 2023/1190, 2023.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[3] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015.

[4] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1467–1481. ACM Press, November 2023.

[5] Diego F. Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez. HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part V*, volume 15488 of *LNCS*, pages 302–334. Springer, Singapore, December 2024.

[6] Shahla Atapoor, Karim Baghery, Hilder V. L. Pereira, and Jannik Spiessens. Verifiable FHE via lattice-based SNARKs. *CiC*, 1(1):24, 2024.

[7] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699. Springer, Cham, August 2018.

[8] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385. Springer, Cham, September 2018.

[9] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[10] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[11] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019.

[12] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

[13] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 201–218. Springer, Berlin, Heidelberg, February 2010.

[14] Jonas Bertels, Hilder V. L. Pereira, and Ingrid Verbauwhede. FINAL bootstrap acceleration on FPGA using DSP-free constant-multiplier NTTs. Cryptology ePrint Archive, Paper 2025/137, 2025.

[15] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Berlin, Heidelberg, March 2013.

[16] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michal Zajac. Fiat-shamir security of FRI and related SNARKs. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part II*, volume 14439 of *LNCS*, pages 3–40. Springer, Singapore, December 2023.

[17] Alexander R. Block and Pratyush Ranjan Tiwari. On the concrete security of non-interactive FRI. Cryptology ePrint Archive, Paper 2024/1161, 2024.

[18] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Cham, May 2021.

[19] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 384–417. Springer, Cham, August 2023.

[20] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018.

[21] Jean-Philippe Bossuat, Christian Mouchet, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 587–617. Springer, Cham, October 2021.

[22] Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 371–404. Springer, Singapore, December 2023.

[23] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Berlin, Heidelberg, August 2012.

[24] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[25] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Cryptology ePrint Archive, Report 2011/344, 2011.

[26] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Berlin, Heidelberg, August 2003.

[27] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[28] Sylvain Chatel, Christian Mouchet, Ali Utkan Sahin, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. PELTA - shielding multiparty-FHE against malicious adversaries. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 711–725. ACM Press, November 2023.

[29] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US.

[30] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. Zkml: An optimizing system for ml inference in zero-knowledge proofs. In

Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys '24, page 560–574, New York, NY, USA, 2024. Association for Computing Machinery.

[31] Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia. High-precision arithmetic in homomorphic encryption. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 116–136. Springer, Cham, April 2018.

[32] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the INDCPA-d security of exact FHE schemes. Cryptology ePrint Archive, Paper 2024/127, 2024.

[33] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Cham, December 2017.

[34] Alessandro Chiesa and Giacomo Fenzi. zkSNARKs in the ROM with unconditional UC-security. Cryptology ePrint Archive, Paper 2024/724, 2024.

[35] Alessandro Chiesa, Ryan Lehmkuhl, Pratyush Mishra, and Yinuo Zhang. Eos: Efficient private delegation of zkSNARK provers. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 6453–6469. USENIX Association, August 2023.

[36] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Cham, December 2019.

[37] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Cham, May 2020.

[38] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.

[39] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 670–699. Springer, Cham, December 2021.

[40] Eleanor Chu and Alan George. *Inside the FFT black box: serial and parallel fast Fourier transform algorithms*. CRC press, 1999.

[41] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340. Springer, Cham, February / March 2016.

[42] Morten Dahl, Daniel Demmler, Sarah El Kazdadi, Arthur Meyre, Jean-Baptiste Orfila, Dragos Rotaru, Nigel P. Smart, Samuel Tap, and Michael Walter. Noah's ark: Efficient threshold-fhe using noise flooding. In Michael Brenner, Anamaria Costache, and Kurt Rohloff, editors, *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Copenhagen, Denmark, 26 November 2023*, pages 35–46. ACM, 2023.

[43] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Berlin, Heidelberg, August 2012.

[44] Tim Dokchitser and Alexandr Bulkin. Zero knowledge virtual machine step by step. Cryptology ePrint Archive, Report 2023/1032, 2023.

[45] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146. Springer, Cham, August 2019.

[46] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[47] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.

[48] Sanjam Garg, Aarushi Goel, Abhishek Jain, Guru-Vamsi Policharla, and Sruthi Sekar. zkSaaS: Zero-knowledge SNARKs as a service. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 4427–4444. USENIX Association, August 2023.

[49] Sanjam Garg, Aarushi Goel, and Mingyuan Wang. How to prove statements obliviously? In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 449–487. Springer, Cham, August 2024.

[50] Robin Geelen and Frederik Vercauteren. Fully homomorphic encryption for cyclotomic prime moduli. Cryptology ePrint Archive, Paper 2024/1587, 2024.

[51] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Berlin, Heidelberg, August 2010.

[52] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[53] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, Cham, May / June 2022.

[54] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Field switching in bgv-style homomorphic encryption. *J. Comput. Secur.*, 21(5):663–684, 2013.

[55] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Berlin, Heidelberg, August 2012.

[56] Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter B. Rønne, and Tjerand Silde. Verifiable decryption in the head. In Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo, editors, *ACISP 22*, volume 13494 of *LNCS*, pages 355–374. Springer, Cham, November 2022.

[57] Shai Halevi. Homomorphic encryption. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 219–276. Springer International Publishing, 2017.

[58] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, August 2014.

[59] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in HElib. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 93–120. Springer, Cham, August 2018.

[60] Justin Holmgren. On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Report 2019/1261, 2019.

[61] Intak Hwang, Hyeonbum Lee, Jinyeong Seo, and Yongsoo Song. Practical zero-knowledge PIOP for public key and ciphertext generation in (multi-group) homomorphic encryption. Cryptology ePrint Archive, Paper 2024/1879, 2024.

[62] Intak Hwang, Jinyeong Seo, and Yongsoo Song. Concretely efficient lattice-based polynomial commitment from standard assumptions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 414–448. Springer, Cham, August 2024.

[63] Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 212–234. ACM Press, November 2021.

[64] Christian Knabenhans, Alexander Viand, Antonio Merino-Gallardo, and Anwar Hithnawi. vfhe: Verifiable fully homomorphic encryption. In Flávio Bergamaschi, Anamaria Costache, and Kurt Rohloff, editors, *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Salt Lake City, UT, USA, October 14-18, 2024*, pages 11–22. ACM, 2024.

[65] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

[66] Fucai Luo and Kunpeng Wang. Verifiable decryption for fully homomorphic encryption. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *ISC 2018*, volume 11060 of *LNCS*, pages 347–365. Springer, Cham, September 2018.

[67] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012.

[68] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Cham, August 2022.

[69] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 215–241. Springer, Cham, May 2021.

[70] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*,

volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010.

[71] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Berlin, Heidelberg, May 2013.

[72] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy. CCS '24, 2024.

[73] Shafik Nassar and Ron D. Rothblum. Succinct interactive oracle proofs: Applications and limitations. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 504–532. Springer, Cham, August 2022.

[74] Ngoc Khanh Nguyen. *Lattice-Based Zero-Knowledge Proofs Under a Few Dozen Kilobytes.* PhD thesis, ETH Zurich, Zürich, Switzerland, 2022.

[75] Alex Ozdemir and Dan Boneh. Experimenting with collaborative zk-SNARKs: Zero-knowledge proofs for distributed secrets. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 4291–4308. USENIX Association, August 2022.

[76] Prashant Palvia and Salvatore T. March. Approximating block accesses in database organizations. *Inf. Process. Lett.*, 19(2):75–79, 1984.

[77] Michael Rosenberg, Jacob D. White, Christina Garman, and Ian Miers. zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. In *2023 IEEE Symposium on Security and Privacy*, pages 790–808. IEEE Computer Society Press, May 2023.

[78] Tsz-Wo Sze. Schönhage-strassen algorithm with mapreduce for multiplying terabit integers. In Marc Moreno Maza, editor, *SNC 2011, Proceedings of the 2011 Internation Workshop on Symbolic-Numeric Computation, San Jose, California, USA, June 7-9, 2011*, pages 54–62. ACM, 2011.

[79] Louis Tremblay Thibault and Michael Walter. Towards verifiable FHE in practice: Proving correct execution of TFHE's bootstrapping using plonky2. Cryptology ePrint Archive, Report 2024/451, 2024.

[80] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. DIZK: A distributed zero knowledge proof system. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 675–692. USENIX Association, August 2018.

[81] Xinxuan Zhang, Ruida Wang, Zeyu Liu, Binwu Xiang, Yi Deng, and Xianhui Lu. FHE-SNARK vs. SNARK-FHE: From analysis to practical verifiable computation. Cryptology ePrint Archive, Paper 2025/302, 2025.

A Supplementary preliminaries

A.1 Number fields, rings and coefficient embedding

For any positive integer m, let $\Phi_m(X)$ denote the m-th cyclotomic polynomial of degree $n = \phi(m)$, where $\phi(\cdot)$ is the Euler totient function. Specifically, when m is a power-of-two, $\Phi_m(X) = X^{m/2} + 1$. The m-th cyclotomic number field is $\mathcal{K}_m = \mathbb{Q}[X]/(\Phi_m(X))$ and the m-th cyclotomic ring is $\mathcal{R}_m = \mathbb{Z}[X]/(\Phi_m(X))$. For $g = \sum_{i=0}^{n-1} g_i X^i \in \mathcal{K}_m$, its coefficient vector $[g_0 \ g_1 \ \dots \ g_{n-1}]^\intercal \in \mathbb{Q}^n$ is denoted as $\overrightarrow{g}$, and its coefficient-wise norms $\|g\|_p = \|\overrightarrow{g}\|_p$, e.g.

$$\|g\|_1 = \sum |g_i|, \quad \|g\|_2 = (\sum g_i^2)^{\frac{1}{2}}, \quad \|g\|_\infty = \max\{|g_i|\}.$$

For $c(X), s(X), b(X) \in \mathcal{R}_m$ and $b(X) = c(X) \cdot s(X)$, their coefficient representations satisfy $\overrightarrow{b} = \mathsf{Rot}_m(c) \cdot \overrightarrow{s}$, where

$$\mathsf{Rot}_m(c) \in \mathbb{Z}^{n \times n} = \begin{bmatrix} | & | & & | \\ \overrightarrow{c_{(0)}} & \overrightarrow{c_{(1)}} & \dots & \overrightarrow{c_{(n-1)}} \\ | & | & & | \end{bmatrix}$$

and $c_{(i)} = X^i \cdot c(X) \bmod \Phi_m(X)$. The expansion factor with respect to the infinity norm is defined as

$$\delta_m = \sup\left\{ \frac{\|g \cdot f \bmod \Phi_m\|_\infty}{\|g\|_\infty \cdot \|f\|_\infty} \mid g, f \in \mathbb{Z}[X] \setminus 0 \text{ and } \deg(g), \deg(f) \leq (n-1) \right\}.$$

For elements in $\mathsf{Rot}_m(c)$, let $\left\|\overrightarrow{c_{(i)}}\right\|_\infty \leq EF_m \cdot \|c\|_\infty$, which consecutively gives $\delta_m \leq n \cdot EF_m$. Specifically, when m is a power-of-two, $EF_m = 1$ and $\delta_m = n$.

For the ring $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, we use $[-\frac{q}{2}, \frac{q}{2})$ as the representative interval, and for $x \in \mathbb{Z}$, we denote the centered reduction modulo q by $[x]_q \in \mathbb{Z}_q$. Let $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the flooring and ceiling functions respectively, and let $\lfloor \cdot \rceil$ denote the rounding function that rounds half up. All these notations are extended to elements in $\mathcal{K}_m$ and $\mathcal{R}_m$ coefficient-wise.

For a non-zero element $t(X) \in \mathcal{R}_m$, denote the quotient ring of $\mathcal{R}_m$ modulo $t(X)$ as $\mathcal{R}_{m,t(X)} = \mathcal{R}_m / t\,\mathcal{R}_m$. Specifically, for $q \in \mathbb{Z}$, the quotient ring of $\mathcal{R}_m$ modulo q is denoted as $\mathcal{R}_{m,q}$. Notations for coefficient vectors and norms in $\mathcal{R}_m$ naturally extend to $\mathcal{R}_{m,q}$ using representatives in $[-\frac{q}{2}, \frac{q}{2})$. For $d(X) \in \mathcal{R}_{m,q}$, the rotation matrix $\mathsf{Rot}_{m,q}(d)$ contains columns $\overrightarrow{d_{(i)}}$ where $d_{(i)} = X^i \cdot d(X) \bmod (q, \Phi_m)$, which are bounded as

$$\left\|\overrightarrow{d_{(i)}}\right\|_\infty \leq \min\{EF_m \cdot \|d\|_\infty, \frac{q}{2}\}, \ 0 \leq i \leq n-1.$$

Moreover, for an explicit power-of-two cyclotomic order 2^k, let R denote the ring $\mathcal{R}_{2^k} = \mathbb{Z}[X]/\left(X^d + 1\right)$ where $d = 2^{k-1}$, and $\mathsf{R}_q \coloneqq \mathsf{R}/q\,\mathsf{R}$.

A.2 Probability distributions

Given a probability distribution χ, the notation $a \leftarrow \chi$ implies that a is sampled from χ. Let $\mathcal{U}(\mathbb{Z}_q)$ and $\mathcal{U}(\mathcal{R}_{m,q})$ denote the uniform distribution over $\mathbb{Z}_q$ and over $\mathcal{R}_{m,q}$, respectively. For example, $a \leftarrow \mathcal{U}(\mathcal{R}_{m,3})$ is a uniformly random polynomial in $\mathcal{R}_m$ with ternary coefficients.

Let D_σ denote the discrete Gaussian distribution with standard deviation σ over the integers, then the following properties are satisfied [7, 67]

$$\Pr\left[|z| > k\sigma \mid z \leftarrow D_\sigma\right] \leq 2e^{-k^2/2} \tag{9}$$

$$\Pr\left[\|\mathbf{z}\|_2 > t\sqrt{r} \cdot \sigma \mid \mathbf{z} \leftarrow D_\sigma^r\right] \leq \left(te^{\frac{1-t^2}{2}}\right)^r. \tag{10}$$

The notation naturally extends to the ring $\mathcal{R}_m$, i.e. $D_{\mathcal{R}_m,\sigma}$ denotes the discrete Gaussian distribution with standard deviation σ over $\mathcal{R}_m$.

Let Bin_κ denote the binomial distribution parameterized by κ, i.e. the distribution $\sum_{i=0}^{\kappa}(a_i - b_i)$ where $a_i, b_i \leftarrow \{0,1\}$. For example, for $c \leftarrow \mathsf{Bin}_1$, $\Pr(c=0) = \frac{1}{2}$ and $\Pr(c=1) = \Pr(c=-1) = \frac{1}{4}$.

B Background on the GBFV Scheme

B.1 Canonical embedding

For polynomials in $\mathcal{K}_m$, defining norms on coefficient vectors provides a straightforward measure of sizes. However, analyzing the coefficient norm growth upon multiplication requires the expansion factor δ_m, which depends heavily on the polynomial modulus $\Phi_m(X)$ and often results in loose bounds. This leads to the broad use of canonical norm $\|\cdot\|^{\text{can}}$ [41, 43, 55, 70, 71], which is defined from the canonical embedding into $\mathbb{C}^n$. Recall that the canonical embedding is

$$\tau : \mathcal{K}_m \hookrightarrow \mathbb{C}^n \colon a(X) \mapsto \{a(\xi_m^j)\}_{j \in \mathbb{Z}_m^\times},$$

where $\xi_m = \exp(2\pi i/m)$ is a primitive complex m-th root of unity. The canonical norm is $\|a\|_p^{\text{can}} = \|\tau(a)\|_p$, and common values of p are $1, 2, \infty$.

Lemma B.1 (Adapted from [43]). *For all $a, b \in \mathcal{K}_m$, the following properties are satisfied*

- $\|a\|_\infty^{\text{can}} \leq \|a\|_1$
- $\|a\|_\infty \leq c_m \cdot \|a\|_\infty^{\text{can}}$, *where c_m is a constant determined by the cyclotomic order m*
- $\|a \cdot b\|_\infty^{\text{can}} \leq \|a\|_\infty^{\text{can}} + \|b\|_\infty^{\text{can}}$

- $\|a \cdot b\|_p^{\text{can}} \leq \|a\|_\infty^{\text{can}} \cdot \|b\|_p^{\text{can}}$

Specifically, $c_m = 1$ for power-of-two m, and for $m = p_1^{e_1} \cdots p_k^{e_k}$, if $p_1 \cdots p_k \leq 400$ then $c_m \leq 8.6$ [43].

B.2 The inherent noise bound in the GBFV Scheme

Let $\Delta = q/t(X) \in \mathcal{K}_m$ denote the scaling factor in GBFV. The inherent noise in the GBFV Scheme [50] can be defined in the same way as for BFV as follows.

Definition B.1. Let $(c_0, c_1) \in \mathcal{R}_{m,q}^2$ be a ciphertext in the GBFV scheme that decrypts to $m \in \mathcal{R}_{m,t}$, then its inherent noise $v_{inh} \in \mathcal{R}_m$ is the polynomial with the lowest infinity norm such that

$$c_0 + c_1 \cdot \mathtt{sk} = \lfloor \Delta \cdot m \rceil + v_{inh} + aq \in \mathcal{R}_m \tag{11}$$

for some polynomial $a \in \mathcal{R}_m$.

For correct decryption, we present the following inherent noise bound $\mathcal{B}_q$.

Lemma B.2. *The ciphertext $(c_0, c_1) \in \mathcal{R}_{m,q}^2$ in the GBFV scheme decrypts to message m correctly if its inherent noise v_{inh} satisfies $\|v_{inh}\|_\infty < \mathcal{B}_q \coloneqq \frac{q}{2 \cdot EF_m \cdot h_t \cdot \|t\|_\infty} - \frac{1}{2}$, where h_t is the number of non-zero terms in $t(X)$.*

Proof. The decryption procedure requires computing

$$\begin{aligned}\left\lfloor \frac{t(X)}{q}(c_0 + c_1 \cdot s) \right\rceil \bmod t(X) &= \left\lfloor \frac{t(X)}{q}(\lfloor \Delta \cdot m \rceil + v_{inh} + aq) \right\rceil \bmod t(X) \\ &= \left\lfloor m + \frac{t(X)}{q}(\epsilon + v_{inh}) \right\rceil,\end{aligned}$$

where $\|\epsilon\|_\infty < \frac{1}{2}$, and the decryption is correct as long as

$$\left\| \frac{t(X)}{q}(\epsilon + v_{inh}) \right\|_\infty < \frac{1}{2}. \tag{12}$$

Let h_t is the number of non-zero terms in $t(X)$, then $\|t(X) \cdot (\epsilon + v_{inh})\|_\infty \leq EF_m \cdot h_t \cdot \|t\|_\infty \cdot (\frac{1}{2} + \|v_{inh}\|_\infty)$, relation (12) is guaranteed by

$$\|v_{inh}\|_\infty < \frac{q}{2 \cdot EF_m \cdot h_t \cdot \|t\|_\infty} - \frac{1}{2}.$$

□

B.3 Modulus switching

Let $\mathtt{ct}[m] = (c_0, c_1) \in \mathcal{R}^2_{m,q}$ denote a ciphertext with ciphertext modulus q and inherent noise v_{inh}, i.e. it satisfies $c_0 + c_1 \cdot \mathtt{sk} = \lfloor \Delta \cdot m \rceil + v_{inh} + aq$ for some $a \in \mathcal{R}_m$. Switching ciphertext modulus to q' amounts to computing

$$\mathtt{ct}' = \left(\lfloor \frac{q'}{q} c_0 \rceil, \lfloor \frac{q'}{q} c_1 \rceil \right) \in \mathcal{R}^2_{m,q'}.$$

The derived ciphertext satisfies

$$\begin{aligned}
\lfloor \frac{q'}{q} c_0 \rceil + \lfloor \frac{q'}{q} c_1 \rceil \cdot \mathtt{sk} &= \frac{q'}{q} (c_0 + c_1 \cdot \mathtt{sk}) + (\epsilon_0 + \epsilon_1 \cdot \mathtt{sk}) \\
&= \frac{q'}{q} (\lfloor \Delta \cdot m \rceil + v_{inh} + aq) + (\epsilon_0 + \epsilon_1 \cdot \mathtt{sk}) \\
&= \frac{q'}{q} \left(\frac{q}{t} \cdot m + \epsilon_3 + v_{inh} + aq \right) + (\epsilon_0 + \epsilon_1 \cdot \mathtt{sk}) \\
&= \frac{q'}{q} \left(\frac{q}{t} \cdot m + \epsilon_3 + v_{inh} + aq \right) + (\epsilon_0 + \epsilon_1 \cdot \mathtt{sk}) \\
&= \lfloor \Delta' \cdot m \rceil + \epsilon_4 + \frac{q'}{q} (\epsilon_3 + v_{inh}) + (\epsilon_0 + \epsilon_1 \cdot \mathtt{sk}) + q' \cdot a
\end{aligned}$$

for $\Delta' = \frac{q'}{t}$ and $\|\epsilon_i\|_\infty \leq \frac{1}{2},\ i \in [4]$. Its inherent noise is

$$v'_{inh} = \frac{q'}{q} \cdot v_{inh} + (\epsilon_4 + \frac{q'}{q} \epsilon_3 + \epsilon_0 + \epsilon_1 \cdot \mathtt{sk}), \tag{13}$$

which can be bounded as $\|v'_{inh}\|_\infty \leq \frac{q'}{q} \|v_{inh}\|_\infty + \mathcal{B}_{ms}$ and $\mathcal{B}_{ms} = 1 + \frac{q'}{2q} + \frac{1}{2} \delta_m \cdot \|\mathtt{sk}\|_\infty$. Moreover, for a ternary secret key with hamming weight h, the bound $\mathcal{B}_{ms}$ can be lower into $(1 + \frac{q'}{2q} + \frac{1}{2} EF_m \cdot h)$ in the worst-case, and $(1 + \frac{q'}{2q} + EF_m \cdot 3 \cdot \sqrt{\frac{h}{12}})$ heuristically.

B.4 Performance of GBFV

Here we present the runtimes used for the microbenchmark in Section 8. They were measured on a MacBook Pro (2021) equipped with an Apple M1 Max processor (10 cores: 8 performance and 2 efficiency), 64 GB of RAM and running macOS Sonoma 14.7.1.

$\log q$ (bits)	$T_{\texttt{add}}$ (ms)	$T_{\texttt{ctct}}$ (ms)	$T_{\texttt{aut}}$ (ms)	$T_{\texttt{ptct}}$ (ms)
120	0.02	5.58	0.84	0.06
180	0.04	9.44	1.66	0.1
240	0.04	9.46	1.66	0.1
300	0.1	18.12	3.98	0.24
360	0.12	23.06	5.52	0.28
420	0.14	28.4	7.24	0.34

Table 3: Timings of operations in GBFV for $n = 2^{14}$ and different sizes for ciphertext modulus q.

C The LNP22 Proof System

This section provides an overview of the LNP22 proof system, including the ABDLOP commitment, commit-and-prove protocols of qudratic relations and approximate proofs of bounded norms (ANP). The latter is extended into proving relations in the coefficient encoding in C.5, and parameters for our instantiation are provided in D.1. For future works, it would be interesting to prove relations with other encodings, such as the new CLPX-like encoding in [62].

Remark C.1. *The modulus q in this appendix section corresponds to the LNP-friendly modulus q'' in the main text.*

C.1 Module-SIS, Module-LWE and the ABDLOP commitment scheme

For some integer k let R denote the ring $\mathcal{R}_{2^k} = \mathbb{Z}[X]/\left(X^d + 1\right)$ where $d = 2^{k-1}$, and $\mathsf{R}_q = \mathsf{R}/q\,\mathsf{R}$. The ABDLOP commitment scheme [68] is defined over the ring R_q and relies on the hardness of the Module-SIS (MSIS) problem and the Module-LWE (MLWE) problem over R_q, as defined below [65].

Definition C.1 ($\mathsf{MSIS}_{\kappa,m,q,B}$)**.** Given $A \leftarrow \mathsf{R}_q^{\kappa\times m}$, the $\mathsf{MSIS}_{\kappa,m,q,B}$ problem is to find $z \in \mathsf{R}_q^m$ such that $A \cdot z = 0^\kappa \bmod q$ and $\|z\|_2 \leq B$.

Definition C.2 ($\mathsf{MLWE}_{\kappa,m,q,\chi}$)**.** Given a distribution χ and parameters κ, the $\mathsf{MLWE}_{\kappa,m,q,\chi}$ problem is to distinguish $(A, A\cdot s+e)$ for $A \leftarrow \mathsf{R}_q^{m\times\kappa}$, secret vector $s \leftarrow \chi^\kappa$ and error vector $e \leftarrow \chi^m$, from $(A, b) \leftarrow \mathsf{R}_q^{m\times\kappa} \times \mathsf{R}_q^m$.

The hardness of $\mathsf{MSIS}_{\kappa,m,q,B}$ and $\mathsf{MLWE}_{\kappa,m,q,\chi}$ are estimated using $\mathsf{SIS}_{\kappa\cdot d,q,B}$ and $\mathsf{LWE}_{\kappa\cdot d,q,\chi}$ in the lattice estimator by Albrecht et al. [3].

The ABDLOP commitment scheme [68]. The ring modulus in ABDLOP is $q = \prod_i q_i$ where $q_i = 5 \bmod 8$ is a prime and q_1 is the smallest factor. Let σ_i denote an automorphism in R_q where $\sigma_i(X) = X^i$ for odd i. This notation extends to arbitrary vectors $\mathbf{m} \in \mathsf{R}^k$ element-wise, i.e. $\sigma_i(\mathbf{m}) = (\sigma_i(\mathbf{m}[j]))_{1\le j\le k}$.

In the ABDLOP commitment scheme, the public parameters pp are generated as

$$\mathsf{pp} = (\mathbf{A_1}, \mathbf{A_2}, \mathbf{B}) \leftarrow \mathsf{R}_q^{\omega\times m_1} \times \mathsf{R}_q^{\omega\times m_2} \times \mathsf{R}_q^{u\times m_2}.$$

In order to commit to a small message $\mathbf{s_1} \in \mathsf{R}_q^{m_1}$ where $\|\mathbf{s_1}\| \le \alpha$ and an arbitrarily large message $\mathbf{m} \in \mathsf{R}_q^{u}$, one samples a small randomness $\mathbf{s_2} \leftarrow \chi^{m_2}$ where χ is a distribution over R_q with bounded infinity norm ν and computes

$$\mathsf{ABDLOP.Com}(\mathsf{pp}, (\mathbf{s_1}, \mathbf{m}, \mathbf{s_2})) = \begin{bmatrix}\mathbf{t_A}\\ \mathbf{t_B}\end{bmatrix} = \begin{bmatrix}\mathbf{A_1}\\ \mathbf{0}\end{bmatrix}\cdot\mathbf{s_1} + \begin{bmatrix}\mathbf{A_2}\\ \mathbf{B}\end{bmatrix}\cdot\mathbf{s_2} + \begin{bmatrix}\mathbf{0}\\ \mathbf{m}\end{bmatrix} \mod q.$$

As such, the ABDLOP scheme not only allows the commitment of large messages $\mathbf{m}$ as in the BDLOP commitment [8], but also compresses small messages $\mathbf{s_1}$ as in the Ajtai commitment [2]. The commitment $\mathbf{t_B}$ of $\mathbf{m}$ and $\mathbf{t_A}$ of $\mathbf{s_1}$ are referred as the BDLOP part and the Ajtai part of ABDLOP, respectively.

Moreover, the commitment does not reveal messages if $\left(\begin{bmatrix}\mathbf{A_2}\\ \mathbf{B}\end{bmatrix}, \begin{bmatrix}\mathbf{A_2}\\ \mathbf{B}\end{bmatrix}\cdot\mathbf{s_2}\right)$ is indistinguishable from uniform. In other words, if the $\mathsf{MLWE}_{m_2-(\omega+u),\omega+u,q,\chi}$ problem is hard, then ABDLOP is computationally hiding.

For the proof of opening, with fixed parameters ξ, η and a power-of-two k, the challenge space $\mathcal{C}h$ is defined as

$$\mathcal{C}h = \left\{c \in \mathsf{R}_q : \|c\|_\infty \le \xi,\ \sigma_{-1}(c) = c \text{ and } \sqrt[2k]{\|c^{2k}\|_1} \le \eta\right\},$$

and it should be exponentially large in the security parameter for soundness purposes. Its set of differences is denoted as $\overline{\mathcal{C}h} = \{c - c' : c, c' \in \mathcal{C}h \text{ and } c \ne c'\}$, and elements in $\overline{\mathcal{C}h}$ are invertible if $\xi < \frac{q_1}{2}$. Example parameters for the challenge space taken from [19] are listed in Table 4.

As in other lattice-based commitment schemes [2, 8], the opening algorithm in ABDLOP is *relaxed*. For an ABDLOP commitment $[\mathbf{t_A}\ \mathbf{t_B}]^\intercal$, its relaxed opening with respect to the commitment key ck is a tuple $(\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}, \bar{c}) \in$

d	ξ	η	k	$\lvert\mathcal{C}\rvert$
64	8	140	32	2^{129}
128	2	59	32	2^{147}

Table 4: Example parameters in [19] to instantiate the challenge space $\mathcal{C}$ assuming $q_1 > 16$

$\mathsf{R}_q^{m_1} \times \mathsf{R}_q^{u} \times \mathsf{R}_q^{m_2} \times \overline{\mathcal{C}h}$ that satisfies

$$\mathsf{ABDLOP.Com}\left(\mathsf{ck}, (\mathbf{s_1}, \mathbf{m}, \mathbf{s_2})\right) = \begin{bmatrix} \mathbf{t_A} \\ \mathbf{t_B} \end{bmatrix}$$

$$\left\|\bar{c}\mathbf{s_1}\right\|_2 \leq B_1 \text{ and } \left\|\bar{c}\mathbf{s_2}\right\|_2 \leq B_2,$$

where $B_1 = B_1(\alpha)$ and $B_2 = B_2(\nu)$ are pre-determined constants. Furthermore, as explained in [68, Lemma 3.1] and in [19, Lemma 5.2], if $\mathsf{MSIS}_{\omega, m_1+m_2, 4\eta\sqrt{B_1^2+B_2^2}}$ is hard, then ABDLOP is computationally binding with respect to the relaxed openings.

C.2 Commit-and-prove of elementary relations

Let $\mathcal{G} = \{g : \mathsf{R}_q^{2(m_1+u)} \to \mathsf{R}_q\}$ denote the set of quadratic functions over R_q, i.e. any $g \in \mathcal{G}$ can be explicitly written as

$$g(\mathbf{a}) = \mathbf{a}^\intercal \mathbf{G_2} \mathbf{a} + \mathbf{g_1} \mathbf{a} + g_0, \quad \forall \mathbf{a} \in \mathsf{R}_q^{2(m_1+u)}$$

for some $\mathbf{G_2} \in \mathsf{R}_q^{2(m_1+u)\times 2(m_1+u)}$, $\mathbf{g_1} \in \mathsf{R}_q^{2(m_1+u)}$ and $g_0 \in \mathsf{R}_q$.

Given an ABDLOP commitment $(\mathbf{t_A}, \mathbf{t_B})$ to the message $(\mathbf{s_1}, \mathbf{m})$ with randomness $\mathbf{s_2}$, the commit-and-prove protocol in [68, Figure 8] (together with the optimization in [68, Section 4.4]) allows one to prove the knowledge of the message

$$\mathbf{s} = \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \\ \sigma_{-1}(\mathbf{s_1}) \\ \sigma_{-1}(\mathbf{m}) \end{bmatrix} \in \mathsf{R}_q^{2(m_1+u)}$$

such that evaluations of public functions $g_1, \ldots, g_N$ in $\mathcal{G}$ at $\mathbf{s}$ satisfy

$$g_j(\mathbf{s}) = 0 \in \mathsf{R}_q, \forall j \in [N] \tag{14}$$

and evaluations of public functions $G_1, \ldots, G_M$ in $\mathcal{G}$ at $\mathbf{s}$ satisfy

$$\overrightarrow{G_j(\mathbf{s})}[1] = 0 \bmod q, \forall j \in [M], \tag{15}$$

where $\overrightarrow{G_j(\mathbf{s})}[1]$ denotes the constant term of $G_j(\mathbf{s}) \in \mathsf{R}_q$. For convenience, this protocol is denoted as

$$\Pi^{(2)}_{eval}\left((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), \sigma_{-1}, (g_1, \ldots, g_N), (G_1, \ldots, G_M)\right).$$

In other words, condition (14) allows proving quadratic relations of committed messages over R_q, and the vanishing constant condition in (15) allows proving inner products between coefficient vectors of committed messages over $\mathbb{Z}_q$ using the following map T.

Inner product from the T map Given two vectors $\vec{a} = (a_0, \ldots, a_{kd-1})$, $\vec{b} = (b_0, \ldots, b_{kd-1}) \in \mathbb{Z}_q^{kd}$, define the following map

$$\mathsf{T} : \mathbb{Z}_q^{kd} \times \mathbb{Z}_q^{kd} \longrightarrow \mathsf{R}_q$$

$$(\vec{a}, \vec{b}) \to \sum_{i=0}^{k-1} \sigma_{-1}\left(\sum_{j=0}^{d-1} a_{id+j} X^j\right) \cdot \left(\sum_{j=0}^{d-1} b_{id+j} X^j\right).$$

Then the constant coefficient of $\mathsf{T}(\vec{a}, \vec{b})$ is equal to the inner product of $\vec{a}$ and $\vec{b}$ modulo q.

C.3 Approximate range proofs

The approximate range proofs [53,68] allow one to prove smallness of a message $\vec{w} \in \mathbb{Z}^m$, with respect to the proof system modulus q. Firstly, the prover computes a projection $\vec{v} = R\vec{w}$, where $R \leftarrow \mathsf{Bin}_1^{256 \times m}$ is a random challenge from the verifier. Note that [68, Lemma 2.8] provides a probabilistic bound for $\vec{v}$

$$\Pr_{R \leftarrow \mathsf{Bin}_1^{256 \times m}}\left[\|\vec{v}\|_2^2 > 337\beta^2\right] \leq 2^{-128},$$

where β is an upper bound on $\|\vec{w}\|_2$. Secondly, by using rejection sampling, the prover generates a vector $\vec{z} = \vec{v} + \vec{y}$ whose distribution is independent of $\vec{v}$ and indistinguishable from the masking vector $\vec{y}$. The standard deviation of $\vec{y}$ (hence also $\vec{z}$) is $\mathfrak{s} = \gamma\|\vec{v}\|_2 = \gamma\sqrt{337}\beta$, where γ is a constant defining the rejection sampling repetition rate. The following lemma shows that if $\vec{z}$ is small, then the vector $\vec{w}$ is small with high probability.

Lemma C.1 ([68, Lemma 2.9]). *Given q, m, a fixed bound $b \leq q/41m$ and $\vec{w} \in \mathbb{Z}_q^m$ such that $\|\vec{w}\|_2 \geq b$, then for arbitrary $\vec{y} \in \mathbb{Z}_q^{256}$, the following holds*

$$\Pr_{R \leftarrow \mathsf{Bin}_1^{256 \times m}} \left[\|R\vec{w} + \vec{y} \bmod q\|_2 < \frac{1}{2}\sqrt{26}b \right] < 2^{-128}.$$

Following the tail bound in Equation (10) for $t \geq 1.64$, the verifier check $\|\vec{z}\|_2 \leq t\sqrt{256} \cdot \mathfrak{s}$ will hold with overwelming probability for a $\|\vec{w}\|_2 \leq \beta$. By rewriting this check as

$$\begin{aligned} \|\vec{z}\|_2 \leq t\sqrt{256} \cdot \mathfrak{s} &= t\sqrt{256} \cdot \gamma\sqrt{337}\beta \\ &= \frac{1}{2}\sqrt{26}\left(2\sqrt{\frac{256}{26}}t\gamma\sqrt{337}\right)\beta, \end{aligned}$$

it is clear that the vector $\vec{w}$ is proven to be small with negligible soundness error. More precisely, if the prover knows a small $\vec{w}$ where $\|\vec{w}\|_2 \leq \beta$ and computes $\vec{z}$ as described, then the verifier can extract a vector $\overrightarrow{w^*}$ such that $\left\|\overrightarrow{w^*}\right\|_2 \leq \psi^{(L2)} \cdot \beta$, assuming $\psi^{(L2)} \cdot \beta < \frac{q}{41m}$, where the factor $\psi^{(L2)} = 2\sqrt{\frac{256}{26}}t\gamma\sqrt{337}$ is called the slack.

The procedure above can also be applied to generate approximate infinity norm proofs with slack $\psi^{(\infty)} = \psi^{(L2)}\sqrt{m}$. Specifically, consider a prover that knows $\vec{w} \in \mathbb{Z}_q^m$ satisfying $\|\vec{w}\|_\infty \leq \alpha$, then its L2 norm is bounded by $\sqrt{m}\alpha$. The previous procedure allows the verifier to extract a vector $\overrightarrow{w^*}$ where

$$\left\|\overrightarrow{w^*}\right\|_\infty \leq \left\|\overrightarrow{w^*}\right\|_2 \leq \psi^{(L2)} \cdot \|\vec{w}\|_2 \leq \psi^{(L2)}\sqrt{m}\alpha,$$

resulting in a slack $\psi^{(\infty)} = \psi^{(L2)}\sqrt{m}$.

C.4 Approximate proofs of bounded norms

In the Approximate Norm bound Proofs (ANP), the prover knows the secret message $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ which satisfies

$$\left\| \mathbf{E} \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \end{bmatrix} + \mathbf{e} \right\|_\infty \leq B_e \tag{16}$$

for public elements $\mathbf{E} \in \mathsf{R}_q^{\ell_e \times (m_1+u)}$, $\mathbf{e} \in \mathsf{R}_q^{\ell_e}$ and a public bound B_e. After committing $(\mathbf{s_1}, \mathbf{m})$ into $(\mathbf{t_A}, \mathbf{t_B})$, the commit-and-prove protocol

convinces the verifier that the prover knows $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ such that $\mathcal{O}^{\mathcal{CT}}((\mathbf{t_A}, \mathbf{t_B}), (\mathbf{s_1}, \mathbf{m})) = \mathsf{acc}$ and

$$\left\| \mathbf{E} \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \end{bmatrix} + \mathbf{e} \right\|_\infty \leq \psi^{(\infty)} \cdot B_e, \tag{17}$$

where the infinity norm slack is $\psi^{(\infty)} = \psi^{(L2)}\sqrt{d\ell_e} = 2\sqrt{\frac{256}{26}}t\gamma\sqrt{337}\sqrt{d\ell_e}$. Moreover, approximate proofs are only complete for bounds $B_e \leq \frac{q}{41(d\ell_e)^{3/2}\psi^{(L2)}}$, as explained in Section C.3.

The protocol. Let $B_b \leftarrow \mathsf{R}_q^{1\times m_2}$, $B_y \leftarrow \mathsf{R}_q^{256/d\times m_2}$, $\mathfrak{s} := \gamma\sqrt{337}\sqrt{d\ell_e} \cdot B_e$ and rej_0 denote the optimized bimodal rejection sampling [68]. The protocol

$$\Pi_{\mathrm{ANP}}((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), (\mathbf{E}, \mathbf{e}, B_e))$$

gives an approximate bounded norm proof for $\mathbf{u} \in \mathsf{R}_q^{\ell_e} := \mathbf{E}\left[\begin{array}{cc}\mathbf{s_1} & \mathbf{m}\end{array}\right]^\top + \mathbf{e}$ and is presented in Figure 7.

Specifically, line 1 samples for a sign b used for bimodal rejection sampling and line 2 samples a vector $\mathbf{y}$ used for masking. In line 3-4, elements b and $\mathbf{y}$ are committed to in the BDLOP part, i.e. the secret messages become

$$\mathbf{s}' := (\mathbf{s_1}, (\mathbf{m}, b, \mathbf{y})),$$

and $\dim(\mathbf{s}') = m_1 + u + 1 + 256/d$. The correct computation of $\vec{\mathbf{z}}$ in line 8 and that $b \in \{-1, 1\}$ are proven by calling the $\Pi_{eval}^{(2)}$ in line 12 with appropriate public functions v and V. These public functions are elements of $\mathcal{V} = \{v : \mathsf{R}_q^{2\cdot\dim(\mathbf{s}')} \to \mathsf{R}_q\}$.

To prove $\vec{\mathbf{z}}$ was computed correctly, public quadratic functions $H_j \in \mathcal{V}$ for $j \in [256]$ are constructed such that their evaluations at $(\mathbf{s}', \sigma_{-1}(\mathbf{s}'))$ satisfy

$$H_j(\mathbf{s}', \sigma_{-1}(\mathbf{s}')) := \mathsf{T}\left(b\vec{R}_j, \vec{\mathbf{u}}\right) + \mathsf{T}(\vec{\mathbf{e}_j}, \vec{\mathbf{y}}) - \vec{\mathbf{z}}[j],\ j \in [256], \tag{18}$$

where $\vec{R}_j$ denote the j-th row of R and $\vec{\mathbf{e}_j}$ is the j-th unit vector for $j \in [256]$. The construction of $H_j \in \mathcal{V}$ is detailed later on. Then, $\vec{\mathbf{z}}$ was computed correctly iff the constant term of all equations in (18) are zero modulo q.

Similarly, to prove $b \in \{-1, 1\}$, public quadratic functions $g \in \mathcal{V}$ and $J_k \in \mathcal{V}$ for $k \in [d-1]$ are constructed such that their evaluations at $(\mathbf{s}', \sigma_{-1}(\mathbf{s}'))$ satisfy

$$g(\mathbf{s}', \sigma_{-1}(\mathbf{s}')) = (b-1)(b+1) \tag{19}$$

$$J_k(\mathbf{s}', \sigma_{-1}(\mathbf{s}')) = \mathsf{T}\left(\vec{b}, \overrightarrow{X^k}\right), k \in [d-1]. \tag{20}$$

Then $b \in \{-1, 1\}$ iff Equation (19) gives the zero element in R_q and constant terms of all equations in (20) are zero modulo q.

Therefore, for line 12, we define $\mathsf{v} \coloneqq \{g\}$ and $\mathsf{V} \coloneqq \{(J_k)_{k\in[d-1]}, (H_j)_{j\in[256]}\}$ as inputs for the subprotocol $\Pi^{(2)}_{eval}$.

	Prover		**Verifier**
1 :	$b \leftarrow \{-1, 1\} \subset \mathsf{R}_q$		
2 :	$\mathbf{y} \leftarrow D^{256/d}_{\mathsf{R}_q, \mathfrak{s}}$		
3 :	$t_b = B_b \mathbf{s_2} + b$		
4 :	$\mathbf{t_y} = B_y \mathbf{s_2} + \mathbf{y}$		
5 :		$t_b, \mathbf{t_y} \longrightarrow$	
6 :			$R \leftarrow \mathsf{Bin}_1^{256\times \ell_e}$
7 :		$\longleftarrow R$	
8 :	$\vec{\mathbf{z}} = bR\vec{\mathbf{u}} + \vec{\mathbf{y}}$		
9 :	If $\mathsf{rej}_0(\vec{\mathbf{z}}, bR\vec{\mathbf{u}}, \mathfrak{s})$		
10 :	Then continue, else abort		
11 :		$\vec{\mathbf{z}} \longrightarrow$	
12 :	Run $\Pi = \Pi^{(2)}_{eval}\left((\mathbf{s}', \mathbf{s_2}), \sigma_{-1}, \mathsf{v}, \mathsf{V}\right)$		return acc iff
13 :			• $\|\vec{\mathbf{z}}\|_2 \leq t\sqrt{256}\mathfrak{s}$
14 :			• Π verifies.

Figure 7: The protocol $\Pi_{\mathrm{ANP}}((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), (\mathbf{E}, \mathbf{e}, B_e))$ that provides an approximate norm proof for $\mathbf{u} = \mathbf{E}\begin{bmatrix}\mathbf{s_1}\\ \mathbf{m}\end{bmatrix} + \mathbf{e}$.

The construction of H_j from (18). We follow the construction in [74, Section 6.4.4] to derive quadratic functions $H_j \in \mathcal{V}$ that satisfy

$$H_j\left(\mathbf{s}', \sigma_{-1}(\mathbf{s}')\right) \coloneqq \mathsf{T}\left(b\vec{R}_j, \vec{\mathbf{u}}\right) + \mathsf{T}(\vec{\mathbf{e}_j}, \vec{\mathbf{y}}) - \vec{\mathbf{z}}[j],\ j \in [256]. \tag{21}$$

Let $\mathbf{K}_s \in \mathsf{R}_q^{(m_1+u)\times 2\cdot\dim(\mathbf{s}')}$, $\mathbf{K}_b \in \mathsf{R}_q^{1\times 2\cdot\dim(\mathbf{s}')}$ and $\mathbf{K}_y \in \mathsf{R}_q^{256/d\times 2\cdot\dim(\mathbf{s}')}$ denote projection matrices such that

$$\begin{bmatrix}\mathbf{s_1}\\ \mathbf{m}\end{bmatrix} = \mathbf{K}_s \begin{bmatrix}\mathbf{s}'\\ \sigma_{-1}(\mathbf{s}')\end{bmatrix}$$

$$b = \mathbf{K}_b \begin{bmatrix}\mathbf{s}'\\ \sigma_{-1}(\mathbf{s}')\end{bmatrix}$$

$$\mathbf{y} = \mathbf{K}_y \begin{bmatrix}\mathbf{s}'\\ \sigma_{-1}(\mathbf{s}')\end{bmatrix}.$$

Let $\mathbf{r_j} \in \mathsf{R}_q^{\ell_e}$ denote a vector of polynomials such that $\vec{\mathbf{r_j}}$ equals $\vec{R}_j$, the j-th row of R. Let $\mathbf{e_j} \in \mathsf{R}_q^{256/d}$ denote a vector of polynomials such that $\vec{\mathbf{e_j}}$ equals the j-th unit vector of dimension 256.

Then the quadratic function $H_j \in \mathcal{V}$ can be explicitly written as

$$H_j(\mathbf{a}) = \mathbf{a}^\intercal \mathbf{G_j}\mathbf{a} + \mathbf{g_j}\mathbf{a} + g_j, \quad \forall \mathbf{a} \in \mathsf{R}_q^{2(\dim(\mathbf{s}'))}$$

where

$$\mathbf{G_j} = \mathbf{K}_b^\top \cdot \sigma_{-1}(\mathbf{r_j})^\top \cdot \mathbf{E} \cdot \mathbf{K}_s$$

$$\mathbf{g_j} = \mathbf{e}^\top \cdot \sigma_{-1}(\mathbf{r_j}) \cdot \mathbf{K}_b + \sigma_{-1}(\mathbf{e_j})^\top \mathbf{K}_y$$

$$g_j = -\vec{\mathbf{z}}[j].$$

C.5 Our vectorized description of the approximate norm bound proof in LNP22

In our vectorized version of the approximate norm bound proof (vec-ANP), the prover knows the secret message $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ which satisfies

$$\left\| \mathbf{W} \begin{bmatrix}\vec{\mathbf{s_1}}\\ \vec{\mathbf{m}}\end{bmatrix} + \mathbf{w} \right\|_\infty \leq B_w, \tag{22}$$

for public elements $\mathbf{W} \in \mathbb{Z}_q^{\ell_w \times (m_1+u)d}$, $\mathbf{w} \in \mathbb{Z}_q^{\ell_w}$ and a public bound $B_w \leq \frac{q}{41\ell_w^{3/2}\psi^{(L2)}}$. After committing $(\mathbf{s_1}, \mathbf{m})$ into $(\mathbf{t_A}, \mathbf{t_B})$, the commit-and-prove

protocol convinces the verifier that the prover knows $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ such that $\mathcal{O}^{\mathcal{CT}}((\mathbf{t_A}, \mathbf{t_B}), (\mathbf{s_1}, \mathbf{m})) = \mathsf{acc}$ and

$$\left\| \mathbf{W} \begin{bmatrix} \vec{\mathbf{s_1}} \\ \vec{\mathbf{m}} \end{bmatrix} + \mathbf{w} \right\|_\infty \leq \psi^{(\infty)} \cdot B_w, \tag{23}$$

where the slack is $\psi^{(\infty)} = \psi^{(L2)}\sqrt{\ell_w} = 2\sqrt{\frac{256}{26}}t\gamma\sqrt{337}\sqrt{\ell_w}$.

In contrast, ANP in Appendix C.4, proves the knowledge of secret messages $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ such that

$$\left\| \mathbf{E} \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \end{bmatrix} + \mathbf{e} \right\| \leq B_e, \tag{24}$$

for public bound B_e and public elements $\mathbf{E} \in \mathsf{R}_q^{\ell_e \times (m_1+u)}$, $\mathbf{e} \in \mathsf{R}_q^{\ell_e}$ with slack $\psi^{(\infty)}$. Note that relation (24) is a special case of the relation (22) by taking $\ell_w = \ell_e \cdot d$ and taking $\mathbf{W}$ and $\mathbf{w}$ as concatenations of rotation matrices for elements in $\mathbf{E}$ and $\mathbf{e}$, respectively.

In the vec-ANP, we define $\vec{\mathbf{u}} \coloneqq \mathbf{W}\left[\, \vec{\mathbf{s_1}} \quad \vec{\mathbf{m}} \,\right]^\top + \mathbf{w} \in \mathbb{Z}_q^{\ell_w}$. The approximate norm proof for $\|\vec{\mathbf{u}}\|_\infty \leq B_w$ is denoted as

$$\Pi_{\text{vec-ANP}}\left((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), (\mathbf{W}, \mathbf{w}, B_w)\right),$$

which contains the same steps as ANP in Figure 7, except that the standard deviation in line 2 is $\mathfrak{s} \coloneqq \gamma\sqrt{337}\sqrt{\ell_w} \cdot B_w$, the projection matrix R in line 6 is $R \leftarrow \mathsf{Bin}_1^{256\times\ell_w}$, and the quadratic functions H_j as inputs for the subprotocol are derived differently.

The construction of H_j in vec-ANP. To derive quadratic functions $H_j \in \mathcal{V}$ that satisfy

$$H_j\left(\mathbf{s}', \sigma_{-1}(\mathbf{s}')\right) \coloneqq \mathsf{T}\left(b\vec{R}_j, \vec{\mathbf{u}}\right) + \mathsf{T}(\vec{\mathbf{e}_j}, \vec{\mathbf{y}}) - \vec{\mathbf{z}}[j],\ j \in [256], \tag{25}$$

we define $\mathbf{K}_s$, $\mathbf{K}_b$ and $\mathbf{K}_y$ as in Appendix C.4.

Let $\mathbf{r_j}^{(W)} \in \mathsf{R}_q^{(m_1+u)}$ denote a vector of polynomials such that $\overrightarrow{\mathbf{r_j}^{(W)}}$ equals $\vec{R}_j \cdot \mathbf{W} \in \mathbb{Z}_q^{d(m_1+u)}$, and $r_j^{(w)}$ denote $\vec{R}_j \cdot \mathbf{w} \in \mathbb{Z}_q$.

Then the quadratic function $H_j \in \mathcal{P}$ can be explicitly written as

$$H_j(\mathbf{a}) = \mathbf{a}^\intercal \mathbf{G_j} \mathbf{a} + \mathbf{g_j}\mathbf{a} + g_j, \quad \forall \mathbf{a} \in \mathsf{R}_q^{2(\dim(\mathbf{s}'))}$$

where

$$\mathbf{G_j} = \mathbf{K}_b^{\intercal} \cdot \sigma_{-1}(\mathbf{r_j}^{(W)})^{\intercal} \cdot \mathbf{K}_s$$

$$\mathbf{g_j} = r_j^{(w)} \cdot \mathbf{K}_b + \sigma_{-1}(\mathbf{e_j})^{\intercal} \cdot \mathbf{K}_y$$

$$g_j = -\vec{\mathbf{z}}[j].$$

D Our instantiation of the PoD protocol

Let us first discuss how we estimate the amount of ciphertexts to decrypt. This is based on the following lemma.

Lemma D.1. *For a set B constructed by taking m random values (with repetition) from a set A, it holds that $f_m(n) \coloneqq \mathbb{E}[|B|] = n(1-(1-1/n)^m)$ with $n = |A|$.*

Now notice that for a FRI query phase that is repeated ℓ times over a domain $|L|$ that is packed into vectors of size P, we can compute the expected number of values to open as

$$1 + 2 \cdot 5 \cdot f_\ell\left(\frac{|L|}{2P}\right) + 2 \cdot \sum_{i=2}^{\log_2\left(\frac{|L|}{P}\right)} f_\ell\left(\frac{|L|}{2^i P}\right)$$

since for each of the ℓ queries, we are taking a random evaluation point in half of each evaluation domain and, in Fractal, opening to the first evaluation domain requires opening to 5 polynomials at the same evaluation point. For the parameters discussed in Section 8,

- $P = 64$ results in on average 2514 ciphertexts to open
- $P = 256$ results in on average 2105 ciphertexts to open

For reference, in the non-blind setting, i.e. $P = 1$, one would open to on average 3728 values.

D.1 Parameters for our instantiation of the PoD

parameters	description	value
$\log q''$	# bits of ciphertext and proof system modulus	48
n''	GBFV ring dimension for PoD	1536
r	average number of ciphertext-plaintext pairs	2514
B_{PoD}^{SZ}	noise bound	$2^{16.9}$
d	proof ring dimension	64
ω	height of $\mathbf{A_1}$, $\mathbf{A_2}$ in ABDLOP	11
m_1	length of the Ajtai message $\mathbf{s_1}$	24
u	length of the BDLOP message $\mathbf{m}$	0
λ	2·(# of $g_j \in \mathcal{R}_{d,q'}$ for boosting soundness)	4
m_2	length of the randomness $\mathbf{s_2}$ in ABDLOP	43
γ	rejection sampling constant for Π_{ANP}	5
$\mathfrak{s}_{\text{ANP}}$	standard deviation for Π_{ANP}	614147325
$\mathfrak{s}_1$	standard deviation for $\Pi_{eval}^{(2)}$	1587.2
$\mathfrak{s}_2$	standard deviation for $\Pi_{eval}^{(2)}$	50790.4
ξ	max. coeff. of a challenge in $\mathcal{C}h$	8
D	number of low-order bits cut from $\mathbf{t_A}$	8

Table 5: Parameters for our instantiation of the proof of decryption protocol from Figure 6 with 100-bit security.

Bibliography

[1] Blindenbach, J., Kang, J., Hong, S., Karam, C., Lehner, T., and Gürsoy, G. SQUiD: ultra-secure storage and analysis of genetic data for the advancement of precision medicine. *Genome Biology 25*, 314 (2024).

[2] Cong, K., Geelen, R., Kang, J., and Park, J. Revisiting oblivious top-k selection with applications to secure k-nn classification. In *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part I* (2024), M. Eichlseder and S. Gambs, Eds., vol. 15516 of *Lecture Notes in Computer Science*, Springer, pp. 3–25.

[3] Cong, K., Kang, J., Nicolas, G., and Park, J. Faster private decision tree evaluation for batched input from homomorphic encryption. In *SCN 24, Part II* (Sept. 2024), C. Galdi and D. H. Phan, Eds., vol. 14974 of *LNCS*, Springer, Cham, pp. 3–23.

[4] Gama, M., Beni, E. H., Kang, J., Spiessens, J., and Vercauteren, F. Blind zkSNARKs for private proof delegation and verifiable computation over encrypted data. *IACR Communications in Cryptology 2*, 3 (2025).

[5] Geelen, R., Iliashenko, I., Kang, J., and Vercauteren, F. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In *EUROCRYPT 2023, Part III* (Apr. 2023), C. Hazay and M. Stam, Eds., vol. 14006 of *LNCS*, Springer, Cham, pp. 257–286.

Statement on the use of Generative AI

I did not use generative AI assistance tools during the research/writing process of my thesis, except for mere language assistance.
The text, code, and images in this thesis are my own (unless otherwise specified). Generative AI has only been used in accordance with the KU Leuven guidelines and appropriate references have been added. I have reviewed and edited the content as needed and I take full responsibility for the content of the thesis.

Curriculum vitae

Jiayi Kang obtained a Bachelor's degree in Physics from Xi'an JiaoTong University. Afterwards, she obtained a Master of Physics from the University of Manchester and a Master of Science in Mathematics from KU Leuven.

In September 2021, Jiayi joined the COSIC research group at KU Leuven as a PhD candidate under the supervision of Prof. Frederik Vercauteren, Prof. Nigel Smart and Dr. Ilia Iliashenko. During her PhD, she spent two months as a research intern at Intel Labs in 2022. Jiayi's research revolves around the theoretical and practical aspects of privacy-enhancing technologies, with a focus on fully homomorphic encryption.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COSIC
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
jiayi.kang@esat.kuleuven.be
http://www.esat.kuleuven.be/cosic