Universidade Estadual de Campinas
Instituto de Computação

Pedro Geraldo Morelli Rodrigues Alves

Cryptographic Engineering of Privacy-preserving Algorithms

Engenharia Criptográfica de Algoritmos que Preservam a Privacidade

CAMPINAS
2023

Pedro Geraldo Morelli Rodrigues Alves

Cryptographic Engineering of Privacy-preserving Algorithms

Engenharia Criptográfica de Algoritmos que Preservam a Privacidade

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Orientador: Prof. Dr. Diego de Freitas Aranha
Coorientador: Prof. Dr. Edson Borin

Este exemplar corresponde à versão final da Tese defendida por Pedro Geraldo Morelli Rodrigues Alves e orientada pelo Prof. Dr. Diego de Freitas Aranha.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

AL87c

Alves, Pedro Geraldo Morelli Rodrigues, 1988-
Cryptographic engineering of privacy-preserving algorithms / Pedro Geraldo Morelli Rodrigues Alves. – Campinas, SP : [s.n.], 2023.

Orientador: Diego de Freitas Aranha.
Coorientador: Edson Borin.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Criptografia. 2. FHE (Criptografia completamente homomórfica). 3. Unidade de processamento gráfico. 4. CUDA (Arquitetura de computador). 5. Computação de alto desempenho. 6. Programação paralela (Computação). 7. Proteção de dados. I. Aranha, Diego de Freitas, 1982-. II. Borin, Edson, 1979-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: Engenharia criptográfica de algoritmos que preservam a privacidade
Palavras-chave em inglês:
Cryptography
FHE (Fully Homomorphic Encryption)
Graphics processing unit
CUDA (Computer architecture)
High performance computing
Parallel programming (Computer science)
Data protection
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Diego de Freitas Aranha [Orientador]
Hilder Vitor Lima Pereira
Fábio Borges de Oliveira
Julio César López Hernández
Guido Costa Souza de Araújo
Data de defesa: 03-03-2023
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-7175-8383
- Currículo Lattes do autor: http://lattes.cnpq.br/2963997438105619

Universidade Estadual de Campinas
Instituto de Computação

Pedro Geraldo Morelli Rodrigues Alves

Cryptographic Engineering of Privacy-preserving Algorithms

Engenharia Criptográfica de Algoritmos que Preservam a Privacidade

Banca Examinadora:

- Prof. Dr. Diego de Freitas Aranha
  Instituto de Computação (IC) – UNICAMP
- Dr. Hilder Vitor Lima Pereira
  KU Leuven
- Dr. Fábio Borges de Oliveira
  Laboratório Nacional de Computação Científica (LNCC)
- Prof. Dr. Julio Cesar Lopez Hernandez
  Instituto de Computação (IC) – UNICAMP
- Prof. Dr. Guido Costa Souza de Araújo
  Instituto de Computação (IC) – UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 03 de março de 2023

Privacy is not about having something to hide, it's about the right to control what you want to keep to yourself. (Bruce Schneier)

Agradecimentos – Acknowledgements

Ao Diego, por ter sido o melhor orientador que eu poderia querer e por ter aceitado caminhar comigo durante 8 anos por terrenos inexplorados e perigosos. Sem a sua paciência, sabedoria, e companheirismo, nada disso teria sido possível

Aos meus pais, pelos sacrifícios que fizeram na intenção de investir no meu futuro.

À Joyce, por ter sido uma importante companheira e me apoiado durante a execução desta pesquisa.

À Jheyne, uma preciosa amiga que desde os tempos de mestrado ofereceu os ouvidos para as reclamações, conselhos para as decisões difíceis, e não desistiu de me inspirar com sua disciplina e dedicação incomparáveis.

À Raiza Kirk, minha professora, que ajudou a encurtar a distância cultural e a me fazer sentir mais em casa, mesmo em terras estranhas.

Aos amigos, que às vezes mesmo sem saber me ajudaram a cuidar da saúde mental e ofereceram apoio durante as inúmeras dificuldades que alguém pode encontrar ao decidir fazer doutorado.

Aos colegas do LMCAD, que me ofereceram espaço e recursos para trabalhar, café para passar os dias, e o apoio que apenas colegas de trabalho podem oferecer.

To my fellow colleagues from Aarhus Crypto Group, who received me on my visit to Aarhus in the best warmful way and made me feel welcome in a strange new land. In particular, I would like to thank Malene, who assisted us in solving all the challenges in the best and quickest way.

Aos docentes do Instituto de Matemática, Estatística, e Computação Científica da UNICAMP, que ajudaram a forjar o arcabouço teórico necessário para a execução deste trabalho.

Aos demais que de alguma maneira me ajudaram na conclusão deste trabalho: docentes, funcionários e colegas do Instituto de Computação da UNICAMP.

Por fim, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo financiamento por meio dos processos 144265/2019-2 e 203175/2019-0, e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), com o código de financiamento 001.

Resumo

Dados coletados de usuários são o ouro da era moderna, e a capacidade de coletá-los é tão crucial quanto a de armazená-los e manipulá-los com segurança. Esta é uma tese no formato de coletânea, composta por artigos publicados ou em processo de revisão, que exploram diferentes aspectos da computação que preserva a privacidade, como a implementação eficiente de primitivas, protocolos e aplicações. Nosso trabalho oferece um *framework* para um banco de dados sempre cifrado, que pode armazenar criptogramas e responder a *queries* cifradas sem necessidade de decifração. Na mesma direção, também estudamos o caso de coleta de dados em larga escala de medidores inteligentes. Nesse contexto, uma entidade, como o fornecedor de eletricidade, coleta dados do usuário que podem ser usados em métodos estatísticos como aprendizado de máquina, e realiza a computação multiparte através de uma rede sem revelar as informações do usuário aos nós. Por outro lado, também apresentamos trabalhos que exploram a implementação eficiente da aritmética usada por esquemas modernos de criptografia completamente homomórfica, como BFV e CKKS. Experimentamos diferentes métodos visando a arquitetura CUDA e mostramos como os criptossistemas podem ser acelerados através da escolha adequada da estrutura de dados, localidade e algoritmo usado na multiplicação polinomial. Quatro trabalhos são apresentados tratando desses tópicos, assim como uma discussão que conecta o trabalho.

Abstract

Data is the gold of the modern era, and the capability of collecting it is as crucial as securely storing and handling it. This is a compilation thesis composed of published or under revision papers that explore different aspects of privacy-preserving computing, such as the efficient implementation of primitives, protocols, and applications. Our work offers a framework for an always-encrypted database, which can store ciphertexts and answer encrypted queries without decryption. In the same direction, we also study the case of large-scale data collection from smart meters. In this case, an entity, such as the electricity provider, collects user data that can be used in statistical methods, such as machine learning, and splits the computation through a network without revealing user information. On the other hand, we also present papers that explore the efficient implementation of the arithmetic used by modern fully homomorphic encryption schemes, such as BFV and the CKKS. We experiment with different methods targeting the CUDA architecture and show how the cryptosystems can be accelerated through the proper choice for the data structure, locality, and algorithm used on the polynomial multiplication. Four papers are presented on these topics, as well as a discussion that connects our work during the years of research.

Contents

Chapter 1

Introduction

The need of user data for behavior forecast and advertising outbreaks the valuation of information, turning it into modern-era gold. Thus, as important as being capable of processing and extracting useful information from data is to protect it from breaches. Accomplishing this task implies looking after the business itself and its users.

How to efficiently collect and compute user data without undermining their privacy is an open problem. As remarked by Narayanan and Felten, "data privacy is a hard problem" [24]. System breaches may happen even when data holders choose the most conservative practice and never share data.

The Breach Level Index compiles information from surveys of security professionals and executives of IT teams around the globe. It reports that in 2016 about 75% of data breaches led to financial fraud. Almost 7000 records per minute were leaked only in the first half of 2017, totaling 2 billion. Of these, only 4.6% were encrypted. That is, most emails, vehicle registration details, payment information, medical records, and other leaked information were in plaintext [29, 20]. In 2019, over 40% of the companies stated that they stored sensitive data on Software as a Service (Saas), Infrastructure as a Service (Iaas), or Platform as a Service (Paas) environments, and a large fraction used other third-party solutions that are also out of the direct control of the main company, as social media and mobile payment solutions. Nevertheless, less than 56% used basic security solutions such as file system and database encryption, and about 30% reported intentions to improve that in the following 12 months [33].

In 2020 and 2021, most companies had to adapt their work environment to fit the COVID-19 pandemic constraints. This implied that a considerable fraction of the employees shifted to remote work and the accelerated use of cloud-based infrastructure to accommodate the related needs. However, only 20% of the companies indicated that their security infrastructure was ready to deal with this. Moreover, almost half were not confident in the reliability of their access security system. Conflicting with what was observed in the previous years, only 17% reported that most of their sensitive data in the cloud were encrypted. But still, about half of the respondents say that more than 41% of their workloads and data resides in external clouds [34]. In other words, companies that were not competently handling the security of the data they stored had to accommodate even higher security constraints rapidly. Large amounts of data have been collected every year, leakage has not been sufficiently mitigated, and still, the trend in the industry is to

embrace solutions that increase the attack surface.

Even data anonymization may not be sufficient to protect data secrecy. The literature has shown that information contained in different social networks may be used to recognize the same person in different networks [23], which considerably increases the knowledge acquisition regarding particular individuals, as demographics and interests. Some works claim more than 80% success ratio on de-anonymization [31]. The severity of such an issue fast increased during the current social-network era, when users became used to give up their data freely. Still, it was already feasible even before that with less rich and simpler datasets, as shown by Narayanan and Shmatikov [25].

Brazil recently sanctioned the General Data Protection Law (LGPD) [1], which is closely inspired by the European General Data Protection Regulation (GDPR) [27]. It clearly states that companies are fully responsible for protecting the collected data, which should only be collected with explicit user consent. Moreover, the companies are accountable for the security of the collected, transmitted, stored, or processed data. Thus, even a third party hired to execute computation on collected data may be subject to legal sanctions in the case of a break of data secrecy.

An approach to develop solutions that preserve privacy is to keep data encrypted throughout its lifespan: transportation, storage, and processing. Doing this, a new security fence is set, tying data secrecy to formal guarantees. This can be done by adopting functional encryption schemes, which assert that a particular mathematical property of plaintexts is conserved after encryption, allowing its processing without decryption.

1.1 Fully homomorphic encryption

A well-known example of functional encryption schemes is homomorphic encryption. Originally proposed in 1978, such concept allows the evaluation of ciphertexts through addition or multiplication operations without the possession of decryption keys and without leaking information about the encrypted messages [30].

Schemes that support either addition or multiplication are called *partially homomorphic*, and have been known for decades, such as the Paillier and Elgamal proposals [28, 18]. Only in 2009 a construction that provides support to an unlimited number of both operations was proposed by Gentry [21]. His work depends on first building a *somewhat homomorphic encryption* scheme, which supports a limited amount of consecutive operations. This construction requires the handling of the noise of a ciphertext, an element sampled from a carefully chosen probabilistic distribution. Homomorphic operations increases it and, after a certain point, decryption is not able to recover the message. A bootstrap procedure is used to reduce the noise level in a ciphertext, allowing further computation. Gentry's work motivated several following further works that pursued improvements on security levels and performance, and standardization [21]. Such schemes are classified as *fully homomomorphic*.

BFV, CKKS, and TFHE are modern fully homomorphic encryption (FHE) lattice-based schemes with unique properties. BFV is a variant of Brakerski's proposal that

[1] Translated from Portuguese "*Lei Geral de Proteção de Dados Pessoais*".

relies on the hardness of the Ring-Learning with Errors (RLWE). It's plaintext domain is defined over a ring of polynomials with coefficients in a integer finite field. CKKS enables computation on encrypted complex numbers and can be used for privacy-preserving machine learning applications [12]. It is also based on the RLWE but is considered an approximate FHE scheme, since it is only able to decrypt an approximation of the protected message. TFHE is an FHE scheme that operates on the torus mathematical space, which makes it suitable for Boolean operations [13]. It is based on the problem of Learning with errors (LWE) and it offers the faster bootstrap procedure among the current FHE proposals. TFHE is particularly useful for privacy-preserving machine learning and data analysis applications.

In summary, BFV can perform integer, CKKS enables computations on encrypted complex numbers, and TFHE is well-suited for Boolean operations. Each one has its own unique properties and the choice of the scheme depends on the specific requirements of the application.

1.2 CUDA and FHE

CUDA (Compute Unified Device Architecture) is a parallel computing architect developed by NVIDIA that allows developers to accelerate computationally intensive tasks on NVIDIA GPUs. In particular, tasks with high data parallelism can benefit from CUDA, as polynomial arithmetic and large vector operators, as used in current FHE schemes.

CUDA's processing paradigm considers the existence of a host, which is basically the CPU and the machine's main memory, from the device, so-called GPU. Thus, before each execution on the GPU data must be copied from the main memory to the GPU global memory, and the computation outcome must be copied back. This communication constraint imposes high latency and implies that any computation capable of taking advantage of the GPU computing power needs to rely on high throughput, absorbing the slowdown caused by memory copies.

Many works in the literature investigate implementation techniques of FHE schemes [22, 1, 35, 16], and show performance improvements by accelerating the scheme arithmetic or the scheme operations on GPUs.

1.3 Purpose and scope

This work discusses different perspectives of the development of privacy-preserving techniques. We consider a wide range of cases: from the low-level implementation of schemes, as the design of polynomial arithmetic required by current homomorphic encryption schemes; to the design of a network of devices running complex algorithms over ciphertexts. The scope of the thesis is the development of existing privacy-preserving techniques to improve security or performance.

1.4 Contributions

In this Section, I summarize in a non-chronological order the studied problems and results obtained during the execution of this PhD. These contributions are related to research publications in conferences and journals, and as software made available publicly to the scientific community.

1.4.1 Efficient implementation of RLWE-based HE cryptosystems on GPUs

My master's research investigated methods for the efficient implementation of YASHE [11], a promising LHE cryptosystem, on GPUs [3]. YASHE relies both on the Ring Learning with Errors (RLWE) and the Decisional Small Polynomial Ratio (DSPR) problem as its underlying mathematical problem, which provided a fast homomorphic multiplication when compared to other HE schemes. In that work, we conclude that the Fast Fourier Transform (FFT) and the Chinese Remainder Theorem (CRT) are useful tools on the implementation of polynomial arithmetic with large coefficients on GPUs, but also that the precision issues caused by the floating-point arithmetic of the FFT could increase significantly the ciphertext's noise and affect computation accuracy. After Albrecht et al. work proposed an efficient attack against schemes based on the DSPR, as YASHE, the viability of these schemes as secure constructions for HE was invalidated [2]. Thus, it was natural to port those results to other schemes during my PhD.

A considerable amount of time was applied discussing the challenges of developing an efficient CUDA implementation of HE schemes, such as BFV and CKKS [19, 12]. In recent years, this became an important topic and several libraries were developed and made available, such as SEAL [17], cuHE [15], Lattigo [32], and Concrete [14]. Some common points of discussions consider techniques to accelerate polynomial multiplication, such as the applicability of DFT [2]-variants such as the FFT and the Number-theoretic Transform (NTT).

As results obtained related to this branch, we mention the following works:

- **P. G. M. R. Alves**; J. N. Ortiz; D. F. Aranha. Faster homomorphic encryption over GPGPUs via hierarchical DGT. In: International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2021. p. 520-540.

- **P. G. M. R. Alves**; J. N. Ortiz; D. F. Aranha. Performance of Hierarchical Transforms in Homomorphic Encryption: A case study on Logistic Regression inference. *Submitted to the Journal of Cryptographic Engineering.*

The first paper extends the techniques proposed in my master's to BFV; adjusts the state machine to more efficiently explore the GPU capability, especially regarding memory locality efficiency; and explores new directions, such as the replacement of NTT, widely used in the literature, by the Discrete Gaussian Transform (DGT) as the arithmetic engine to accelerate polynomial multiplication. Some authors argue that it could perform

[2]Acronym of Discrete Fourier Transform.

better on the GPU than the NTT since it implies a higher arithmetic density [9]. That work offered benchmarks comparing our implementation of the BFV using the DGT with the state of the art, and it was clear that those combined techniques effectively offered a performance improvement, but we couldn't isolate the contribution of the transform itself. Moreover, we couldn't say how relevant is the contribution of NTT or DGT to the performance of the implementation, assuming that some benefit can be observed. Hence, in the second paper, we developed two mirror libraries that implement CKKS, a cryptosystem similar to BFV, one using the DGT and the other running the NTT. We ran several experiments, measuring and comparing the latency on both in different scenarios, from the basic direct comparison to higher-level contexts that assist the understanding of the impact these transforms cause on the application. In particular, we study the case of logistic regression inference, an important machine learning algorithm that can be used to classify data. We conclude that considerable differences can be observed in each case. This line of research was completely novel and relevant to solve open questions in the literature.

These papers aim for low-level contributions that reduce the overhead related to modern HE schemes on the execution happening on fog nodes. Combined, they show that CUDA is a vital architecture for the low-latency implementation of these schemes as long as the developer finds ways to reinforce memory locality and use efficiently the fast memories available on the GPUs. Lastly, they present a deep comparison of the most efficient methods available to accelerate polynomial multiplication. Such discussion is scarce in the literature, and often developers simply pick a solution without understanding its aptitude to the target hardware.

1.4.2 Protocols for privacy-preserving computation

A different branch of my work involves the proposal of high-level protocols to accommodate privacy-preserving techniques to real-world applications.

One of our approaches in that branch was the construction and management of encrypted databases. First, we investigate open problems, such as how one could efficiently find records without having the decryption key. We propose a relational algebra to build a framework that addresses this and other relevant issues described in the related works.

The original version of this work was published at the XVI Brazilian Symposium on Information and System Security (SBSeg 2016), when it received an honorable mention award. As consequence, we were invited by the Journal of Internet Services and Applications (JISA) to extend it on a special issue.

In the extended version, we explore the Netflix Grand Prize contest as a use case. It was a famous event in which researchers and developers tried for three consecutive years to propose new methods to learn from previous interactions of users with the Netflix catalog and become more efficient to predict if a particular content would be likable or not by that user. In our work, we pick the winner solution of the contest and implement the primitives needed for its execution on a real database using our relational algebra.

That problem involves the computation of complex functions locally and management of an encrypted database. A different direction is to consider the effect of a privacy-

oriented solution that requires network communication, as an electricity smart-meter grid capable of processing encrypted user data without breaking the data secrecy. In a recent research we explore the case of such a network composed by a user, producing data; a centralized control center that may have limited access to decrypted data but does not have computational power to support the entire network needs; and several fog nodes capable of splitting the computation but not being trustworthy to handle plaintexts. The fog nodes would need to compute parts of a complex function, such as a machine learning algorithm, capable of, for instance, consumption forecasting, without access to decryption keys. FHE is a natural solution but its impact on the network performance might be high considering the ciphertext expansion factor, which might be of thousands, and computational complexity. Thus, our work explores the network overhead of handling large operators as FHE ciphertexts.

- **P. G. M. R. Alves**; D. F. Aranha. A framework for searching encrypted databases. In: Proceedings of the XVI Brazilian Symposium on Information and Computational Systems Security. Niterói: SBC: 2016.
- **P. G. M. R. Alves**; D. F. Aranha. A framework for searching encrypted databases. Journal of Internet Services and Applications, v. 9, n. 1, p. 1-18, 2018.
- S. A. Marandi; **P. G. M. R. Alves**; D. F. Aranha; R. H. Jacobsen. Lattice-based Homomorphic Encryption for Privacy-Preserving Smart Meter Data Analytics. *Submitted to the IEEE Transactions on Dependable and Secure Computing.*

1.4.3 Software development

Several computational libraries and relevant code repositories were produced and made publicly available during this work development. In particular, we mention:

- A proof-of-concept MongoDB wrapper which provides encrypted functionality to the database. Relates to [4, 5]:
  - Alves P. A proof-of-concept searchable encryption backend for MongoDB. 2016. `https://github.com/pdroalves/encrypted-mongodb`.
- A CUDA-based library that implements the basic polynomial arithmetic needed for the implementation of RLWE schemes, as BFV and CKKS, and a implementation of the BFV over it. Relate to [6].
  - Alves P. `https://github.com/pdroalves/cupoly`.
  - Alves P. `https://github.com/pdroalves/spog-bfv`.
  - Alves P. `https://github.com/pdroalves/spog-ckks`.
  - Alves P. `https://github.com/pdroalves/spogfaces`.
  - Alves P. `https://github.com/pdroalves/spog-svm`.

- Two mirror implementations of the CKKS on CUDA, differing only on the underlying solution to accelerate polynomial multiplication, one uses the NTT as commonly found in the literature, while the other uses the DGT. Relate to [7].
  - Alves P. `https://github.com/pdroalves/aoa`.
  - Alves P. `https://github.com/pdroalves/aoa-logistic-regression`

1.4.4 Visiting period at Aarhus University

Between September 2021 and September 2022, I visited the Aarhus University, where I could work closely with the Aarhus Crypto Group. There, I continued the investigation on accelerating GPU implementations of HE schemes, but also pursued new research lines.

Besides the direct benefits to my work, as publications and software development mentioned at Section 1.4, my time in Aarhus also led the development of a research line towards the practical use of RNS on the AVX512 architecture in accelerating modular arithmetic involving large numbers applied on public-key cryptography. The prototypes I wrote [3] were used as starting point for the grant-awarded project "*RENAIS: Residue Number Systems for Cryptography*", which targets discovering new algorithms and implementation techniques to accelerate modular arithmetic of cryptographic interest, with a focus on developing formal verification for correctness and implementation security guarantees [8].

1.5 Organization of the document

This is a compilation thesis, in which the body of the document is a verbatim compilation of published or submitted to publication articles. At Chapter 2 we present the works. They are sorted by publishing date, in the case of those already published, or submission-for-publishing date, otherwise, and Chapter 3 discusses the research line that characterizes the collection of papers presented. Lastly, Chapter 4 concludes this thesis.TFHE is an FHE scheme that operates on the torus mathematical space, which makes it suitable for boolean operations [13]. It is based on the problem of Learning with errors (LWE) and it offers the faster bootstrap procedure among the current FHE proposals. TFHE is particularly useful for privacy-preserving machine learning and data analysis applications.

[3] Source code available at `https://github.com/pdroalves/rnsmr_avx512`.

Chapter 2

Published works

2.1 Searchable encryption

This publication is entitled “A framework for searching encrypted databases” and was published originally at the XVI Brazilian Symposium on Information and Computational Systems Security in 2016, where received the honorable mention award [4]; and latter as an extended version at the Journal of Internet Services and Applications in 2018 [5]. This thesis presents the second and most complete version of the work.

A framework for searching encrypted databases

Pedro G. M. R. Alves Diego F. Aranha

University of Campinas

January 2018

Abstract

Cloud computing is a ubiquitous paradigm responsible for a fundamental change in the way distributed computing is performed. The possibility to outsource the installation, maintenance and scalability of servers, added to competitive prices, makes this platform highly attractive to the computing industry. Despite this, privacy guarantees are still insufficient for data processed in the cloud, since the data owner has no real control over the processing hardware. This work proposes a framework for database encryption that preserves data secrecy on an untrusted environment and retains searching and updating capabilities. It employs order-revealing encryption to perform selection with time complexity in $\Theta(\log n)$, and homomorphic encryption to enable computation over ciphertexts. When compared to the current state of the art, our approach provides higher security and flexibility. A proof-of-concept implementation on top of the MongoDB system is offered and applied in the implementation of some of the main predicates required by the winning solution to Netflix Grand Prize.

Keywords— cryptography, functional encryption, homomorphic encryption, order revealing encryption searchable encryption, databases

1 Introduction

The massive adoption of cloud computing is responsible for a fundamental change in the way distributed computing is performed. The possibility to outsource the installation, maintenance and scalability of servers, added to competitive prices, makes this service highly attractive [13, 60]. From mobile to scientific computing, the industry increasingly embraces cloud services and takes advantage of their potential to improve availability and reduce operational costs [29, 19]. However, the cloud cannot be blindly trusted. Malicious parties may acquire full access to the servers and consequently to data. Among the threats there are external entities exploiting vulnerabilities, intrusive governments requesting information, competitors seeking unfair advantages, and even possibly malicious system administrators. The data owner has no real control over the processing hardware and therefore cannot guarantee the secrecy of data [64]. The risk of confidentiality breaches caused by inadequate and insecure use of cloud computing is real and tangible.

This is the extended version of a paper by the same name that appeared in XVI Brazilian Symposium on Information and Computational Systems Security in November, 2016.

The importance of privacy preservation is frequently underestimated, as well as the damage its failure represents to society, as the unfolding of a privacy breach may be completely unpredictable. A report from Javelin Advisory Services found a distressing correlation between individuals who were victims of data breaches and later victims of financial fraud. About 75% of total fraud losses in 2016 had this characteristic, corresponding to U$ 12 billion [44]. This could be avoided with the use of strong encryption at the user side, never revealing data even to the application or the cloud.

The problem of using standard encryption in an entire database is that it eliminates the capability of selecting records or evaluating arbitrary functions without the cryptographic keys, reducing the cloud to a complex and huge storage service. For this reason, alternatives have been proposed to solve this problem, starting from anonymization and heuristic operational measures which do not provide formal privacy guarantees. Encryption schemes tailored for databases such as searchable encryption are a promising solution with perhaps more clear benefits [47, 46, 1, 58]. Searchable encryption enables the cloud to manipulate encrypted data on behalf of a client without learning information. Hence, it solves both of aforementioned problems, keeping confidentiality in regard to the cloud but retaining some of its interesting features.

1.1 The frustration of data anonymization

In 2006, Netflix shared their interest in improving the recommendation system offered to their users with the academic community. This synergy was directed to an open competition during 3 rounds which offered financial prizes for the best recommendation algorithms. An important feature of Netflix's commercial model is to efficiently and assertively guide subscribers in finding content compatible to their interests. Doing this correctly may reinforce the importance of the product for leisure activities, consolidate Netflix's commercial position, and ensure clients' loyalty [7].

The participants of the contest received a training set with anonymized movie ratings collected from Netflix subscribers between 1999 and 2005. There are approximately half million customers and about 17 thousands movies classified in the set, totalling over 100 million ratings. This dataset is composed by movie titles, the timestamp when the rating was created, the rating itself, and an identification number for relating same-user records. No other information about customers was shared, such as name, address or gender. The objective of the participants was to predict with good accuracy how much someone would enjoy a movie based on their previously observed behavior in the platform.

In the same year, America Online (AOL) took a similar approach and released millions of search queries made by 658,000 of its users with the goal of contributing to the scientific community by enabling statistical work over real data [35]. As Netflix, AOL applied efforts on anonymizing the data before publishing. All the obviously sensitive data, such as usernames and IP addresses, were suppressed, being replaced by unique identification numbers.

The ability to understand user's interests and predict their behavior based on collected data is desirable in several commercial models and consequently a hot topic in the scientific literature [50, 61, 45]. However, the importance of privacy-preserving practices is still underestimated, a challenge to overcome. For instance, despite the anonymization efforts of Netflix, Narayanan and Shmatikov brilliantly demonstrated how to break anonymity of the Netflix's dataset by cross-referencing information with public knowledge bases, as those provided by the Internet Movie Database (IMDB) [37]. Using a similar approach, New York Times' reporters were capable of relating a subset of queries to a particular person by joining apparently innocent queries to non-anonymous real state public databases [4].

1.2 "Unexpected" leaks

These events raised a still unsolved discussion about how to safely collect and use data without undermining user privacy. As remarked by Narayanan and Felten, "data privacy is a hard problem" [36]. Even when data holders choose the most conservative practice and never share data, system breaches may happen.

In 2013, a large-scale surveillance program of the USA government was revealed by Edward Snowden, a former NSA employee. Named PRISM, it was structured as a massive data interception effort to collect information for posterior analysis. Their techniques arguably had support of the US legal system and were frequently applicable even without knowledge of the data-owner companies [25, 62].

Two years later, in 2015, stolen personal data of millions of users of the website Ashley Madison was leaked by malicious parties exploiting security vulnerabilities [57]. As consequence, several reports of extortion and even a suicide, illustrating how increasingly sensitive data breaches are becoming.

In the same year, the Sweden's Transport Agency decided to outsource its IT operations to IBM. To fulfill the contract, the latter chose sites in Eastern Europe to place these operations. This resulted in Swedish confidential data being stored in foreign data centers, in particular Czech Republic, Serbia and Romania. As expected, this decision led to a massive data leak, containing information about all vehicles throughout Sweden, including police and military vehicles. Thus, names, photos and home addresses of millions of Swedish citizens, military personal, people under the witness relocation program, were exposed [41].

In 2016, Yahoo confirmed that a massive data breach, possibly the largest known, affected about 500 million accounts and revealed to the world a dataset full of names, addresses, and telephone numbers [5].

These occurrences take us to the disturbing feeling that, despise all efforts, the risk of data deanonymization increases in worrying ways following how much of it is made available [56, 24]. Hence, a seemingly obvious strategy to avoid such issue is to simply stop any kind of dataset collection.

1.3 Privacy by renouncing knowledge

History has proven that the task of collecting and storing data from third parties should be treated as risky. The chance of compromising user privacy by accident is too big and possibly with extreme consequences. This way, the concept of security by renouncing knowledge has attracted adepts, as the search engine DuckDuckGo that states in a blog post that "the only truly anonymised data is no data", and because of that claims to forego the right to store their users' data [21, 51].

A more financial-realistic approach for dealing with this issue is not to give up completely of knowledge but reduce the entities with access by keeping it encrypted during all its lifespan: transportation, storage, and processing, staying secret to the application and the cloud. Thus, a new security fence is set, tying data secrecy to formal guarantees.

1.4 Our contributions

This work follows the state of the art and proposes directives to the modeling of a searchable encrypted database [11]. We detect the main primitives of a relational algebra necessary to keep the database functional, while adding enhanced privacy-preserving properties. A set of cryptographic tools is used to construct each of these primitives. It is composed by order-revealing encryption to enable data selection, homomorphic encryption for evaluation of arbitrary functions, and a standard symmetric scheme to protect and add flexibility to the handling of general data. In particular, our proposed selection primitive achieves time complexity of $\Theta(\log n)$ on the

dataset size. Moreover, we provide a security analysis and performance evaluation to estimate the impact on execution time and space consumption, and a conceptual implementation that validates the framework. It works on top of MongoDB, a popular document-based database, and is implemented as a wrapper over its Python driver. The source code was made available to the community under a GNU GPLv3 license [2].

When compared to CryptDB [47], our proposal provides stronger security since it is able to keep confidentiality even in the case of a compromise of the database and application servers. Since CryptDB delegates to the application server the capability to derive users' cryptographic keys, it is not able to provide such security guarantees. Furthermore, our work is database-agnostic, it is not limited to SQL but can be applied on different key-value databases.

This work is structured as follows. Section 2 describes the cryptographic building blocks required for building our proposed solution. Sections 3 and 4 define searchable encryption, discuss related threats, and present existing implementations. Section 5 proposes our framework, while Section 6 discusses its suitability in a recommendation system for Netflix. Our experimental validation results are presented in Section 7 and Section 8 concludes the paper.

2 Building blocks

The two main classes of cryptosystems are known as symmetric and asymmetric (or public-key) and defined by how users exchange cryptographic keys. Symmetric schemes use the same secret key for encryption and decryption, or equivalently can efficiently compute one from the other, while asymmetric schemes generate a pair of keys composed by public and private keys. The former is distributed openly and is the sole information needed to encrypt a message to the key owner, while the latter should be kept secret and used for decryption.

Besides this, cryptosystems that produce always the same ciphertext for the same message-key input pair are known as deterministic. The opposite, when randomness is used during encryption, are known as probabilistic. We next recall basic security notions and special properties that make a cryptosystem suitable to a certain application. Later, we shall make use of these concepts to analyze the security of our proposal.

2.1 Security notions

Ciphertext indistinguishability is a useful property to analyze the security of a cryptosystem. Two scenarios are considered, when an adversary has and does not have access to an oracle that provides decryption capabilities. Usually these are evaluated through a game in which an adversary tries to acquire information from ciphertexts generated by a challenger [6].

Indistinguishability under chosen plaintext attack – IND-CPA. In the IND-CPA game the challenger generates a pair (PK,SK) of cryptographic keys, makes PK public and keeps SK secret. An adversary has as objective to recognize a ciphertext created from a randomly chosen message from a known two-element message set. A polynomially bounded number of operations is allowed, including encryption (but not decryption), over PK and the ciphertexts. A cryptosystem is indistinguishable under chosen plaintext attack if no adversary is able to achieve the objective with non-negligible probability.

Indistinguishability under chosen ciphertext attack and adaptive chosen ciphertext attack – IND-CCA1 and IND-CCA2. This type of indistinguishability differs from IND-CPA due to the adversary having access to a decryption oracle. In this game the challenge is again to recognize a ciphertext as described before, but now the adversary is able to use decryption results. This new game has two versions, non-adaptive and adaptive. In the non-adaptive version, IND-CCA1, the adversary may use the decryption oracle until it

receives the challenge ciphertext. On the other hand, in the adaptive version he is allowed to use the decryption oracle even after that event. For obvious reasons, the adversary cannot send the challenge ciphertext to the decryption oracle. A cryptosystem is indistinguishable under chosen ciphertext attack/adaptive chosen ciphertext attack if no adversary is able to achieve the objective with non-negligible probability.

Indistinguishability under chosen keyword attack and adaptative chosen keyword attack – IND-CKA and IND-CKA2. This security notion is specific to the context of keyword-based searchable encryption [17]. It considers a scenario in which a challenger builds an index with keyword sets from some documents. This index enables someone to use a value $\mathcal{T}_w$, called trapdoor, to verify if a document contains the word w. This game imposes that no information should be leaked from the remotely stored files or index beyond the outcome and the search pattern of the queries. The adversary has access to an oracle that provides the related trapdoor for any word. His objective is to use this oracle as training to apply the acquired knowledge and break the secrecy of unknown encrypted keywords. As well as in the IND-CCA1/IND-CCA2 game, the non-adaptative version, IND-CKA, of this game forbids the adversary to use the trapdoor oracle once the challenge trapdoor is sent by the challenger. On the other hand, the adaptative version allows the use of the trapdoor oracle even after this event.

A cryptosystem is indistinguishable under chosen keyword attack if every adversary has only a negligible advantage over random guessing.

Indistinguishability under an ordered chosen plaintext attack – IND-OCPA. Introduced by Boldyreva *et al.*, this notion supposes that an adversary is capable of retrieving two sequences of ciphertexts resulting of the encryption of any two sequences of messages [8]. Furthermore, he knows that both sequences have identical ordering. The objective of this adversary is to distinguish between these ciphertexts. A cryptosystem is indistinguishable under an ordered chosen plaintext attack if no adversary is able to achieve the objective with non-negligible probability.

2.2 Functional encryption

Cryptographic schemes deemed "functional" receive such name because they support one or more operations over the produced ciphertexts, hence becoming useful not only for secure storage.

Order-revealing encryption (ORE) Order-revealing encryption schemes are characterized by having, in addition to the usual set of cryptographic functions like *keygen* and *encrypt*, a function capable of comparing ciphertexts and returning the order of the original plaintexts, as shown by Definition 1.

Definition 1 (ORE)**.** *Let E be an encryption function, C be a comparison function, and m_1 and m_2 be plaintexts from the message space. The pair (E, C) is defined as an encryption scheme with the order-revealing property if:*

$$C(E(m_1), E(m_2)) = \begin{cases} \textit{LOWER}, & \textit{if } m_1 < m_2, \\ \textit{EQUAL}, & \textit{if } m_1 = m_2, \\ \textit{GREATER}, & \textit{otherwise.} \end{cases}$$

This is a generalization of order-preserving encryption (OPE), that fixes C to a simple numerical comparison [10].

Security As argued by Lewi and Wu, the "best-possible" notion of security for ORE is IND-OCPA, which means that it is possible to achieve indistinguishability of ciphertexts and with a much stronger security guarantee than OPE schemes can have [32]. Furthermore, differently from OPE, ORE is not inherently deterministic [31]. For example, Chenette *et al.* propose an ORE scheme that applies a pseudo-random function over an OPE scheme, while Lewi and Wu propose an ORE scheme completely built upon symmetric primitives, capable of limiting the use of the comparison function and reducing the leakage inherent to this routine [14, 32]. Their solution works by defining ciphertexts composed by pairs (ct_L, ct_R). To compare ciphertexts ct_A and ct_B, it requires ct_{A_L} and ct_{B_R}. This way, the data owner is capable of storing only one side of those pairs in a remote database being certain that no one will be able to make comparisons between those elements. Nevertheless, any scheme that reveals numerical order of plaintexts through ciphertexts is vulnerable to inference attacks and frequency analysis, as those described by Naveed *et al.* over relational databases encrypted using deterministic and OPE schemes [38]. Although ORE does not completely discard the possibility of such attacks, it offers stronger defenses.

Homomorphic encryption (HE) Homomorphic encryption schemes have the property of conserving some plaintext structure during the encryption process, allowing the evaluation of certain functions over ciphertexts and obtaining, after decryption, a result equivalent to the same computation applied over plaintexts. Definition 2 presents this property in a more formal way.

Definition 2 (HE)**.** *Let E and D be a pair of encryption and decryption functions, and m_1 and m_2 be plaintexts. The pair (E, D) forms an encryption scheme with the homomorphic property for some operator $\diamond$ if and only if the following holds:*

$$E(m_1) \circ E(m_2) \equiv E(m_1 \diamond m_2).$$

The operation $\circ$ in the ciphertext domain is equivalent to $\diamond$ in the plaintext domain.

Homomorphic cryptosystems are classified according to the supported operations and their limitations. *Partially homomorphic encryption* schemes (PHE) hold on Definition 2 for either addition or multiplication operations, while *fully homomorphic encryption* schemes (FHE) support both addition and multiplication operations.

PHE cryptosystems have been known for decades [42, 22]. However, the most common data processing applications, as those arising from statistics, machine learning or genomics processing, frequently require support for both addition and multiplication operations simultaneously. This way, such schemes are not suitable for general computation.

Nowadays, FHE performance is prohibitive, so weaker variants, such as SHE [1] and LHE [2], have the stage for solving computational problems of moderate complexity [23, 12].

Security In terms of security, homomorphic encryption schemes achieve at most IND-CCA1, which means that the scheme is not secure against an attacker with arbitrary access to a decryption oracle [6]. This is a natural consequence of the design requirements, since these cryptosystems allow any entity to manipulate ciphertexts. Most of current proposals, however, reach at most IND-CPA and stay secure against attackers without access to a decryption oracle [34].

[1] SHE stands for "Somewhat homomorphic encryption".

[2] LHE stands for "Leveled fully homomorphic encryption".

3 Searchable encryption

We now formally define the problem of searching over encrypted data. We present three state-of-the-art implementations of solutions to this problem, namely the CryptDB, Arx, and Seabed database systems.

3.1 The problem

Suppose a scenario where Alice keeps a set of documents in untrusted storage maintained by an also untrusted entity Bob. She would like to keep this data encrypted because, as defined, Bob cannot be trusted. Alice also would like to occasionally retrieve a subset of documents accordingly to a predicate without revealing any sensitive information to Bob. Thus, sharing the decryption key is not an option. The problem lies in the fact that communication between Alice and Bob may (and probably will) be constrained. Hence, a naive solution consisting of Bob sending all documents to Alice and letting her decrypt and select whatever she wants may not be feasible. Alice must then implement some mechanism to protect her encrypted data so that Bob will be able to identify the desired documents without knowing their contents or the selection criteria [54].

An approach that Alice can take is to create an encrypted index as in Definition 3.

Definition 3 (Encrypted indexing)**.** *Suppose a dataset $\mathcal{DB} = (m_1, \ldots, m_n)$ and a list $\mathcal{W} = (W_1, \ldots, W_n)$ of sets of keywords such that W_i contains keywords for m_i. The following routines are required to build and search on an encrypted index:*

BuildIndex$_K(\mathcal{DB}, \mathcal{W})$: *The list $\mathcal{W}$ is encrypted using a searchable scheme under a key K and results in a searchable encrypted index $\mathcal{I}$. This process may not be reversible (e.g., if a hash function is used). The routine outputs $\mathcal{I}$.*

Trapdoor$_K(\mathcal{F})$: *This function receives a predicate $\mathcal{F}$ and outputs a trapdoor $\mathcal{T}$. The latter is defined as the information needed to search $\mathcal{I}$ and find records that satisfy $\mathcal{F}$.*

Search$_\mathcal{I}(\mathcal{T})$: *It iterates through $\mathcal{I}$ applying the trapdoor $\mathcal{T}$ and outputs every record that returns* TRUE *for the input trapdoor.*

This way, if the searchable cryptosystem used is IND-CKA then Alice is able to keep her data with Bob and remain capable of selecting subsets of it without revealing information [11].

3.2 Threat modeling

The development of efficient and secure solutions for management of datasets depends on the awareness of the threats we intent to mitigate. For such, this work follows Grubbs' definitions of adversaries for a database [28].

Active attacker. The worst case scenario is when the attacker acquires full control over the server, being capable of performing arbitrary operations. Thus, he is not committed to follow any protocol.

Snapshot attacker. The adversary obtains a snapshot of the dataset containing the primary data and indexes but no information about issued queries and how they access the encrypted data.

Persistent passive attacker. Another possibility is a scenario in which the attack cannot interfere with the server functionality but can observe all of its operations. We do not consider only attackers that inspect issued queries in real-time, but also those that are able to recover them later. As demonstrated by Grubbs, the data contained in a real-world database goes far beyond the primary dataset (names, addresses, ...). It also includes logs, caches, and auxiliary tables (as MySQL's diagnostic tables) used, for instance, to guarantee *ACID* [3] and enable the server to undo incomplete queries after a power-break. It is very likely that an attacker competent to subjugate the security protocols of the system will be capable to also recover these secondary datasets.

The idea of a *snapshot attacker* is very popular among solutions and researchers intended to develop encrypted databases. Nevertheless, it underestimates the attacker and the many side-attacks a motivated adversary can execute. As Rogaway remarks, we cannot make the mistake to reduce the adversary to the lazy and abstract Bob, but we must remember that it can go far beyond that and take the form of a military-industrial-surveillance program with a billionaire budget and capability to surpass the obvious [49].

4 Related work

The management of a dataset is made by a database management system (DBMS). It is composed by several layers responsible for coordinating read and write operations, guarantee data consistency and integrity, and user access. The engineering of such a system is a complex task and requires smart optimizations to be able to store data, process queries and return the outcome with minimum latency and good scalability.

This way, searchable encryption solutions usually are implemented not inside but on top of these systems as a middleware to translate encrypted queries to the DBMS without revealing plaintext data and decrypt the outcome, as shown in Figure 1. This strategy enables the use of decades of optimizations incorporated in nowadays DBMSs and portable to encrypted data. It is important to state that, ideally, security features should be designed in conjunction with the underlying database. Long-term solutions are expected to assimilate those strategies internally in the DBMS core.

4.1 CryptDB

CryptDB is a software layer that provides capabilities to store data in a remote SQL database and query over it without revealing sensitive information to the DBMS. It introduces a proxy layer responsible to encrypt and adjust queries to the database and decrypt the outcome [47].

The context in which CryptDB stands is a typical structure of database-backed applications, consisting of a DBMS server and a separate application server. To query a database, a predicate is generated by the application and processed by the proxy before it is sent to the DBMS server. The user interacts exclusively with the application server and is responsible for keeping his password secret. This is provided on login to the proxy (via application) that derives all the cryptographic keys required to interact with the database. When the user logs out, it is expected that the proxy deletes its keys.

Data encryption is done through the concept of "onions", which consist of layers of encryption that are combined to provide different functionalities, as shown in Figure 2. Such layers are revealed as necessary to process the queries being performed. Modeling a database involves evaluating the meaning of each attribute and predicting the operations it must support. In

[3] Relative to a set of desirable properties for a database. Acronym to "Atomicity, Consistency, Isolation, Durability".

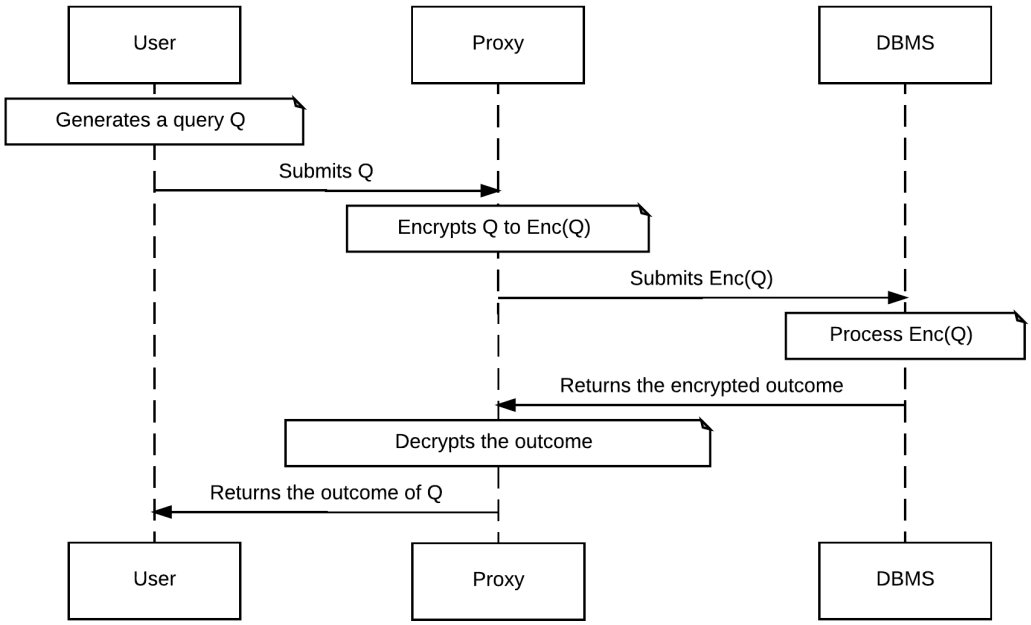


Figure 1: Sequence diagram representing the process of generating and processing an encrypted query. The proxy is positioned between the user and the DBMS in a trusted environment. Its responsibility is to receive a plaintext query, apply an encryption protocol, submit the encrypted query to the DBMS, and decrypt the outcome.

particular, keyword-searching as described in Definition 3 is implemented as proposed in Song's work [54]. The performance overhead over MySQL measured by the authors is up to 30%.

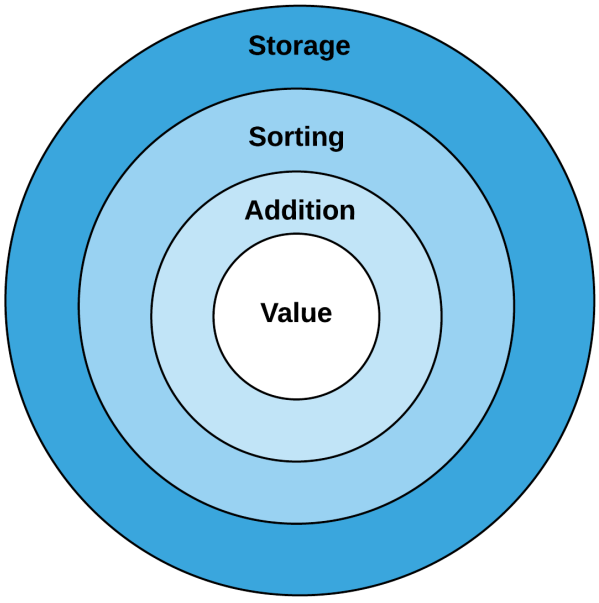


Figure 2: Representation of the data format used by CryptDB. The current value to be protected lies in the center, and a new encryption layer is overlapped to it according to the need of a particular functionality.

Two types of threats are treated in CryptDB: curious database administrators who try to snoop and acquire information about client's data but respect the established protocols (a

persistent passive attacker); and an adversary that gains complete control of application and DBMS servers (an *active attacker*). The authors state that the first threat is mitigated through the encryption of stored data and the ability to query it without any decryption or knowledge about its content; while the second applies only to logged-in clients. In the considered scenario, the cryptographic keys relative to data in the database are handled by the application server. Thus, if the application server is compromised, all the keys it possesses at that moment (that are expected to be only from logged-in users) are leaked to the attacker. Such arguments were revisited after works by Naveed and Grubbs *et al.* demonstrated how to explore several weaknesses of the construction, such as the application of OPE [39, 27].

4.2 Arx

Arx is a database system implemented on top of MongoDB [46]. It targets much stronger security properties and claims to protect the database with the same level of regular AES-based encryption[4], achieving IND-CPA security. This is a direct consequence of the almost exclusively use of AES to construct selection operators, even on range queries, and not only brings strong security but also good performance due to efficiency of symmetric primitives, sometimes even benefiting from hardware implementations. The authors report a performance overhead of approximately 10% when used to replace the database of ShareLatex. The building blocks used for searching follow those described in Definition 3. Furthermore, they apply a different AES key for each keyword when generating the trapdoor, requiring the client to store counters, as explained in the next paragraph.

At its core, Arx introduces two database indexes, Arx-Range for range and order-by-limit queries and Arx-EQ for equality queries, both built on top of AES and using chained garbled circuits. The former uses an obfuscation strategy to protect data, while enabling searches in logarithmic time. The latter embeds a counter into each repeating value. This ensures that the encryption of both are different, protecting them against frequency analysis. Using a token provided by the client, the database is able to expand it in many search tokens and return all the occurrences desired, allowing an index to be built over encrypted data.

The context in which Arx stands is similar to CryptDB. However, the authors consider the data owner as the application itself. This way, it simplifies the security measures and considers the responsibility to keep the application server secure outside of its scope.

4.3 Seabed

Seabed was developed by Papadimitriou *et al.* and aims at Business Intelligence (BI) applications interested in keeping data secure on the cloud [43]. As well as CryptDB and Arx, Seabed was built consisting of a client-side query translator (to SQL), a query planner, and a proxy that connects to a Apache Spark instance [53]. Its main foundations are two new cryptographic constructions, *additively symmetric homomorphic encryption* (ASHE) and *Splayed ASHE* (SPLASHE). The former is used to replace Paillier as the additively homomorphic encryption scheme, stating that their construction is up to three orders of magnitude faster. The latter is used to protect the database against inference attacks [38].

SPLASHE works by splitting sensitive data into multiple attributes, obscuring the low-entropy of deterministic encryption. Formally, let C be a sensitive attribute of a dataset that can be filled with d possible discrete values. The approach taken by SPLASHE is to replace this attribute in the encrypted database by $\{\mathrm{C}_1, \mathrm{C}_2, \cdots, \mathrm{C}_d\}$ such that $\mathrm{C}_v = 1$ and $\mathrm{C}_t = 0$ for $t \neq v$ if $\mathrm{C} = v$. When encrypted by ASHE the ciphertexts will look random to the adversary.

[4]The Advanced Encryption Standard (AES) is a well-established symmetric block cipher enabling high performance implementation in hardware and software [18].

Seabed's authors argue that SPLASHE is strong enough to mitigate frequency analysis, enabling the use of deterministic encryption whenever it is required in the database model. However, Grubbs states that SPLASHE's protection may be deflected through the auxiliary data stored by the database [28]. Their work demonstrates how state-of-the-art databases store metadata that can be used to reconstruct issued queries and, this way, recognize access patterns on the attributes. Such patterns leak the information that SPLASHE intended to hide. Considering this, the only threat really mitigated by SPLASHE against the deterministic encryption of Seabed is from a *snapshot attacker*.

5 Proposed framework

The goal of the proposed framework is to develop a database model capable of storing encrypted records and applying relational algebra primitives on it without the knowledge of any cryptographic keys or the need for decryption. A trade-off between performance and security is desirable, however we completely discard deterministic encryption whenever possible for security reasons. The only exception are contexts with unique records, which avoid by definition weaknesses intrinsic to deterministic encryption. The applicability of this framework goes beyond SQL databases. Besides the relational algebra hereby used to describe the framework, it can be extended to key-value, document-oriented, full text and several other databases classes that keep the same attribute structure.

The three main operations needed to build a useful database are insertion, selection and update. Once data is loaded, being able to select only those pieces that correspond to an arbitrary predicate is the basic block to construct more complex operations, such as grouping and equality joins. This functionality is fundamental when there is a physical separation between the database and the data owner, otherwise high demand for bandwidth is incurred to transmit large fractions of the database records. Furthermore, real data is frequently mutable and thus the database must support updates to remain useful.

We define as *secure* a system model that guarantees that the data owner is the only entity capable of revealing data, which can be achieved by his exclusive possession of the cryptographic keys. Thus, a fundamental aspect of our proposal is the scenario in which the database and the application server handle data with minimum knowledge.

Lastly, the framework does not ensure integrity, freshness or completeness of results returned to the application or the user, since an adversary that compromises the database in some way can delete or alter records. We consider this threat to be outside the scope of this framework.

5.1 Classes of attributes

Records in an encrypted database are composed by attributes. These consist of a name and a value, that can be an integer, float, string or even a binary blob. Values of attributes are classified according to their purpose:

static An immutable value only used for storage. It is not expected to be evaluated with any function, so there is no special requirement for its encryption.

index Used for building a single or multivalued searchable index. It should enable one to verify if an arbitrary term is contained in a set without the need to acquire knowledge of its content.

computable A mutable value. It supports the evaluation with arithmetic circuits and ensures obtaining, after decryption, a result equivalent to the same circuit applied over plaintexts.

The implementation of each attribute must satisfy the requirements without leaking any vital information beyond those related directly with the attribute objective (e.g.: order for *index* attributes). Since the name of an attribute reveals information, it may need to be protected as well. However, the acknowledgement of an attribute is done using its name, so even anonymous attributes must be traceable in a query. An option for anonymizing the attribute name is to treat it as an *index*.

The aforementioned cryptosystems are natural suggestions to be applied within these classes. Since *static* is a class for storage only, which has no other requirements, any scheme with appropriate security level and performance may be used, as AES. On the other hand, *index* and *computable* attributes are immediate applications of ORE and HE schemes. Particularly, the latter defines the HE scheme according to the required operations. Attributes that require only one operation can be implemented with a PHE scheme, which provides good performance; while those that require arbitrary additions and multiplications must use FHE and deal with the performance issues.

Definition 4 (Secure ORE)**.** *Let E and C be, respectively, an encryption and a comparison function. The pair (E, C) forms an encryption scheme with the order-revealing property defined as "secure" if and only if it satisfies Definition 1; the encryption of a message m can be written as $E(m) = (c_L, c_R) = (E_L(m), E_R(m))$, where E_L and E_R are complementary encryption functions; and the comparison between two ciphertexts c_1 and c_2 is done by $C(c_{L1}, c_{R2})$. This way, C may be applied without the complete knowledge of the ciphertexts.*

In order to build a secure and efficient *index*, an ORE scheme that corresponds to Definition 4 should be used. We define the search framework as in Definition 5.

Definition 5 (Encrypted search framework)**.** *Let S be a set of words, sk a secret key, and an ORE scheme (ENC, CMP) that satisfies Definition 4. The operations required for an encrypted search over S are defined as follows:*

BuildIndex$_{sk}(S)$**:** *Outputs the set*

$$S^* = \{c_R \mid (c_L, c_R) = \textsc{Enc}_{sk}(w), \forall w \in S\}.$$

Trapdoor$_{sk}(w)$**:** *Outputs the trapdoor*

$$\mathcal{T}_w = (c_L \mid (c_L, c_R) = \textsc{Enc}_{sk}(w)).$$

Search$_{S^*,\, r}(\mathcal{T}_w)$**:** *To select all records in S^* with the relation $r \in \{$*LOWER, EQUAL, GREATER*$\}$ to word w, one computes the trapdoor $\mathcal{T}_w$ and iterates through S^* looking for the records $w^* \in S^*$ that satisfy*

$$\textsc{Cmp}(\mathcal{T}_w, w^*) = r.$$

The set $\hat{S}$ with all the elements in S^* that satisfy this equation is returned.

5.2 Database operations

Let us consider a model composed by an encrypted dataset stored in a remote server and a user that possesses the secret cryptographic keys. The latter would like to perform queries on data

without revealing sensitive information to the server, as defined in Section 3.1.

In 1970, Codd proposed the use of a relational algebra as a model for SQL [16]. This consists of a small set of operators that can be combined to execute complex queries over the data.

Through the functions defined in Definition 5, a relational algebra for encrypted database operations can be built. The basic operators for such algebra are defined as follows.

1. **Projection** (π_A): Returns a subset A of attributes from selected records. This subset may be defined by attribute names that may or may not be encrypted.

   (a) *encrypted*: If encrypted, a deterministic scheme is used or they are treated as *index* values.
   deterministic scheme: The user computes $A^* = \{Enc_{Det}(a) \mid a \in A\}$. A^* is sent to the server, which picks the projected attributes using a standard algorithm.
   index attributes: The user computes $A^* = \{Trapdoor_{sk}(a) \mid a \in A\}$. A^* is sent to the server, which picks the projected attributes using Search.

   (b) *unencrypted*: Unencrypted selectors are sent to and selected by the server using a standard algorithm.

2. **Selection** (σ_φ): Given a predicate φ, returns only the records satisfying it.

   - Handles exclusively *index*, hence φ must be equivalent to a combination of comparative operators supported by Search.
   - Let $w \diamond x \leftarrow \varphi$, where $\diamond$ is a compatible comparative operator, w an *index* attribute, and x the operand to be compared (e.g.: $\sigma_{age>30}$ signals for records which the attribute named "age" value is greater than 30). The trapdoor $\mathcal{T}_\varphi = Trapdoor_{sk}(\varphi)$ is sent to the server that executes Search.

3. **Cartesian product** ($\times$): The Cartesian product of two datasets is executed using a standard algorithm.

4. **Difference** ($-$): The difference between two datasets A and B encrypted with the same keys is defined as $A - B = \{x \mid x \in A \text{ and } x \notin B\}$.

5. **Union** ($\cup$): The union of two datasets A and B encrypted with the same keys is defined as $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.

Union and difference are defined over datasets with the same set of attributes. The opposite is expected for Cartesian product, so that no attribute may be shared between operands.

Ramakrishnan calls these "basic operators" in the sense that they are essential and sufficient to execute relational operations [48]. Additional useful operators can be built over those. For instance: rename, join-like, and division. The same observation applies in the encrypted domain, and complex operators can be constructed given basic operators defined over the encrypted domain.

6. **Rename** ($\rho_{a,b}$): Renames attributes. Their names may or may not be encrypted.

   (a) *encrypted*: Encryption shall be executed using a deterministic cryptosystem or names treated as *index* values.

   deterministic scheme: Let a be an attribute name to be replaced by b. The user computes $a^* \leftarrow Enc_{Det}(a)$ and $b^* \leftarrow Enc_{Det}(b)$, and sends the output to the server, which applies a standard replacement algorithm.

index attributes: Let a be an attribute name to be replaced by b. The user computes $a^* \leftarrow Trapdoor(a)$ and $b^* \leftarrow c_R \mid (c_L, c_R) = \text{Enc}_{index}(b)$ and sends the output to to the server, which selects attributes related to a^* as EQUAL through the operation SEARCH and renames the result to b^*.

(b) *unencrypted*: Unencrypted attribute names may be renamed by the server using a standard algorithm.

7. **Natural join** ($\bowtie$): Let A and B be datasets with a common subset of attributes. The natural join between A and B is defined as the selection of all elements that lies in A and B and match all the values in those attributes. More formally, let $c_1, c_2, \ldots, c_n$ be attributes common to A and B; $x_1, x_2, \ldots, x_n$ attributes not contained in A or in B; $a_1, a_2, \ldots, a_m$ be attributes unique to A; $b_1, b_2, \ldots, b_k$ be attributes unique to B; and $\mathbb{K} = \mathbb{N}^*_{n+1}$. We have that,
$$A \bowtie B \equiv \sigma_{c_i = x_i} \left(\rho_{(c_i, x_i)}(A) \times B\right), \forall i \in \mathbb{K}.$$

8. **Equi-join** ($\bowtie_\theta$): Let A and B be datasets. The equi-join between A and B is defined as the selection of all elements that lie in A and B and satisfy a condition θ. More formally, $A \bowtie B = \sigma_\theta(A \times B)$.

9. **Division** ($/$): Let A and B be datasets and C the subset of attributes unique to A. The division operator joins the operands by common attributes but projects only those unique to the dividend. Hence, $A/B = \pi_C (A \bowtie B)$.

Finally, it is important to define data insertion and update despite these cannot be properly defined as relational operators.

- **Insert**: Encrypted data is provided and inserted into the database using a standard algorithm.
- **Update**: An update operation is defined as a selection followed by the evaluation of a *computable* attribute by a supported homomorphic operation.

This set of operators enables operating over an encrypted database without the knowledge of cryptographic keys or acquiring sensitive information from user queries.

5.3 Security analysis

We assume the scenario in which the data owner has exclusive possession of cryptographic keys. This way, insertions to the database must be locally encrypted before being sent to the server. The database or the application never deal with plaintext data. Our framework thus has the advantage over CryptDB of preserving privacy even in the outcome of a compromised database or application server.

Despite being conceptually similar to OPE, ORE is able to address several of its security limitations. ORE does not necessarily generate ciphertexts that reveal their order by design, but allows someone to protect this information by only revealing it through specific functions. ORE is able to achieve the IND-OCPA security notion and adds randomization to ciphertexts. Those characteristics make it much safer against inference attacks [38]. The proposal of Lewi and Wu goes even beyond that and is capable of limiting the use of the comparison function [32]. Their scheme generates a ciphertext that can be decomposed into left and right components such that a comparison between two ciphertexts requires only a left component of one ciphertext and the right component of the other. This way, the authors argue that robustness against such attacks is ensured since the database dump may only contain the right component, that is encrypted

using semantically-secure encryption. Their scheme satisfies our notion of a *Secure ORE* and, therefore, provides strong defenses against *Snapshot attackers*.

An eavesdropper (*Active* or *Persistent passive attacker*) is not capable of executing comparisons by himself in a *Secure ORE*. However he may learn the result of these and recognize repeated queries by observing the outcome of a selection. This weakness may still be used for inference attacks, that can breach confidentiality from related attributes. This issue can get worse if the trapdoor is deterministic, when there is no other solution than implementing a key refreshment algorithm. Besides that, the knowledge of the numerical order between every pair of elements in a sequence may leak information depending on the application. This problem manifests itself in our proposal on the σ primitive if it uses a weak index structure, like a naive sequential index. A balanced-tree-based structure, on the other hand, obscures the numerical order of elements in different branches. This way, an attacker is capable of recovering the order of up to $O(\log n)$ database elements and infer about the others, in a database with n elements.

Schemes used with *computable* attributes are limited to IND-CCA1, and typically reach only IND-CPA. Moreover, homomorphic ciphertexts are malleable by design. Thus, an attacker that acquires knowledge about a ciphertext can use it to predictably manipulate others.

Finally, BuildIndex is not able to hide the quantity of records that share the same index. This way, one is able to make inferences about those by the number of records. There is also no built-in protection for the number of entries in the database. A workaround is to fix the size of each *static* attribute value and round the quantity of records in the database using padding. This approach increases secrecy but also the storage overhead.

5.4 Performance analysis

The application of ORE as the main approach to build a database index provides an extremely important contribution to selection queries. Search does not require walking through all the records testing a trapdoor, but only a logarithmic subset of it when implemented over an optimal index structure, such as an AVL tree or B-tree based structure [52]. This characteristic is highlighted on union, intersection and difference operations, which work by comparing and selecting elements in different groups. Moreover, current proposals in the state of the art of ORE enjoy good performance provided by symmetric primitives and does not require more expensive approaches such as public-key cryptography [14, 32, 10]. In particular, although fully homomorphic cryptosystems promise to fulfill this task and progress has been made with new cryptographic constructions [20], it is still prohibitively expensive for real-world deployments [9].

Space consumption is also affected. Ciphertexts are computed as a combination of the plaintext with random data. This way, a non-trivial expansion rate is expected. Differently from speed overheads which are affected by a single attribute type, all attributes suffer with the expansion rate of encryption.

5.5 Capabilities and limitations

Our framework is capable of providing an always-encrypted database that preserves secrecy as long as the data owner keeps the cryptographic keys secure. One is able to select records through *index* and apply arbitrary operations on attributes defined as *computable*. Furthermore, it increases the security of data but maintaining the computational complexity of standard relational primitives, achieving a fair trade-off between security and performance.

Although the framework has no constraints about attributes classified as both *index* and *computable*, there is no known encryption scheme in the literature capable of satisfying all the requirements. This way, the relational model of the database must be as precise as possible when assigning attributes to each class, specially because the costs of a model refactor can be prohibitive.

Some scenarios appear to be more compatible with an encrypted database as described than others. An e-mail service, for example, can be trivially adapted. The e-mails received by a user are stored in encrypted form as *static* and some heuristic is applied on its content to generate a set of keywords to be used on BuildIndex. This heuristic may use all unique words in the e-mail, for example. The sender address may be an important value for querying as well, so it may be stored as an *index*. To optimize common queries, a secondary collection of records may be instantiated with, for example, counters. The quantity of e-mails received from a particular sender, how often a term appears or how many messages are received in a time frame. Storing this metadata information in a secondary data collection avoids some of the high costs of searching in the main dataset.

However, our proposal fails when the user wants to search for something that was not previously expected. For example, regular expressions. Suppose a query that searches for all the sentences that start with "Attack" and end with "dawn", or all the e-mails on the domain "mail.com". If these patterns were not foreseen when the keyword index was built, then no one will be able to correctly execute this selection without the decryption of the entire database. Since the format of the strings is lost on encryption, this kind of search is impossible in our proposal.

Lastly, relational integrity is a desired property for a relational database. It connects two or more sets using same-value attributes in both sets (e.g.: every value of a column in a table A exists in a column in table B), and establishes a *primary-foreign* key relationship. This way, the existence of a record in an attribute classified as *foreign* key depends on the existence of the related record on the other set, in which the *primary* key is equal to that *foreign* key. To implement such feature one must provide to the DBMS capabilities to reinforce relational integrity rules. In other words, the server must be able to recognize such a relationship to guarantee it will be respected by issued queries.

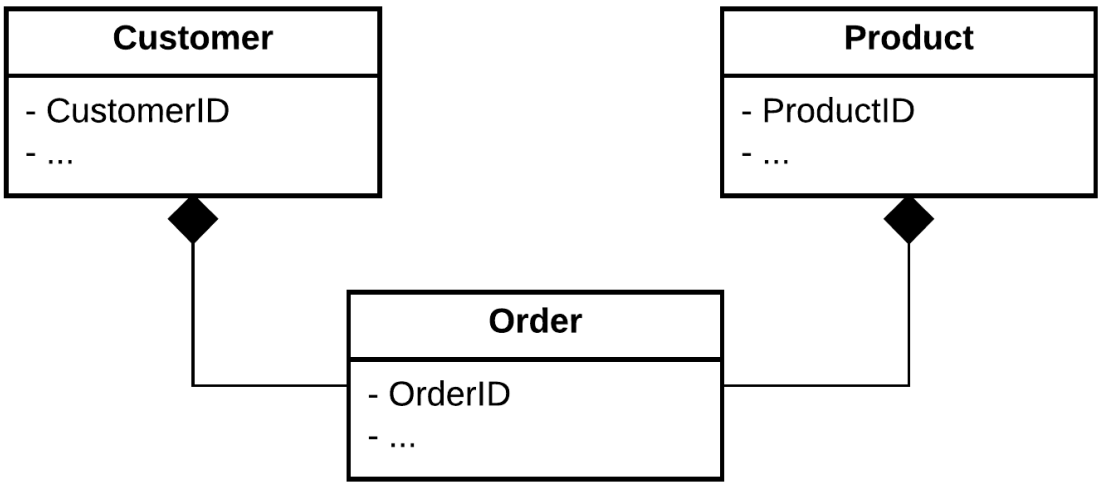


Figure 3: Simple diagram describing the interaction between users and products composing the information regarding an order. Notice that the existence of users and products is independent, but there is a dependence for orders.

An example of the applicability of this concept is an e-commerce database. Best practices dictate that user data should be stored separately from products and orders. Thus, one may model it as in Figure 3. When a new order arrives, it is clear that a user chose some product and informed the store about his intent to buy it. Users and products are concrete elements. However, a sale is an abstract object and cannot happen without a buyer and a product. This

way, to maintain the consistency of the database the DBMS must assure that no sale record will exist without relating user and product. This can be achieved by constructing the sales table such that records contain *foreign* keys for the user and product tables (implying that these contain attributes classified as *primary* keys). By definition this feature imposes an inherent requirement that the DBMS has knowledge about this relationship between records on different tables. Any approach to protect the attributes against third parties will affect the DBMS itself and will never really achieve the needed protection. Thus, any effort on implementing secure relational integrity is at best security through obscurity [5].

6 The winner solution of Netflix's prize

The winner of Netflix Grand Prize was BellKor's Pragmatic Chaos team, who built a solution over the progress achieved in the 2007 and 2008 Progress Prizes [59]. Several machine learning predictors were combined in the final solution with the objective of anticipating the suitability of Netflix content for some user considering previous behavior in the platform. The foundation used for this considered diverse factors, such as:

- What is the general behavior of users when rating? What is the average rating?
- How critic is a user and how this changes over time?
- Does the user demonstrate preference for a specific movie or gender?
- Does the user demonstrate preference for blockbusters or non-mainstream content?
- What property of a movie affects the rating? Is there a correlation between the rating of a user and the presence of a particular actor in a particular gender?

The strategy used to combine these factors (and many others) escapes the scope of this work. We should attend only to the necessity of extracting data from the dataset to feed the learning model.

6.1 Searching the encrypted Netflix's database

An interesting application of our framework is enabling an entity to maintain an encrypted database on third party hardware with a similar structure of Netflix's dataset and being able to implement a prediction algorithm with minimum data leakage to the DBMS. The database should be capable of answering the requested predicates regarding user behavior.

Two scenarios must be considered: the recommendation system running on Netflix's infrastructure, and the dataset becoming public. The former offers an execution environment apparently honest (no one would share data with an openly malicious party) but that can be compromised at some point. To mitigate the damage, the data owner can implement different strategies to reduce the usefulness of any leakage that might happen. Thus, data being handled exclusively in encrypted form on the server is a natural option, since security breaches would reveal nothing but incomprehensible ciphertexts. This is the best case scenario since the data owner has as much control of the execution machine as possible, so our framework proposal can be applied in its full capacity.

As an example of the latter, an important feature required for running the Netflix's prize is the capability of in-dataset comparisons. This time any security solution should find the balance between protecting data secrecy and offering conditions for experimentation. Moreover, we must

[5] When the security of a system relies only in the lack of knowledge by adversaries about its implementation details and flaws.

consider that the execution environment cannot be considered honest anymore. This way, the suitability of our framework depends on the relaxation of the indexing method. *index* values must be published to enable comparisons. For instance, both sides of Lewi-Wu's ciphertexts should be published, or even an OPE scheme may be used on the encryption of the index. From the perspective of the secrecy of ciphertexts, if a IND-OCPA scheme is used then there will be no security reduction beyond what the corresponding threat model expects, as discussed in Section 2.1. The adversary learns the ciphertext order but has restricted ability to make inferences using information acquired from public databases. The only strategy that can be applied uses the data distribution in the dataset (that can be retrieved by enabling comparisons), which puts an attacker in this scenario in a very similar position than the persistent passive attacker.

Given the boundary conditions for privacy preservation, we cannot precisely state the robustness of our framework in the context of the Netflix prize. It clearly increases the hardness against an inference attack, since the adversary is unable to observe the plaintext, but the distribution leaked will give him hints about its content. For instance, the correlation of age groups and most watched (or better rated) movies. It is a fact that all these are expressed as ciphertexts, but as previously stated, a motivated adversary may be able to combine such hints and defeat our security barriers.

Our framework performs much better in the more conservative scenario, where a production server provides recommendations to users with comparisons controlled by the data owner through the two-sided *index* attributes. The impossibility for arbitrary comparisons makes snapshot attacks completely infeasible.

As previously discussed, a motivated adversary with access to the database may be able to also retrieve logs and auxiliary collections. Consequently, previous queries may leak the second side of *index* ciphertexts and recall the danger of persistent passive attacks. So, an important feature for future work is the development of a key refreshment algorithm to nullify the usefulness of such information.

6.2 Data structure

The dataset shared by Netflix is composed by more than 100 million real movie ratings from 480,000 users about 17,000 movies, made between 1999 and 2005, and formatted as a training test set [7, 59]. It contains a subset of 4.2 million of those ratings, with up to 9 ratings per user. It consists of:

- *CustomerID:* A unique identification number per user,
- *MovieID:* A unique identification number per movie,
- *Title:* The English title of the movie,
- *YearOfRelease:* The year the movie was released,
- *Rating:* The rating itself,
- *Date:* The timestamp informing when the rating happened.

6.3 Constructing queries of interest over encrypted data

Following we rewrite some of the main predicates required for BellKor's solution using the relational algebra of Section 5.2, thus enabling their execution over an encrypted dataset.

Let

- $\mathcal{DB}$ be a dataset as described in Section 6.2,

- AID be the CustomerID related to a particular user (that we shall call Alice),
- BID be the CustomerID related to a particular user different to Alice (that we shall call Bob),
- MID be the MovieID related to an arbitrary movie in the dataset (that we should refer as $\mathcal{M}$),
- $\mathcal{T} = (\mathcal{T}_{\text{start}}, \mathcal{T}_{\text{end}})$ be a time interval of interest,
- $\mathcal{T}_{\text{first-alice}}$ be the timestamp of the first rating Alice ever made,
- $\mathcal{C}()$ be a function that receives a set and returns the quantity of items contained,
- r_{H} and r_{L} be thresholds for extreme ratings characterizing users that hated or loved a movie,
- $\sigma_{Date \in \mathcal{T}}(\mathcal{DB}) \equiv \sigma_{Date \geq \mathcal{T}_{\text{start}}}(\mathcal{DB}) + \sigma_{Date < \mathcal{T}_{\text{end}}}(\mathcal{DB})$,
- $f(X) = \frac{\sum_{x \in X} \pi_{Rating}(x)}{\mathcal{C}(X)}$.

Then, some of the required predicates for BellKor's solution are:

- **Movies rated by Alice:** Returns all movies that received some rating from Alice. For

$$\text{U}(X) = \sigma_{CustomerID=X}(\mathcal{DB}),$$

we have the query

$$\pi_{\text{MovieID}}(U(\text{AID})). \tag{1}$$

- **Users who rated M:** Returns all users that sent some rating for MID. For

$$M(X) = \sigma_{MovieID=\text{MID}}(\mathcal{DB}),$$

we have the query

$$\pi_{CustomerID}(M(\text{MID})). \tag{2}$$

- **Average of Alice's ratings over time:** Computes the average of all rates sent by Alice during a particular time interval $\mathcal{T}$. For

$$\mathcal{A}_{\text{AID},\mathcal{T}} = \sigma_{Date \in \mathcal{T}}(U(\text{AID})),$$

we have that

$$\text{avg}(\text{AID}, \mathcal{T}) = \begin{cases} f(\mathcal{A}_{\text{AID},\mathcal{T}}) \text{ if } \mathcal{C}(\mathcal{A}_{\text{AID},\mathcal{T}}) > 0, \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

- **Average of ratings for a particular movie M in a timeset:** Computes the average of all rates sent by all users during a particular time interval $\mathcal{T}$ for a movie $\mathcal{M}$. For

$$\mathcal{M}_{\text{MID},\mathcal{T}} = \sigma_{Date \in \mathcal{T}}(M(\text{MID}))$$

we have that

$$\text{avg}(\text{MID}, \mathcal{T}) = \begin{cases} f(\mathcal{M}_{\text{MID},\mathcal{T}}) \text{ if } \mathcal{C}(\mathcal{M}_{\text{MID},\mathcal{T}}) > 0, \\ 0, \text{otherwise.} \end{cases} \tag{4}$$

- **Number of days since Alice's first rating:** Computes how many days have been since the Alice submitted the first rating of movie, relative to a moment $\mathcal{I}$.

$$\text{dsf}(AID, \mathcal{I}) = \mathcal{I} - \pi_{Date}(\sigma_{min(Date)}(U(\text{AID}))). \tag{5}$$

- **Quantity of users who hated $\mathcal{M}$:** Counts the quantity of very bad ratings $\mathcal{M}$ received since its release.
$$\mathcal{C}_H(\mathcal{M}) = \mathcal{C}\left(\sigma_{\text{MovieID}=\text{MID}}(\sigma_{\text{Rating}\leq r_{\text{H}}}(\mathcal{DB}))\right). \tag{6}$$
- **Quantity of users who loved $\mathcal{M}$:** Counts the quantity of very good ratings $\mathcal{M}$ received since its release.
$$\mathcal{C}_L(\mathcal{M}) = \mathcal{C}\left(\sigma_{MovieID=\text{MID}}(\sigma_{\text{Rating}\geq r_{\text{L}}}(\mathcal{DB}))\right). \tag{7}$$
- **Users that are similar to Alice:** The similarity assessment between users require the derivation of a specific metric according to the boundary-conditions. The winning solution developed a sophisticated strategy, building a graph of neighborhoods considering similar movies and users and computing a weighted mean of the ratings. For simplicity, we shall condense two factors that can be used for this objective: the set of common rated movies, and how close the ratings are. To query the movies rated both by Alice and Bob, let
$$\alpha_{\text{AID}} = \pi_{\text{MovieID,RatingA}}(\rho_{Rating,RatingA}(U(\text{AID})))$$
and
$$\beta_{\text{BID}} = \pi_{\text{MovieID,RatingB}}(\rho_{Rating,RatingB}(U(\text{BID}))).$$
Then
$$\textsc{SimilaritySet}\,(\text{AID}, \text{BID}) = \alpha_{AID} \bowtie \beta_{BID} \tag{8}$$
returns a sequence of tuples of ratings made by Alice and Bob. A simple approach for evaluating proximity is to compute the average of the difference of ratings for each movie returned by Equation 8, as shown in Equation 9.
$$\frac{\sum_{\textsc{SimilaritySet}(\text{AID,BID})}|\text{RatingA} - \text{RatingB}|}{\mathcal{C}\left(\textsc{SimilaritySet}\,(\text{AID}, \text{BID})\right)} \tag{9}$$

7 Implementation

A proof-of-concept implementation of the proposed framework was developed and made available to the community under a *GNU GPLv3* license [2]. It runs upon the popular document-based database MongoDB and was designed as a wrapper over its Python driver [15]. Hence, we are able to evaluate its competence as a search framework as well as the compatibility with a state-of-the-art DBMS. Moreover, running as a wrapper makes it database-agnostic and restricts the server to dealing with encrypted data. We choose to implement our wrapper over a NoSQL database so we could avoid dealing with the SQL interpreter and thus reduce the implementation complexity. However, our solution should be easily portable to any SQL database because of its strong roots in relational algebra. Table 1 provides the schemes used for each attribute class, the parameter size and its security level.

Table 1: Chosen cryptosystems for each attribute presented in Section 5.

Attribute	**Cryptosystem**	**Parameters**	**Sec. level**
static	AES	128 bits	128 bits
index	Lewi-Wu	128 bits	128 bits
computable $(+)$	Paillier	3072 bits	128 bits
computable $(\times)$	ElGamal	3072 bits	128 bits

Lewi-Wu's ORE scheme relies on symmetric primitives and achieves IND-OCPA. The authors claim that this is more secure than all existing OPE and ORE schemes which are practical [32]. Finally, Paillier and ElGamal are well-known public-key schemes. Both achieve

IND-CPA and are based on the hardness of solving integer factorization and discrete logarithm problems, respectively. Paillier supports homomorphic addition, while ElGamal provides homomorphic multiplication. Both are classified as PHE schemes [42, 22]. The implementation of AES was provided by the *pycrypto* toolkit [33]; we wrote a Python binding over the implementation of Lewi-Wu provided by the authors [63]; and we implemented Paillier and ElGamal schemes. An AVL tree was used as the index structure. It is important to notice that performance was not the main focus in this proof-of-concept implementation.

The machines used to run our experiments are described in Tables 2 and 3. The former specifies the machine used to host the MongoDB server, and latter describes the one used to run the client. Both machines were connected by a Gigabit local network connection.

Table 2: Specifications of the machine used for running the MongoDB instance.

CPU	2 x Intel Xeon E5-2670 v1 @ 2.60GH
OS	CentOS 7.3
Memory	16 x DDR3 DIMM 8192MB @ 1600MHz
Disk	7200RPM Western Digital HDD (SATA)

Table 3: Specifications of the machine used for running the queries described in this document.

CPU	2 x Intel Xeon E5-2640 v2 @ 2.60GH
OS	Ubuntu 16.04.2
Memory	4 x DDR3 DIMM 8192MB @ 1600MHz
Disk	7200RPM Western Digital HDD (SATA)

While it was trivial to index the plaintext dataset natively, it was not so simple with the encrypted version. MongoDB is not friendly to custom index structures or comparators, so we decided to construct the structure with Python code and then insert it into the database using pointers based on MongoDB's native identity codes. Walking through the index tree depends on a database-external operation at Python-side, calling MongoDB's FIND method to localize documents related to left/right pointers starting from the tree root. Such limitation brings a major performance overhead that especially affects range queries.

7.1 Netflix's prize dataset

We used the Netflix's dataset to measure the computational costs of managing an encrypted database.

We consider the two threat scenarios discussed in Section 6.1, a recommendation system running in production, and the disclosure of a real ratings dataset. Both require the ability of running all queries presented in Section 6.3, differing only in the content that must be inserted in the encrypted dataset (for instance, how much of the *index* ciphertexts may be stored). Hence, to demonstrate the suitability of our framework as a strategy to fulfill the development and execution of a good predictor in such contexts, and being capable of mitigating damages to user privacy, we implemented those queries in an encrypted instance of the dataset.

As shown in Table 4, the four attributes chosen are classified as *static*, which use the faster encryption and decryption available. *Rating* is tagged *computable* for addition and multiplication, thus being compatible with Equations 3 and 4. We use *CustomerID*, *MovieID*, and *Date* for indexing. Encrypting the document structure takes 540μs per record.

There is no way to implement integer division over Paillier ciphertexts. Thus, the predictor may be adapted to use the non-divided result on Equations 3 and 4. Otherwise, a division oracle

Table 4: Attribute structure of elements in the Netflix's prize dataset.

Name	**Value type**	**Class**
CustomerID	integer	*index*, *static*
MovieID	integer	*index*, *static*
Rating	integer	*static*, *computable*
Date	integer	*index*, *static*

must be provided, to which one could submit their homomorphically added values and ask for a ciphertext equivalent to its division by an arbitrary integer. This approach does not reduce security for an IND-CPA homomorphic scheme.

Handling such a large dataset was not an easy task. The ciphertext expansion factor caused by AES, Paillier and ElGamal cryptosystems was relatively small, but the Lewi-Wu implementation is very inefficient in this regard, having an expansion of about 400×. This directly affects the index building and motivated us to explore different strategies to encrypt and load the dataset to a MongoDB instance in reasonable time.

Again, MongoDB is not friendly for custom indexing. A contribution by Grim, Wiersma and Turkmen to our code enables us to manage the AVL tree inside the database through JavaScript code stored inside MongoDB's engine (the only way to execute arbitrary code in MongoDB) [26]. Thus, our primary approach to feed the DBMS with the dataset was quite simple: encrypt each record in our wrapper, insert in the database, and update the index and balance the tree inside the DBMS. The two first operations suffered from an extremely high memory consumption and by far surpassed our available RAM capacity. However, an even worse problem we faced was to build the AVL tree. For the first thousand records we could do the node insertion and tree balancing with a transfer rate of about 600 documents per second, but it dropped quickly as the tree height increases, reaching less than 1 document per second before insertion of the 10,000th record.

We found out that the initial insertions required a novel approach. We completely decoupled the *index* from the *static* data encryption and chose to first feed the database with the *static* ciphertexts, constructing the entire AVL tree using the plaintext on client-sided memory, and then inserting it in the database. Moreover, to speed up the index construction we followed Algorithms 1 and 2 to construct the AVL tree. It takes a sorted list of inputs and builds the tree with time complexity of $O(n)$ on the list size. As a result of this approach we were able to build the encrypted database and the index by 3000 documents per second during the entire procedure.

Algorithm 1 Build an AVL tree using an array of documents.

1: **procedure** BUILD_INDEX(docs)
2: $docs_{sort} \leftarrow sort(docs)$;
3: $docs_{group} \leftarrow group(docs_{sort})$; ▷ Combine equal elements
4: return $build\_aux(docs_{group}, 0, lenght(docs_{group}) - 1)$;
5: **end procedure**

Table 5 shows the latency of each step we observed during the construction of the AVL tree-based indexes. The total time to build those 3 indexes was 40 minutes.

The queries we derived in Section 6.3 were ported to our encrypted database, and the latency for each one can be seen in Table 6. The parameters used for each Equation were arbitrarily selected. The *CustomerID*s for Alice and Bob (AID and BID) were 1061110 and 2486445 respectively, while MID was fixed as 6287. The time interval used was 01/01/2003 to

Algorithm 2 Recursively builds an AVL tree with a sorted array of documents without repeated elements. Receives the array itself, and the indexes for the leftmost and rightmost elements to be handled in each recursive call.

```
1: procedure BUILD_AUX(docs, L, R)
2:     if L = R then
3:         return new_node(docs[L]);
4:     else if L + 1 = R then
5:         left_node ← new_node(docs[L]);
6:         right_node ← new_node(docs[R]);
7:         left_node.right = right_node;
8:         left_node.height = 1;
9:         return left_node;
10:    else
11:        M ← L + ⌊(R − L)/2⌋;
12:        middle_node ← new_node(docs[M]);
13:        middle_node.left ← build_aux(docs, L, M − 1);
14:        middle_node.right ← build_aux(docs, M + 1, R);
15:        lh ← middle_node.left.height;
16:        rh ← middle_node.right.height;
17:        middle_node.height = 1 + max(lh, rh);
18:        return middle_node;
19:    end if
20: end procedure
```

Table 5: Latency for each step in the construction of an AVL tree following Algorithm 1 for each *index* attribute specified in 4.

Attribute	**sort** (s)	**group** (s)	**build_index** (s)
CustomerID	329	459	129
MovieID	270	161	2
Date	187	197	5

01/01/2004. Lastly, we defined a "loved" rating as those greater than 3, and "hated" rating as those lower than 3. We applied some efforts in optimizing the execution, however these results can still be improved.

As it can be seen, complex queries composed by range selections, as well as those with numerous outcomes, suffered from the slow communication between server and the client. The latter influenced even the plaintext results. The outcome of Equation 1 is quite small, requiring much less time to return than the outcome of Equation 2 (the number of movies rated by a user is much smaller than the number of users that rated a movie).

The time interval selection in Equations 3 and 4 required our implementation to visit many nodes in the index tree for *Date*. Because each iteration requires a back and forth between the server and the client, this dramatically impacted the performance. The latencies for Equations 1 and 5 were only 1.4 times higher in the encrypted database, however it reached 710 times for Equation 3. Lastly, Equations 6 and 7 depend on Paillier's homomorphic additions. This implied in a factor-12 slowdown.

Table 6: Execution times for implementations of the Equations presented in Section 6.3 on an encrypted MongoDB collection and an equivalent plaintext version. Each row contains the latency for the entire circuit required by the respective Equation and returning the outcome to the client. Times are computed as the average for 100 independent executions. The machine and parameters used in each cryptosystem follow those defined in Section 7.

Equation	**Encrypted**	**Plaintext**
1	16.6 ms	11.9 ms
2	2 s	850 ms
3	2.7 s	3.8 ms
4	2.7 s	1.0 s
5	16.8 ms	11.8 ms
6 and 7	12 ms	1.0 ms
9	603 ms	200 ms

The implementation of queries based on Equations 3 and 4 took the previous suggestion and skipped the final division. We believe this does not undermine any procedure that eventually consumes this outcome.

The optimal implementation of Equations 6 and 7 requires indexing of *MovieID* and *Rating* attributes. However, due to limitations in our implementation, rather than indexing the latter we use linear search over the outcome of the movie selection on client-side. Our approach for building indexes use the set data structure of MongoDB documents. Yet, in the most recent release such structure holds up to 16MB of data, much smaller than the required for indexing the entire dataset for *Rating* with our strategy.

Lastly, Equation 8 was implemented aiming at the joining of data regarding two users, Alice and Bob. We let the evaluation of such information by a similarity-evaluation function as future work.

8 Conclusion

We presented the problem of searching in encrypted data and a proposal of a framework that guides the modeling of a database with support to this functionality. This is achieved by combining different cryptographic concepts and using different cryptosystems to satisfy the requirements of each attribute, like order-revealing encryption and homomorphic encryption. Over this approach, a relational algebra was built to support encrypted data composed by: projection, selection, Cartesian product, difference, union, rename, and join-like operators.

An overview of the security provided is discussed, as well as a performance analysis about the impact in a realistic database. As a case study we explored the Netflix prize, which published an anonymized dataset with real-world information about user behavior which was later deanonymized through correlation attacks involving public databases.

We offered a proof-of-concept implementation in Python over the document-based database MongoDB. To demonstrate its functionality, we selected and ran some of the main predicates required by the winning solution of the Netflix Grand Prize and measured the performance impact of the execution in a encrypted version of the dataset. We conclude that our proposal offers robustness against a compromised server and we discuss how it would help to avoid the deanonymization of the Netflix dataset. In comparison with CryptDB, our proposal provides higher security, since it delegates exclusively to the data owner the responsibility of encrypting and decrypting data. This way, privacy holds even in a scenario of database or application compromise.

As future research objectives we can mention:

- *Extend the scope to associative arrays*: Despite being powerful on SQL, Codd's relational algebra is not completely applicable for non-relational databases. For instance, NoSQL and NewSQL databases lack the concept of joining. A more convenient foundation for such context is algebra of associative arrays [30]. Hence, the formalization of our primitives in such algebra would be an interesting work.

- *Reduce the leakage of index construction in the database*: Our proposal leaks both sides of *index* ciphertexts to enable the index construction. At this moment, an eavesdropper monitoring queries would learn all information required to freely compare the exposed ciphertexts. As discussed in this document, such capability must be restricted, under risk of enabling an inference attack.

- *Key refreshment algorithm*: A persistent passive attacker is capable of learning the required information to perform comparisons through the entire database, just by observing issued queries and its outcome. Thus, the framework primitives must be improved to support an algorithm capable of avoid any damage caused by the knowledge of such information.

- *Hide repeated queries*: Even with encrypted queries and outcomes, the access pattern in a database may indicate repeated queries and the associated records. A technique such as ORAM could be useful to protect such information [55].

- *Explore different databases*: As stated, MongoDB is a very popular NoSQL database. However, it is not friendly to custom indexing or third party code running in its engine. Thus, to replace it by a more appropriate database could provide a more productive system.

- *Improve performance of our implementation*: Our implementation had as objective to be a proof-of-concept and demonstrate how the proposal works. The development of a space and speed-optimized versions is an important next step.

Author's contributions

The first author developed the study design, carried out the implementation efforts and wrote most the paper. The second author contributed with discussions about the proposal and its validation with the case study. Both authors read and approved the final manuscript.

Acknowledgements

A prior version of this paper was presented at the XVI Brazilian Symposium on Information and Computational Systems Security (SBSeg16) [3].

The authors thank Proof. André Santanchè for the initial opportunity to develop this work, the Multidisciplinary High Performance Computing Lab (LMCAD) for providing the required infrastructure, and the anonymous reviewers that helped improving this work.

Funding

This research was partially founded by CNPq and the Google Research Awards Latin America.

Availability of data and materials

Our proof-of-concept implementation is publicly available on GitHub, as well as a generator for a synthetic dataset used for testing [2]. The Netflix dataset is available in academic repositories [40].

Competing interests

The authors declare that they have no competing interests.

References

[1] *A Secure Coprocessor for Database Applications*, 2013.

[2] Pedro Alves. A proof-of-concept searchable encryption backend for mongodb. `https://github.com/pdroalves/encrypted-mongodb`, 2016. Accessed July 2017.

[3] Pedro G. M. R. Alves and Diego F. Aranha. A framework for searching encrypted databases. In *Proceedings of the XVI Brazilian Symposium on Information and Computational Systems Security*, Nov 2016.

[4] Michael Barbaro and Tom Zeller. A Face Is Exposed for AOL Searcher No. 4417749, 2006. Accessed 05 April 2017.

[5] BBC News. Yahoo 'state' hackers stole data from 500 million users, 2016. Accessed 23 September 2016.

[6] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. *Advances in Cryptology — CRYPTO '98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings*, chapter Relations among notions of security for public-key encryption schemes, pages 26–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[7] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[8] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. *Lecture Notes in Computer Science*, 5479:224–241, 2009.

[9] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. Private database queries using somewhat homomorphic encryption. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security*, ACNS'13, pages 102–118, Berlin, Heidelberg, 2013. Springer-Verlag.

[10] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. *Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation*, pages 563–594. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[11] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2):18:1–18:51, August 2014.

[12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[13] Rajkumar Buyya. Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing As the 5th Utility. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 1–, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical order-revealing encryption with limited leakage. In FSE, 2016.

[15] Kristina Chodorow and Michael Dirolf. *MongoDB: The Definitive Guide*. O'Reilly Media, Inc., USA, 1st edition, 2010.

[16] E F Codd. A relational model of data for large shared data banks. *Commun. ACM 26*, (6):64–69, 1983.

[17] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[18] Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael, 1999.

[19] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, 13(18):1587–1611, 2013.

[20] Yarkın Doröz, Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, Berk Sunar, William Whyte, and Zhenfei Zhang. Fully homomorphic encryption from the finite field isomorphism problem. Cryptology ePrint Archive, Report 2017/548, 2017.

[21] DuckDuckGo. Privacy Mythbusting #3: Anonymized data is safe, right? (Er, no.). `https://spreadprivacy.com/dataanonymization-e1e2b3105f3c`. Accessed 24 July 2017.

[22] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[23] Craig Gentry. Computing Arbitrary Functions of Encrypted Data. *Commun. ACM*, 53(3):97–105, March 2010.

[24] Philippe Golle. Revisiting the uniqueness of simple demographics in the us population. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, WPES '06, pages 77–80, New York, NY, USA, 2006. ACM.

[25] Glenn Greenwald and Ewen MacAskill. NSA Prism program taps in to user data of Apple, Google and others, 2013.

[26] M.W. Grim, A.T. Wiersma, and F. Turkmen. Security and Performance Analysis of Encrypted NoSQL Databases. Technical report, University of Amsterdam, 2017.

[27] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built on top of encrypted data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1353–1364, New York, NY, USA, 2016. ACM.

[28] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. Why your encrypted database is not secure. Cryptology ePrint Archive, Report 2017/468, 2017. `http://eprint.iacr.org/2017/468`.

[29] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the Use of Cloud Computing for Scientific Workflows. In *eScience 08. IEEE Fourth International Conference on*, pages 640–645, Dec 2008.

[30] Jeremy Kepner, Vijay Gadepally, Dylan Hutchison, Hayden Jananthan, Timothy G. Mattson, Siddharth Samsi, and Albert Reuther. Associative array model of sql, nosql, and newsql databases. *CoRR*, 2016.

[31] Vladimir Kolesnikov and Abdullatif Shikfa. On the limits of privacy provided by Order-Preserving Encryption. *Bell Labs Technical Journal*, 2012.

[32] Kevin Lewi and David J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. Cryptology ePrint Archive, Report 2016/612, 2016.

[33] Dwayne Litzenberger. Python Cryptography Toolkit. `http://www.pycrypto.org/`, 2016. Accessed 03 July 2016.

[34] Jake Loftus, Alexander May, Nigel P. Smart, and Frederik Vercauteren. On CCA-Secure Somewhat Homomorphic Encryption. In *Proceedings of the 18th International Conference on Selected Areas in Cryptography*, SAC'11, pages 55–72, Berlin, Heidelberg, 2012. Springer-Verlag.

[35] Michael Arrington. AOL Proudly Releases Massive Amounts of Private Data. `https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/`, 2006. Accessed 24 July 2017.

[36] Arvind Narayanan and Edward W Felten. No silver bullet: De-identification still doesnt work. Technical report, 2014.

[37] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[38] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 644–655, New York, NY, USA, 2015. ACM.

[39] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 644–655, New York, NY, USA, 2015. ACM.

[40] Netflix prize data set. `http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a`, 2009. Accessed July 2017.

[41] Niklas Magnusson and Niclas Rolander. Sweden Tries to Stem Fallout of Security Breach in IBM Contract, 2017.

[42] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[43] Antonis Papadimitriou, Ranjita Bhagwan, Nishanth Chandran, Ramachandran Ramjee, Andreas Haeberlen, Harmeet Singh, Abhishek Modi, and Saikrishna Badrinarayanan. Big data analytics over encrypted datasets with seabed. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 587–602, Berkeley, CA, USA, 2016. USENIX Association.

[44] Al Pascual. 2017 Data Breach Fraud Impact Report: Going Undercover and Recovering Data. Technical report, Javelin Advisory Services, 2017.

[45] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[46] Rishabh Poddar, Tobias Boelter, and Raluca Ada Popa. Arx: A strongly encrypted database system. Cryptology ePrint Archive, Report 2016/591, 2016.

[47] 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, 2011. ACM.

[48] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 3 edition, 2003.

[49] Phillip Rogaway. The moral character of cryptographic work. *IACR Cryptology ePrint Archive*, page 1162, 2015.

[50] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136. ACM, 2014.

[51] Bruce Schneier. Data is a toxic asset, 2016.

[52] Robert Sedgewick. *Algorithms*, chapter 15, page 199. Addison-Wesley, 1983.

[53] Abdul Ghaffar Shoro and Tariq Rahim Soomro. Big data analysis: Apache spark perspective. *Global Journal of Computer Science and Technology*, 15(1), 2015.

[54] Dawn Xiaodong Song, David Wagner, Adrian Perrig, and A Perrig. Practical techniques for searches on encrypted data. *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 44–55, 2000.

[55] Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: an extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 299–310. ACM, 2013.

[56] Latanya Sweeney. Simple Demographics Often Identify People Uniquely. 2000.

[57] Simon Thomsen. Extramarital affair website Ashley Madison has been hacked and attackers are threatening to leak data online, 2015. Accessed 25 May 2016.

[58] Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. *Proc. VLDB Endow.*, 6(5):289–300, March 2013.

[59] Andreas Töscher, Michael Jahrer, and Robert M. Bell. The bigchaos solution to the netflix grand prize, 2009.

[60] C. Vecchiola, S. Pandey, and R. Buyya. High-Performance Cloud Computing: A View of Scientific Applications. In *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, pages 4–16, Dec 2009.

[61] Zhibo Wang, Jilong Liao, Qing Cao, Hairong Qi, and Zhi Wang. Friendbook: a semantic-based friend recommendation system for social networks. *IEEE Transactions on Mobile Computing*, (3):538–551, 2015.

[62] Harrison Weber. How the NSA & FBI made Facebook the perfect mass surveillance tool, 2014. Venture Beat. Published on 05/15/2014.

[63] David J. Wu and Kevin Lewi. Fastore. `https://github.com/kevinlewi/fastore`, 2016. Accessed July 2017.

[64] Zhifeng Xiao and Yang Xiao. Security and Privacy in Cloud Computing. *IEEE Communications Surveys Tutorials*, 15(2):843–859, Second 2013.

2.2 Efficient implementation of homomorphic encryption

This publication is entitled "Faster Homomorphic Encryption over GPGPUs via Hierarchical DGT" and was published at the Financial Cryptography and Data Security 2021 conference, on March 2021. This paper is a natural follow up of my master's dissertation, which investigates the efficient implementation of homomorphic encryption schemes on GPUs [3]. This time, we aim the Fan-Vercauraten (BFV) proposal, and combine novel methods in the literature with classical, as the hierarchical design of DFT-based algorithms.

Faster Homomorphic Encryption over GPGPUs via hierarchical DGT

Pedro G. M. R. Alves[1] Jheyne N. Ortiz[1] Diego F. Aranha[2]

University of Campinas, Campinas, Brazil[1]
Aarhus University, Aarhus, Denmark[2]

March 2021

Abstract

Privacy guarantees are still insufficient for outsourced data processing in the cloud. While employing encryption is feasible for data at rest or in transit, it is not for computation without remarkable performance slowdown. Thus, handling data in plaintext during processing is still required, which creates vulnerabilities that can be exploited by malicious entities. Homomorphic encryption schemes enable computation over ciphertexts without knowing the related plaintexts or the decryption key. This work focuses on the challenge of developing an efficient implementation of the BFV scheme on CUDA. This is done by combining and adapting different literature approaches, as the *double*-CRT representation and the Discrete Galois Transform. Moreover, we propose and implement an improved formulation of the DGT inspired by classical algorithms, which computes the transform up to 2.6 times faster than the state-of-the-art. By using these approaches, we obtain up to 3.6 times faster homomorphic multiplication.

Keywords— Fully Homomorphic Encryption, BFV, CUDA, Polynomial multiplication, Privacy-preserving computing

1 Introduction

With the growing data collection by governments and companies, protecting its secrecy becomes as important as processing and extracting useful information. However, how to efficiently collect and compute user data without undermining their privacy is an open problem. System breaches may happen even when data holders choose the most conservative practices and never share data intentionally.

The Breach Level Index provides distressful statistics about data leakage. It states that most breaches occur by accidental loss on leaving plaintext data exposed inadvertently. Attacks from malicious parties, which explore vulnerabilities to subvert security mechanisms, are also far from negligible [29]. Data can be protected by encryption even in case of leakage. However, encryption-decryption cycles during its lifespan create a weak point in the system's security. Thus, building the system roots attached to mathematical guarantees and dispensable decryption is the only way to achieve a more reliable security.

Homomorphic Encryption (HE) schemes enable data processing while protecting its confidentiality. They allow the evaluation of arithmetic circuits over ciphertexts by a third party

without any knowledge of the corresponding plaintexts or the decryption key, preventing the computation's inputs and outcome to be learned. Hence, HE is a natural candidate for solving privacy issues caused by malicious third parties, careless administrators, or other security flaws during the processing, such as side-channel vulnerabilities.

Many of the HE schemes available in the literature rely on the hardness of the Ring-Learning with Errors (RLWE) problem. This assumption offers a strategy for protecting messages, encoded as polynomials in $R_q = \mathbb{Z}_q[x]/(f(x))$, by adding noise in a way that it can only be removed when given a trapdoor. There are several proposals following this approach such as BFV [21], CKKS [13], and TFHE [14]. All depend on polynomial arithmetic as the main building block, so its efficient implementation is critical for adopting HE in the real-world.

CUDA is an important tool for the efficient implementation of polynomial arithmetic. It's a SIMD architecture developed and maintained by NVIDIA for employing the data parallelism potential of a GPU in tasks beyond graphical processing. However, the particularities of CUDA impose challenges for its cryptographic use. Its processing flow demands careful planning to align possible conditional branches with certain thread groups, and its memory paradigm considers several structures with different dimensions and latency characteristics, separated from the machine's main memory. Moreover, at this point, no general-purpose cryptographic library or polynomial arithmetic framework supports CUDA. Hence, these constraints motivate the development of a complete toolkit to work as an arithmetic engine aimed at RLWE-based cryptosystems.

Our contributions. This work presents mathematical tools and techniques for the efficient implementation of the BFV scheme in CUDA. We follow the literature by employing the Residue Number System (RNS) as the best approach for handling the multiprecision arithmetic required, and the Halevi, Polyakov, and Shoup modification of BFV to solve the division and rounding problem in the RNS domain [9, 24]. The main contributions of this study are:

- A novel *hierarchical* formulation of the Discrete Galois Transform (DGT) that offers about two times lower latency on GPUs than the best version available in the literature. Moreover, we collect evidence that suggests it is faster than the commonly used Number Theoretic Transform (NTT). Such formulation is inspired by Bailey's version of the Fast Fourier Transform [7].

- Compatible choice of parameters between the DGT and the RNS representation. We show that the *double*-CRT representation proposed by Gentry *et al.*is a better implementation design than the usual approach of working with Mersenne or Solinas primes in different rings [10].

- A more efficient, GPU-optimized, state machine which reduces the need for moving data in and out of the DGT domain and between the main memory and the GPU global memory.

These contributions are not limited to the BFV cryptosystem and can be easily applied to other RLWE-based schemes, such as CKKS. Moreover, we provide latency benchmarks from a proof-of-concept implementation named SPOG, which was built based on the methods above. Two relevant works employing the DGT are considered for comparison with our results: Badawi, Polyakov, Aung, Veeravalli, and Rohloff [4]; and Badawi, Veeravalli, Mun, and Aung [6]. When considering homomorphic multiplication as the main performance-critical operation, SPOG offers higher performance against these works, surpassing a 3.6-factor performance improvement against the latter.

2 Mathematical background

The efficient implementation of an RLWE-based cryptosystem on CUDA requires carefully designed building blocks for adjusting the operations to the architecture's limitations. The BFV

cryptosystem, as well as other HE proposals, relies on large parameters for achieving proper security levels. This imposes a challenge in the light of GPGPUs' [1] constraints, for both the size of the coefficients, much larger than the native integer instruction set; and the polynomial arithmetic, that requires highly-optimized algorithms to reduce the computational complexity and improve the scalability of expensive operations, such as polynomial multiplication.

This Section describes the Fan and Vercauteren cryptosystem; presents the Residue Number System (RNS) representation, used to avoid the multiprecision arithmetic; and introduces the Discrete Galois Transform (DGT), a more suitable variant of the Fast Fourier transform (FFT) to GPU implementation.

2.1 The BFV cryptosystem

Fan and Vercauteren proposed a variant of Brakerski's homomorphic cryptosystem, nowadays referred to as BFV, that relies on the hardness of the Ring-Learning With Errors (RLWE) problem [21]. Classified as a leveled homomorphic encryption scheme (LHE), it is currently one of the most efficient cryptosystems of its class concerning speed and memory consumption and remains untouched by recent advances in cryptanalysis [1, 16].

Let $p > 1$ be an integer and n a power-of-2. BFV's basic arithmetic is built upon polynomial rings of the form $R_p = \mathbb{Z}_p[X]/(X^n + 1)$. The scheme defines the following parameter set: a security parameter λ; a decomposition base $\omega > 1$; the modulus $t \geq 2$ that determines the plaintext domain R_t; and the modulus $q \gg t$ that determines the ciphertext domain R_q. Moreover, it makes use of an error distribution χ_{err}, usually a zero-mean discrete Gaussian distribution parameterized by the standard deviation σ.

Let $l = \lfloor \log_\omega q \rfloor$. The main procedures of BFV are the following:

`KeyGen(`λ, ω`):` Let $\texttt{sk} \leftarrow R_3$ be the secret key. Sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi_{err}$, and define the public key $\texttt{pk} = (b, a) = \left([-(a \cdot \texttt{sk} + e)]_q, a\right)$. Generate the evaluation key `evk` as: Sample $\mathbf{a}_i \leftarrow R_q$ uniformly at random, $\mathbf{e}_i \leftarrow \chi_{err}$, and compute $\gamma_i = \left([-(\mathbf{a}_i \cdot \texttt{sk} + \mathbf{e}_i) + \omega^i \cdot \texttt{sk}^2]_q, \mathbf{a}_i\right)$. Define $\texttt{evk} = \bigcup_{i=0}^{l} \gamma_i$. Output $(\texttt{sk}, \texttt{pk}, \texttt{evk})$.

`Encrypt(`m`, pk):` for a plaintext message $m \in R_t$ and a public key $\texttt{pk} = (b, a)$, sample $u \leftarrow R_2$ uniformly at random and $e_1, e_2 \leftarrow \chi_{err}$, and compute the ciphertext $\mathbf{c} = \left([\Delta m + b \cdot u + e_1]_q, [a \cdot u + e_2]_q\right)$, where $\Delta = \lfloor q/t \rfloor$.

`Decrypt(c, sk):` for a ciphertext $\mathbf{c} = (c_0, c_1)$ and the secret key $\texttt{sk} = s$, recover the plaintext $m = \left[\left\lfloor \frac{t}{q} [c_0 + c_1 \cdot s]_q \right\rfloor\right]_t$.

`Add(`$\mathbf{c}_0$`, `$\mathbf{c}_1$`)` : for ciphertexts $\mathbf{c}_0 = (c_{0,0}, c_{0,1})$ and $\mathbf{c}_1 = (c_{1,0}, c_{1,1})$, compute $\mathbf{c}_{\texttt{add}} = ([c_{0,0} + c_{1,0}]_q, [c_{0,1} + c_{1,1}]_q)$.

`Relin(`(c_0, c_1, c_2)`, evk)` : for $c_0, c_1, c_2 \in R_q$, $\texttt{evk} = (\mathbf{b}, \mathbf{a})$, and a decomposition of c_2 in base w such that $c_2 = \sum_{i=0}^{l} c_2^{(i)} w^i$, return $\left(\left[c_0 + \sum_{i=0}^{l} \mathbf{b}_i \cdot c_2^{(i)}\right]_q, \left[c_1 + \sum_{i=0}^{l} \mathbf{a}_i \cdot c_2^{(i)}\right]_q\right)$.

`Mul(`$\mathbf{c}_0$`, `$\mathbf{c}_1$`, evk)` : for ciphertexts $\mathbf{c}_0 = (c_{0,0}, c_{0,1})$ and $\mathbf{c}_1 = (c_{1,0}, c_{1,1})$, compute $c = \left(\left[\left\lfloor \frac{t}{q} \cdot c_{0,0} \cdot c_{1,0} \right\rfloor\right]_q, \left[\left\lfloor \frac{t}{q} \cdot (c_{0,0} \cdot c_{1,1} + c_{0,1} \cdot c_{1,0}) \right\rfloor\right]_q, \left[\left\lfloor \frac{t}{q} \cdot c_{0,1} \cdot c_{1,1} \right\rfloor\right]_q\right)$ and return $\mathbf{c}_{\texttt{mul}} = \texttt{Relin}(c, \texttt{evk})$.

[1] GPGPU, acronym for General-Purpose Graphics Processing Unit.

2.2 Residue Number System

As can be observed in Section 2.1, BFV depends upon computationally expensive polynomial operations. Moreover, the literature reveals that big integer arithmetic is required to offer proper security levels [28]. A common strategy in implementations of BFV is to use the Chinese Remainder Theorem (CRT) on the Residue Number System (RNS) to map large integers to a set of smaller residues capable of being evaluated by processor's native instructions [19, 9].

Definition 1 (CRT) *Let x be a polynomial in R_q, and $\{p_0, \ldots, p_{\ell-1}\}$ a set of pairwise coprimes. The CRT decomposition results in a set X with ℓ residues such that $CRT(x) = \left\{[x]_{p_0}, \ldots, [x]_{p_{\ell-1}}\right\}$. The inverse $CRT(X)$ is defined as: $\left[\sum_{i=0}^{\ell-1} \frac{M}{p_i} \cdot \left[\left(\frac{M}{p_i}\right)^{-1} X_i\right]_{p_i}\right]_M = x$, where $M = \prod_{i=0}^{\ell-1} p_i$.*

Addition and multiplication in the RNS domain work by applying the operation residue-wise. However, division and modular reduction are more complicated and require a more advanced technique, as described next.

2.3 Division and rounding inside the RNS domain

Some parts of BFV are hardly compatible with RNS, such as the coefficient-wise division and rounding used in decryption and homomorphic multiplication. Motivated by that, two variants of BFV can be found in the literature, BEHZ-BFV and HPS-BFV, which propose modifications to the cryptosystem to support them in the RNS domain [8, 24].

Let $Q = \{q_0, q_1, \ldots, q_{\ell-1}\}$ be a RNS basis which we can use to represent any ciphertext, as described in Section 2.2. BEHZ-BFV and HPS-BFV claim that the division and rounding can be computed by extending base Q to a new basis $B = \{b_0, b_1, \ldots, b_{k-1}\}$ such that $\prod q_i < \prod b_j$. While BEHZ-BFV looks for an exact rounding, HPS-BFV shows how to build operations to minimize the error and merge it into the natural cryptosystem noise. This allows a much simpler procedure, with a lower computational cost, to be used. HPS-BFV's authors present an analysis that demonstrates that their procedures are simpler and have lower complexity and noise growth than those in BEHZ-BFV.

The HPS-BFV methods are composed by a basis extension procedure, which computes a polynomial representation in a base B from its representation in base Q; and two methods to scale down and round an integer in its RNS representation by t/q, one to be used on decryption, which is a more straightforward scenario that requires the output to be in base $\{t\}$, and one for homomorphic encryption, which is a bit more complicated since the outcome must lie in base B.

Both variants of BFV take the fact that q is not defined as a prime integer. Thus, they represent and work with R_q polynomials in an RNS base composed by a factorization of q, i.e. $q = \prod_{i=0}^{\ell-1} q_i$. One of the advantages of doing this is the automatic merge of the RNS bounds with the ciphertext coefficient domain.

2.4 Discrete Galois Transform

The Fast Fourier Transform (FFT) is a well-known method that offers linear computational cost for polynomial multiplication when the operands lie in its domain and quasi-linear when considering the computation of the transform itself. However, the FFT is defined on $\mathbb{C}$, which makes it harder for its direct applicability in the context of RLWE-based cryptosystems, defined on integer domains. Thus, variations offering the same functionality but built upon integer arithmetic were proposed in the literature, such as the Number Theoretic Transform (NTT) over $GF(p)$, and the Discrete Galois Transform (DGT) over $GF(p^2)$, for some convenient choice of a prime number p [26, 17].

The main difference of DGT over NTT is caused by their domains, which results in memory bandwidth savings, as deeply discussed in Sections 3 and 4. Despite this, they are sufficiently similar so that they share most of the computation data paths and their efficient implementation strategies. Furthermore, as $GF(p^2)$ can be represented in the set of Gaussian integers $\mathbb{Z}_p[i] = \{a + ib \mid a, b \in \mathbb{Z}_p\}$, it uses finite field arithmetic with $\mathbb{Z}_p$ elements as building blocks, which resonates with the representation used by RNS and BFV. In Definition 2 we introduce the base formulation, as done in by Badawi *et al.*[5].

Definition 2 (Discrete Galois Transform) *Let $p \geq 3$ be a prime number, $x = \{x_0, \ldots, x_{n-1}\}$ be a vector of length n such that $x_k \in GF(p^2)$ for $0 \leq k < n$, and g be an n-th primitive root of unity in $GF(p)$. Then, the DGT and its inverse are defined as: $X_k = \sum_{j=0}^{n-1} x_j g^{-jk} \in GF(p^2)$ and $x_k = n^{-1} \sum_{j=0}^{n-1} X_j g^{jk} \in GF(p^2)$, respectively.*

3 Efficient CUDA operation on cyclotomic rings

An efficient implementation of the arithmetic of cyclotomic polynomial rings requires a convenient approach for polynomial multiplication and a proper data representation, not only with low computational complexity but also that fits well in the processing hardware. This Section provides optimization strategies for implementing polynomial arithmetic on CUDA.

3.1 Fast polynomial multiplication

The complexity to compute a polynomial multiplication using a textbook formula is $\Theta\left(n^2\right)$ for n-degree polynomials, which means that performance will be seriously affected with the increase of the degree.

In the context of cryptosystems based on RLWE, as observed by Lindner and Peikert, security is strongly related to the degree of the polynomial ring [25]. Specifically on BFV, Player concludes that a parameter set nowadays considered secure, with an estimated security upper bound close to $\lambda = 128$, requires n between 2^{11} and 2^{15} [28]. Hence, an efficient implementation of polynomial multiplication for operands with a large degree is vital for performance.

FFT-based transforms, such as the NTT, provide a domain in which the polynomial multiplication complexity is reduced to $\Theta\left(n\right)$, and among those, the DGT is a promising variant defined over $GF(p^2)$. As introduced in Section 2.4, this field can be represented as the set of Gaussian integers $\mathbb{Z}_p[i] = \{a + ib \mid a, b \in \mathbb{Z}_p\}$, which enables the polynomial folding of inputs and consequently halves their degree. This folding works such that, for a polynomial $P(x) = \sum_{j=0}^{n-1} a_j \cdot x^j$, we have $fold(P(x)) = \sum_{j=0}^{n/2-1} (a_j + i \cdot a_{j+n/2}) \cdot x^j$, for $i = \sqrt{-1}$ and n even.

Considering the use of Gaussian integer arithmetic [3], a first impression may be that the increased cost of the arithmetic nullifies the reduction of the polynomial degree due to the quadratic extension. However, it is important to notice that, by working with half the coefficients, only half the roots, like those in Definition 2, are required compared to the FFT or NTT. In this way, in a memory-constrained scenario, this property implies a speedup caused by fewer memory transactions and enables a more coalesced pattern. In the case of CUDA, such operations may target the GPU's global memory, which is significant in size but has high latency, or even shared or constant memories, which are fast but very small. The resulting increased arithmetic density favors GPU implementations.

Badawi *et al.*propose Algorithm 1 for polynomial multiplication through the DGT. It first folds both input signals and then applies a twisting by powers of $n/2$-th primitive roots of i, which provides a negacyclic convolution. This equips the algorithm with a free polynomial reduction by a cyclotomic polynomial [17]. Finding these roots is a complex computational task usually performed by brute force when p is sufficiently small. Otherwise, numerical methods may be used. We offer in Appendix B a suggestion for their construction.

Algorithm 1: Polynomial multiplication in $\mathbb{Z}_p[x]/(x^n+1)$ via DGT

Input: Polynomials $a, b \in \mathbb{Z}_p[x]/(x^n+1)$, p a prime number, n a power-of-two integer, and h a primitive $\frac{n}{2}$-th root of i modulo p.
Output: $c = a \cdot b \in \mathbb{Z}_p[x]/(x^n+1)$.

1 **for** $j = 0; j < n/2; j = j+1$ **do**
2 $\quad a'_j = a_j + ia_{j+n/2}$ `// Folding the input polynomials`
3 $\quad b'_j = b_j + ib_{j+n/2}$
4 **for** $j = 0; j < n/2; j = j+1$ **do**
5 $\quad a'_j = h^j \cdot a'_j \pmod p$ `// Applying the right-angle convolution`
6 $\quad b'_j = h^j \cdot b'_j \pmod p$
7 $a' = \mathrm{DGT}(a')$ `// Computing the DGT of both operands`
8 $b' = \mathrm{DGT}(b')$
9 **for** $j = 0; j < n/2; j = j+1$ **do**
10 $\quad c'_j = a'_j \cdot b'_j \pmod p$ `// Component-wise multiplying in` $\mathbb{Z}_p[i]$
11 $d' = \mathrm{IDGT}(c')$ `// Computing the IDGT of the multiplication result`
12 **for** $j = 0; j < n/2; j = j+1$ **do**
13 $\quad u = h^{-j} \cdot d'_j \pmod p$ `// Removing the twisting factors`
14 $\quad c_j = u_{re}$ `// Unfolding the result`
15 $\quad c_{j+\frac{n}{2}} = u_{im}$
16 **return** c

There is no need for the bit-reversal procedure in the context of implementing a polynomial multiplication. Thus, an efficient implementation avoids it by selecting a decimation-in-frequency (DIF) algorithm for the forward transform and a decimation-in-time (DIT) algorithm for the inverse, as defined by Chu and George [15]. In this work, we follow the proposal of Badawi *et al.*and choose the Gentleman-Sande, a DIF, and the Cooley-Tukey, a DIT, data-paths for the forward and inverse versions of the DGT, respectively [5].

The canonical formulation of these contains a combination of three nested loops, which increases the complexity of its implementation, especially on the CUDA architecture. This structure creates dependencies between the loops and disturbs parallel execution. So, for better compatibility with the programming model, they have to be rewritten by wiping out one layer of nesting and leaving only two loops, an outer loop related to the stride and an inner loop that asserts the access patterns. For each outer loop iteration, the inner one can be completely parallelized. Our proposals have a much weaker dependency between iterations and can be seen in Algorithms 2 and 3.

3.2 An improved and hierarchical DGT

The procedures described in Algorithms 2 and 3 require synchronization at the end of each iteration of the outer loop. On CUDA, this enforces a limitation on the polynomial degree at the cost of latency, since the only data structure that provides such synchronicity at the hardware level is a Thread Block, and its dimension is limited to 1024 threads in modern hardware. An alternative implementation involves calling a different CUDA kernel for each iteration, forcing a CPU-sided synchronization. This incurs a considerable overhead caused by several kernel calls.

In this scenario, we propose a technique for splitting the DGT transform into smaller blocks that better fit the processing hardware and does not require synchronizing large sets of threads, called hierarchical DGT. It is an adaptation of the four-step FFT algorithm, initially proposed by David H. Bailey and later on revisited by Govindaraju *et al.* [7, 23].

Algorithm 2: Rewritten forward DGT via Gentleman-Sande

Input: A folded vector $x \in \mathbb{Z}[i]^k$, p a prime number, k a power-of-two integer, and g a primitive k-th root of unity modulo p.
Output: $x \leftarrow \mathrm{DGT}(x)$ in bit-reversed ordering.

for $s = 0; s < \lfloor \log(k) \rfloor; s = s + 1$ **do**
  $m = \frac{k}{2^{(s+1)}}$
  for $l = 0; l < k/2; l = l + 1$ **do**
    $j = \frac{2ml}{k}$
    $i = j + \left(l \mod \frac{k}{2m}\right) \cdot 2m$
    $a = g^{j \cdot \frac{k}{2^{(\log(k)-s)}}} \pmod p$
    $(u, v) = (x[i], x[i+m])$
    $(x[i], x[i+m]) = (u + v, a \cdot (u - v)) \pmod p$
return x

Algorithm 3: Rewritten inverse DGT via Cooley-Tukey

Input: A vector $x \in \mathbb{Z}[i]^k$ in bit-reversed order, p a prime number, k a power-of-two integer, and g a primitive k-th root of unity modulo p.
Output: $x \leftarrow k \cdot \mathrm{IDGT}(x)$ in standard ordering.

$m = 1$
for $s = 0; s < \lfloor \log(k) \rfloor; s = s + 1$ **do**
  for $l = 0; l < k/2; l = l + 1$ **do**
    $j = \frac{2ml}{k}$
    $i = j + \left(l \mod \frac{k}{2m}\right) \cdot 2m$
    $a = g^{-j \cdot \frac{k}{2^{s+1}}} \pmod p$
    $(u, v) = (x[i], x[i+m])$
    $(x[i], x[i+m]) = (u + a \cdot v, u - a \cdot v) \pmod p$
  $m = 2 \cdot m$
return x

The general idea of the hierarchical DGT and hierarchical inverse DGT, referred to respectively as HDGT and HIDGT, is to split the DGT computation over $\mathbb{Z}_p[x]/(x^n + 1)$ into computations in smaller rings with optimal degree near $\sqrt{n}$. In practice, the vector of coefficients is treated as a matrix and the DGT is performed over the columns and rows of this matrix. The objective of this is to avoid the case in which one is unable to compute the DGT of an entire polynomial in a single CUDA kernel call. We move to a higher granularity approach in which we apply the transform multiple times over arbitrary small polynomials that can perfectly fit in our processing architecture.

The HDGT is described in Algorithm 4. Firstly, the polynomial $a(x)$ is represented by taking its coefficient embedding as $a = (a_0, a_1, \ldots, a_{n-1})$. To be represented in the DGT domain $GF(p^2)$, $a \in \mathbb{Z}_p^n$ is folded as a $(n/2)$-size vector of Gaussian integers $\tilde{a} \in \mathbb{Z}_p[i]^{n/2}$, as described in Section 3.1. In the Algorithm, the "right-angle" convolution is given by multiplying the j-th coefficient of $\tilde{a}$ by h^j, for $j \in \mathbb{Z}_{n/2}$, where h is the $(n/2)$-th primitive root of i in $\mathbb{Z}_p[i]$.

After the folding and twisting procedures, the $(n/2)$-length vector of Gaussian integers $\tilde{a}$ is treated as a matrix with dimensions (N_r, N_c). These dimensions shall be chosen so that each coefficient's subset fits in the processing hardware. In our case, the objective is to find a subset that fits in the GPU's shared memory so that the DGT can be performed in a single Thread

Block.

Since the bit-reversal is not used in Algorithm 2, the called "step-2" of Bailey's method has to be rewritten. In line 8, the twiddle factors are the powers of g, the $(n/2)$-th root of unity modulo p. Since the output of the DGT is not corrected from the bit-reversed order, the twiddle factors become $g^{\texttt{bit-reversal}(j)\cdot k}$ instead of $g^{j\cdot k}$, which matches the position of the corresponding element in $\tilde{a}$ when it is seen as a matrix.

Algorithm 4: Hierarchical forward DGT

Input: A polynomial $a \in \mathbb{Z}_p[x]/(x^n+1)$, p a prime number, $n = 2 \cdot N_r \cdot N_c$ a power-of-two integer, h a primitive $n/2$-th root of i modulo p, and g a primitive $n/2$-th root of unity modulo p.

Output: $\tilde{a} = \text{HDGT}(a)$.

1 **for** $j = 0; j < n/2; j = j + 1$ **do**
2 $\quad \tilde{a}_j = a_j + i a_{j+n/2}$ `// Fold the input polynomial`
3 $\quad \tilde{a}_j = \tilde{a}_j \cdot h^j \pmod p$ `// Twist the folded polynomial`
4 **for** $k = 0; k < N_c; k = k + 1$ **do**
5 $\quad \tilde{a}_{\_,k} = \text{DGT}(\tilde{a}_{\_,k})$ `// Step 1: Apply the DGT through` N_c `columns`
6 **for** $j = 0; j < N_r; j = j + 1$ **do**
7 $\quad$ **for** $k = 0; k < N_c; k = k + 1$ **do**
8 $\quad\quad \tilde{a}_{j,k} = \tilde{a}_{j,k} \cdot g^{\texttt{bit-reversal}(j)\cdot k} \pmod p$ `// Step 2: Multiplication by the twiddle factors in bit-reversal order`
9 **for** $j = 0; j < N_r; j = j + 1$ **do**
10 $\quad \tilde{a}_{j,\_} = \text{DGT}(\tilde{a}_{j,\_})$ `// Step 3: Apply the DGT through the` N_r `rows`
11 **return** $\tilde{a}$

The inverse counterpart of the hierarchical DGT simply executes the inverse steps of the forward transform, and is described in Algorithm 5. It adopts the IDGT transform via Cooley-Tukey, described in Algorithm 3, without bit-reversing the input vector. The algorithm executes the inverse steps of the forward transform by first applying the IDGT over the rows of $\tilde{a}$. The twiddle factors are removed by multiplying $\hat{a}_{j,k}$ by $g^{-\texttt{bit-reversal}(j)\cdot k}$, since the column indexes of the output of the previous step still are in bit-reversed order. Considering that the powers of g can be precomputed, they can be multiplied by N_c^{-1}, avoiding the additional multiplication. Finally, the IDGT is applied over the columns of $\hat{a}$ and the matrix indexes are back to standard ordering. Following the same approach, the powers of h^{-1} can be precomputed already multiplied by the scalar N_r^{-1}. This avoids the multiplication by the scaling factor when applying the IDGT over the columns of $\hat{a}$.

As in FFT and NTT, the two operands are evaluated using the HDGT for further point-wise multiplication. The polynomial corresponding to $a \cdot b$ in $\mathbb{Z}_p[x]/(x^n+1)$ is obtained by computing the HIDGT.

3.3 Polynomial representation and memory locality

The usability of an RLWE-based cryptosystem requires the careful selection of a parameter set that satisfies all the security constraints of the application. For instance, with BFV one must select q, t, n, and σ such that a security level λ is achieved. More than that, these parameters together determine the multiplicative depth supported by the scheme. Thus, as discussed by Fan and Vercauteren, the selection of such parameters is too complex to be affected by the particularities of the implementation [21].

A constraint for choosing those is the hardware instruction set. By selecting a big q one may be confronted by the lack of hardware support for native processing of the coefficients.

Algorithm 5: Hierarchical inverse DGT

Input: $\tilde{a} = \text{HDGT}(a)$, p a prime number, $n = 2 \cdot N_r \cdot N_c$ a power-of-two integer, h a primitive $n/2$-th root of i modulo p, and g a primitive $n/2$-th root of unity modulo p.

Output: A polynomial $a \in \mathbb{Z}_p[x]/(x^n + 1)$.

1 **for** $j = 0; j < N_r; j = j + 1$ **do**
2 $\quad \hat{a}_{j,\_} = \text{IDGT}(\tilde{a}_{j,\_})$ `// Step 3: Apply IDGT to each of` N_r `rows`
3 **for** $j = 0; j < N_r; j = j + 1$ **do**
4 $\quad$ **for** $k = 0; k < N_c; k = k + 1$ **do**
5 $\quad\quad \hat{a}_{j,k} = \hat{a}_{j,k} \cdot g^{-\texttt{bit-reversal}(j)\cdot k} \cdot N_c^{-1} \pmod{p}$ `// Step 2: Remove twiddle factors`
6 **for** $k = 0; k < N_c; k = k + 1$ **do**
7 $\quad \hat{a}_{\_,k} = \text{IDGT}(\hat{a}_{\_,k})$ `// Step 1: Apply IDGT to each of` N_c `columns`
8 **for** $j = 0; j < n/2; j = j + 1$ **do**
9 $\quad \hat{a}_j = \hat{a}_j \cdot h^{-j} \cdot N_r^{-1} \pmod{p}$ `// Remove the twisting`
10 $\quad a_j = \hat{a}_{j_{re}}$ `// Unfold the output polynomial`
11 $\quad a_{j+\frac{n}{2}} = \hat{a}_{j_{im}}$
12 **return** a

Through RNS, as described in Section 2.2, we handle this by splitting big integers in small residues following the limits of the underlying machine.

The link between the cryptosystem and RNS must be carefully designed so that data secrecy is provided with suitable performance. For that, Gentry *et al.*suggested the *double*-CRT representation, which encapsulates data into two layers [22]. The first layer is the RNS representation, as described in Definition 1. After that, a set of polynomial residues with full support for native hardware evaluation of addition and multiplication is obtained. However, we still need a second layer for the latter, since the multiplication of polynomials can achieve a quite high computational complexity without some well-designed algorithm, as discussed in Section 3.1. Because of that, the second layer consists of moving each residue, individually, to a different domain with a convenient property for efficient polynomial multiplication. The original proposal of *double*-CRT is the use of the NTT as this transform, but a similar approach using the FFT would also be expected. This work, however, proposes that the second layer of the *double*-CRT should use the DGT instead of the NTT since the former appears to suit much better the cyclotomic ring arithmetic in GPUs and presents more efficient memory access patterns [5].

Another design decision, widespread to HE implementations, is the selection of a single special prime p for the application of the transform over all RNS residues [18, 20]. For instance, let x be a polynomial and $\{q_0, \ldots, q_{\ell-1}\}$ a set of ℓ pairwise coprimes, then $\left\{\text{DGT}_p([x]_{q_0}), \ldots, \text{DGT}_p([x]_{q_{\ell-1}})\right\}$ is the set of transformed residues. By using such a prime, one is capable of taking advantage of their intrinsic mathematical properties, as in the selection of a Mersenne or Solinas prime, which enables the use of a very efficient modular reduction. Nonetheless, this approach does not interplay well with the RNS layer and requires algorithmic efforts to correct these modular reductions and keep consistency for each residue. In this way, we propose a simpler solution by computing the transform layer using the coprime related to each residue, at the cost of a more expensive modular reduction since, in most cases, there are not enough special primes for the required number of residues. Thus, in this representation, the set of residues becomes $\left\{\text{DGT}_{q_0}([x]_{q_0}), \ldots, \text{DGT}_{q_{\ell-1}}([x]_{q_{\ell-1}})\right\}$. Moreover, without the need for those corrections, we become capable of increasing RNS' residues to the biggest supported word size of the target architecture, reducing the number of residues needed. By choosing $q = \prod_{i=0}^{\ell-1} q_i$ we establish a bond between BFV, RNS, and the DGT.

Lastly, our state machine proposal targets the insistent maintenance of data in our version of the *double*-CRT representation in GPU's memory. Data copy between the main memory and the GPU's memory has high latency and must be avoided.

4 Experimental results

In this Section we present SPOG[2], a proof-of-concept implementation that consolidates the aforementioned techniques by exploring parallel processing on GPGPUs through CUDA.

Designed from scratch, SPOG is a modular implementation in which the arithmetic operations are separated from the cryptosystem. The polynomial operations were implemented on a sister library named cuPoly, while BFV was implemented separated on SPOG-BFV. Both are based on CUDA and closely follow the sketch provided in Section 3, pursuing low-latency methods with a simple API and stretching the size of the residues to the highest supported by modern CUDA-supported GPUs, which is 63-bit residues with 1 bit for storing the sign. By doing this, we guarantee that BFV can be easily replaced by any other scheme based on the RLWE; thus, our work is not restricted to a single scheme. The entire arithmetic implementation can also be replaced without affecting the cryptosystem code. Hence, SPOG is flexible enough to encourage future work to develop and test different setups using the presented libraries.

cuRAND, a NVIDIA probabilistic library, was used for the sampling required by the BFV. This library offers sampling directly to the GPU memory, avoiding the cost of data copy. Sampling uniformly at random from R_x is implemented through its uniform sampler and the result is reduced by x. On the other hand,the discrete Gaussian distribution is not supported by this library. Because of that, an alternative implementation works by truncating a normal distribution, natively supported by cuRAND. The statistical validity of this design still needs to be asserted at the cost of compromising the security. Moreover, to the best of our knowledge, cuRAND lacks sufficient scrutiny by the scientific community so that it can be seen as cryptographic secure. However, this is a common implementation decision in the literature and is also done by the related works cited in Section 4.1.

SPOG and cuPoly source code are available to the community under a *GNU GPLv3* license [2].

4.1 Related work

We consider Badawi, Polyakov, Aung, Veeravalli, and Rohloff, work, referred as BPAVR, the state-of-the-art implementation in GPUs for BFV [4]. It complements Halevi, Polyakov, and Shoup proposal and provides the first implementation of the HPS-BFV method on a high-end NVIDIA Tesla V100 GPU, demonstrated by the authors to be the fastest and most scalable variant of the scheme when compared to BEHZ-BFV [24, 8].

BPAVR do not describe all details regarding their performance results, only presenting latency measurements for decryption and homomorphic multiplication. Because of that, and the fact of their source code is not publicly available, we also consider a similar work of Badawi, Veeravalli, Mun, and Aung, which offers timings for encryption, decryption, homomorphic addition, and homomorphic multiplication for a CUDA-based BFV implementation, denoted by BVMA[6]. The authors compare BVMA with Microsoft SEAL, a reference on the field with support for HPS-BFV [12]; and NFLlib-FV, an equally important work implementing the BEHZ-BFV variant; with impressive speedups on all scenarios [27]. Despite of their efforts for parallel computation, the other libraries presented in that work are CPU-based implementations and thus show a significant slowdown, up to 27 times, when compared to BVMA. Hence, we do not believe that the direct comparison with SPOG is relevant to this paper.

[2]SPOG, acronym for "Secure Processing on GPGPUs".

Lastly, both works apply the DGT as the underlying solution to handle polynomial multiplication. So, by comparing SPOG with them, we can collect evidence about the suitability of the HDGT over the DGT for such task.

4.2 Execution environment, methodology, and BFV parameters

The experimental results presented in the next Sections for BPAVR or BVMA are those reported by the authors in their corresponding publications. We do not re-execute the benchmarks provided in the related work. This decision is based on the fact that the implementations and benchmarking tools were not made available to the community. Because of that, we decided to collect our measurements in a similar processing hardware adopted in the related works using the same parameters.

We used Google Cloud's virtual machines (VMs) for emulating the computational environment described in those works. Two instances were considered: *gc.k80* and *gc.v100*, which provide a NVIDIA Tesla K80 GPU, used on BVMA measurements; and a NVIDIA Tesla V100 GPU, used on BPAVR. We precisely followed the execution environment described in each work, running GCC 7.2.1 and CUDA 8.0 at *gc.k80*; and GCC 7.3.1 and CUDA 9.0 at *gc.v100*. CUDA events were used to measure execution time, following the common methodology from the literature.

Our benchmark targets the most relevant primitives for HE. Regarding BFV, implemented in SPOG, we consider encryption, decryption, homomorphic addition, and homomorphic multiplication (including the relinearization cost). On the polynomial arithmetic side, implemented in cuPoly, we focus on the performance gains caused by the replacement of the canonical DGT by the HDGT.

In our measurements, we do not include initialization steps, which are performed only once and have negligible effect on long term runs. Because of that, the latency for generating cryptographic keys is not described in this work. Similarly, sampling is not explicitly considered in the benchmarks, despite of being included in the timings for encryption.

Two different setups are considered for compatibility with each work, both choosing $t = 256$ for the plaintext domain.

BPAVR parameters: Different polynomial ring settings are used identified by the pairs $(\lceil \log(q) \rceil, \log(n)) \in \{(60, 11), (60, 12), (120, 13), (360, 14), (600, 15)\}$ for the ciphertext coefficient domain and the ring degree, respectively. These offer a security level of at least 128 bits [4].

BVMA parameters: Different polynomial ring settings are used identified by the pairs $(\lceil \log(q) \rceil, \log(n)) \in \{(62, 11), (186, 12), (372, 13), (744, 14), (744, 15)\}$ for the ciphertext coefficient domain and the ring degree, respectively. These offer a security level of 80 bits [6].

4.3 Memory consumption

Let $\hat{q}$ and $\hat{b}$ be the main and auxiliary RNS bases used to represent elements of R_q and used by the HPS-BFV methods described in Section 2.3, respectively; and $\texttt{nres}_{qb}$ the quantity of elements in $\hat{q} \cup \hat{b}$. A BFV ciphertext on SPOG is composed by two N-degree polynomials represented as $\texttt{nres}_{qb}$ residues with 63-bits coefficients, thus requiring $s(N, \texttt{nres}_{qb}) \coloneqq 63 \cdot (2 \cdot N \cdot \texttt{nres}_{qb})$ bits for storage.

The ciphertext expansion factor, however, depends also on its slot occupancy. Through batching, a single ciphertext can store up to N integer plaintexts [11]. Hence, the expansion factor is given by $\frac{s(N, \texttt{nres}_{qb})}{63 \cdot \texttt{batch\_size}}$.

4.4 SPOG operations

In Table 1 we compare SPOG with BVMA on *gc.k80*, and with BPAVR on *gc.v100*. As mentioned in Section 4.1, The authors of BPAVR offer measurements for decryption and homomorphic multiplication only, what inhibits the comparison with SPOG for encryption and homomorphic addition.

One of the major motivations for using a FHE scheme is the applicability of its homomorphic primitives, and because of that, we focus on improving the performance of these. As can be seen, homomorphic multiplication, a critical and known expensive operation, reports speedup between 2.0 and 3.6 times when compared to the BVMA. When compared to the BPAVR these speedups lies between 2 and 2.4. The different characteristics between both setups, considering the processing hardware and the cryptosystem parameters, makes the direct comparison between both data sets impossible, however the performance gains are consistent.

Homomorphic addition, a much simpler operation, presented gains between 2 and 5.2 times when compared to the BVMA. The latter is probably not related to the HDGT, since this procedure is essentially a coefficient-wise addition, but to the better state machine, as described in Section 3.3.

Despite our focus in this work does not being on encryption and decryption, the faster polynomial multiplication strategy and the improved state machine offered up to 4.6 times faster encryption and about 2 times faster decryption.

Table 1: Comparison between SPOG and two state-of-the-art implementations, BVMA and BPAVR. Average running time of 100 independent executions, in milliseconds, for the most relevant BFV operations for the setups described in Section 4.2.

		gc.k80				*gc.v100*				
	$\log n$	11	12	13	14	$\log n$	12	13	14	15
Encrypt	SPOG	0.303	0.309	0.575	1.630	-	-	-	-	-
	BVMA	0.541	1.440	2.645	6.657	-	-	-	-	-
	Ratio	**1.785**	**4.660**	**4.600**	**4.084**	-	-	-	-	-
Decrypt	SPOG	0.089	0.098	0.191	0.542	SPOG	0.029	0.031	0.049	0.099
	BVMA	0.151	0.194	0.252	0.610	BPAVR	0.054	0.059	0.087	0.111
	Ratio	**1.697**	**1.980**	**1.319**	**1.125**	Ratio	**1.862**	**1.903**	**1.776**	**1.121**
Hom. Add.	SPOG	0.009	0.010	0.021	0.066	-	-	-	-	-
	BVMA	0.037	0.052	0.068	0.127	-	-	-	-	-
	Ratio	**4.111**	**5.200**	**3.238**	**1.924**	-	-	-	-	-
Hom. Mul.	SPOG	0.926	1.214	3.061	13.914	SPOG	0.423	0.472	0.823	2.325
	BVMA	3.343	3.873	7.700	28.953	BPAVR	0.859	1.012	2.010	4.826
	Ratio	**3.610**	**3.190**	**2.516**	**2.081**	Ratio	**2.031**	**2.144**	**2.442**	**2.076**

4.5 Efficiency of the HDGT

A major contribution of this work is the HDGT, a novel formulation of the DGT which better explores the parallel capability of GPUs and compensate its memory limitations. However, a carefully evaluation of its quality must be done to understand the performance gains on realistic scenarios. Thus, at this Section, we provide a comparison between the HDGT and the best implementation designs for the canonical DGT.

As discussed before, the HDGT works by splitting a high-degree polynomial, which does not fit in the processing hardware, and applying the DGT in a divide-and-conquer approach through blocks of arbitrarily small size. To evaluate this design, we implemented the canonical DGT adopting two different strategies, namely DGT-I and DGT-II. The former uses a multi-kernel

design which executes the loop synchronization employing a different CUDA kernel for each iteration. This way, the transformation requires $\log \frac{n}{2}$ kernels to process an n-degree polynomial. The latter uses a single-kernel design, which is only compatible with polynomial rings with degree smaller or equal than 4096 since these are the only that fit GPU's shared memory. These strategies are better described in Section 3.2. We verified the impact of this change in two important procedures direct affected by the DGT, encryption and homomorphic multiplication.

Table 2 presents the latency measurements. The HDGT is about 2 times faster than the DGT-I, which results in speedups ranging from 1.4 to 2.2 times on BFV's primitives. The DGT-II, though, presents a slowdown in most cases. This relates to the need for serialization within HDGT's steps, which was implemented by splitting the algorithm into 4 sequential kernels. DGT-II is always executed by a single kernel, implying a much smaller overhead. This suggests that the single-kernel design better accommodates smaller instances. Such effect doesn't sustain on *gc.v100* that better handles the high-granularity of the HDGT. Unfortunately, DGT-II is not scalable to bigger rings.

Table 2: Comparison between SPOG running the canonical DGT using a multi-kernel and a single-kernel strategy, called DGT-I and DGT-II, respectively; and the HDGT. The first row group compares the transform alone. Average running time of 100 independent executions, in milliseconds, for the setups described in Section 4.2.

		gc.k80					*gc.v100*				
	$\log n$	11	12	13	14	15	11	12	13	14	15
DGT	HDGT	0.059	0.071	0.146	0.432	0.651	0.018	0.019	0.020	0.031	0.073
	DGT-I	0.114	0.131	0.281	0.711	1.637	0.035	0.034	0.040	0.078	0.188
	Ratio	**1.934**	**1.864**	**1.925**	**1.644**	**2.517**	**1.934**	**1.815**	**2.040**	**2.487**	**2.593**
	DGT-II	0.052	0.091	-	-	-	0.026	0.047	-	-	-
	Ratio	0.881	**1.292**	-	-	-	**1.423**	**2.492**	-	-	-
Encrypt	HDGT	0.303	0.309	0.575	1.630	3.127	0.103	0.098	0.099	0.153	0.315
	DGT-I	0.571	0.499	0.861	2.597	5.835	0.144	0.146	0.159	0.287	0.704
	Ratio	**1.882**	**1.614**	**1.499**	**1.593**	**1.866**	**1.395**	**1.498**	**1.615**	**1.883**	**2.238**
	DGT-II	0.276	0.377	-	-	-	0.120	0.188	-	-	-
	Ratio	0.910	**1.220**	-	-	-	**1.163**	**1.921**	-	-	-
Hom. Mult.	HDGT	0.926	1.214	3.061	13.914	28.990	0.436	0.423	0.472	0.823	2.325
	DGT-I	1.795	2.031	4.231	19.952	42.800	0.795	0.783	0.913	1.609	4.078
	Ratio	**1.938**	**1.673**	**1.382**	**1.434**	**1.476**	**1.825**	**1.850**	**1.934**	**1.956**	**1.754**
	DGT-II	0.642	0.983	-	-	-	0.362	0.466	-	-	-
	Ratio	0.693	0.810	-	-	-	0.830	**1.102**	-	-	-

5 Conclusion

This work investigates strategies to achieve an efficient implementation of the leveled homomorphic encryption scheme BFV on the CUDA architecture. To fulfill this objective, we explored different approaches for the utilization of the DGT in the reduction of the computational complexity of polynomial multiplications. The outcome is an optimized version of the hierarchical DGT, a high granularity implementation of DGT that better fits the GPU processing. Furthermore, the *double*-CRT concept is revisited and an efficient state machine is proposed so we can avoid the costs to alternate between DGT and RNS domains, and between the machine's main memory and GPU's memory.

Our implementation of BFV, named SPOG, is compared with two other works in the literature, BVMA and BPAVR, that represent the state-of-the-art implementations on CUDA.

Homomorphic addition, in spite of being a simple and usually fast operation, presented speedups between 2 and 5.2 times over the BVMA. Furthermore, SPOG's homomorphic multiplication showed itself between 2.0 and 3.6 times faster over the BVMA.

As future work, we intend to verify the gains of applying our methods on other relevant RLWE-based cryptosystems such as the CKKS [13], and SPOG as a tool for the acceleration of privacy-focused deep learning algorithms.

Acknowledgements

This work was supported in part by CNPq, grants number 164489/2018-5, 203175/2019-0, and 44265/2019-2; and CAPES grant number 1591123. We specially thank LG for financial support within project "*Privacy-preserving analytics*", project number 5296; Google for GCP Research Credits Program under number 106101194491; and the Concordium Blockchain Research Center at Aarhus University, Denmark.

References

[1] Martin Albrecht, Shi Bai, and Léo Ducas. A Subfield Lattice Attack on Overstretched NTRU Assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 153–178, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[2] Pedro Alves. SPOG: Secure processing on GPGPUs. `https://github.com/spog-library`, 2021.

[3] Pedro Alves, Jheyne N. Ortiz, and Diego F. Aranha. Faster Homomorphic Encryption over GPGPUs via hierarchical DGT. Cryptology ePrint Archive, Report 2020/861, 2020. `https://eprint.iacr.org/2020/861`.

[4] Ahmad Al Badawi, Yuriy Polyakov, Khin Mi Mi Aung, Bharadwaj Veeravalli, and Kurt Rohloff. Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme. *IACR Cryptol. ePrint Arch.*, 2018:589, 2018.

[5] Ahmad Qaisar Al Badawi, Bharadwaj Veeravalli, and Khin Mi Mi Aung. Efficient Polynomial Multiplication via Modified Discrete Galois Transform and Negacyclic Convolution. In *AISC*, volume 886, pages 666–682, Cham, 2019. Springer.

[6] Ahmad Qaisar Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun, and Khin Mi Mi Aung. High-Performance FV Somewhat Homomorphic Encryption on GPUs: An Implementation using GPUs. *TCHES*, 1(2):70–95, 2018.

[7] David H. Bailey. FFTs in external or hierarchical memory. *J. Supercomput.*, 4(1):23–35, 1990.

[8] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *SAC*, volume 10532 of *Lecture Notes in Computer Science*, pages 423–442. Springer, 2016.

[9] Jean-Claude Jc Bajard, Nicolas Meloni, and Thomas Plantard. Efficient RNS bases for Cryptography. *IMACS World Congress: Scientific Computation, Applied Mathematics and Simulation*, 2005.

[10] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, 2014.

[11] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin E. Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2018:462, 2018.

[12] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library - SEAL v2.1. *IACR Cryptol. ePrint Arch.*, 2017:224, 2017.

[13] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.

[14] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.*, 33(1):34–91, 2020.

[15] Eleanor Chu and Alan George. *Inside the FFT black box: serial and parallel fast Fourier transform algorithms*. CRC press, 1999.

[16] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In *CT-RSA*, volume 9610 of *Lecture Notes in Computer Science*, pages 325–340. Springer, 2016.

[17] Richard E. Crandall. Integer convolution via split-radix fast Galois transform. *Center for Advanced Computation Reed College*, 1999.

[18] Wei Dai and Berk Sunar. cuHE: A Homomorphic Encryption Accelerator Library. In *BalkanCryptSec*, volume 9540 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 2015.

[19] Cunsheng Ding, Dingyi Pei, and Arto Salomaa. *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific, 1996.

[20] Niall Emmart and Charles C. Weems. High precision integer multiplication with a GPU using strassen's algorithm with multiple FFT sizes. *Parallel Process. Lett.*, 21(3):359–375, 2011.

[21] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[22] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[23] Naga K. Govindaraju, Brandon Lloyd, Yuri Dotsenko, Burton Smith, and John Manferdelli. High performance discrete fourier transforms on graphics processors. In *SC*, page 2. IEEE/ACM, 2008.

[24] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. In *CT-RSA*, volume 11405 of *Lecture Notes in Computer Science*, pages 83–105. Springer, 2019.

[25] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

[26] Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *CANS*, volume 10052 of *Lecture Notes in Computer Science*, pages 124–139, 2016.

[27] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. NFLlib: NTT-Based Fast Lattice Library. In *CT-RSA*, volume 9610 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 2016.

[28] Rachel Player. *Parameter selection in lattice-based cryptography.* PhD thesis, PhD thesis, Royal Holloway, University of London, 2018.

[29] Thales. 2019 Thales Data Threat Report. https://go.thalesesecurity.com/rs/480-LWA-970/images/2019-DTR-Global-USL-Web.pdf, USA, 2019.

[30] Christian Wuthrich. Further Number Theory. `https://www.maths.nottingham.ac.uk/plp/pmzcw/download/fnt_chap5.pdf`, 2011. Last accessed: 2020/06/18.

A Properties of Gaussian integers

This Appendix presents important properties of Gaussian integers and useful results that can be applied on their implementation. In the following, we recall some important properties stated by Wuthrich that are useful to this work [30].

Definition 3 (Norm) *The norm of a Gaussian integer is defined as its product with its conjugate*[3]*. That is,* $N(a+ib) = (a+ib)\cdot(a-ib) = a^2+b^2$*, so* $N(\alpha) = \alpha\cdot\overline{\alpha}$*.*

Proposition 1 (Wuthrich's Proposition 5.7) *For each prime number* $p \equiv 1 \mod 4$ *there are exactly two Gaussian primes* π *and* $\overline{\pi}$ *of norm* p*.*

Lemma 1 (Wuthrich's Lemma 5.4) *If* $\pi \in \mathbb{Z}[i]$ *is such that* $N(\pi)$ *is a prime number, then* π *is a Gaussian prime.*

Lemma 2 (Wuthrich's Lemma 5.6) *Let* p *be a prime number with* $p \equiv 1 \mod 4$*. Then there exists a Gaussian prime* π *such that* $p = \pi.\overline{\pi}$*.*

Lemma 3 (Wuthrich's Lemma 5.10) *Any prime* $p \equiv 1 \mod 4$ *can be written as a sum of two squares. This is a manifestation of Fermat's theorem on sums of two squares.*

From Lemma 2 and Proposition 1, if p is prime such that $p \equiv 1 \mod 4$, then we know that it can be factored as a product of exactly two Gaussian primes that are the conjugate of each other. Lemma 3 is a direct consequence since we know that a prime $p \equiv 1 \mod 4$ can be factored as $p = \pi\cdot\overline{\pi}$ and, assuming that $\pi = a+bi$, we obtain that $\pi\cdot\overline{\pi} = a^2+b^2$.

B Generating k-th primitive roots of i modulo p

The use of the DGT for polynomial multiplication in a cyclotomic polynomial ring requires the computation of a k-th root of i modulo a prime p, discussed in Section 3.1. This element is used for achieving a cyclotomic polynomial reduction for free when n is a power of two. When p is a Mersenne prime, the literature presents efficient analytic methods; for other choices of p, the best option still is a trial-and-error approach.

Badawi *et al.*state that a naive implementation of such approach takes 156 hours to find a 2^{14}-th primitive root of i for $p = 2^{64} - 2^{32} + 1$ [5]. Because of that, they propose a more efficient strategy, when $p \equiv 1 \mod 4$, by factoring p in two Gaussian primes, namely f_0 and f_1. This decomposition of p is quite simple and relies on Lemma 2 and Proposition 1.

[3]Let $x = a+ib$ be a Gaussian integer. If y is x's conjugate then $y = a-ib$.

Algorithm 6: `decompose_in_gaussian_primes`: Decomposes a prime.

Input: A prime p
Output: Gaussian integers f_0 and f_1 such that $f_0 \cdot f_1 = p$

do
  $n = \texttt{sample}(\mathbb{Z}_p)$
while $n^{(p-1)/2} \not\equiv -1 \mod p$
$k = n^{(p-1)/4} \mod p$
$u = \texttt{gcd}(p, k+i)$
return $(f_0, f_1) = (u, \overline{u})$

Algorithm 6 starts from the Fermat's Little Theorem, which states that if p is a prime then $n^{p-1} \equiv 1 \mod p$ for all $n \in \mathbb{Z}_p$. Hence, the square root of that must be equivalent to either 1 or -1. In the latter case, we can find a number k^2 such that $k \equiv n^{(p-1)/4} \equiv i \mod p$. In other words, if $k^2 \equiv -1 \mod p$ then $k^2 + 1 \equiv 0 \mod p$ and p divides $k^2 + 1$. Since $k^2 + 1$ factors in $(k+i) \cdot (k-i)$, we found a factorization of p.

At this point, there is no guarantee that $k + i$ is a Gaussian prime. By Lemma 4, we find that the greatest common divisor of p and $k + i$ is either $k + i$ or that there exists some u such that $u \mid p$ and $u \mid k + i$. Thus, since $u = \texttt{gcd}(p, k + i)$ results in a Gaussian prime, we take it as the first factor of p. From Lemma 2, $\overline{u}$ is the second factor.

Lemma 4 *Let p be an odd prime such that $p \equiv 1 \mod 4$ and $k \in \mathbb{Z}_p$. The greatest common divisor of p and $k + i$ is $k + i$ or a Gaussian prime u such that $u \mid p$ and $u \mid k + i$.*

Proof 1 *By the Fermat's theorem on sums of two squares, we have that an odd prime p can be expressed as $p = x^2 + y^2$, with $x, y \in \mathbb{Z}$, if, and only if, $p \equiv 1 \mod 4$. Since $x^2 + y^2 = (x + iy)(x - iy)$ and $N(x + iy) = N(x - iy) = p$, then $x + iy$ and $x - iy$ are Gaussian primes and $p = (x + iy)(x - iy)$ is the unique factorization of p in $\mathbb{Z}[i]$, not considering the order of the factors*[4]*.*

On the other hand, we have that $(k + i)(k - i) \equiv p \mod p$, by construction. Combining the two facts, we obtain that $p = (x + iy)(x - iy) \equiv (k + i)(k - i)$, which is equivalent to $(k + i)(k - i) = \ell(x + iy)(x - iy)$, for some $\ell \in \mathbb{Z}$.

When $\ell = 1$, we have an equality and we find that $(k + i)$ and $(k - i)$ are indeed the factors of p. When $\ell \neq 1$, $(k + i)$ is not a Gaussian prime and still can be factored in $\mathbb{Z}[i]$; otherwise, it would be a factor of p. We know that p divides $(k + i)(k - i)$ but not $k + i$, or its conjugate, since $k < p$ and $(k + i)/p$ is not a Gaussian integer. Then, $k + i$ and p must share a common factor u that can be found as the greatest common divisor. Since the two factors of p are $x + iy$ and $x + iy$, u must be one of them.

Finally, the factors of p can be found by computing the greatest common divisor of p and $k+i$ and then computing its conjugate. Since $p = x^2 + y^2$ and $N(x + iy) = N(x - iy) = x^2 + y^2$, by Lemma 1, the factors are Gaussian primes.

Given a method for factoring a prime number $p \equiv 1 \mod 4$ in $\mathbb{Z}[i]$, Badawi *et al.*propose Algorithm 7, which makes much faster the step of precomputing a k-th root of i for a prime $p \equiv 1 \mod 4$ [5]. The method starts by finding the factorization $p = f_0 \cdot f_1 \in \mathbb{Z}_p[i]$ using the Algorithm 6. Thus, we have that each Gaussian prime f_j, with $j = \{0, 1\}$, defines a cyclic group corresponding to the set of Gaussian integers modulo f_j. Then, a k-th root of i modulo p, denoted as h, is constructed via CRT using that $h_j = \zeta_j^{\frac{(p-1)}{4n}} \mod f_j$, with $j = \{0, 1\}$, where ζ_j is a generator for the cyclic group j.

[4]Wuthrich proves in Theorem 5.8 that every $0 \neq \alpha \in \mathbb{Z}[i]$ has a unique factorization [30].

Algorithm 7: Compute the k-th primitive root of $i \mod p$, for a prime number $p \equiv 1 \mod 4$.

Input: An integer k and a prime $p \equiv 1 \mod 4$.
Output: The k-th primitive root of $i \mod p$.

$f_0, f_1 = \texttt{decompose\_in\_gaussian\_primes}(p)$
while *True* **do**
    for $j = 0; j < 2; j = j + 1$ **do**
        $\zeta_j = \texttt{sample\_generator}(f_j)$; $h_j = \zeta_j^{\lfloor (p-1)/(4k) \rfloor} \mod f_j$
    $h = f_1 \cdot \left(f_1^{-1} \cdot h_0 \mod f_0\right) + f_0 \cdot \left(f_0^{-1} \cdot h_1 \mod f_1\right) \mod p$
    if $h^k \equiv i \mod p$ **then**
        return h

2.3 Efficient polynomial multiplication on GPUs

The work presented in Section 2.2 was followed by questions from the community regarding the lack of a formal investigation of the real benefits of the DGT over the NTT. Aiming that, we proposed a methodology to evaluate the advantage of each on different scenarios when running on GPUs.

This publication is entitled "Performance of Hierarchical Transforms in Homomorphic Encryption" and is under reviewing in the Journal of Cryptographic Engineering.

Performance of Hierarchical Transforms in Homomorphic Encryption:
A case study on Logistic Regression inference

Pedro G. M. R. Alves[1] Jheyne N. Ortiz[1] Diego F. Aranha[2]

Institute of Computing, University of Campinas, Campinas, Brazil[1]
Department of Computer Science, Aarhus University, Aarhus, Denmark[2]

To be published

Abstract

Recent works challenged the Number-Theoretic Transform (NTT) as the most efficient method for polynomial multiplication in GPU implementations of Fully Homomorphic Encryption schemes such as CKKS and BFV. In particular, these works argue that the Discrete Galois Transform (DGT) is a better candidate for this particular case. However, these claims were never rigorously validated, and only intuition was used to argue in favor of each transform. This work brings some light on the discussion by developing similar CUDA implementations of the CKKS cryptosystem, differing only in the underlying transform and related data structure. We ran several experiments and collected performance metrics in different contexts, ranging from the basic direct comparison between the transforms to measuring the impact of each one on the inference phase of the logistic regression algorithm. Our observations suggest that, despite some specific polynomial ring configurations, the DGT in a standalone implementation does not offer the same performance as the NTT. However, when we consider the entire cryptosystem, we noticed that the effects of the higher arithmetic density of the DGT on other parts of the implementation is substantial, implying a considerable performance improvement of up to 15% on the homomorphic multiplication. Furthermore, this speedup is consistent when we consider a more complex application, indicating that the DGT suits better the target architecture.

Keywords— NTT, DGT, Fully Homomorphic Encryption, CKKS, CUDA, Polynomial multiplication, Privacy-preserving computing

1 Introduction

In 1978, Rivest *et al.* first conceived the notion of Homomorphic Encryption (HE) schemes [41]. Their objective was to preserve some mathematical structure after encryption that enables the evaluation of arithmetic circuits over ciphertexts without decryption or knowledge of the secret

Results were partially obtained while visiting the Department of Computer Science at Aarhus University.

key. The computation outcome is naturally encrypted, and an observer learns nothing regarding the operands, the result, or the decryption key. The first HE schemes had limited capability, supporting only additions or multiplications, and because of that they were called Partially Homomorphic Encryption (PHE) schemes. ElGamal's and Paillier's are notorious examples of such PHE schemes [24, 37]. Since they cannot support both operations simultaneously, their suitability to real-world implementations is limited. It took around 30 years after this initial work for the first practical construction of a Fully Homomorphic Encryption (FHE) scheme supporting an unlimited number of both operations to be introduced [26]. Unfortunately, the first proposals did not stand out for performance regarding latency or memory consumption. Still, Gentry was successful in drawing a blueprint that has guided many FHE schemes. His main contribution [26] was the proposal of a bootstrapping operation that homomorphically evaluates the decryption procedure to remove the upper bound on the complexity of supported functions. The following decade was dedicated to security and performance improvements [27, 13, 12, 34, 22, 15, 14].

Modern schemes have significantly reduced the performance overhead imposed on computation over ciphertexts [23, 16]. Their implementation relies on polynomial arithmetic, so developers have to find efficient ways to handle known costly operators, such as polynomial multiplication and division. Concerning the former, the literature has established the suitability of the Number-Theoretic Transform (NTT), a variant of the Discrete Fourier Transform (DFT) that operates over integers, to compute the polynomial multiplication with linear complexity within the transform domain. Nonetheless, some recent works suggest that the Discrete Galois Transform (DGT) may be a better candidate when the target hardware is a CUDA-enabled GPU [2, 6, 4]. CUDA is a SIMD architecture developed and maintained by NVIDIA to employ GPUs' potential for data parallelism in tasks beyond graphical processing. In particular, the suitability of current devices for polynomial arithmetic made CUDA an essential tool for the efficient implementation of FHE schemes [21, 32].

However, the particularities of the architecture impose some challenges. For example, its processing flow demands careful planning to align possible conditional branches with certain thread groups, and its memory paradigm considers several structures with different dimensions and latency characteristics, apart from the machine's main memory. So, part of the difficulty of its use involves tailoring classical methods into variants that conveniently fit the GPU.

1.1 Related Work

Current HE schemes are built on top of Gentry's proposal of a bootstrappable cryptosystem, a scheme that homomorphically evaluates its own decryption circuit [26]. The bootstrap procedure enables these proposals to be used as fully homomorphic since they can perform an unlimited number of additions and multiplications. Some of the primary schemes available in the literature are BFV [22], CKKS [14], and TFHE [15]. All these schemes share a common core for polynomial arithmetic, which allows using a DFT-based method to accelerate polynomial multiplication. For this task, one may use the Fast Fourier Transform (FFT), the NTT, or the DGT. In particular, we target the application of these methods to the implementation of CKKS on GPUs.

The applicability of DFT-based algorithms to efficiently implement polynomial multiplication is ubiquitous in the FHE literature. In 1966, Gentleman and Sande [25] proposed the hierarchical FTT, and it remained forgotten in the interim until 1989 when Bailey rediscovered it as the "four-step" FFT algorithm [7]. Later on, in 2008, Govindaraju et al. implemented the four-step hierarchical FFT in the context of GPUs [28]. A few years later, Harvey developed arithmetic techniques for reducing the number of reductions modulo p during the computation of the NTT [30]. The NTT is a DFT variant that works in a finite field, so its arithmetic better suits cryptographic contexts, avoiding the use of floating-point arithmetic that is inherent to the FFT. Because of that, the NTT became the norm in the literature. In 2018, Dai et al. recursively applied the four-step Cooley-Tukey algorithm [17] to obtain NTTs with size 64 instead of

performing the transform on integer vectors of 2048 and 4096 coordinates [20]. Recently, Jung et al. applied a hierarchical implementation of the NTT in the fastest up-to-date implementation of logistic regression over GPUs [31]. Following a slightly different branch, Badawi et al. explored the suitability of the DGT for FHE [2]. The DGT works in the finite field $\mathbb{F}_{p^2}$ for prime p. Thus, it is similar to the NTT but also adds the possibility of halving the operands' degree. In this same direction, the DGT was used to accelerate polynomial multiplication on a BFV implementation on GPUs [6], and lately, Alves et al. improved the use of the DGT for polynomial multiplication in GPUs through a hierarchical implementation [4].

The DGT is favored by the fact that arithmetic in the field $\mathbb{F}_{p^2}$ is performed in the set of Gaussian integers $\mathbb{Z}_p[i]$, for some convenient choice of prime p. By working with Gaussian integers, the operands are converted from a N-degree to a $N/2$-degree polynomial ring via a folding procedure. Consequently, the degree of the inputs used inside the transform is halved and so is the number of roots that will be loaded and stored into memory during the transform computation. Moreover, a CUDA implementation can also reduce the number of required threads, which avoids the parallel performance degradation in bigger instances [2, 6]. Nevertheless, the representation in $\mathbb{Z}_p[i]$ implies denser arithmetic operations, which saves in memory bandwidth consumption. Notice that arithmetic in $\mathbb{F}_p$ consists in coefficient-wise operations modulo p between two N-degree polynomials. On the other hand, when the base field is $\mathbb{F}_{p^2}$, addition and multiplication operations are similar to the ones over complex numbers. This means that an increased arithmetic density is natural to the DGT and may improve performance if the implementation explores the processing hardware special capabilities.

1.2 Our contributions

In this paper, we describe our efforts to resume the investigation started by Badawi et al. on the possible advantages of replacing the NTT with the DGT for the implementation of polynomial multiplication in FHE cryptosystems [2]. In particular, we target results claiming that the DGT is more suitable than NTT for GPUs and memory-bounded platforms [5]. To the best of our knowledge, no previous work provided a deep analysis of the advantages of each transform in the context of GPU execution.

We developed two implementations of the CKKS scheme using DGT and NTT as the underlying transform to perform multiplication in the ring $\mathbb{Z}_q[x]/(x^N + 1)$. The implementations are referred to as AOA-DGT and AOA-NTT, respectively. We compare the latencies of the transforms executed independently and also within CKKS' homomorphic primitives. Moreover, we present a case study verifying how the performance of the logistic regression inference is affected by each. For that, we compute the inference score using both implementations for a trained model over the MNIST database considering the case when the model is encrypted, protecting its secrecy, and when it is handled as plaintext [33]. Furthermore, we analyze how the problem scales for approaches found in the literature that run the inference with and without the computation of the activation function.

Our experiments reveal that the NTT offers a clear performance advantage over the DGT in most cases and especially in smaller instances. For 8192-degree polynomial rings, however, the DGT more efficiently takes advantage of the processing hardware and can overcome the NTT.

Nonetheless, the AOA-DGT's homomorphic multiplication performs better even though its related transform implementation does not. We found procedures in the critical path of that primitive not directly related to the transform itself but that are impacted by the associated implementation decisions. For instance, a 10% slowdown on the Logistic Regression inference executed on AOA-NTT is observed, caused by the less efficient basis extension methods. We show that this result can be reversed by increasing arithmetic density in AOA-NTT to match the one in AOA-DGT.

Organization This document is organized as follows. Section 2 describes the adopted notation and relevant basic building blocks at Sections 2.2 and 2.4; and the target FHE scheme, at Section 2.3. Section 3 presents our experiments and methodology, and discusses the obtained results. Furthermore, Section 4 presents a case study on how the selection of each transform affects the performance on the inference of a homomorphic logistic regression implementation.

2 Background

Most implementations of cryptosystems based on the Ring Learning-With-Errors (RLWE) problem requires the construction of basic building blocks that offer polynomial arithmetic on a cyclotomic ring. An example is the cryptosystem CKKS [14], a leveled homomorphic encryption scheme. In particular, the implementation of algorithms for polynomial multiplication has been a challenging task. A simple schoolbook algorithm requires quadratic complexity on the operands degree, and because of that it is utterly unsuitable on instances of cryptographic size. The typical approach in the literature involves variants of the DFT, which offer linear complexity when the operands lie in their domains. In this work, we focus on the efficient implementation of the NTT and DGT.

In this context, this section describes the notation used throughout the document, defines the relevant primitives of CKKS, and discusses formulations for the NTT and the DGT.

2.1 Notation

We use bold letters to denote vectors e.g., $\mathbf{a}$ and $\mathbf{A}$. For a vector $\mathbf{a}$, we refer by a_i the i-th element. We denote by $[x]_q$ the reduction of an integer x modulo q, that is, $[x]_q := x \mod q$, for some integer q. Furthermore, let $\mathbf{X}$ be a matrix, then $x_{*,i}$ is the set of all the elements on column i of $\mathbf{X}$. In the same way, $x_{i,*}$ is the set of all the elements on row i of $\mathbf{X}$. Also, we use $\lfloor x \rceil$ to define the rounding to the nearest integer operation.

Let K be the $2N$-th cyclotomic number field and $R = \mathcal{O}_K$ its ring of integers. We represent R in its polynomial form, that is, $R = \mathbb{Z}[x]/(x^N + 1)$. Moreover, for an integer $q \geq 2$, R_q denotes the quotient ring $R_q = \mathbb{Z}_q[x]/(x^N + 1)$.

Consider that $\mathcal{C} = \{q_0, q_1, \ldots, q_\ell\}$ is a set of coprime integers and $q = \Pi_{i=0}^{\ell} q_i$. If $s \in R_{\mathcal{C}}$, then $\exists S \in R_q$ such that $s \coloneqq \{[S]_{q_0}, [S]_{q_1}, \ldots, [S]_{q_\ell}\}$. Arithmetic operations as addition and multiplication over elements in $R_{\mathcal{C}}$ are taken coefficient-wise, that is, if $a, b \in R_{\mathcal{C}}$ then $a + b = \left\{[a_0 + b_0]_{q_0}, \ldots, [a_\ell + b_\ell]_{q_\ell}\right\}$. Furthermore, we denote by $[x]_{q_0}$ the operation that selects the 0-th residue of x.

2.2 DFT-based Transforms

NTT is a variation of the DFT that replaces the primitive N-th complex root of unity by a primitive N-th root of unity ω_N in a ring $\mathbb{Z}_p$ [40]. For N a power of two, the NTT requires p to be a prime number and that $N \mid (p-1)$. In this case, Pollard proved that there exists a primitive N-th root of unity in $\mathbb{Z}_p$ that can be computed as $r^{(p-1)/N}$, where r is the primitive root modulo p. For a polynomial $a(x) = \sum_{j=0}^{N-1} a_j x^j \in \mathbb{Z}[x]$, the N-point NTT computes

$$\mathrm{NTT}_{\omega_N}(a(x)) = \left(a(\omega_N^0), \ldots, a(\omega_N^{N-1})\right). \tag{1}$$

Conversely, the inverse transform is

$$\mathrm{INTT}_{\omega_N^{-1}}(a(x)) = \left[N^{-1} \cdot \mathrm{NTT}_{\omega_N^{-1}}(a(x))\right]_p, \tag{2}$$

where N^{-1} is the multiplicative inverse of N modulo p. The polynomial multiplication $c(x) = a(x) \cdot b(x) \mod x^N + 1$ can be done via NTT as

$$c(x) = \omega_{2N}^{-1} \cdot \mathrm{INTT}_{\omega_N^{-1}}(\mathrm{NTT}_{\omega_N} \hat{a}(x)) \odot \mathrm{NTT}_{\omega_N}(\hat{b}(x))),$$

in which $\hat{a}(x) = \omega_{2N} \cdot a(x)$. This technique of scaling the operands by powers of ω_{2N} is known as the negative wrapped convolution [35].

DGT is an alternative to the NTT over the finite field $\mathbb{F}_{p^2}$, where p is a prime. It was presented as a method for integer convolution by Crandall [18] and was further considered for polynomial multiplication by Badawi et al. [6]. The DGT and its inverse transform are defined in the same way as NTT in Equations 1 and 2, but now the operands are vectors with elements in $\mathbb{F}_{p^2}$.

In this context, the field elements may be represented using the set of Gaussian integers modulo p, denoted $\mathbb{Z}_p[i]$, which is defined as $\mathbb{Z}_p[i] = \{a + ib \mid a, b \in \mathbb{Z}_p\}$, for $i = \sqrt{-1}$. The arithmetic in $\mathbb{Z}_p[i]$ is similar to the one in $\mathbb{C}$ but both real ($\Re$) and imaginary ($\Im$) parts are taken modulo p. When the polynomial ring is $\mathbb{Z}[x]/(x^N + 1)$, Crandall [18] defines a transform on a vector $\mathbf{a}$ with size $N \equiv 0 \mod 2$ to a vector $\mathbf{A}$ with size $N/2$, denoted as *folding*, such that

$$A_j = a_j + i a_{j+\frac{N}{2}} \in \mathbb{Z}_p[i].$$

This transform maps the coefficients of the polynomial from $\mathbb{Z}_p^N$ to $\mathbb{Z}_p[i]^{\frac{N}{2}}$. The inverse transform from $\mathbb{Z}_p[i]^{\frac{N}{2}}$ to $\mathbb{Z}_p^N$ is denoted as *unfolding* and it is given by

$$a_j = \Re(A_j) \quad \text{and} \quad a_{j+\frac{N}{2}} = \Im(A_j),$$

for $0 \leq j \leq N/2 - 1$. Crandall [18] also defines a "right-angle" convolution that multiplies the folded vector $\mathbf{A}$ by powers of $\tau = \tau_{\frac{N}{2}}$, an $\frac{N}{2}$-th root of i modulo p. We refer to this convolution as *twisting*, since powers of τ are the twisting factors defined as

$$A_j = A_j \cdot \tau^j.$$

The corresponding inverse convolution is given by the multiplication of the vector $\mathbf{a}$, which is the output of the unfolding procedure, by growing powers of the inverse $\frac{N}{2}$-th root of i, denoted τ^{-1}.

For completeness, we present the polynomial multiplication in the ring $\mathbb{Z}_p[x]/(x^N + 1)$ via DGT introduced by Badawi et al. [6] in Algorithm 1. Notice that it operates on the coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ of two polynomials $a(x), b(x) \in \mathbb{Z}_p[x]/(x^N + 1)$. Similarly, the algorithm outputs the coefficient vector $\mathbf{c}$ corresponding to the computation $c(x) = a(x) \cdot b(x) \in \mathbb{Z}_p[x]/(x^N + 1)$.

2.3 CKKS Scheme

Cheon-Kim-Kim-Song proposed a leveled homomorphic encryption scheme known as CKKS [14] in which the plaintext domain is composed of complex numbers. The array of complex numbers is mapped into elements of the ring R using an encoding method that works as follows.

Let $\mathbf{z}$ be a vector of N complex numbers and Δ a scalar. In practice, we have that $\mathsf{decode}(\mathbf{z}) = \lfloor \mathrm{FFT}(\mathbf{z}) \cdot \Delta^{-1} \rceil$. In this sense, encode is defined simply as the inverse procedure. Through this approach, CKKS becomes capable of operating with non-integer numbers using fixed-point arithmetic. In this representation, Δ is called the *scaling factor* and is responsible for setting its precision.

A CKKS ciphertext, denoted $\mathsf{ct} = (c_0, c_1)$, is a pair of elements in $R_{\mathcal{C}}$, for $\mathcal{C} = \{q_0, \ldots, q_L\}$ a RNS basis. In other words, $\mathsf{ct} = \{(c_0^{(i)}, c_1^{(i)})\}_{0 \leq i \leq L}$ such that $(c_0^{(i)}, c_1^{(i)}) \in R_{q_i} \times R_{q_i}$. At this

Algorithm 1 Polynomial multiplication in $\mathbb{Z}_p[x]/(x^N+1)$ via DGT

Require: Vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^N$, p a prime number, N a power-of-two integer, and τ a primitive $\frac{N}{2}$-th root of i modulo p.
Ensure: A coefficient vector $\mathbf{c} \in \mathbb{Z}_p^N$.

for $j = 0;\ j < N/2;\ j = j + 1$ **do**
  $a'_j = a_j + i a_{j+N/2}$
  $b'_j = b_j + i b_{j+N/2}$
end for
for $j = 0;\ j < N/2;\ j = j + 1$ **do**
  $a'_j = \tau^j \cdot a'_j \pmod{p}$
  $b'_j = \tau^j \cdot b'_j \pmod{p}$
end for
$a' = \mathrm{DGT}(a')$
$b' = \mathrm{DGT}(b')$
for $j = 0;\ j < N/2;\ j = j + 1$ **do**
  $c'_j = a'_j \cdot b'_j \pmod{p}$
end for
$c' = \mathrm{IDGT}(c')$
for $j = 0;\ j < N/2;\ j = j + 1$ **do**
  $\mathrm{aux} = \tau^{-j} \cdot c'_j \pmod{p}$
  $c_j = \Re(\mathrm{aux})$
  $c_{j+\frac{N}{2}} = \Im(\mathrm{aux})$
end for
return c

moment, we say that this ciphertext has level $\ell = L + 1$. We have that c_0 and c_1 are built by the CKKS encryption algorithm such that $[c_0 + s \cdot c_1]_{q_0} \approx m$, for a secret key s.

A decryption imprecision is expected, and its error is considered part of the cryptosystem's inherent noise. Let $\mathrm{ct}_0 = (\mathrm{ct}_{0,0}, \mathrm{ct}_{0,1})$ and $\mathrm{ct}_1 = (\mathrm{ct}_{1,0}, \mathrm{ct}_{1,1})$ be encryptions of m_0 and m_1 under a secret key s, respectively. Then, $m_0 \cdot m_1 \approx [(\mathrm{ct}_{0,0} + s \cdot \mathrm{ct}_{0,1}) \cdot (\mathrm{ct}_{1,0} + s \cdot \mathrm{ct}_{1,1})]_{q_0}$. Therefore, the property that is expected to be conserved to offer homomorphic multiplication is

$$\begin{aligned} m_2 = m_0 \cdot m_1 \approx [&\mathrm{ct}_{0,0} \cdot \mathrm{ct}_{1,0} \\ &+ s \cdot (\mathrm{ct}_{0,1} \cdot \mathrm{ct}_{1,0} + \mathrm{ct}_{1,1} \cdot \mathrm{ct}_{0,0}) \\ &+ s^2 \cdot \mathrm{ct}_{0,1} \cdot \mathrm{ct}_{1,1}]_{q_0}. \end{aligned}$$

The problem with this construction is that the outcome of a ciphertext multiplication would be a ciphertext composed of three parts, which are the coefficients of powers of s. This is not desirable for storage efficiency, and can also become a significant computational problem for following homomorphic operations, especially for homomorphic multiplications. Thus, a procedure to recover the ciphertext's linearity, i.e. write it as a linear combination of $\{1, s\}$, is crucial in this context.

The relinearization procedure for this scheme extends the polynomial representation of the quadratic coefficient from an element of $R_{\mathcal{C}}$ to an element of $R_{\mathcal{C}+\mathcal{D}}$, for a secondary basis $\mathcal{D} = \{p_0, p_1, \ldots, p_k\}$ coprime to $\mathcal{C}$. A multiplication by an evaluation key, evk, is done in this bigger basis, and then the representation is shrank back to the basis $\mathcal{C}$. These basis conversion steps are done through approximate modulus switching functions referred to as ModUp and ModDown.

Bajard et al. [8] define a fast basis extension procedure from $\mathcal{C}$ to $\mathcal{D}$ as follows:

$$\mathrm{Conv}_{\mathcal{C}\to\mathcal{D}}(a) = \left(\left[\sum_{j=0}^{\ell-1} [a^{(j)} \cdot \hat{q}_j^{-1}]_{q_j} \cdot \hat{q}_j \right]_{p_i} \right)_{0 \le i \le k}$$

where $\hat{q}_j = \prod_{j' \neq j} q_{j'}$. This procedure can be used for ModUp, computing the approximated representation of a in a bigger basis. This approximation, in the context of the CKKS, is close enough to add negligible noise to the cryptosystem.

The inverse procedure, ModDown, aims at computing $b \approx P^{-1} \cdot \tilde{b} \in \mathbb{Z}_{\mathcal{C}}$ for $P = \prod_{p \in \mathcal{D}} p$, given as input the representation $\tilde{b} \in \mathbb{Z}_{\mathcal{C}+\mathcal{D}}$.

$$\begin{aligned} &\mathrm{ModDown}_{\mathcal{C}+\mathcal{D}\to\mathcal{C}}\left(\left\{\tilde{b}_{\mathcal{C}}, \tilde{b}_{\mathcal{D}}\right\}\right) = \\ &\left(P^{-1} \cdot \left(\tilde{b}_{\mathcal{C}}^{(j)} - \mathrm{Conv}_{\mathcal{D}\to\mathcal{C}}(\tilde{b}_{\mathcal{D}})^{(j)}\right)\right)_{0 \le j \le \ell}. \end{aligned}$$

Notice that, after a homomorphic multiplication, the encoding scaling factor of the outcome is Δ^2. For maintaining the representation precision, CKKS uses a rescaling method to restore the original scaling factor (or an approximation of it). In this sense, one of the residues used to represent the ciphertext is consumed, leading to a $\ell' = (\ell - 1)$-level ciphertext. When $\ell' = 0$, no further rescaling is possible.

Let χ_{key} be a secret key distribution, and χ_{err} an encryption key distribution over the ring R. In practice, χ_{key} is usually defined as a narrow distribution, sampling uniformly from $\{-1, 0, 1\}$, and χ_{err} is taken as a discrete Gaussian. Also, let $\mathcal{C} = \{q_0, q_1, \ldots, q_L\}$ and $\mathcal{D} = \{q_{L+1}, q_{L_2}, \ldots, q_{L+k}\}$ be two RNS basis coprime to each other. In the following we define some primitives relevant for this work:

- **CKKS.SecKeyGen**(1^λ): Sample $s \leftarrow\!\!\$\ \chi_{key}$ and set the secret key as $\mathsf{sk} \coloneqq (1, s)$.
- **CKKS.PubKeyGen**(sk): Sample $\left(a^{(0)}, \ldots, a^{(L)}\right) \leftarrow\!\!\$\ R_{\mathcal{C}}$ and $e \leftarrow\!\!\$\ \chi_{err}$. Set the public key as $\mathsf{pk} \coloneqq \left(\mathsf{pk}^{(j)} = (-a^{(j)} \cdot s + e \mod q_j, a^{(j)})_{0 \le j \le L}\right)$.

- **CKKS.RelinKeyGen**(sk): Sample $\left(a^{(0)}, \ldots, a^{(k+L)}\right) \hookleftarrow\$ R_{\mathcal{C}\bigcup\mathcal{B}}$ and $e \hookleftarrow\$ \chi_{err}$. Let $b^{(j)} \coloneqq -a^{(j)} \cdot s + \left[\prod_{i=0}^{k-1} p_i\right]_{g^j} + e \mod g^j$, for $g_j \in \mathcal{C}\bigcup\mathcal{B}$. Set the relinearization key $\mathsf{rlk} \coloneqq \left(\mathsf{rlk}^{(j)} = (b^{(j)}, a^{(j)})\right)_{0\leq j\leq L}$.
- **CKKS.Encrypt**(m, pk): For $m \in R_{\mathcal{C}}$, sample $v \hookleftarrow\$ \chi_{enc}$ and $e_0, e_1 \hookleftarrow\$ \chi_{err}$. Output the ciphertext $\mathsf{ct} \coloneqq \left(c^{(j)} = \left[v \cdot \mathsf{pk}^{(j)} + (m + e_0, e_1)\right]_{q_j}\right)_{0\leq j\leq L}$
- **CKKS.Decrypt**(ct, sk): Output $[\mathsf{ct} \cdot \mathsf{sk}]_{q_0}$.
- **CKKS.Add**(c_a, c_b): Let $c_a = (a_0^{(j)}, a_1^{(j)})$ and $c_b = (b_0^{(j)}, b_1^{(j)})$ for $j \in \{0, \ldots, L\}$. Output $([a_0^{(j)} + b_0^{(j)}]_{q_j}, [a_1^{(j)} + b_1^{(j)}]_{q_j})_{0\leq j\leq L}$
- **CKKS.DR2**(c_a, c_b): Let $c_a = (a_0^{(j)}, a_1^{(j)})$ and $c_b = (b_0^{(j)}, b_1^{(j)})$. Output $\left(d_0^{(j)}, d_1^{(j)}, d_2^{(j)}\right) = \left(a_0^{(j)}b_0^{(j)}, a_0^{(j)}b_1^{(j)} + a_1^{(j)}b_0^{(j)}, a_1^{(j)}b_1^{(j)}\right) \in R_{q_j}^3$, for $j \in \{0, \ldots, L\}$.
- **CKKS.Mul**(c_a, c_b): Let $c_a = (a_0^{(j)}, a_1^{(j)})$ and $c_b = (b_0^{(j)}, b_1^{(j)})$.
  1. $\left(d_0^{(j)}, d_1^{(j)}, d_2^{(j)}\right) = \text{CKKS.DR2}(c_a, c_b)$,
  2. Having the representation of d_2 in base $\mathcal{C}$, compute its representation in base $\mathcal{D}$: $\tilde{d_2} \coloneqq \textsc{ModUp}_{\mathcal{C}_\ell \leftarrow \mathcal{D}_\ell}(d_2)$
  3. $\tilde{ct}^{(j)} \coloneqq \tilde{d_2}^{(j)} \cdot \mathsf{evk}^{(j)} \mod q_j$ for $q_j \in \mathcal{C} + \mathcal{D}$.
  4. Having $\tilde{ct}^{(j)}$ in base $\mathcal{C} + \mathcal{D}$, compute its representation in base $\mathcal{C}$: $\hat{c}^{(j)} \coloneqq \textsc{ModDown}_{\mathcal{D}_\ell \leftarrow \mathcal{C}_\ell}(\tilde{ct}^{(j)})$
  5. Output $\left(\hat{c}_0^{(j)} + d_0^{(j)}, \hat{c}_1^{(j)} + d_1^{(j)}\right)_{0\leq j\leq L}$.
- **CKKS.Rescale**(c_a): Let a ℓ-level ciphertext $c_a = (a_0^{(j)}, a_1^{(j)})$ for $j \in \{0, \ldots, \ell\}$. Compute $c_b \coloneqq q_\ell^{-1} \cdot \left(a_0^{(j)} - a_0^{\ell}, a_1^{(j)} - a_1^{\ell}\right)_{0\leq j\leq \ell-1}$. The result, c_b, is a $(\ell-1)$-level ciphertext.
- **CKKS.AddPlain**(c, p): Let $c = (a^{(j)}, b^{(j)})$ be a ℓ-level ciphertext and $p \in R_{\mathcal{C}}$ a plaintext, for $j \in \{0, \ldots, \ell\}$. Output $(a^{(j)} + p^{(j)}, b^{(j)})_{0\leq j\leq L} \in R_{\mathcal{C}}$
- **CKKS.MulPlain**(c, p): Let $c = (a^{(j)}, b^{(j)})$ be a ℓ-level ciphertext and $p \in R_{\mathcal{C}}$ a plaintext, for $j \in \{0, \ldots, \ell\}$. Output $(a^{(j)}p^{(j)}, b^{(j)}p^{(j)})_{0\leq j\leq L} \in R_{\mathcal{C}}$.

2.4 Hierarchical Transforms

The memory paradigm of GPUs involves different layers that should be considered prior to the computation. Usually, the processing workflow starts with the data copy from the machine's main memory to the GPU global memory, which is the largest but also slowest memory space accessible by CUDA threads. Thus, before computation really starts, data needs to be copied from the global memory to a faster memory. Each CUDA thread is member of a three-dimensional block of threads, which shares a fast but small memory space, called *shared memory*. By doing that, the performance is considerably increased. Yet, this also imposes a constraint on the space consumption.

As discussed by Alves et al., the implementation of the DGT or NTT in memory-constrained devices, such as GPUs, is not straightforward [4]. The synchronization calls needed by these algorithms limits the dimension of the input to the size of a block of threads, or requires the

use of a software-based mechanism like Cooperative Threads. To avoid such issue, a hierarchical strategy can be adopted to reduce the size of the transforms to a feasible dimension, which can be easily supported by the processing hardware. Originally proposed by Bailey and revisited by Govindaraju et al. for the FFT, the idea is to split the input polynomial, of degree N, into smaller instances as close as to $\sqrt{N}$ [7, 28].

We present a general description of a forward hierarchical transform, which in our case is either the hierarchical NTT or hierarchical DGT. The forward hierarchical transform is done by executing the following four steps on an N-length integer vector $\mathbf{a}$. We also consider that the arithmetic operations are taken modulo a prime number p. When dealing with the hierarchical NTT transform, we require that $p \equiv 1 \pmod{2N}$. But, when the transform is the DGT, we require that $p \equiv 1 \pmod{4N}$ in order to the $\frac{N}{2}$-th primitive root of i exist modulo p.

1. Apply the weight corresponding to either NTT or DGT to the operand $\mathbf{a}$. When the NTT is the transform, the weight is the negative wrapped convolution, which multiplies the coefficients of $\mathbf{a}$ by powers of the $2N$-th primitive root of unity modulo p. When the transform is the DGT, the weight consists of folding the input vector followed by a multiplication by powers of the $N/2$-th primitive root of i modulo p, denoted $h \pmod{p}$. Despite the transform, the result of this step is assumed to be an N'-length vector.

2. By treating $\mathbf{a}$ as an $N_r \times N_c$-matrix, denoted $\mathbf{A}$, perform N_c simultaneous N_r-length transforms on each column of $\mathbf{A}$.

3. Apply the twisting factor g, which is the N'-th primitive root of unity modulo p, to $\mathbf{a}$ by multiplying each element $\mathbf{A}_{i,j}$ by $g^{i \cdot j} \pmod{p}$.

4. Finally, perform N_r simultaneous N_c-length transforms on each row of $\mathbf{A}$.

We summarize the basic steps of a forward hierarchical transform in Algorithm 2. Notice that $\mathbf{A}_j$ denotes the j-th column of the matrix $\mathbf{A}$. The inverse hierarchical transform is obtained by executing the above four steps in reversed order. This is done by replacing the forward transform in steps 2 and 4 with their inverse counterparts and substituting the primitive roots in steps 1 and 3 by their inverse modulo p.

Algorithm 2 Hierarchical forward transform

Require: An N-length integer vector $\mathbf{a}$.
Ensure: The operand $\mathbf{A}$ in the hierarchical transform domain.

```
a = ApplyWeight(a)
for j = 0; j < N_c; j = j + 1 do
    A_j = PerformForwardTransform(A_j)
end for
A = ApplyTwiddleFactor(A)
A = TransposeMatrix(A)
for i = 0; i < N_r; i = i + 1 do
    A_i = PerformForwardTransform(A_i)
end for
return A
```

Consider that we want to perform the polynomial multiplication $c(x) = a(x) \cdot b(x) \in \mathbb{Z}[x]/(x^N + 1)$ by adopting the hierarchical transforms. We consider that $\mathbf{a}$ and $\mathbf{b}$ contain the coefficients of the integer polynomials $a(x)$ and $b(x)$, respectively. Thus, the polynomial multiplication is done by executing the four-step algorithm into both $\mathbf{a}$ and $\mathbf{b}$, by computing

the component-wise multiplication on the operands, and by finally computing the inverse hierarchical transform. As a result, we obtain $\mathbf{c}$, which holds the coefficients of the polynomial $c(x)$. Notice that, in the NTT domain, the component-wise multiplication is performed in $\mathbb{Z}_p$. On the other hand, in the DGT domain, the product is performed in $\mathbb{Z}_p[i]$ using arithmetic similar to performed over complex numbers.

3 Performance Evaluation of Hierarchical Transforms

This work investigates whether the DGT outperforms the NTT in its hierarchical form as a mechanism to accelerate the CKKS arithmetic in the CUDA architecture. The main differences between them are the input folding and the field arithmetic of the DGT, which is more expensive than in the NTT. However, doing arithmetic operations in $\mathbb{F}(p^2)$ instead of $\mathbb{F}(p)$ offers a higher computational density. These characteristics may improve performance if they use the processing hardware more efficiently. To examine this hypothesis, we conceived two CUDA-based implementations of CKKS using the hierarchical versions of both DGT and NTT, which we refer to as AOA-DGT and AOA-NTT. Both follow the same design decisions, except for their basic data type.

We represent a polynomial as a single array by concatenating its residues. In AOA-NTT, the coefficients are stored as 64-bit unsigned integers, and in AOA-DGT we use a structure composed of two of those. All other implementation decisions follow Alves et al.'s blueprint, as the use of the *double*-CRT representation, encapsulating data simultaneously in the RNS representation and within the transform domain; and the GPU-optimized state machine, avoiding data move between memories and the transform domain [4].

Let $\mathcal{C} = \{q_0, \ldots, q_L\}$ and $\mathcal{D} = \{q_{L+1}, \ldots, q_{L+k}\}$ be the main and secondary RNS basis, as defined in Section 2.3. All time measurements were obtained with a 63-bit prime q_0, and 52-bit primes q_i, for $i > 0$, fixing the scaling factor at 2^{52}. We selected this scaling factor aiming the required precision to execute the logistic regression inference, described at Section 4.

We collected time measurements of both implementations in two distinct scenarios, comparing the transforms as a standalone but also as part of more complex algorithms. The implementations were analyzed using NVIDIA's recommended profiling tool, namely NVIDIA Nsight Systems version 2021.2.1.2 [36]. All executions were performed on a Google cloud instance and the code was compiled using GCC and G++ 8.4.0, and CUDA 11.3. Experiments were executed on either NVIDIA Tesla V100 or Tesla A100 GPUs.

3.1 Direct Comparison: DGT versus NTT

Following the hierarchical strategy, presented in Section 2.4, N-degree polynomials are processed as N_r or N_c-degree, such that $N = N_r \cdot N_c$. Thus, we need N_c blocks of $\lceil N_r/2 \rceil$ threads each to compute step 2 of the algorithm, and then N_r blocks of $\lceil N_c/2 \rceil$ threads for step 4.

The effect of the hierarchical approach can be observed in rings with degrees 4096 and 8192. Table 1 shows execution times for computing the DGT and the NTT on these instances represented in different RNS bases, which offer the best and the worst performance for the DGT compared to the NTT, respectively. As it can be seen in the table, DGT exhibits a consistent slowdown in the smaller instance. Due to DGT's folding procedure, 4096-degree polynomials are folded into 2048-degree polynomials with Gaussian integer coefficients. Since 2048 can be written as $64 \cdot 32$, it means that there will be blocks with $32/2 = 16$ threads running in the GPU.

Modern GPUs' streaming multiprocessors (SMs) process groups of 32 threads at a time, called warps, which are the primary processing unit in a GPU. In this sense, 16-thread blocks are too small and do not reach the CUDA warp size. Since warps are only composed of threads contained in the same block, blocks smaller than 32 imply that SM resources are being wasted, explaining the performance observed on instances of the DGT with a size smaller or equal to 32.

Table 1: Latencies for the computation of the DGT and NTT on their hierarchical formulation on 4096- and 8192-degree polynomials represented in RNS bases composed of r elements. Measurements in microseconds were computed as the average of 100 independent executions on an NVIDIA Tesla V100 GPU.

	$N = 4096$						
r	1	5	10	20	30	40	50
DGT	16.2	17.1	18.4	24.7	32.6	39.6	48.4
NTT	13.4	14.3	17.0	22.0	28.5	35.2	43.1
Ratio	1.21	1.20	1.08	1.12	1.14	1.13	1.12
	$N = 8192$						
r	1	5	10	20	30	40	50
DGT	17.7	18.9	23.0	35.0	47.8	60.4	77.0
NTT	14.2	17.2	23.5	36.9	52.2	64.3	76.4
Ratio	1.25	1.10	0.98	0.95	0.92	0.94	1.01

In comparison with NTT that does not use folding, the operands are processed as 4096-degree polynomials. Thus, $4096 = 64 \cdot 64$, and all blocks are set with 32 threads, fitting in a warp perfectly. For $N = 8192$, the opposite happens, and the DGT benefits from the SM processing. In this case, some NTT thread blocks have 64 elements, but DGT enters in its optimal setup with 32-thread blocks. Figure 1 expands this analysis for further configurations. When $N = 4096$, a considerable slowdown of the DGT is observed, related to its inefficiency in exploiting the hardware's resources. The NTT starts to move from its optimal configuration when $N = 8192$, losing execution efficiency. At the same time, the DGT finds its best performance, in which a speedup in the interval between 10 and 45 residues is observed.

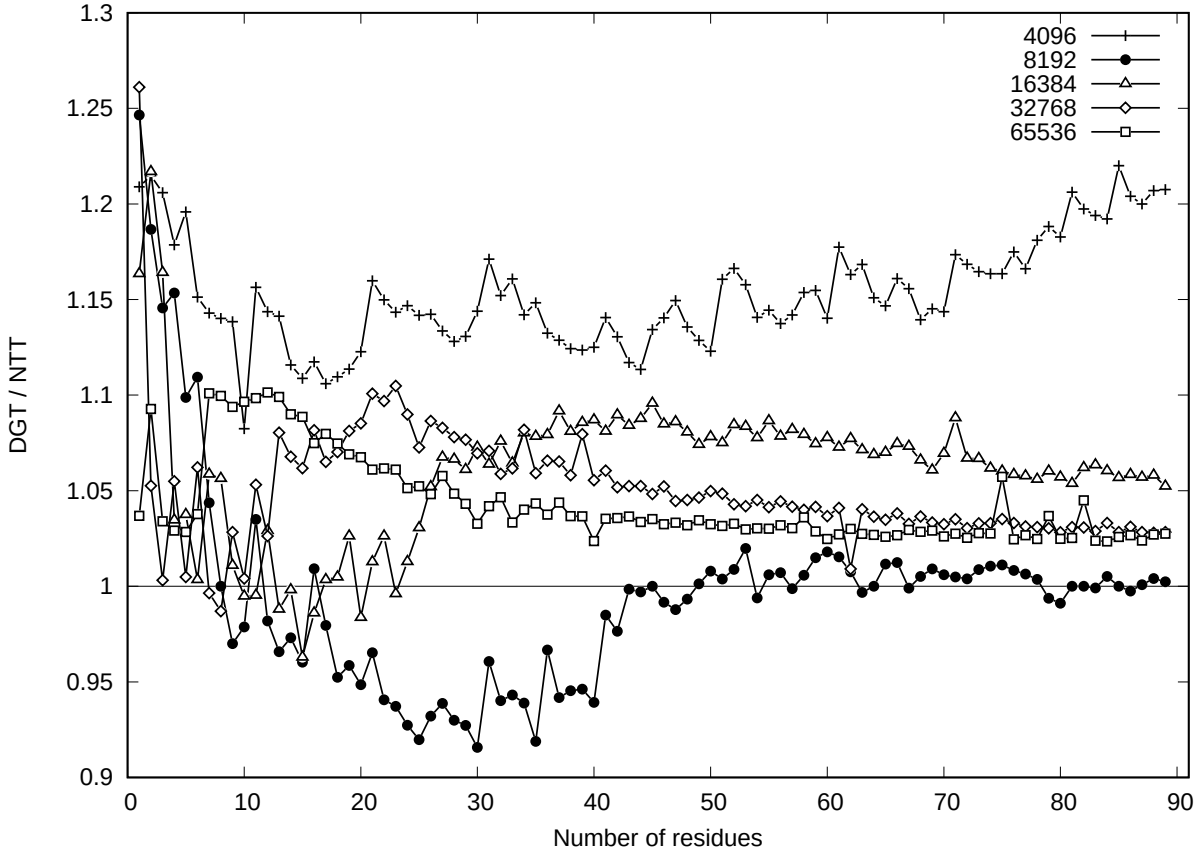
In larger instances, AOA-DGT and AOA-NTT suffer from the increasing consumption of shared memory, which rises bank conflicts, and thread block synchronization becomes more expensive, since blocks must be split among several warps.

3.2 Impact on CKKS Homomorphic Primitives

The NTT and DGT transforms are applicable to the CKKS context for reducing the complexity of polynomial multiplications. However, the consequences go beyond that. For instance, the basis extension methods ModUp and ModDown, described in Section 2.3, cannot be expressed in an arithmetic circuit that can be evaluated in the domains of the transforms. Thus, certain operations, such as homomorphic multiplication, require converting between distinct domains. One would prefer to keep the data structure associated with each transform to avoid conversion costs. In that case, the implementation of those methods would have to consider that data structure and would be affected by its particularities. Thus, Table 1 is not sufficient to decisively conclude about the suitability of each method regarding its employment to CKKS.

Table 2 compares the CKKS' homomorphic addition and multiplication using each transform. When the homomorphic addition is implemented in AOA-DGT, it presents a slowdown of around 10%. This is a straightforward procedure and, intuitively, one would expect the same performance in both implementations since the required number of modular additions is the same. However, this is an extremely memory-bound procedure. Apart from memory transactions, it only requires integer additions. Even the related reductions modulo p can be implemented with a single integer addition by constant-time selection of $a+b$ and $a+b-p$. Hence, the memory overhead outweighs the computational cost. Some of the kernel launch cost can be amortized by executing the entire homomorphic addition in a single CUDA kernel. However, the profiler still indicates that both implementations achieve very low occupancy and warps may stall waiting for load and store

Figure 1: Ratio (DGT/NTT) of the DGT and NTT execution time for different polynomial degrees and varying sizes of RNS basis. Measurements computed as the average of 100 independent executions on an NVIDIA Tesla V100 GPU.



transactions to the device's memory. Notice that the implementation in AOA-DGT takes twice the number of input and output operands, thus being more affected by warps stalls.

Algorithm 3 Pseudo-code of CKKS' homomorphic multiplication

Require: $\mathsf{ct}_0 \equiv \text{CKKS.Encrypt}(m_0)$ and $\mathsf{ct}_1 \equiv \text{CKKS.Encrypt}(m_1)$, and evk as an evaluation key.

Ensure: ct_2 such that $\mathsf{ct}_2 \equiv \text{CKKS.Encrypt}(m_0 \times m_1)$.

$\widehat{\mathsf{ct}}_0 = \mathsf{Transform}(\mathsf{ct}_0)$
$\widehat{\mathsf{ct}}_1 = \mathsf{Transform}(\mathsf{ct}_1)$
$\hat{d} = \mathsf{CKKS.DR2}(\widehat{\mathsf{ct}}_0, \widehat{\mathsf{ct}}_1)$
$d = \mathsf{InverseTransform}(\hat{d})$
$e_2 \coloneqq \mathsf{ModUp}_{\mathcal{C}_\ell \leftarrow \mathcal{D}_\ell}(d_2)$
$\hat{e}_2 = \mathsf{Transform}(e_2)$
$\widehat{\mathsf{ct}} \coloneqq \hat{e}_2 \times \mathsf{evk}$
$\mathsf{ct} = \mathsf{InverseTransform}(\widehat{\mathsf{ct}})$
$a \coloneqq \mathsf{ModDown}_{\mathcal{D}_\ell \leftarrow \mathcal{C}_\ell}(\mathsf{ct})$
$\mathsf{ct}_2 \coloneqq (a_0 + d_0, a_1 + d_1)$
return ct_2

In contrast, homomorphic multiplication is a much more complex operation, composed of

Table 2: Comparison of homomorphic operations using the DGT and the NTT to perform the multiplication of 2^n-degree polynomials with $\log q = 323$, providing at least 80-bit security level when $n \geq 13$. Rescaling is not considered for homomorphic multiplication. Measurements, in microseconds, computed as the average of 100 independent executions on an NVIDIA Tesla V100 GPU.

	Hom. Add.			Hom. Mul.		
n	**DGT** (μs)	**NTT** (μs)	**DGT/NTT**	**DGT** (μs)	**NTT** (μs)	**DGT/NTT**
12	4.4	4.1	1.07	188.6	175.8	1.07
13	5.7	5.5	1.04	237.9	251.6	0.95
14	8.8	8.3	1.06	380.7	395.1	0.96
15	15.9	14.2	1.12	652.4	725.4	0.90
16	30.8	25.5	1.21	1232.9	1402.8	0.88

Table 3: Comparison of the latency needed for each component of a homomorphic multiplication algorithm using the DGT and the NTT to perform polynomial multiplication. We also compare them with two distinct methods for basis conversion, a canonical and an optimized version applied to AOA-NTT to increase its arithmetic density. The canonical fast basis extension algorithms are implemented following Algorithm 4 whereas the optimized uses Algorithm 5. These values were obtained through NVIDIA's Nsight Systems 2021.2.1.2 on a machine with an NVIDIA Tesla V100 GPU. The results were evaluated on 2^{16}-degree polynomials with $\log q = 1831$, composed by 53- and 54-element main and secondary basis, respectively.

	DGT (μs)	**NTT** (μs)	**Ratio**	**Opt.** (μs)	**Ratio**
ModUp	2377.0	4928.9	0.48	2592.8	0.92
ModDown	1659.2	1854.7	0.89	1896.6	0.87
Transforms	1131.2	1054.8	1.07	1062.8	1.06
Integer Op.	354.3	227.3	1.56	203.2	1.74
Total	5521.7	8065.7	0.68	5755.5	0.96

arithmetic operations, basis extensions, and transformations between the polynomial and transform domain, as shown in Algorithm 3 and discussed in Section 2.3. In Table 2, AOA-NTT presents a slowdown that increases with the ring degree. This result appears to contradict the observation presented in Section 3.1, when we observed a better performance for the NTT on these parameters.

In Table 3, we provide experimental results for each component of the homomorphic multiplication, allowing them to be observed separately on a much bigger instance. NVIDIA's profiler tool reveals that the basis extension procedures occupy a critical role in homomorphic multiplication. In AOA-NTT, ModUp takes 61% of the primitive's computing time, while ModDown takes 23%. Also, the procedures ModUp and ModDown consumes 43% and 30% of the overall running time of AOA-DGT, respectively. For the results in the first two columns, both basis switching methods were implemented as in Algorithm 4. Specially, the profiling tool indicated that ModUp is roughly 2× slower in AOA-NTT in comparison with AOA-DGT, and that ModDown presents a non-negligible slowdown of 11%.

Our implementation of ModUp for the NTT issues 2.2× more instructions than the DGT approach, suggesting that the processor scheduler is being less efficient. We verified this hypothesis by refactoring our implementation according to Algorithm 5. Experimental results for this optimized version are referred to as "Opt." in Table 3. Now, each thread handles two coefficients, and same-instruction operations are called sequentially, inducing the processor into dual

Algorithm 4 Canonical basis extension

Require: $a_{\mathcal{C}}$ an N-degree polynomial represented in the main RNS basis $\mathcal{C} = \{q_0, \ldots, q_\ell\}$.
Ensure: $a_{\mathcal{D}}$ an N-degree polynomial represented in the secondary RNS basis $\mathcal{D} = \{p_0, \ldots, p_k\}$.

```
for i = 0; i ≤ k; i = i + 1 do
    a_D^(i) = {0, ..., 0}
    for j = 0; j ≤ ℓ; j = j + 1 do
        for z = 0; z < N; z = z + 1 do
            x = a_C^(j)[z]
            aux = x · q̂_j^-1
            aux = aux mod q_j
            aux = aux · q̂_j
            aux = aux mod p_i
            aux = aux + a_D^(i)[z]
            aux = aux mod p_i
            a_D^(i)[z] = aux
        end for
    end for
end for
return a_D
```

issue mode. This operation emulates the behavior of the DGT, which benefits from the input's folding. Table 3 shows this optimization partially solves the slowdown for ModUp, although not affecting ModDown. No benefit could be measured for this technique on DGT, suggesting that the previous approach already saturates the processor's dual-issue capability.

The effect of this optimized version of ModUp on homomorphic multiplication is summarized in Table 4. By replacing that method we can observe a considerable performance gain that matches DGT's implementation in most cases. This result goes towards Badawi et al.'s claim that the DGT better fits CUDA's processing paradigm [6]. The folding property of DGT naturally increases the arithmetic density, which benefits the execution on GPUs.

Table 4: Comparison of homomorphic operations using the DGT and the NTT to accelerate polynomial multiplication for 2^n-degree polynomials and $\log q = 323$, providing at least 80-bit security level when $n \geq 13$. Rescaling is not considered for homomorphic multiplication. This experiment uses the Algorithm 5 on AOA-NTT, which increases arithmetic density per thread for the basis extension algorithms. Measurements, in microseconds, computed as the average of 100 independent executions on an NVIDIA Tesla V100 GPU.

	Hom. Mul. (μs)		
n	**DGT**	**NTT**	**Ratio**
12	188.6	175.4	1.08
13	237.9	240.8	0.99
14	380.7	381.0	1.00
15	652.4	681.6	0.96
16	1232.9	1318.6	0.94

Algorithm 5 Optimized basis extension

Require: $a_{\mathcal{C}}$ an N-degree polynomial represented in the main RNS basis $\mathcal{C} = \{q_0, \ldots, q_\ell\}$, for $N \equiv 0 \mod 2$.

Ensure: $a_{\mathcal{D}}$ an N-degree polynomial represented in the secondary RNS basis $\mathcal{D} = \{p_0, \ldots, p_k\}$, for $N \equiv 0 \mod 2$.

$N_h \coloneqq N/2$
for $i = 0;\ i \leq k;\ i = i + 1$ **do**
  $a_{\mathcal{D}}^{(i)} = \{0, \ldots, 0\}$
  for $j = 0;\ j \leq \ell;\ j = j + 1$ **do**
    for $z = 0;\ z < N_h;\ z = z + 1$ **do**
      $x_0\ ,\ x_1 = a_{\mathcal{C}}^{(j)}[z]\ ,\ a_{\mathcal{C}}^{(j)}[z + N_h]$
      $\text{aux}_0\ ,\ \text{aux}_1 = x_0 \cdot \hat{q_j}^{-1}\ ,\ x_1 \cdot \hat{q_j}^{-1}$
      $\text{aux}_0\ ,\ \text{aux}_1 = \text{aux}_0 \mod q_j\ ,\ \text{aux}_1 \mod q_j$
      $\text{aux}_0\ ,\ \text{aux}_1 = \text{aux}_0 \cdot \hat{q_j}\ ,\ \text{aux}_1 \cdot \hat{q_j}$
      $\text{aux}_0\ ,\ \text{aux}_1 = \text{aux}_0 \mod p_i\ ,\ \text{aux}_1 \mod p_i$
      $\text{aux}_0\ ,\ \text{aux}_1 = \text{aux}_0 + a_{\mathcal{D}}^{(i)}[z]\ ,\ \text{aux}_1 + a_{\mathcal{D}}^{(i)}[z + N_h]$
      $\text{aux}_0\ ,\ \text{aux}_1 = \text{aux}_0 \mod p_i\ ,\ \text{aux}_1 \mod p_i$
      $a_{\mathcal{D}}^{(i)}[z]\ ,\ a_{\mathcal{D}}^{(i)}[z + N_h] = \text{aux}_0\ ,\ \text{aux}_1$
    end for
  end for
end for
return $a_{\mathcal{D}}$

4 Case Study: Homomorphic Logistic Regression

Logistic Regression (LR) is a learning algorithm widely used to solve classification problems. Basically, LR tries to model dependence between variables [19]. For instance, in a binary classification problem, LR takes a dataset composed by n records of the form $(y_i, \mathbf{x}_i)$, with $y_i \in \{0, 1\}$ and $\mathbf{x}_i \in \mathbb{R}^d$. Its objective is to predict the value of y given $\mathbf{x}$. For that, it assumes that the distribution of y given $\mathbf{x}$ is

$$\Pr[y = 1 \mid \mathbf{x}] \coloneqq \sigma(-\mathbf{x'}^{\top}\mathbf{w}), \tag{3}$$

for some fixed vector $\mathbf{w}$ of weights, $\mathbf{x}'_i = (1 \mid \mathbf{x}_i) \in \mathbb{R}^{d+1}$, and the Sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Thus, by having an approximation $\mathbf{w}_{\text{approx}}$ for $\mathbf{w}$, we can infer y with a predictable accuracy by evaluating $\sigma(-\mathbf{x'}^{\top}\mathbf{w}_{\text{approx}})$.

LR can be seen as a neural network composed by a single hidden unit that uses the Sigmoid as the activation function. Its implementation resembles some of the challenges of implementing more complex neural networks, as selecting homomorphic compatible approximations of those functions [11]. Hence, this is a relevant application for homomorphic encryption and, in particular, to evaluate using the DGT or the NTT in an implementation of CKKS.

The computation of $\mathbf{w}$ is done in the training phase. A training dataset is used to compute a $\mathbf{w}_{\text{approx}}$ that sufficiently approximates $\mathbf{w}$, and a test dataset is used to evaluate the quality of the approximation. The Gradient Descent method is a common strategy for training [42]. It selects an initial $\mathbf{w}^0_{\text{approx}}$ and repeatedly computes $\mathbf{w}^j_{\text{approx}} = \mathbf{w}^{j-1}_{\text{approx}} - s \cdot \Delta \cdot J(\mathbf{x}_i)$ until a certain objective is satisfied (e.g. a certain number of iterations is executed), for s an arbitrary step size and J the loss function related to the problem. By modifying $\mathbf{w}_{\text{approx}}$ in the negative direction of the gradient of J, we minimize this function, obtaining a better approximation for the weight vector. However, this is computationally costly and a delicate procedure that may require thousands of multiplications to achieve a suitable $\mathbf{w}_{\text{approx}}$. Moreover, it can also require human-supervised iterations to adjust the network topology if we consider more complex neural

networks. Hence, since training is done much less frequently than inference, in this work, we focus only on the effects of the hierarchical transforms in the inference phase, when predictions are done by evaluating Equation 3 using $\mathbf{w}_{\text{approx}}$.

Lastly, it is important to notice that the Sigmoid function cannot be computed homomorphically. In this sense, similar works that also implement LR inference on FHE schemes avoid its computation by just taking the outcome of $\mathbf{x'}^{\top}\mathbf{w}$ as the classification result [10]. An alternative approach is to approximate the computation of the Sigmoid function using the related Taylor series expansion [29]. The former strategy can retain the classification result but fails to compute the probability of a specific record belonging to a particular class. Conversely, approximating the Sigmoid function requires several additional multiplicative levels per ciphertext since such approximation is usually made using around 4 and 8-degree polynomials according to the required precision.

Implementation We assume a scenario in which both the training and the model was already computed. It could be done over plaintexts through a standard library as `scikit-learn` or `PyTorch` [39, 38], or over ciphertexts [19, 11, 10]. Then, CUDA-enabled nodes need to classify encrypted data using the model given as input. This model could be made available as plaintext, contemplating cases in which the model owner is contracted to evaluate third-party data, or in encrypted form, when the computation is performed by an entity that should not have access to the model.

We follow other works in the literature that apply learning algorithms to the MNIST dataset [33]. The MNIST dataset is a classical data collection of handwritten digits, composed of black and white images representing digits between 0 and 9, each having $28 \times 28 = 784$ pixels. These images are split into a training and a test set with $60,000$ and $10,000$ records, respectively. Moreover, each image is unique regarding the handwritten style and the expected complexity for its interpretation. So, the digit recognition problem involves classifying m images among $d = 10$ classes of digits, each image having its pixel columns serialized, composing $n = 784$-element arrays.

We trained a model using a simple Python script that applied the `scikit-learn` library's LR implementation to the training set of images. The outcome is a model $\mathcal{M}$ that indicates an accuracy of 0.9167 when evaluated over the test set. This model, as Equation 3 suggests, is simply a $d \times n$ matrix of real numbers, represented using the `float` data type, such that each row relates to a classification index. So, it follows that $d = 10$ and $n = 784$.

In this MNIST context, two ciphertext designs were evaluated, as follows.

Direct: In a simple implementation, we encrypt each row of the model in a ciphertext, having their columns distributed through the slots. So, a single ciphertext stores all the columns related to a particular digit. This implies that d ciphertexts are needed to fully encrypt the model $\mathcal{M}$. We perform a similar process to encrypt images, encrypting each image into a single ciphertext. Thus, a set of m images becomes a set of m ciphertexts, each one using n slots, as shown in Algorithm 7.

Transposed: The direct approach has memory consumption efficiency but requires a sequence of slots rotations to execute LR's inference. An alternative to that is the transposition of the operands. An n-pixel image dataset with m records becomes a matrix of n rows and m columns in which each row is encrypted to a single ciphertext. However, the model encryption requires each element to be encrypted in a single ciphertext, so we can compute the inner product. This implies a considerable increase in memory consumption. The model encryption in this case requires $n \cdot d$ m-slot ciphertexts. This design is presented in Algorithm 8.

In more detail, Algorithm 7 receives an encrypted set of images such that each image is encapsulated in a single ciphertext. Moreover, it also receives the trained model, which can be encrypted or *exposed* (in plaintext). Let $\mathcal{M} \in \mathbb{R}^d \times \mathbb{R}^n$ be the model. If encrypted,

the algorithm receives a vector $\mathbf{W} = \{\text{CKKS.Encrypt}(w_{i,*}) \mid\ w_{i,*} \in \mathcal{M}\}$. Otherwise, it receives $\mathbf{W} = \{w_{i,*} \mid\ w_{i,*} \in \mathcal{M}\}$.

The input of Algorithm 8 is transposed regarding Algorithm 7. This time, the set of images is encrypted such that each ciphertext stores a single pixel of all images. Thus, a m-sized set of n-pixel images becomes n ciphertexts of m slots. When executed with the exposed model, it takes as input a matrix $\mathbf{W} = \left\{w_{i,j} \mid\ w_{i,j} \in \mathcal{M}^\top\right\}$. Otherwise, it receives a matrix of ciphertexts such that $\mathbf{W} = \left\{\text{CKKS.Encrypt}(\{w_{j,i}, \ldots, w_{j,i}\}) \mid\ w_{i,j} \in \mathcal{M}^\top\right\}$.

Initially, we consider that the inference is executed without computing the Sigmoid function. In this case, the direct strategy requires $d+m$ ciphertexts with at least n available slots to encrypt images and the model. The transposed strategy requires $n \cdot (d+1)$ ciphertexts containing at least m slots. Hence, the direct and the transposed ciphertext designs require $d + m = 10 + 10,000 = 10,010$ and $n \cdot (d+1) = 784 \cdot 11 = 8624$ ciphertexts, respectively. Also, the direct method performs two homomorphic multiplications to compute the inner product whereas the transposed requires only one homomorphic multiplication.

In the transposed strategy, the number of multiplications and the encoding of the MNIST test set of images require 10,000 slots. Thus, since our implementation only supports power-of-2 polynomials, we need that $N = 32,768$ and $\log q = 115$, composed by a 2-element RNS basis. We use the LWE estimator of Albrecht, Player, and Scott [3] to estimate the hardness of the proposed parameter set against the fastest LWE solvers currently known. For that, we obtain that the cost of running lattice attacks against the underlying LWE instance is at least 2^{1343}, comprehending the cost to run the uSVP variant of the primal lattice attack.

Considering that a maximum number of 784 slots are needed per ciphertext in the direct design, we would require that $N = 2,048$ and $\log q = 167$, composed by a 3-element RNS basis. However, the estimated hardness of such a parameter set is equivalent to a 47-bit security level, not surpassing the 100-bit security level threshold. By choosing $N = 8,192$, we obtain a parameter set with 181-bit security. Consequently, given the high-degree of the polynomial rings, the memory consumption for executing the LR's inference over the entire test set of the MNIST dataset is 3.42 GB and 8.1 GB for direct and transposed versions, respectively.

Moreover, the computation of $p_{i,j}$, the probability of the image i being classified as the digit j, is partially solved by a single homomorphic multiplication between the i-th encrypted image and the j-th encrypted row of $\mathcal{M}$. After that, each slot of the resulting ciphertext will contain the multiplication between each slot of the operands. To conclude the inner product, we need to sum all the slots of a ciphertext. That can be done as shown in Algorithm 6, which depends on a slot rotation done homomorphically [14]. Let $c' = \mathsf{rotate}(c, i)$. If c' is a rotation of k slots of c, and s_i is the i-th slot of c, then $s'_{i+k \mod n} = s_i$, where s'_j is the j-th slot of c'.

Algorithm 6 sumslots – Sum all slots of a ciphertext

Require: A ciphertext ct with n slots.
Ensure: A ciphertext ct′ with n slots such that each slot is the summation of every ct slot.

```
ct' = copy(ct)
for i = n/2; i ≥ 1; i = i/2 do
    aux = rotate(ct', i)
    ct' = ct' + aux
end for
return ct'
```

Algorithm 6, however, has a high inherent cost due to the rotation procedure. Table 5 shows the latencies measured in our implementations. When compared to the costs for homomorphic

Table 5: Execution times of our implementations of Algorithm 6 on AOA-DGT and AOA-NTT for cyclotomic polynomial rings with dimension $N = 2^{n+1}$. The optimized basis extension approach is used in AOA-NTT. We consider $\log q = 167$, that offers 80-bit security in all cases, and $\log q = 323$, that achieves this security level when $n \geq 13$. Measurements taken on a Google Cloud instance with an NVIDIA Tesla V100 GPU.

Sumslots

$\log(q)$	323			167		
n	**DGT**	**NTT**	**Ratio**	**DGT**	**NTT**	**Ratio**
12	1.34	1.21	1.11	1.05	0.98	1.08
13	1.92	1.89	1.01	1.31	1.22	1.07
14	3.30	3.27	1.01	2.02	1.90	1.07
15	6.44	6.23	1.03	3.44	3.41	1.01
16	13.34	12.84	1.04	6.87	6.59	1.04

multiplication, presented in Table 3, it becomes clear that the slots summation is the most costly part of the LR inference with the direct approach in both AOA-DGT and AOA-NTT.

Let $s = \{s_0, s_1, \ldots, s_{n-1}\}$ be the slots of a ciphertext c. The outcome of the c-slot summation will be a ciphertext c' such that all its slots will be equal to $\sum_{i=0}^{n-1} s_i$. To improve space efficiency, we would like to store information about the suitability of an image i to all candidate digits in a single ciphertext, with d being the slot related to the digit d. We use $\mathsf{DiscardSlotsExcept}(c, d)$ for that. It returns the homomorphic multiplication of c by the encryption of $a = \{a_0, a_1, \ldots, a_{n-1}\}$ such that $a_i = 1$, if $i = d$, and 0, otherwise.

By following the direct approach, we can compute homomorphically a vector of ciphertexts, denoted **pred**, such that element with index i stores the encryption of the inner product between the image i and the weight vector related to each class, as shown in Algorithm 7. By design, **pred** supports n slots but only d will be possibly different than zero.

Algorithm 7 Direct version of an encrypted LR inference

Require: $\mathbf{W}$ as a n-slot d-element representation of the trained model; $\mathbf{X} \in \mathbb{R}^m \times \mathbb{R}^n$ as a set of images; and $X_i = \text{CKKS.Encrypt}(x_i)$ for $x_i \in \mathbf{X}$.

Ensure: $c \in \mathbb{Z}^n$ such that $c_i = d$, if x_i is classified as the digit d.

Online phase

for $j = 0;\ j < d;\ j = j + 1$ **do**
  for $i = 0;\ i < m;\ i = i + 1$ **do**
    $p = \mathsf{sumslots}(X_i \cdot W_j)$
    $p = \mathsf{DiscardSlotsExcept}(p, d)$
    $\text{pred}_i = \text{pred}_i + p$
  end for
end for

Offline phase

$\text{pred} = \text{CKKS.Decrypt}(\text{pred})$
for $i = 0;\ i < m;\ i = i + 1$ **do**
  $c_i = \mathsf{argmax}(\text{pred}_i)$
end for
return c

On the other hand, the transposed design does not need the sumslots procedure to compute the inner product. Instead, the operands' transposition enables its replacement by a sequence of

homomorphic multiplications and additions, as seen in Algorithm 8. As we avoid the expensive Algorithm 6, a considerable performance improvement of 70× is observed when the model is encrypted. When the model is given as plaintext, the performance is improved by 700×, as presented in Table 6.

Algorithm 8 Transposed version of an encrypted LR inference

Require: $\mathbf{W}$ as a $n \times d$ matrix representing the transposed trained model; $\mathbf{X}^\top \in \mathbb{R}^m \times \mathbb{R}^n$ as a set of transposed images; and $X_i = \text{CKKS.Encrypt}(x_i)$ for $x_i \in \mathbf{X}^\top$.
Ensure: $c \in \mathbb{Z}^n$ such that $c_i = d$, if x_i is classified as the digit d.

Online phase

for $j = 0;\ j < d;\ j = j + 1$ **do**
  $\text{pred}_j = 0$
  for $i = 0;\ i < n;\ i = i + 1$ **do**
    $\text{pred}_j = \text{pred}_j + X_i \cdot W_{j,i}$
  end for
end for

Offline phase

$\text{pred} = \text{CKKS.Decrypt}(\text{pred})$
for $i = 0;\ i < m;\ i = i + 1$ **do**
  $c_i = \mathsf{argmax}(\text{pred}_{*,i})$
end for
return c

Algorithms 7 and 8 are composed of two phases. The first one, called *online phase*, is the more computationally intensive and, as aforementioned, can be executed homomorphically on a powerful device. The dataset is kept encrypted, and no knowledge of the decryption key is needed. The second, called *offline phase*, is the result delivery phase when the matrix of probabilities is decrypted, and the prediction is made by selecting the index of the maximum element of the array. This is a much less intensive step that can be executed even on a low-power device. However, as discussed by Bajard et al., the `sign` or `argmax` works also as filters that truncate data that can be used to retrieve confidential information. Thus, by not executing these functions homomorphically, an adversary may be able to leak the entire trained model [9]. Moreover, in some cases, as with the Support Vector Machine algorithm, this could also imply leakage of the input data. Thus, the solution presented in those algorithms assumes a secure executing environment for the decryption keys and the decrypted matrix of probabilities.

Table 6 presents the latencies for the minimum parameter set that offers at least a 128-bit security level and provides the required multiplicative depth for each approach. The accumulated slowdown for sumslots on AOA-DGT, presented in Table 5, impacts its latency on the direct approach. Nonetheless, the transpose approach does not depend on it and thus AOA-DGT and AOA-NTT show similar execution times.

No impact was detected for the AOA-NTT implementation done with the optimization discussed on Section 3.2. The ModUp function performance is directly impacted by the basis sizes, and in these instances, with 3 and 2-sized bases, that optimization does not seem relevant.

When we consider the canonical formulation of LR inference, which depends on the Sigmoid function, something different can be observed. We follow the literature and approximate this function using a polynomial obtained by the truncation at the 8-th term of its Taylor series approximation [1]. Evaluated through Horner's rule, it increases by 8 the number of homomorphic multiplications needed for inference, affecting performance and memory consumption. Table 7 presents our measurements. Motivated by the increased memory requirement, we executed this experiment on an NVIDIA Tesla A100. In the direct design, the additional multiplications imply

Table 6: Comparison of the latency per record required to compute the online phase of the LR inference over records in the test set of the MNIST database with the model encrypted and as plaintext. The basic design runs with $N = 2^{13}$ and $\log q = 167$ while the transposed version runs with $N = 2^{15}$ and $\log q = 115$, offering 172-bit and 1343-bit security level, respectively, according to Albrecht's estimator [3]. Measurements in microseconds were taken on a Google Cloud instance with an NVIDIA Tesla V100 GPU.

Model	**Encrypted**	**Exposed**	**Encrypted**$^\top$	**Exposed**$^\top$
DGT	15461.2	13885.9	226.8	15.6
NTT	14396.8	12827.6	238.5	15.2
DGT/NTT	1.07	1.08	0.95	1.03

Table 7: Comparison of the latency per record required to compute the online phase of the LR inference over records in the test set of the MNIST database with the model encrypted and as plaintext following the textbook algorithm, with the homomorphic computation of the Sigmoid function. The basic design runs with $N = 2^{14}$ and $\log q = 583$ while the transposed version runs with $N = 2^{15}$ and $\log q = 531$, offering 94-bit and 225-bit security level, respectively, according to Albrecht's estimator [3]. Measurements in microseconds were taken on a Google Cloud instance with an NVIDIA Tesla A100 GPU.

Model	**Encrypted**	**Exposed**	**Encrypted**$^\top$	**Exposed**$^\top$
DGT	60279.7	56013.2	475.7	33.1
NTT	65998.5	60968.6	562.6	33.4
DGT/NTT	0.91	0.92	0.85	0.98
NTT-Opt	58200.0	56053.7	517.6	32.5
DGT/NTT-Opt	1.04	1.00	0.92	1.02
NTT/NTT-Opt	1.13	1.09	1.09	1.03

a reduction of the security level of about 50 bits if we choose $N = 2^{13}$. Thus, by setting $N = 2^{14}$, we could obtain a security level of at least 80 bits.

In bigger instances, the higher arithmetic density of AOA-DGT implied in speedups in all cases when compared to the non-optimized AOA-NTT. However, when the optimization of the ModUp function takes place, AOA-NTT is capable of reversing that scenario. The transposed design, implemented with a plaintext model, presents a consistent similarity between both implementations. Homomorphic addition and multiplication between ciphertexts and plaintexts are coefficient-wise operations, as shown in Section 2.3.

The folding procedure used by the DGT does not imply the reduction of the number of required operations. However, the scalability of the DGT-based implementation highlights and reduces the execution time when more complex methods are considered as basis extension procedures rotation. This behavior agrees with the conclusion in Section 3.2, which suggests that the DGT implementation better fits CUDA's processing paradigm and that its characteristics have to be ported to AOA-NTT so we can observe a similar, or even superior, performance.

5 Conclusion

Most implementations based on the RLWE problem use the NTT to efficiently compute the polynomial multiplication in the ring $\mathbb{Z}_p[x]/(x^N + 1)$ for N a power of two and p a prime. However, recent works [2, 6, 4] claim that polynomial multiplication can be more efficiently implemented on GPUs using the DGT instead of the NTT.

In this context, we developed two implementations of the CKKS cryptosystem following the

same blueprint but diverging on the transform, using either the DGT or the NTT for polynomial multiplication. We refer to them as AOA-DGT and AOA-NTT. We performed benchmarks on both implementations by first directly comparing the NTT and DGT transforms as standalone, and then their implementations of CKKS' homomorphic multiplication. After that, we extended our evaluation to the context of logistic regression inference.

Considering the latency for the DGT or NTT isolated, we observed an overall similarity between both implementations in bigger instances, i.e., the ring dimension ranging from 16384 to 65536. For smaller instances, the NTT outperforms the DGT but with a clear trend towards equalization in large rings. The exception occurs on 8192-degree polynomial rings when the DGT reaches its warp efficiency peak. In that case, we observed a speedup of up to 10% for AOA-DGT. Nevertheless, we could not observe the same behavior for the homomorphic multiplication in CKKS. In this case, AOA-DGT rapidly surpasses the AOA-NTT performance. A deep analysis through NVIDIA's profiler tool showed that the DGT data structure provides a higher arithmetic density, which efficiently explores the processing hardware, especially on the methods for basis extension. We were able to successfully match the performance of both implementations by porting the DGT data representation to AOA-NTT

Lastly, we evaluated the impact of both AOA-DGT and AOA-NTT on a logistic regression inference performed homomorphically over ciphertexts. We trained a model to classify handwritten digits in the MNIST dataset and measured the execution time per image in different configurations, exploring two ciphertext designs. We concluded that, in bigger instances, AOA-DGT presents a considerable speedup when compared with a standard implementation on the AOA-NTT, i.e. without methods that increase arithmetic density on CUDA kernels.

Hence, our results indicate that the data representation of the DGT on a CKKS implementation increases the implementation's overall performance by assisting the programmer to take implementation decisions that more efficiently explore the GPU hardware.

Acknowledgments

This work was supported in part by the Brazilian National Council for Scientific and Technological Development (CNPq), grants number 164489/2018-5 and 203175/2019-0; and the Brazilian Coordination for the Improvement of Higher Education Personnel Foundation (CAPES) grant number 1591123. We specially thank Google for GCP Research Credits Program under number 106101194491; the Concordium Blockchain Research Center at Aarhus University (COBRA), Denmark; and the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No. 803096 (SPEC).

References

[1] Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

[2] Ahmad Al Badawi, Bharadwaj Veeravalli, and Khin Mi Mi Aung. Efficient polynomial multiplication via modified discrete galois transform and negacyclic convolution. In *Future of Information and Communication Conference*, pages 666–682. Springer, 2018.

[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] Pedro Geraldo M. R. Alves, Jheyne N. Ortiz, and Diego F. Aranha. Faster homomorphic encryption over gpgpus via hierarchical DGT. *IACR Cryptol. ePrint Arch.*, 2020:861, 2020.

[5] Ahmad Al Badawi, Yuriy Polyakov, Khin Mi Mi Aung, Bharadwaj Veeravalli, and Kurt Rohloff. Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme. *IEEE Trans. Emerg. Top. Comput.*, 9(2):941–956, 2021.

[6] Ahmad Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun, and Khin Mi Mi Aung. High-performance FV somewhat homomorphic encryption on gpus: An implementation using CUDA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):70–95, 2018.

[7] David H. Bailey. FFTs in external or hierarchical memory. *J. Supercomput.*, 4(1):23–35, 1990.

[8] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 423–442. Springer, 2016.

[9] Jean-Claude Bajard, Paulo Martins, Leonel Sousa, and Vincent Zucca. Improving the efficiency of SVM classification with FHE. *IEEE Trans. Inf. Forensics Secur.*, 15:1709–1722, 2020.

[10] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[11] Flávio Bergamaschi, Shai Halevi, Tzipora T. Halevi, and Hamish Hunt. Homomorphic training of 30, 000 logistic regression models. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 592–611. Springer, 2019.

[12] Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2013.

[13] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electron. Colloquium Comput. Complex.*, page 111, 2011.

[14] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 347–368. Springer, 2018.

[15] 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, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[16] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander A. Schwarzmann, editors, *Cyber Security Cryptography and*

Machine Learning - 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8-9, 2021, Proceedings, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2021.

[17] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[18] Richard E. Crandall. Integer convolution via split-radix fast Galois transform, 1999.

[19] Jack L. H. Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing real work with FHE: the case of logistic regression. In Michael Brenner and Kurt Rohloff, editors, *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC@CCS 2018, Toronto, ON, Canada, October 19, 2018*, pages 1–12. ACM, 2018.

[20] Wei Dai, Yarkın Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savaş, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy abe scheme. *IEEE Transactions on Information Forensics and Security*, 13(5):1169–1184, 2018.

[21] Wei Dai and Berk Sunar. cuhe: A homomorphic encryption accelerator library. In Enes Pasalic and Lars R. Knudsen, editors, *Cryptography and Information Security in the Balkans*, pages 169–186, Cham, 2016. Springer International Publishing.

[22] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[23] Axel Feldmann, Nikola Samardzic, Aleksandar Krastev, Srini Devadas, Ron Dreslinski, Karim Eldefrawy, Nicholas Genise, Chris Peikert, and Daniel Sánchez. F1: A fast and programmable accelerator for fully homomorphic encryption (extended version). *CoRR*, abs/2109.05371, 2021.

[24] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[25] W. M. Gentleman and G. Sande. Fast fourier transforms: For fun and profit. In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, AFIPS '66 (Fall), page 563–578, New York, NY, USA, 1966. Association for Computing Machinery.

[26] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[27] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[28] Naga K. Govindaraju, Brandon Lloyd, Yuri Dotsenko, Burton Smith, and John Manferdelli. High performance discrete fourier transforms on graphics processors. In *Proceedings of the ACM/IEEE Conference on High Performance Computing, SC 2008, November 15-21, 2008, Austin, Texas, USA*, page 2. IEEE/ACM, 2008.

[29] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data. *IACR Cryptol. ePrint Arch.*, page 662, 2018.

[30] David Harvey. Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation*, 60:113–119, 2014.

[31] Wonkyung Jung, Sangpyo Kim, Jung Ho Ahn, Jung Hee Cheon, and Younho Lee. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. Cryptology ePrint Archive, Report 2021/508, 2021. `https://ia.cr/2021/508`.

[32] Sangpyo Kim, Wonkyung Jung, Jaiyoung Park, and Jung Ho Ahn. Accelerating number theoretic transformations for bootstrappable homomorphic encryption on gpus. In *IISWC*, pages 264–275. IEEE, 2020.

[33] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database, 2010.

[34] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234. ACM, 2012.

[35] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A Modest Proposal for FFT Hashing. In Kaisa Nyberg, editor, *Fast Software Encryption*, pages 54–72, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[36] NVIDIA. NVIDIA Nsight Systems. https://developer.nvidia.com/nsight-systems, 2021. Last accessed: 2021-10-13.

[37] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[39] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[40] John M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25:365–374, 1971. URL: `http://cr.yp.to/bib/entries.html#1971/pollard`.

[41] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[42] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

2.4 Privacy-preserving data analytics

In joint work with Rune Jacobsen and Ali Marandi from the Department of Electrical and Computer Engineering at Aarhus University, we propose an FHE-based homomorphic encryption method to protect privacy while computing smart meters' data. It enables a network of smart meters that collect data from the user and share computation along several nodes in the network without revealing information during the calculation. Moreover, we compare it with Paillier and BGN-based methods and present the practical trade-offs. To prevent the drawbacks of centralized trusted authority, we emphasize the importance of using a distributed key generation and distribution system that works with the proposed CKKS-based method.

This publication title is "Lattice-based Homomorphic Encryption for Privacy-Preserving Smart Meter Data Analytics" and is under review at the The Computer Journal (COMPJ).

Lattice-based Homomorphic Encryption for Privacy-Preserving Smart Meter Data Analytics

Ali Marandi[1] Pedro G. M. R. Alves[2] Diego F. Aranha[3]
Rune Hylsberg Jacobsen[4]

Department of Applied Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark[1],
Institute of Computing, University of Campinas, Campinas, Brazil[2]
Department of Computer Science, Aarhus University, Aarhus, Denmark[3]
Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark[4]

To be published

Abstract

Privacy-preserving smart meter data collection and analysis are critical for optimizing smart grid environments without compromising privacy. Using homomorphic encryption techniques, smart meters can encrypt collected data to ensure confidentiality, and other untrusted nodes can further compute the encrypted data without being able to recover the underlying plaintext. As an illustrative example, this approach can be useful to compute the monthly electricity consumption without violating consumer privacy by collecting fine-granular data through small increments of time. Towards that end, we propose an architecture for privacy-preserving smart meter data collection, aggregation, and analysis based on lattice-based homomorphic encryption. Furthermore, we compare the proposed method with the Paillier and Boneh-Goh-Nissim (BGN) cryptosystems, which are popular alternatives for homomorphic encryption in smart grids. We consider different services with different requirements in terms of multiplicative depth, e.g., billing, variance, and non-linear SVM classification. Accordingly, we measure and show the practical overhead of using the proposed homomorphic encryption method in terms of communication traffic (ciphertext size) and latency. Our results show that lattice-based homomorphic encryption is more efficient than Paillier and BGN for both multiplication and addition operations while offering more flexibility in terms of the computation that can be evaluated homomorphically.

Keywords— Homomorphic encryption, smart meter data analysis, privacy, security

1 Introduction

Smart grids improve traditional power grids with modern information technologies to enable higher efficiency and robustness in electricity production and distribution. To reach these goals, a smart grid requires real-time monitoring of the electricity grid and efficient communications with the consumers and between its components [1]. The expected advantages of adopting the technology are reducing energy consumption and cost, and preventing major service interruptions.

The use of smart meters enables near real-time monitoring, bi-directional communications with the smart grid, demand forecast and planning, and fraud detection. Typically, smart meters communicate with a gateway that aggregates the received smart meter data and forwards it to the control center located multiple hops away. Based on the received aggregated data, the control center can observe the smart grid status. Accordingly, the control center attempts to optimize the energy consumption and balance the electricity load [2].

User privacy should be preserved during smart meter data collection, aggregation, and processing. Otherwise, third parties can detect user behaviors from smart meter data [3, 4]. For example, they can easily spot periods of user inactivity. Privacy preservation is critical specifically when smart meter data is fine-grained. Recently, homomorphic encryption (HE) techniques [5] have become popular for protecting smart meter data privacy. HE techniques allow computation over ciphertexts without decryption, in a way that smart meters or their associated gateways can encrypt the data for untrusted nodes to run statistical functions without learning anything from the operands or the outcome. The latter, also encrypted, can be only decrypted by the entity possessing the secret key.

The Paillier cryptosystem [6] is a classical HE method that supports one multiplication between a ciphertext (CT) and a plaintext (PT) and the addition of CTs. The Boneh-Goh-Nissim (BGN) scheme allows not only additions but also a single multiplication between ciphertexts, making it possible to compute statistical operations up to degree 2 on encrypted data, such as calculating statistical variance [7]. Due to this feature and being relatively efficient, BGN and Paillier cryptosystems became popular for smart meter data analytics [1, 8]. However, these cryptosystems rely on hardness assumptions that are not quantum-safe: integer factoring and discrete logarithm in bilinear pairing groups, respectively, can be solved efficiently in a quantum computer. Therefore, these cryptosystems do not provide long-lasting security, an important requirement for systems deployed in the near future but which need to operate for decades without radical changes. Furthermore, their limitations on the supported types of homomorphic operations considerably reduce the scope of applications that may use them.

We propose our architecture over CKKS [9], a post-quantum scheme that depends on the Ring Learning with Errors (RLWE) assumption over lattices [10] and supports a much higher number of additions and multiplications of ciphertexts. Although it permits doing more than one multiplication on the encrypted smart meter data, the multiplicative depth of the circuit evaluated over ciphertexts needs to be known in advance before encryption to select the parameters accordingly.

Going beyond the choice of HE scheme, key management also needs to be designed carefully. Previous works assume that a *trusted authority* generates HE encryption, evaluation, and decryption keys [1, 8, 11]. This is a centralized design, creating a single point of failure. Therefore, we propose that multiple nodes cooperate based on a distributed HE scheme, implementing a secure multi-party computation (MPC) strategy.

In terms of functionality, the control center typically needs to apply statistical functions, e.g., average, variance, and one-way analysis of variance (ANOVA) to the received smart meters' aggregated data. To balance its computational load and save bandwidth, the control center can outsource part of the computation to other network components closer to smart meters. For that, a fog composed of intermediary nodes may be built between smart meters and the control center. A comprehensive analysis of all the wired and wireless technologies for the smart grid concludes by recommending the LTE as the communication infrastructure for the smart grid [12]. In this paper, smart meters use LTE to communicate their data to the fog node that is connected to the Evolved Node B (eNB). We evaluate the performance of the system including the CT sizes for different statistical operations and the delay to communicate them to the fog node.

In summary, the contributions of our work can be summarized as follows:

- Propose a CKKS-based HE method for privacy-preserving smart meter data analysis.

- Show the practical trade-offs of using the proposed CKKS-based HE method for different services with different requirements in terms of multiplication depth.
- Compare the proposed CKKS HE method with Paillier and BGN cryptosystems [6, 7].
- Propose a decentralized key center based on MPC concepts that work with the proposed CKKS-based HE method and avoids the drawbacks of having a centralized trusted authority.

The remainder of this paper is structured as follows. Section 2 discusses related works. Next, Section 3 describes the system architecture. Section 4 presents the underlying HE method. Then, Section 5 presents the proposed HE method. After, Section 6 assesses system performance. Finally, Section 7 concludes the paper.

2 Related Works

To perform privacy-preserving smart meter data aggregation, previous works (e.g., [1, 8]) make use of Paillier or BGN cryptosystems [6, 7]. The Paillier cryptosystem allows a multiplication between a ciphertext and a plaintext, while BGN permits a multiplication of two ciphertexts.

Chen et al. [1] assumed that a gateway was responsible for authentication, smart meter data aggregation, and forwarding the aggregated smart meter data to the control center. The gateway was enabled to perform statistical functions with degree $\leq$ 2, so it can perform sum, variance, and one-way ANOVA over smart meter data, and communicate their results to the control center. However, if the control center retrieves the aggregated data of smart meters $\{sm_1, sm_2, ..., sm_n\}$ and also the aggregated data of smart meters $\{sm_1, sm_2, ..., sm_{n-1}\}$, it can commit a differential attack and reveal the individual consumption of smart meter sm_n. The threat model assumes that the gateway and the control center are both trusted but the nodes between users and the gateway, and also between the gateway and the control center, might eavesdrop on the communications and act as a malicious adversary. The adversary might compromise the gateway or control center databases. To cope with this, the authors suggested that the gateway uses differential privacy techniques over smart meter data aggregations. The main drawback of [1] is that a trusted authority is responsible for system initialization, i.e., setting up system parameters as well as key generation and distribution. If this entity is successfully attacked, the security of the entire network gets compromised. This single point of failure is avoided in this paper through the use of secure MPC to form a distributed key center, providing robustness to key generation and distribution processes.

The work in [11] emphasizes that fog nodes can assist the utility company and mediate interactions between the control center and the consumers. Therefore, fog nodes can handle service requests and commands from the control center to provide a variety of services, such as applying smart electricity pricing strategies, calculating the overall electricity consumption, smart meter data analysis, and providing the consumers additional electricity when needed. It uses somewhat homomorphic encryption (SHE) schemes to preserve consumers' privacy against fog nodes and enable support to evaluate more complex arithmetic circuits with both additions and multiplications. SHE is more restricted than leveled homomorphic encryption (LHE), only allowing the homomorphic evaluation of a subset of circuits of the wider range of circuits that can be evaluated with LHE. For instance, BGN is a HE scheme that builds ciphertexts that may be homomorphically evaluated for addition and multiplication. However, since it only supports a single multiplication, these ciphertexts can only be evaluated on circuits composed exclusively by additions or which end with a single multiplication [7]. CKKS, on the other hand, produces ciphertexts that can be evaluated for addition and multiplication without restrictions if we consider the use of a bootstrapping method (becoming, then, a fully homomorphic scheme [13]),

or at least may have the encryption parameters adjusted to support an arbitrary circuit, meaning that it is classified as an LHE scheme.

To ascertain fair pricing and consider distributed electricity generation, Zhao et al. [11] further assume that the control center can dynamically apply diversified pricing strategies. It is assumed that every 15 minutes there is an electricity price p_γ. To perform billing based on diversified electricity prices, the fog node can multiply the encryption of p_γ to the encryption of the consumer's electricity consumption during the 15-minute period. This use case satisfies scenarios when the electricity price may be subjected to non-disclosure agreements. The authors considered that the control center can request the fog node to perform a one-way ANOVA on encrypted consumption reports to assess the impact of diversified electricity prices on consumers' electricity consumption and subsequently select win-win pricing strategies. The authors assumed that the control center can be trusted and has access to the decryption keys. However, for example, in the case of temporal aggregation, the fog node performs the aggregation homomorphically and sends the encrypted sum to the control center. Thus, the control center can decrypt the sum, but cannot reveal individual consumption data.

Jokar et al. [14] discussed the problem of energy theft detection. Energy theft events must be detected with high accuracy because the inspection is costly for utility companies. Hence, their work attempts to decouple other seems-to-be-anomalous events from energy theft, e.g., change of appliances, absence of residents, etc. After processing the energy data using the k-means algorithm, the data is classified among multiple classes. Therefore, the work in [14] suggested multi-class SVM classification using the radial basis function, which is one of the most common kernels for SVM. SVM classification on homomorphically encrypted data is investigated in [15] and [16]. Rahulamathavan et al. [15] uses the Paillier cryptosystem [6], which can only support one homomorphic multiplication between a ciphertext and a plaintext. However, SVM classification using radial basis or a polynomial kernel function (as considered by [15]) cannot be done with only one multiplication. To cope with this limitation of the Paillier cryptosystem, the work in [15] suggested a two-phase procedure that sends the ciphertexts to a trusted server, that decrypts them, applies the kernel function, and encrypts the result before returning it. In this paper, we show that the proposed CKKS-based HE method does not require any communications with a trusted server to perform the SVM classification, which is a significant advantage compared to [15]. Bajard et al. [16] presented a detailed description of the problem of instantiating SVM over FHE and describe techniques for its efficient implementation, as the formulation of the sign evaluation, which needs to be expressed as an arithmetic circuit. For that, they proposed a function approximation through the Newton-Raphson procedure. Moreover, their work targeted parallel implementation by decomposing the operands using the Residue Number System (RNS) and doing the computation on a CUDA-enabled GPU. Lastly, they offer a polynomial evaluation algorithm that minimizes multiplicative depth, something considerably important for FHE schemes.

In terms of implementation frameworks, TenSEAL is a generic framework that implements many learning algorithms, including logistic regression and convolutional neural networks [17]. Their goal is to offer an API similar to TensorFlow or PyTorch. In this paper, we used TenSEAL for implementing the proposed CKKS-based HE method.

The work in [12] compared the smart grid system performance using wired (PLC) and wireless (WiMAX, LTE) technologies. The results confirm that using full wireless technology (in this case LTE) results in better performance. More precisely, using PLC in the low voltage area and LTE at the medium/high voltage areas results in higher latencies compared to the case where LTE was used at all voltage areas. Furthermore, their work shows that the requirements of the IEC 61850 standard are satisfied using the LTE communication medium, while the use of PLC combined with LTE leads to longer delays than the case where only LTE was used. Therefore, it is not suitable for mission-critical services, e.g., smart grid protection.

The core property of a homomorphic scheme, of enabling the manipulation of the encrypted

plaintext, causes a weakness in systems built upon it. While a ciphertext created using a non-homomorphic scheme reports tampering if a single bit is modified by a malicious entity, since its decryption becomes completely disconnected from the plaintext domain, homomorphic ciphertexts can be manipulated without leaving evidence. For instance, in the context of this work, a smart meter may submit an encrypted measurement to a fog node expecting that it will compute the electricity bill using a particular price. If the fog node is behaving maliciously, it can use the wrong price table. The outcome, decrypted by the smart meter or the utility company, may be indistinguishable from the expected outcome.

A verifiable computation (VC) scheme, as defined by Bois et al. [18], can be implemented together with a HE scheme to validate that no adulteration occurred when a ciphertext, encrypted by a HE scheme, is submitted to an untrusted entity to be evaluated on a particular circuit. Fiore et al. [19] define a VC scheme using four main procedures: 1) KeyGen(f, λ) that receives a function f and generates a public key encoding f and a secret key to be stored securely; 2) ProbGen$_{\mathsf{sk}}(x)$ that encodes an input x using sk, resulting in a public value σ_x, which is given to the entity that will perform the computation, and a secret value τ_x, which is given to the entity that will verify the outcome; 3) Compute$_{\mathsf{pk}}(\sigma_x)$ that combines σ_x and pk to compute σ_y, an encoded version of $y = f(x)$; 4) Verify$_{\mathsf{sk}}(\tau_x, \sigma_y)$ that returns acc $= 1$ if the verifier accepts that $y = f(x)$, or acc $= 0$ otherwise.

An important property is that the verifier must be able to verify the correctness of y much faster than running the actual computation. Fiore et al.[19] propose a classical instantiation of a VC scheme that supports only the evaluation of quadratic functions on top of the Brakerski and Vaikuntanathan (BV) cryptosystem [20]. Bois et al. [18] and Fiore et al. [21] improve that work by increasing the range of parameters supported, boosting performance, but also the complexity of supported functions. While [19] supports only quadratic functions, [18] and [21] also handle computations of constant multiplicative depth. However, the state of the art still does not support the main HE schemes available in the literature and is under discussion for standardization, such as the CKKS, BFV, or TFHE [9, 22, 23]. Moreover, it also does not provide a robust solution that addresses all the arithmetic circuits supported by those.

3 System Model

This Section presents the system architecture, composed of smart meters measuring the energy consumed by users; a control center, which manages the network; and several supporting entities used to perform computation and manage cryptographic keys used by smart meters and the control center. Also, we present our trust and threat model.

3.1 System Architecture

The system architecture (Figure 1) considers a utility company responsible for the generation, storage, and distribution of electricity. The control center analyzes the smart meter data and monitors the substations by sending commands to them. Substations distribute electricity to the customers. We assume that the control center has deployed fog nodes close to the substations that can process aggregated smart meter data for their geographical area. This reduces the processing load of the control center. Following related work, we also use the LTE technology for communicating smart meters' data to an eNB that is directly connected to a fog node. The fog node is a server that is responsible for performing the required computations on the smart meter data.

Previous works assume that a single node called *trusted authority* generates and distributes the cryptographic keys [1, 11]. However, it is not ideal to generate and distribute those cryptographic keys in a centralized way because the trusted authority becomes a single point of failure. Therefore, if it is compromised by third parties or a malicious administrator, the entire

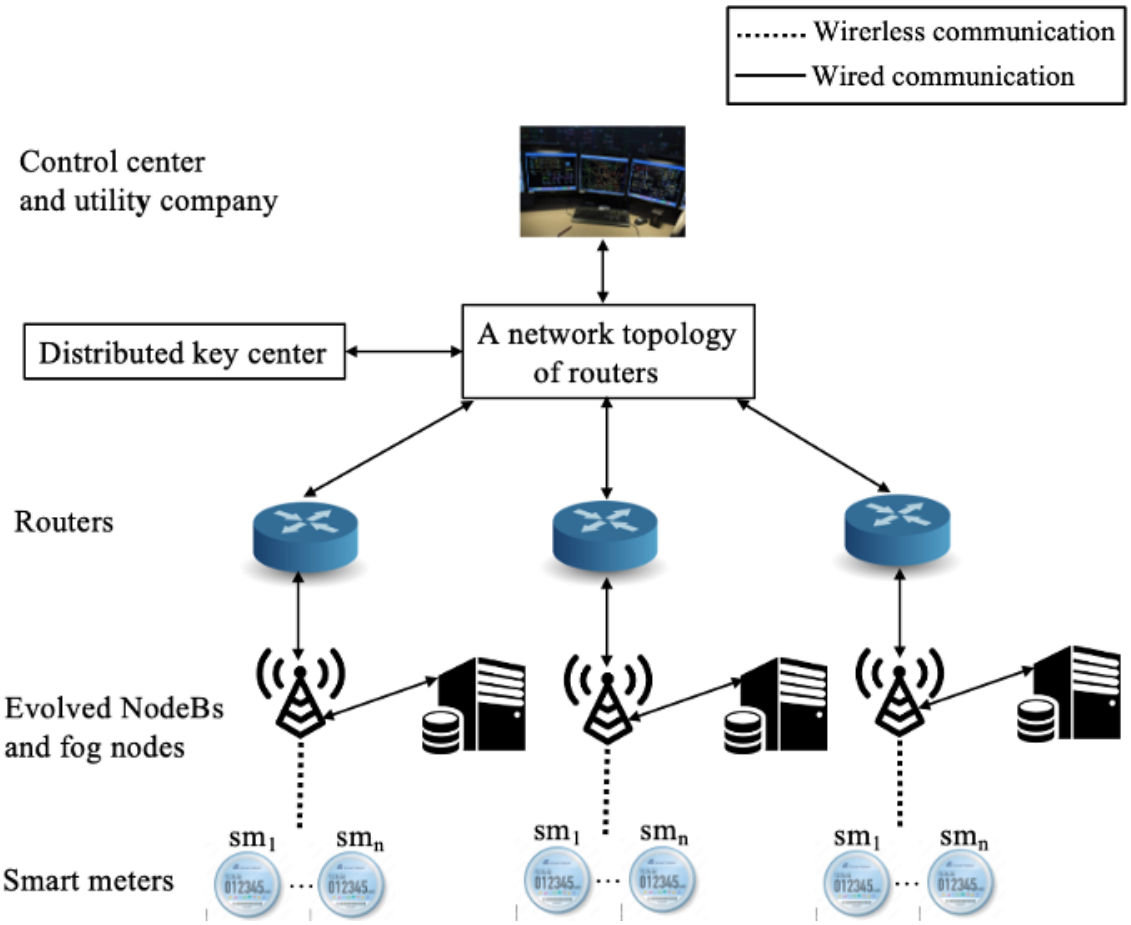


Figure 1: System architecture consisting of smart meters, eNodeBs and fog nodes, routers, key center, control center, and utility company.

network becomes vulnerable. A better design consists of nodes working together to perform the key center tasks with MPC protocols. Mouchet et al. [24] propose a distributed version of BFV, an FHE scheme, that may address that issue [22]. In this paper, we extend their contribution by porting it to the CKKS, as discussed in Section 4.2. We propose a *distributed key center* that generates and distributes the required cryptographic keys among the network nodes involved in the encryption and evaluation processes. The distributed key center consists of a group of nodes that run MPC protocols. We assume a key center gateway node is responsible for communications between the key center structure and other network nodes. Note that the key center gateway could be one of the key center nodes, which can generate and distribute the public data required by the protocols; and conclude the collective decryption protocol. Smart meters use the collective public keys to encrypt their data. Fog nodes use the collective evaluation keys to homomorphically evaluate the aggregated smart meter data. No decryption key is communicated through any nodes in the network. All decryption requests must be submitted to the key center. We assume that the fog nodes are not trusted, and thus no decryption would be authorized for them. The control center might request to decrypt the aggregated smart meter data that it has received from the fog nodes.

In the following, we discuss some models for secure communications. These instances are simple in the sense that they do not offer a mechanism to assert data integrity. We consider NIST's definition of data integrity, which requires the data not to be modified in an unauthorized manner during storage, processing, and while it is transmitted [25]. We cannot guarantee that the data used during the computation came from the user we expect to, or it was evaluated

through the agreed arithmetic circuit (e.g., an end-user cannot be sure that the received bill was calculated by a sequence of additions of measurements made by the user's smart meter). Thus, in all these models, we have to trust that the entities will evaluate data correctly, and the users will not try to contest the outcome.

3.2 Trust Model

A property intrinsic to HE is the malleability of ciphertexts, i.e., anyone in possession of ciphertext can evaluate it on an arbitrary circuit. This may result in a new ciphertext that keeps no trace of what happened. In a non-malleable scheme, any manipulation results in an invalid ciphertext, and the responsible entities could detect that on decryption. Thus, to mitigate the risks, we assume to have secure communication between the system components, asserting data integrity by protecting homomorphically encrypted ciphertexts from malicious parties interested in exploiting their natural malleability properties (i.e., one with access to them would be able to perform arithmetic operations without being noticed). This could be done using a verifiable computation (VC) scheme working as a receipt of all arithmetic operations done with a particular ciphertext, as discussed in Section 2. This protects the ciphertext against malicious nodes, which can be part of a communication path within the network.

Our trust model assumes the following network properties: 1) smart meters are trusted to measure energy consumption correctly, so the utility company is trusted to operate smart meters correctly; 2) the key center cannot collude with other entities to decrypt confidential data that is not supposed to be revealed, although if the key center is implemented through MPC protocols, a subset of colluding parties can be tolerated; 3) the key center must be capable of generating and sharing securely public keys for all relevant cryptographic schemes in use; 4) to provide the functionality needed for HE, the key center generates the public keys and securely shares them with each smart meter cluster. Furthermore, the key center will be available for decrypting the results of computations; 5) the secret keys only need to be generated once or every time the cryptographic keys need to be refreshed, e.g., when a new party joins or leaves the key center party. It is required to refresh all the related public keys when that happens, 6) after any computation, decryption can only be executed by the key center. Thus, decryption requests are forwarded to the key center. If the key center accepts a decryption request, it will decrypt the ciphertext and return its plaintext through a secure channel. When a node sends a decryption request, it has to digitally sign the decryption request so that the key center can verify the signature of requesting node to know whether that node is eligible to send a decryption request.

3.3 Threat Model

We consider an adversary to be a member of the network or a third party that takes control of a member of the network. Moreover, such an adversary may assume control of more than one network member to increase its attack surface by exploring possibilities of collusion.

By collusion, we consider two or more entities combining their data to break data secrecy or produce incorrect computation output masked as correct. For instance, fog nodes receive users' data and are allowed to do computation but not to decrypt. Two or more nodes might collude, combining their shares of the evaluation key to learn something about the decryption key or producing erroneous computation outcomes Another possibility would be the collusion between a fog node and a smart meter, which might cause, for instance, wrong energy consumption being notified to the control center. Any collusion involving the control center has serious implications because it can see the plaintext of aggregated smart meters' data. One important case is the possibility of a differential attack when the control center colludes with fog nodes [1]. This could circumvent privacy mechanisms to protect a single user's data.

In summary, our threat model assumes the following network properties: 1) the control

center is not trusted with access to fine-grained smart meter data, i.e., it may only receive the plaintext of aggregated data to protect individual customers' privacy; 2) parties conducting outsourced computation cannot observe inputs or outputs of computation performed on behalf of the control center, since data is always encrypted; 3) fog nodes are not trusted and thus cannot see the plaintexts; 4) individual party members operating as the parties of the key center are not trusted, and only a quorum of the minimal size can be of all parties or a majority fraction of it; 5) energy retailers are not trusted to have access to the outcome of computation performed by the fog nodes or the control center, otherwise, it can be exploited and result in a differential attack [1].

3.4 Use Cases

In the following we present use cases that can be performed over the described network.

Billing: it could be computed based on monthly aggregated data or based on time-of-use (ToU) price.

Average: to calculate the average of measurements reported by $n > 0$ smart meters, the fog node adds up the n measurements and multiplies the result by $1/n$, which gives the encrypted outcome. It then sends the result to the control center, which may ask the key center for decryption.

Variance: when the fog node has received the homomorphically encrypted measurements m_i of n smart meters and wants to compute the variance over them, the fog node has to compute the variance $\frac{1}{n}\sum_{i=1}^{n}(m_i - \bar{m})^2$ where $\bar{m}$ is the mean for the n records. This expression can be rewritten as $\frac{1}{n}\sum_{i=1}^{n} m_i^2 - \frac{1}{n^2}(\sum_{i=1}^{n} m_i)^2$, thus the fog node only needs to compute homomorphic evaluations of squarings and additions, with a total multiplicative depth of one [1].

One-way ANOVA: When the control center defines $s \geq 3$ different pricing strategies, it might apply one-way ANOVA to the consumers' daily electricity consumption to understand whether these pricing strategies had a considerable impact on the electricity usage [1]. Let m_{ij} be the energy consumption of user U_i under the j-th pricing strategy. For testing the hypothesis, one evaluates if $F = \frac{A_{SS_B}/(n-s)}{A_{SS_W}/(s-1)}$ is above a fixed critical threshold F_C, for the sum of squares between groups $A_{SS_B} = \sum_{j=1}^{s}\sum_{i=1}^{n} m_{ij}^2 - \frac{1}{n}\sum_{j=1}^{s}(\sum_{i=1}^{n} m_{ij})^2$, and the sum of squares within groups $A_{SS_W} = \frac{1}{n}\sum_{j=1}^{s}(\sum_{i=1}^{n} m_{ij})^2 - \frac{1}{ns}(\sum_{j=1}^{s}\sum_{i=1}^{n} m_{ij})^2$. The computation again requires the homomorphic evaluation of squares and additions over the encryptions of values m_{ij}, with a total multiplicative depth of one multiplication.

Non-linear SVM classification: We consider the algorithm proposed by Jokar et al. [14] that leverages SVM classification using the radial basis kernel for energy theft detection. We assume that training has been done using plaintext (inside the key center or the control center) and we want to perform privacy-preserving SVM classification using the radial basis kernel that requires 5 multiplications.

4 Homomorphic Encryption

Homomorphic encryption schemes are defined as those that conserve some mathematical structure of the data during encryption that allows one to perform computation over ciphertexts. In this Section, we define cryptosystems those and discuss relevant aspects to this work.

4.1 Notations and Definitions

We denote the operation of randomly sampling an element a from a probability distribution χ as $a \leftarrow\$ \chi$. When the sampling is uniform from a set X we write it as $a \leftarrow\$ U(X)$.

Let K be the $2N$-th cyclotomic number field and $R = \mathcal{O}_K$ its ring of integers, which is represented in its polynomial form as $R = \mathbb{Z}[x]/(x^n + 1)$. For an integer $q \geq 2$, R_q denotes the quotient ring $R_q = R/qR = \mathbb{Z}_q[x]/(x^n + 1)$. Furthermore, let $\mathcal{C} = \{q_0, q_1, \ldots, q_\ell\}$ be a set of coprime integers, also referred as a basis, and $q = \Pi_{i=0}^{\ell} q_i$. If $p \in R_{\mathcal{C}}$, then $\exists P \in R_q$ such that $p \coloneqq \{[P]_{q_0}, [P]_{q_1}, \ldots, [P]_{q_\ell}\}$. Operations such as addition and multiplication over elements in $R_{\mathcal{C}}$ shall be taken as coefficient-wise. That is, if $a, b \in R_{\mathcal{C}}$ then $a + b = \left\{[a_0 + b_0]_{q_0}, \ldots, [a_\ell + b_\ell]_{q_\ell}\right\}$.

Let $(\mathcal{E}, \mathcal{D})$ be a pair of encryption and decryption functions that compose a cryptographic scheme. Moreover, let $c_0 = \mathcal{E}_{\text{SK}}(m_0)$ and $c_1 = \mathcal{E}_{\text{SK}}(m_1)$ be encryptions related to a certain secret key sk. If this scheme is homomorphic regarding an operator $\diamond$, then there exists an operator $\circ$ such that $\mathcal{D}(c_0 \circ c_1) \equiv m_0 \diamond m_1$ [26]. A scheme that supports this property for an unlimited quantity of additions and multiplications is called fully homomorphic (FHE).

If a homomorphic encryption scheme supports only one of these operations, it is called partially homomorphic. Paillier cryptosystem is a well-known partially homomorphic encryption scheme [6]. Based on the composite residuosity class problem, it is an additive homomorphic encryption scheme, i.e., it supports the addition of encrypted messages, but this implies that it also supports the multiplication of ciphertexts by plaintexts, a natural extension of supporting additions [6]. Nonetheless, the underlying problem is not secure against attacks based on quantum computers [27].

The Boneh-Goh-Nissim (BGN) cryptosystem was the first proposed homomorphic encryption scheme that allows an arbitrary number of homomorphic additions and exactly one multiplication with a constant-size ciphertext [7]. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing between three multiplicatively subgroups of prime order r. For efficiency, we refer to the version proposed by [28] that employs bilinear maps in the so-called asymmetric setting instead of the original composite-order curves [29]. Define a *projecting pairing* $\hat{e} : G \times H \to G_T$ between $G = \mathbb{G}_1^2, H = \mathbb{G}_2^2, G_t = \mathbb{G}_T^4$ that corresponds to a product of 4 invocations of the pairing e. Under the BGN cryptosystem, one can encrypt a message m into either G or H and perform an arbitrary number of homomorphic additions in both G and H. A homomorphic multiplication between two ciphertexts combines the G elements from the first ciphertext with the H elements from the second ciphertext and maps them to G_T, where homomorphic additions can still be performed but no further multiplications. The BGN cryptosystem can be defined as:

- BGN.KeyGen(1^λ): Sample $g \leftarrow\!\$\, \mathbb{G}_1, h \leftarrow\!\$\, \mathbb{G}_2$, set the secret key as $\mathsf{sk} \coloneqq (x, y, z, x', y', z') \in (\mathbb{Z}_r^*)^6$ and compute the public key $pk \coloneqq ((g, g^x), (g^y, g^z), (h, h^{x'}), (h^{y'}, h^{z'}))$.
- BGN.Encrypt(m, pk): Sample $k \leftarrow\!\$\, \mathbb{Z}_r^*$, return $(C_1, C_2) = ((g^{ym+k}, g^{zm+xk}), (h^{y'm+k}, h^{z'm+x'k}))$.
- BGN.Add(C, D) : for ciphertexts $C = (C_1, C_2), D = (D_1, D_2)$, return $(C_1 \cdot D_1, C_2 \cdot D_2)$.
- BGN.Multiply(C, D): for C, D as above, compute and return $\hat{e}(C_1, D_2) = \prod_{1 \leq i,j \leq 2} e(C_{1,i}, D_{2,j})) = \hat{e}(D_1, C_2)$.
- BGN.Decrypt(m, pk): For ciphertext in G, compute the discrete logarithm of $(g^{ym+k})^x (g^{zm+xk})^{-1} = (g^{xy-z})^m$ in base g^{xy-z}. Decryption in H (or G_t after a multiplication) can be carried out similarly.

Another important scheme was proposed by Cheon et al. (CKKS) [9]. This is a recent RLWE-based cryptosystem that follows the blueprint proposed by Gentry to build a quantum-resistant FHE scheme [13]. An important particularity of CKKS is that it aims at non-integer arithmetic, defining the plaintext domain as a ring of polynomials with complex coefficients and tolerating the manipulation of approximations of its elements. The decryption outcome is always an approximation of the expected result, affected by the natural imprecision related to the fixed-precision arithmetic but also by the encryption noise intrinsic to the encryption process.

CKKS uses the Residue Number System (RNS) to define arithmetic that does not require multi-precision operations. RNS states that an integer x can be decomposed as a set of residues in $R_{\mathcal{C}}$, for a basis $\mathcal{C} = \{q_0, q_1, \ldots, q_\ell\}$. A ciphertext, denoted $\mathsf{ct} = (c_0, c_1)$, is a pair of elements in $R_{\mathcal{C}}$. In other words, $\mathsf{ct} = \{(c_0^{(i)}, c_1^{(i)})\}_{0 \leq i \leq \ell}$ such that $(c_0^{(i)}, c_1^{(i)}) \in R_{q_i} \times R_{q_i}$. The CKKS cryptosystem has an embedded encoding method to convert complex numbers into a more convenient format for encryption that supports homomorphic operations. Let z be a vector of n complex numbers and Δ a scalar. In practice, we have that $decode(z) = \lfloor \mathrm{FFT}(z) \cdot \Delta^{-1} \rceil$, where FFT is the Fast Fourier Transform [30]. Thus, by taking the inverse operation we can see that the encoding of z is an element of R.

Let χ_{key} be a secret key distribution, and χ_{err} an encryption key distribution over R. In practice, χ_{key} is usually defined as a narrow distribution, sampling uniformly from $\{-1, 0, 1\}$, and χ_{err} as the discrete Gaussian. Furthermore, let $\mathcal{C} = \{q_0, q_1, \ldots, q_L\}$ and $\mathcal{B} = \{q_{L+1}, q_{L+2}, \ldots, q_{L+k}\}$ be two RNS basis coprime to each other, for an arbitrary integer L. In the following, we present some of the primitives relevant to this work.

- CKKS.SecKeyGen(1^λ): Sample $s \leftarrow\!\$\ \chi_{key}$ and set the secret key as $\mathsf{sk} \coloneqq (1, s)$.
- CKKS.PubKeyGen(sk): Sample $\left(a^{(0)}, \ldots, a^{(L)}\right) \leftarrow\!\$\ U(R_{\mathcal{C}})$ and $e \leftarrow\!\$\ \chi_{err}$. Set the public key as $\mathsf{pk} \coloneqq \left(\mathsf{pk}^{(j)} = (-a^{(j)} \cdot s + e \mod q_j, a^{(j)})_{0 \leq j \leq L}\right)$.
- CKKS.RelinKeyGen(sk): Sample $\left(a^{(0)}, \ldots, a^{(k+L)}\right) \leftarrow\!\$\ U(R_{\mathcal{C} \bigcup \mathcal{B}})$ and $e \leftarrow\!\$\ \chi_{err}$. Let $b^{(j)} \coloneqq -a^{(j)} \cdot s + \left[\prod_{i=0}^{k-1} p_i\right]_{g^j} + e \mod g^j$, for $g_j \in \mathcal{C} \bigcup \mathcal{B}$. Set the relinearization key $\mathsf{rlk} \coloneqq \left(\mathsf{rlk}^{(j)} = (b^{(j)}, a^{(j)})\right)_{0 \leq j \leq L}$.
- CKKS.Encrypt(pk): For $m \in R$, sample $v \leftarrow\!\$\ \chi_{enc}$ and $e_0, e_1 \leftarrow\!\$\ \chi_{err}$. Output the ciphertext $ct \coloneqq \left(ct^{(j)} = v \cdot \mathsf{pk}^{(j)} + (m + e_0, e_1)(\mod q_j)\right)_{0 \leq j \leq L}$.
- CKKS.Decrypt(sk): Output $ct \cdot \mathsf{sk} \mod q_0$.

4.2 Distributed CKKS

To avoid depending on a centralized trusted authority, we assume that a cluster of nodes collectively performs storage and generation of secret keys in MPC. Therefore, a malicious third party interested in subverting the system will need to compromise several nodes before succeeding. Such threshold cryptosystems assert that a subset of the parties involved in the computation must agree to achieve a valid result. Thus, if a certain threshold of agreements is not achieved, the computation is not done. In particular, we call it a distributed scheme when that fraction is the totality of the parties.

Mouchet et al. apply an additive secret-sharing approach to the BFV and build a distributed version of that scheme [24]. Their proposal assumes that a group of parties want to issue public keys so that anyone can execute homomorphic operations but decryption of the outcome is only possible if the entire group collaborates. Thus, they modify the scheme's key generation algorithm so that no party possesses the secret key. Shares of it are split through the nodes so that they must be securely combined before decryption is executed. They argue that the similarities between BFV and CKKS allow their method to be ported easily to the latter [24]. In the following, we describe the modifications required on the original CKKS design to convert it to a distributed version.

4.2.1 Ideal Secret Key

In this context of a threshold scheme, we refer to its secret key as an *ideal secret key*, in the sense that it never materializes but only exists as shares distributed among the parties. Let $\mathcal{C}$

and $\mathcal{B}$ be orthogonal RNS bases and $\mathcal{P}$ a party. Let s_i be a secret key generated through CKKS. SecKeyGen by $P_i \subset \mathcal{P}$. An ideal secret key for $\mathcal{P}$, through the additive secret-sharing method, can be defined as:

$$s = \left(\sum_{P_i \in \mathcal{P}} s_i \right) \mod R_{\mathcal{C} \bigcup \mathcal{B}}. \tag{1}$$

Note that each party must handle its share of the secret key as it would handle the secret key itself. It shall never be made public or shared with other parties. Hence, all operations that depend on it will need to be executed in an MPC protocol, such as decryption and the generation of the public and relinearization keys. An implication of defining the ideal secret key as in Eq. (1) is that its norm grows with $O(|\mathcal{P}|)$, impacting the noise growth of the new scheme [24].

4.2.2 Multi-Party Computation Protocols

Since the secret key is never instantiated, procedures that depend on it must be adapted to work interactively among the party members.

Collective public key (ColPubKeyGen): Each party member computes a share of the public key and distributes it to every other party. With all the shares, one can calculate the group public key.

Let $a \in R_{\mathcal{C}}$ be a public polynomial represented in base $\mathcal{C}$ and s_i a share of the secret key generated by a party $P_i \in \mathcal{P}$. To compute a collective public key cpk, each party must sample $e_i \leftarrow\!\!\$\ \chi_{err}$ and disclose $b_i = \text{PubKeyGen}(a, s_i) = -(a \cdot s_i + e_i) \mod R_{\mathcal{C}}$. From $b = \sum_{P_i \in \mathcal{P}} b_i \mod R_{\mathcal{C}}$, sets the collective public key as $\mathsf{cpk} = (b, a)$.

Collective relinearization key (ColRelinKeyGen): The relinearization key is computed in 3 interactive steps. In the end, a group evaluation key, which is public, is issued.

Let $a \in R^{k+L}_{\mathcal{C} \bigcup \mathcal{B}}$ be a public polynomial represented in base $\mathcal{C} \bigcup \mathcal{B}$ and $P = \Pi_{p_i \in \mathcal{B}} p_i$. Each party P_i must: 1) sample $u_i \leftarrow\!\!\$\ \chi_{key}$, $e_{0,i}, e_{1,i} \leftarrow\!\!\$\ \chi^{L+k}_{err}$ and disclose $(h_{0,i}, h_{1,i}) = \text{Relin1}(a, s_i) = \left(-u_i a + s_i [P]_{g_j} + e_{0,i}, s_i a + e_{1,i}\right)$ for all $g_j \in \mathcal{C} \bigcup \mathcal{B}$, 2) from $h_0 = \sum_{P_j \in \mathcal{P}} h_{0,j}$ and $h_1 = \sum_{P_j \in \mathcal{P}} h_{1,j}$, sample $e_{2,i}, e_{3,i} \leftarrow\!\!\$\ \chi^{L+k}_{err}$ and disclose $(h'_{0,i}, h'_{1,i}) = \text{Relin2}(h_0, h_1, s_i) = (s_i h_0 + e_{2,i}, (u_i - s_i) h_1 + e_{3,i})$, 3) from $h'_0 = \sum_{P_j \in \mathcal{P}} h'_{0,j}$ and $h'_1 = \sum_{P_j \in \mathcal{P}} h'_{1,j}$, outputs $\mathsf{crlk} = \text{RelinKeyGen}(h'_0, h'_1, h_1) = (h'_0 + h'_1, h_1)$.

Combined decryption (ColDecrypt): A share of the decryption result is computed by each party and all the shares are combined to obtain the actual decryption of the ciphertext.

Let $c = (c_0, c_1)$ be a ciphertext. Each party P_i must: 1) sample $e \leftarrow\!\!\$\ \chi_{err}$ and disclose $c'_i = \text{Decrypt1}(s_i, c_1) = s_i \cdot c_1 + e \mod q_0$ and 2) from $c' = \sum_{P_i \in \mathcal{P}} c'_i$, output $m' = \text{Decrypt2}(c_0) = c_0 + c' \mod q_0$.

ColDecrypt depends on the parties disclosing $s_i \cdot c_1$. Since c_1 is public, we need the addition of noise e to protect the secret key share s_i. Thus, this step implies a noise growing of $\mathcal{O}(|P|)$.

Along with these protocols, one can also generalize the ColRelinKeyGen algorithm to define a collective Key-switching procedure, which can be used to convert a ciphertext encrypted regarding a secret key s' to a different key s. However, this is beyond the scope of this paper.

5 Proposed Method

Our proposal assumes that the only entity that holds the decryption keys and is responsible for the key distribution protocol is the key center. The key center works as a decryption oracle to the network since it must store all the required decryption keys. It generates and distributes public keys to smart meters, such as encryption and evaluation keys. We consider that a single

secret key is related to a cluster of smart meters (e.g., in a neighborhood or a building). This cluster may be as small as a single user or as large as needed. Each cluster member receives a different public key generated through the same secret key. This hides the group relationship between smart meters, preventing any entity from linking two smart meters to the same group. Moreover, in this case, one can aggregate multiple user data to, for instance, predict group demand. However, if the key center fails to authenticate smart meters' data, different users in the same cluster may learn about data related to their neighbors. That is, authentication can not be done through the same pair of encryption keys used for HE, since the private key is related to the cluster and not particular users. To assert nonrepudiation, the authentication request must be done through dedicated and specific keys to each entity.

5.1 Distributed Key Center

The distributed key center is instantiated as a cluster of nodes. We need to define protocols for the collective key establishment and decryption, following the primitives described in Section 4.2.2.

Decryption in an MPC protocol depends on the agreement by all members of the key center cluster that the request is valid and that the protocol can be executed. A token σ must be submitted together with the ciphertext to assert correctness for the decryption mechanism. This token may be the public value of a VC scheme or a digital signature asserting that the decryption came from an authorized entity. If a node of the key center cluster cannot validate σ, it rejects the decryption. If the number of rejections surpasses a certain parameter T (e.g., 1 in case of a distributed HE scheme), decryption is denied.

To assist communications, we define an Entry fog node, which is an entity responsible for assisting other members of the key center on the partial computations and relaying of data. It can be abstract, in the sense that it can be replaced by direct communication among the key center members, or concrete, as an independent node.

5.2 Network Behavior

Two evaluation steps are considered, performed by a fog node and by the control center. The first step must be done homomorphically over ciphertexts, but the second step can be executed over the plaintexts as long as the key center accepts to decrypt the ciphertexts for the control center. A smart meter keeps a communication cycle with its related fog node with n iterations consisting of data production (e.g., electricity measurements) and data aggregations at the fog node, for n arbitrary. After that, the encrypted outcome is sent to the control center, which can decide to keep data encrypted and execute more computation (locally or remote), or request decryption to the key center.

Figures 2 and 3 present the key agreement protocol and the decryption protocol, which are internal operations to the key center and depend on the nodes exchanging information. Figure 4 presents the usual communication procedure for the network members.

This use case has the following properties: 1) if the control center is allowed to ask the key center for decryptions, the computation can be done over plaintexts, otherwise the ciphertexts can be rebuilt with different parameters and sent to a third party, 2) data collected from different smart meters can be aggregated at a fog node or control center level, 3) fog nodes do not handle plaintexts, 4) fog nodes are not capable of generating valid ciphertexts, 5) there is limited data secrecy unless a threshold FHE scheme is being used. A centralized key center sees everything. The distributed key center can solve that, so that only the entity requesting data may have access to the decrypted plaintext.

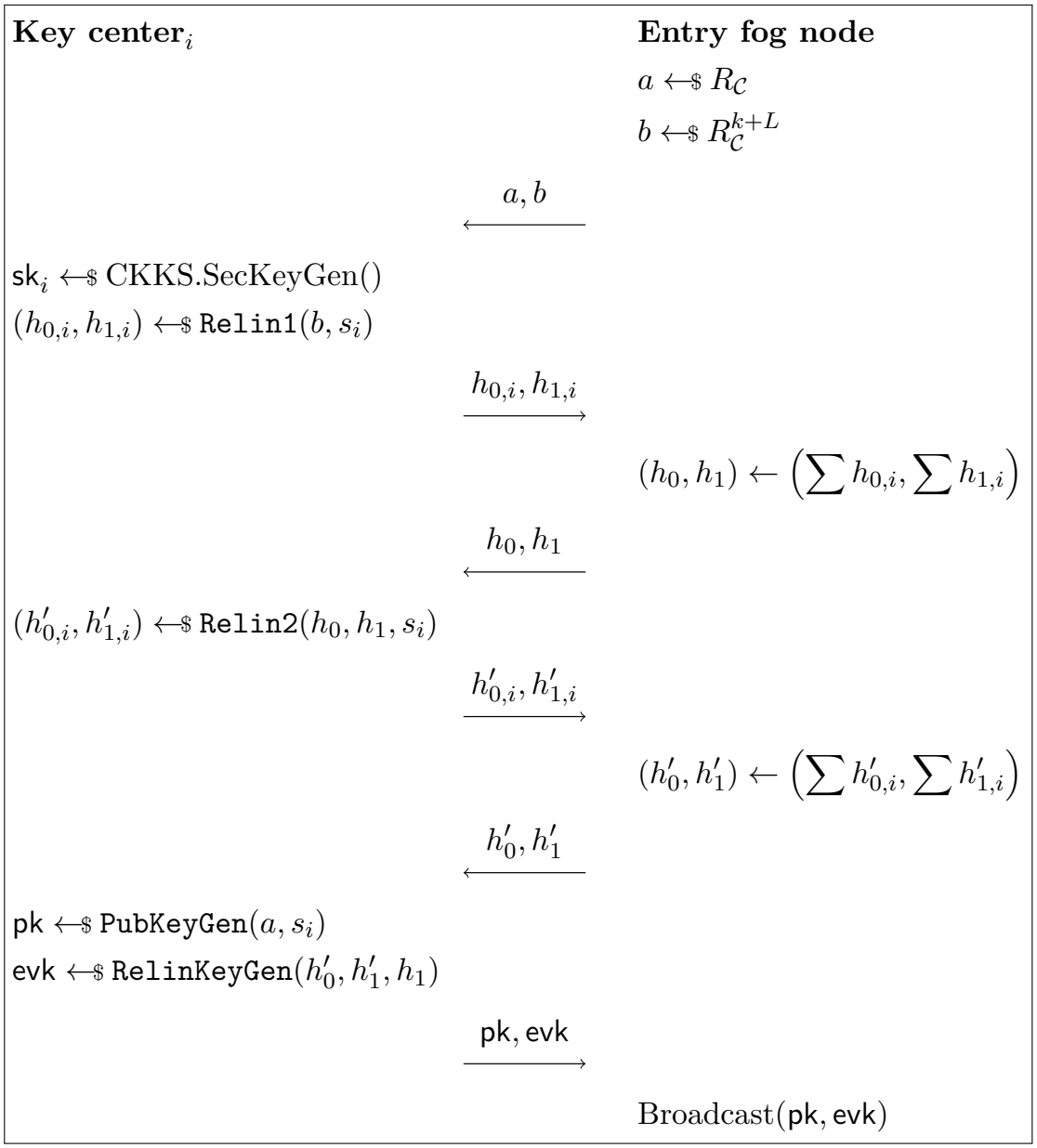


Figure 2: Key agreement protocol (KAP). The members of the key center exchange the data elements needed for agreement of the shared secret key, generating and broadcasting public keys to members of the network.

6 Performance Evaluation

For different numbers of multiplications, we measure the ciphertext size. Then, we try to optimally fill the ciphertext slots by experimenting with different data collection frequencies. This experiment results in suggesting optimal data collection frequencies. To measure the bandwidth consumption and delay, we have implemented the system in the ns-3 environment [31]. Finally, we compare the proposed HE method with Paillier and BGN HE methods in terms of computational overhead.

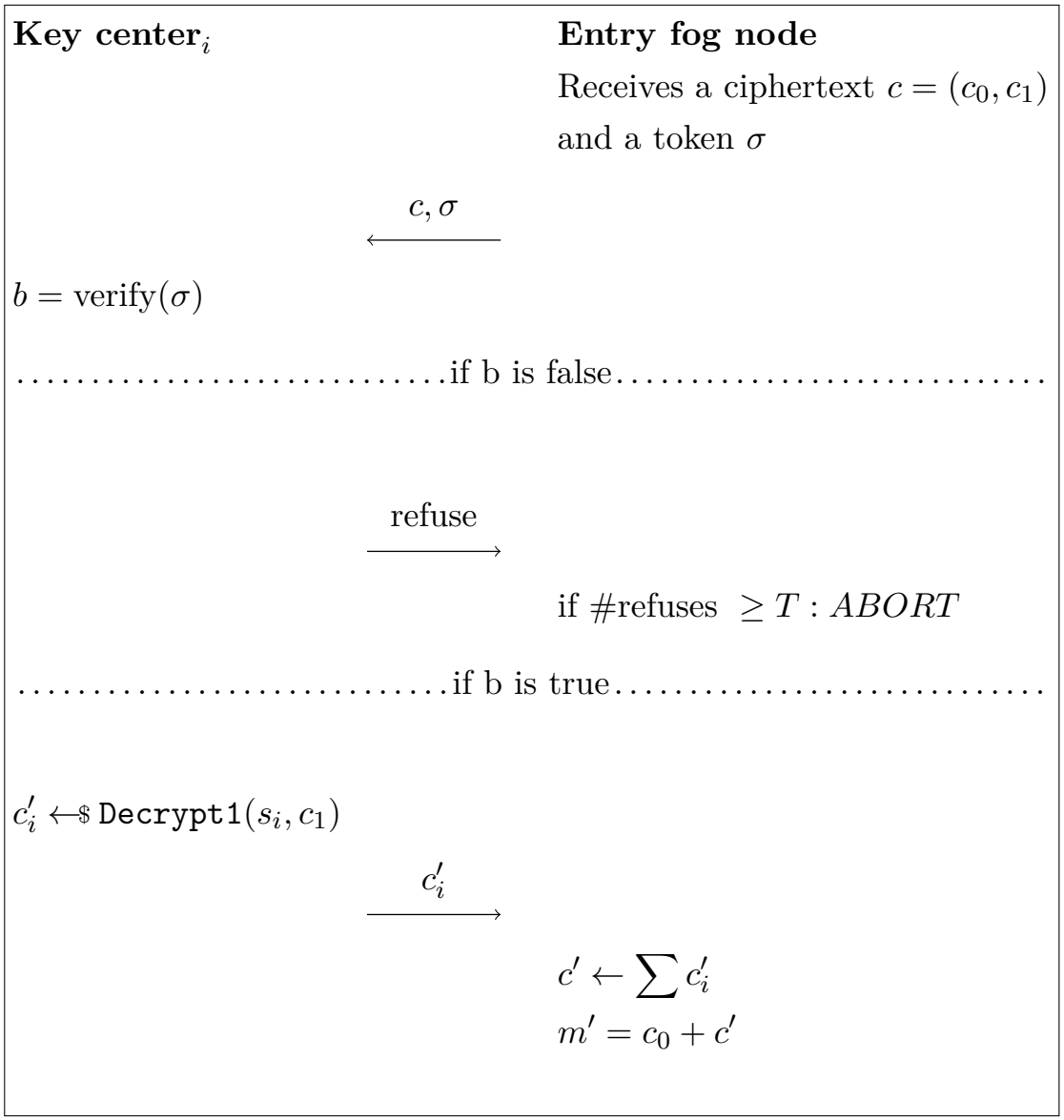


Figure 3: Decryption protocol (ColDecrypt). An entry fog node receives a request of decryption for a particular ciphertext. The validity of this request is assured by a token σ.

6.1 Parameter Selection

For CKKS, we take a power-of-2 degree n, and define $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. This means that our working ring is composed of polynomials with integer coefficients modulo q and degrees lower or equal to $n - 1$. This scheme represents integers using an RNS basis $\mathcal{C} = \{q_0, \ldots, q_L\}$ such that $q = \Pi_{i=0}^{l} q_i$. Moreover, the Discrete Gaussian is usually employed as the error distribution, and the secret key is sampled from the ternary distribution, i.e., each coefficient is chosen uniformly random from $\{-1, 0, 1\}$. However, there are cases, as with threshold schemes, where the secret key may be sampled uniformly from R_q. Table 1 presents a parameter recommendation in case the secret key is sampled from a ternary distribution. To predict the security level offered by a particular set of parameters in CKKS, we rely on Albrecht et al.'s script in Sage to estimate security level considering several possible attacks on instances of the LWE problem [32].

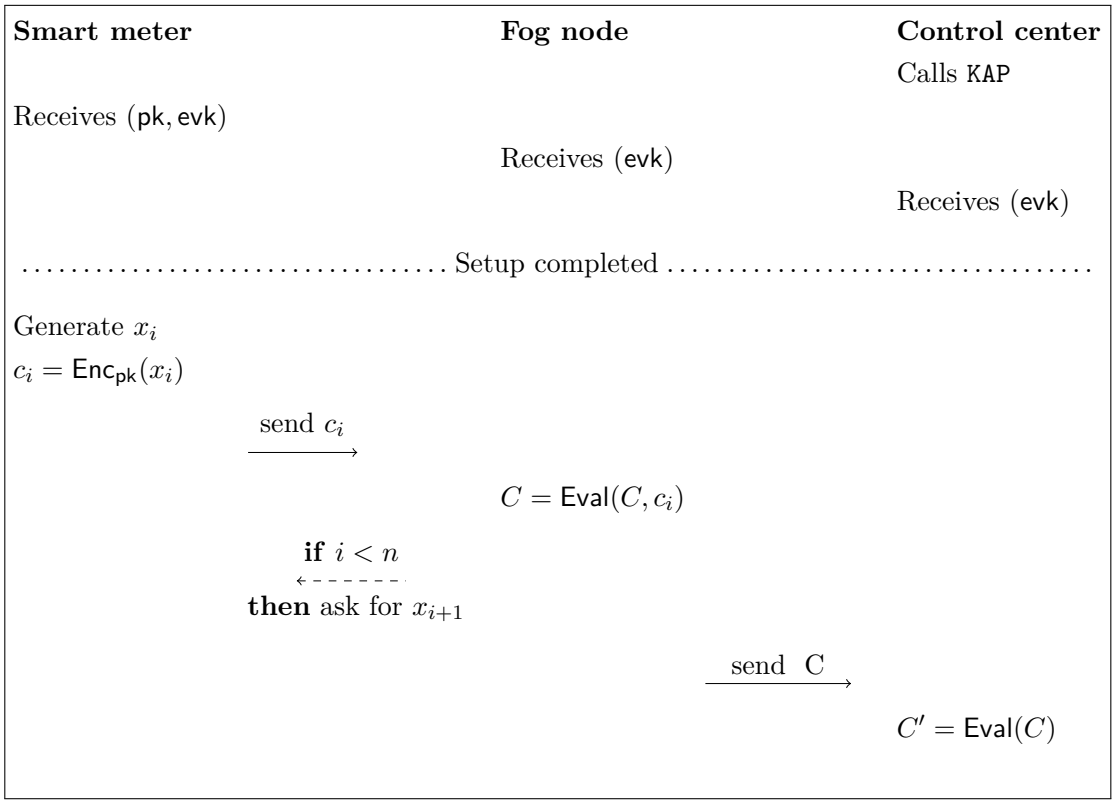


Figure 4: Usual communication behavior. A fog handles intermediary communication between the smart meter and the control center. This includes relaying public keys but also executing computation homomorphically.

6.2 Batching and Ciphertext Size

Many modern HE schemes, such as CKKS and BGV [9, 33, 34, 35], support packing techniques to encrypt multiple messages in a single ciphertext, building homomorphic operations that work similarly to a single instruction, multiple data (SIMD) implementations. CKKS contains its own encoder that maps vectors of complex numbers to elements of R_q. As aforementioned, these elements are represented in a RNS basis $\mathcal{C} = \{q_0, \ldots, q_L\}$. A fresh CKKS ciphertext, denoted $\mathsf{ct} = (c_0, c_1)$, is a pair of elements in $R_{\mathcal{C}}$. That is, $\mathsf{ct} = \{(c_0^{(i)}, c_1^{(i)})\}_{0 \leq i \leq L}$ such that $(c_0^{(i)}, c_1^{(i)}) \in R_{q_i} \times R_{q_i}$. At this moment, we say that this ciphertext has level $L + 1$. Each pair $(c_0^{(i)}, c_1^{(i)})$ will be referred as a residue of ct. After a homomorphic multiplication, a rescaling operation may be required to conserve plaintext precision after decryption. For that, the highest-index residue of the ciphertext is consumed. That is, after ℓ homomorphic multiplications, ct becomes a pair of elements in $R_{\mathcal{C}-\{q_{L-\ell}, \ldots, q_L\}}$, and the ciphertext becomes a ℓ-level ciphertext.

CKKS ciphertexts are capable of storing a certain number of plaintexts, referred as slots. Let n be the ring degree, as defined in Section 6.1. By design, a CKKS ciphertext supports up to $n/2$ slots which can be filled with complex elements.

Based on the above, CKKS ciphertexts require $s(n, \ell) \coloneqq (2 \cdot n \cdot \ell) \cdot 64$ bits for storage. Table 2 presents some examples based on data from Table 1. However, the ciphertext expansion factor also depends on its slot occupancy. Through batching, the expansion factor becomes $\frac{s(n,\ell)}{64 \cdot \texttt{batch\_size}}$.

Table 1: Parameter recommendation for different security levels if the secret key is sampled from a ternary distribution.

n	**log(q)**	λ **(Security level)**
1024	27	128
	19	192
	14	256
2048	54	128
	37	192
	29	256
4096	109	128
	75	192
	58	256
8192	218	128
	152	192
	118	256
16384	438	128
	305	192
	237	256
32768	881	128
	611	192
	476	256

Table 2: Comparison of ciphertext sizes for different parameter selections at 128-bit security level. It considers the batching of n plaintexts for the CKKS ciphertext and the equivalent quantity of Paillier's ciphertexts.

n	**q**	**CKKS (KB)**	**BGN (KB)**	**Pallier (KB)**
1024	27	442	295 (1.50)	786 (0.56)
2048	54	1769	590 (3.00)	1573 (1.13)
4096	109	7143	1180 (6.05)	3146 (2.27)
8192	218	28574	2359 (12.11)	6291 (4.54)

Table 3: Recommendation of parameters and decomposition that supports a certain number of multiplications.

n	log(**q**)	λ **(Sec. level)**	**#mul**	**Decomposition**	**Scale**
1024	43	80	1	{43}	21
	34	100	1	{34}	16
	27	128	1	{27}	13
2048	84	80	2	{35,16,33}	16
	84	80	1	{45,39}	21
	68	100	1	{37, 31}	18
	54	128	1	{31, 23}	15
4096	170	80	4	{40, 21, 21, 21, 21, 40}	21
	135	100	3	{35, 21, 21, 21, 35]	21
	109	128	2	{35, 21, 21, 30}	21
8192	330	80	5	{54, 46, 46, 46, 46, 46, 46}	46
	270	100	5	{54, 36, 36, 36, 36, 36, 36}	36
	218	128	5	{54, 26, 26, 26, 26, 26, 34}	26

Table 4: Size in bits of a Paillier's ciphertext for different security levels [36].

λ **(Security level)**	Ciphertext size (bits)
80	2048
112	4096
128	6144
192	15360

6.3 Simulation Settings

We have implemented a set of 200 smart meters that are in the radio range of an eNB, which receives the smart meter data. It is assumed that the eNB is located at the substation owned by the utility company [37] (cell range is assumed to be 1 km, which is about the same as the range of an electric grid [11]). We assume that a fog node is connected to the eNB that is responsible for the storage and processing of the received smart meter data.

In the following, we show results in terms of ciphertext size when 1, 2, or 5 homomorphic multiplications could be done on it (billing could be done with 1 homomorphic multiplication, the variance may need 2 homomorphic multiplications, and SVM classification using radial basis function requires 5 homomorphic multiplications). We use CKKS and selected the parameters compatible with a 128-bit security level, according to Albrecht's estimator [32]. Furthermore, we fill the CKKS ciphertext slots according to different data collection frequencies and attempt to fill the maximum number of slots such that we avoid wasting bandwidth resources due to having empty slots. We also report results regarding the delay of communicating ciphertexts, which can support 1, 2, or 5 multiplications from the smart meters to the eNB.

6.4 Bandwidth Requirement in Different Scenarios

We consider that each smart meter has to report its measurements every 15 minutes [4]. To encrypt its data, a smart meter needs to know the maximum multiplicative depth presented in the arithmetic circuits in which the fog node will evaluate ciphertexts. This information impacts the parameter selection and the ciphertext size. Larger ciphertext sizes result in higher bandwidth consumption and longer delays.

Figure 5 shows results in terms of communicated ciphertext size for three cases where 1, 2, or 5 multiplications could be done on the ciphertext. We experiment with different data collection frequencies to find the optimal data collection frequencies in terms of bandwidth consumption efficiency. Figure 5 specifies the results for the suggested data collection frequencies using dark-shaded diamonds. The other results from this figure that are specified using solid circles have not used the recommended data collection frequencies, thus ciphertexts were communicated with many more empty slots, which resulted in wasting bandwidth resources. Table 5 summarizes the parameters used in each case.

When only 1 multiplication is required, from Table 1, we select the first row where $n = 1024$. In this case, we attempt to find the data report granularity that results in filling most of the ciphertext slots such that the least ciphertext space is wasted. If the frequency of filling the slots is 1.13, 1017 out of 1024 slots are filled, while if we fill a slot per second, 900 out of 1024 slots are filled. Hence, we recommend the frequency of filling slots to be 1.13. Supporting one multiplication, from Figure 5, we observe that when data collection frequency is up to 1.13, the communicated ciphertext size is 442.368 KB. If one decides to increase the frequency of filling the slots to have finer data report granularity, we recommend the frequencies of filling the slots to be 2.27, 4.55, or 10.24. If the frequency of filling the slots is 2.27, a smart meter requires to send two ciphertexts every 15 minutes, and 2043 out of 2048 slots will be filled; the communicated ciphertext size will be 884.736 KB. If we increase the frequency of filling the slots to 4.55, a smart meter will transmit 4 ciphertexts per 15 minutes where 4095 out of 4096 slots will be

Table 5: Benchmark parameters

Case	**I**	**II**	**III**
# Mul.	1	2	5
n	1024	4096	8192
log(q)	27	109	218

filled, and the communicated ciphertext size is 1769.472 KB. Finally, if we further increase the frequency of filling the slots to 10.24, each smart meter will communicate 9 ciphertexts every 15 minutes and all the 9216 slots will be filled, i.e., no ciphertext slot is wasted; this results in 3981.42 KB of communicated ciphertext size.

If the ciphertext can support 2 multiplications, we select the third row from Table 1 where $n = 4096$. In this case, we recommend the frequencies of filling the slots to be 4.55, 9.10, or 13.65. If this frequency is 4.55, 4095 out of 4096 slots will be filled and the communicated ciphertext size is 7143.424 KB. If the frequency of filling the slots is 9.10, a smart meter will send 2 ciphertexts every 15 minutes where 8190 out of 8192 slots will be filled; the communicated ciphertext size will be 14286.848 KB. Finally, if the frequency of filling the slots is 13.65, 12285 out of 12288 slots will be filled and the communicated ciphertext size is 21430.272 KB. In case the ciphertext can support 5 multiplications, we choose the parameters according to the fourth row of Table 1 where $n = 8192$ and Figure 5 reports the trade-offs for different frequencies of filling the slots. From Figure 5, we suggest the frequencies of filling the slots to be 9.10, 18.00, or 27.00. When we select 9.10, a smart meter will send 1 ciphertext per 15 minutes and 8190 out of 8192 slots of that ciphertext will be filled; the communicated ciphertext size is 28573.696 KB. If the smart meter increases the frequency of filling the slots to 18.00, the smart meter needs to send 2 ciphertexts every 15 minutes and can fill all the 16384 slots; the communicated ciphertext size will be 57147.392 KB. Finally, if the smart meter fills the slots with the higher frequency of 27.00, it will require to communicate 3 ciphertexts per 15 minutes and all the 24576 slots could be filled; the communicated ciphertext size will be 85721.088 KB.

6.5 Communication Delay

We saw from figure 5 that for different data collection frequencies, different number of ciphertexts were communicated. Therefore, Figure 6 reports the communication delay from the smart meters to the eNB when different numbers of ciphertexts are communicated supporting 1, 2, or 5 homomorphic multiplications. We noted from Figure 5 that the ciphertext size increases when more homomorphic multiplications are supported. Using this figure, if we compare the cases where 2 multiplications are supported with the case where 1 multiplication is supported, we observe in average 16 times longer communication delays. Using Figure 5, we can also compare the case of 5 supported multiplications with 2 supported multiplications, and understand that the communication delay increases approximately 4 times in average. Furthermore, for each of the considered cases (supporting 1, 2, or 5 multiplications), when more ciphertexts are communicated, we see from Figure 6 that the communication delays increases.

6.6 Computation Delay

Table 6 presents the latencies of CKKS from SEAL v3.7.2 [38], and Paillier and BGN implemented in RELIC v0.5 [39].

We used the parameter sets described in Table 5 for the CKKS, offering support to respectively 1, 2, and 5 homomorphic multiplications with at least a 128-bit security level, as presented in Table 3. In comparison, we instantiate Paillier with $\log(n^2) = 6144$, which offers a similar security level as presented in Table 4, and BGN over the elliptic curve BLS12-381. CKKS is

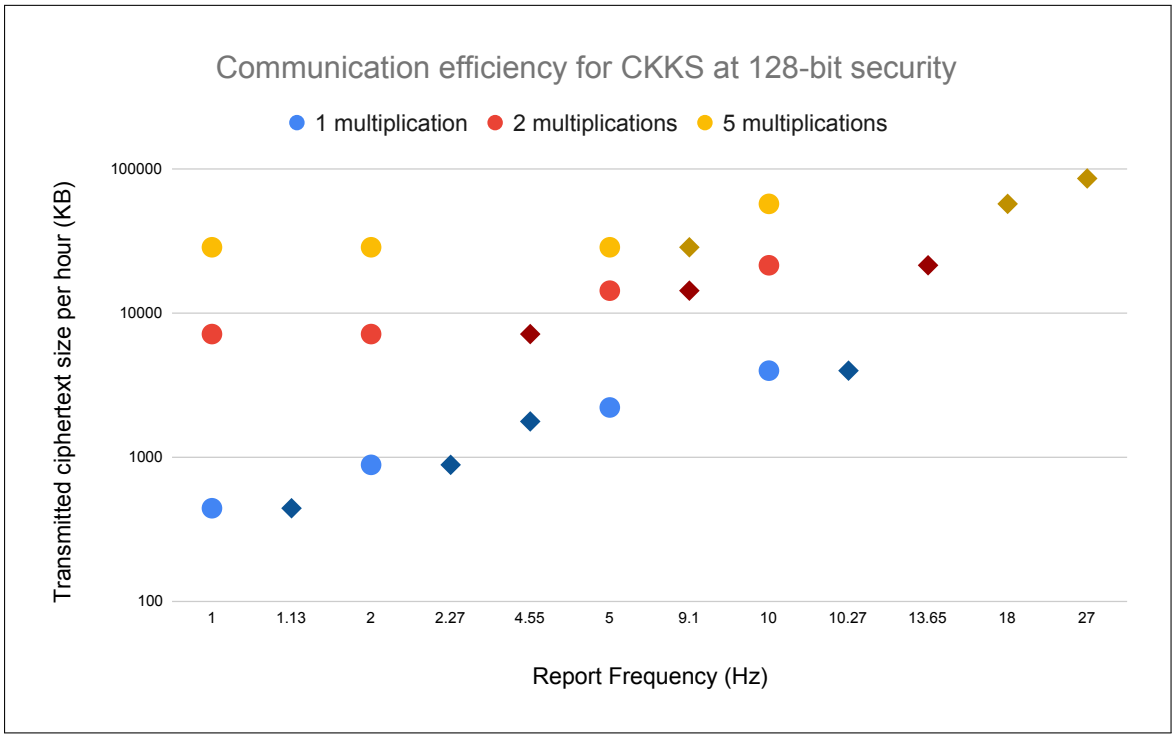


Figure 5: Communicated ciphertext size for the CKKS cryptosystem at the 128-bit security from a smart meter during an hour for 1, 2 and 5 multiplications allowed. The optimal configurations are represented with a dark shaded diamond instead of a circle.

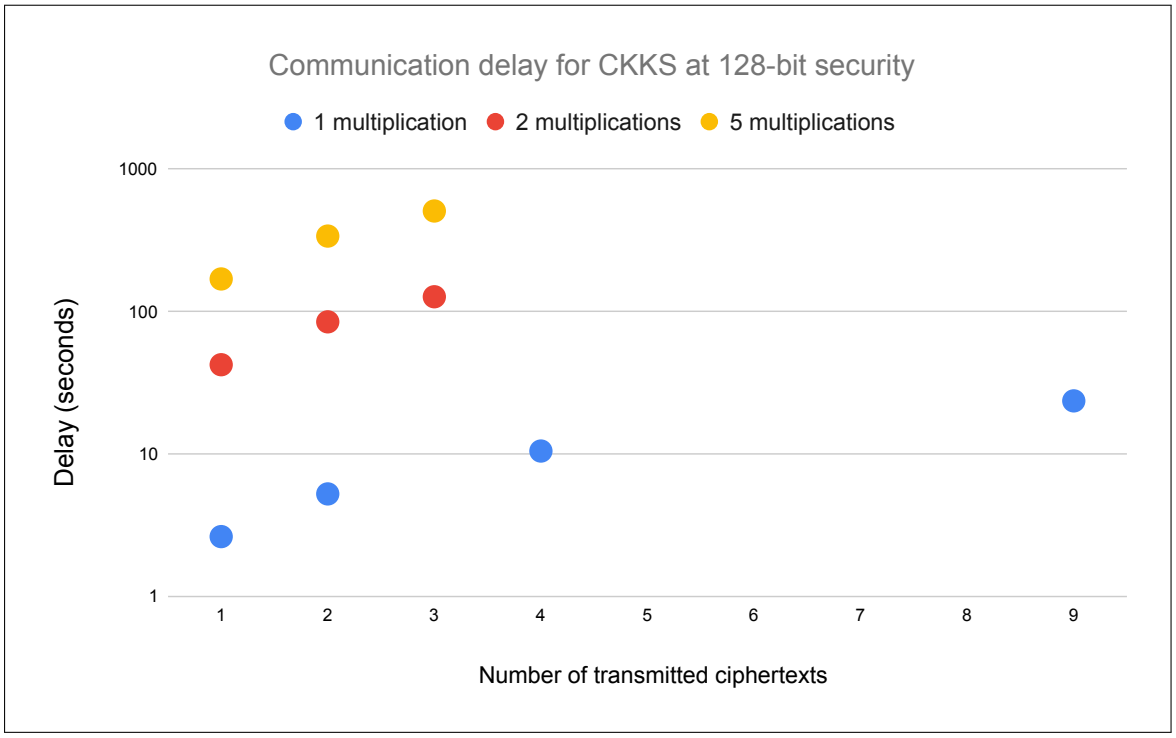


Figure 6: Average delay of sending a ciphertext from a smart meter to the eNB for 1, 2 and 5 multiplications. All the figures at the 128-bit security level.

the clear winner in performance for both homomorphic addition and multiplication. For case I supporting an unlimited number of additions, CKKS is up to 3 times faster than the competi-

Table 6: Number of cycles for the execution of homomorphic operations on Paillier, BGN, and CKKS on a Skylake Core i7 6700K@4GHz for parameters supporting of 1, 2 and 5 multiplications (cases I, II and III respectively). A dash means that the configuration is not supported.

Operation	**Case**		
Addition	**I**	**II**	**III**
Paillier	33,112	-	-
BGN	39,623	-	-
CKKS	13,920	107,600	424,000
Multiplication	**I**	**II**	**III**
Paillier	-	-	-
BGN	12,798,243	-	-
CKKS	70,000	500,000	2,088,000

tion, and up to 6 times faster for case III supporting 5 multiplications in comparison to BGN supporting a single one.

7 Conclusions

We considered a fog-based smart grid architecture in which smart meters use cellular communication infrastructure to communicate with a fog node that receives and processes smart meters' data. We proposed a lattice-based HE method for encryption of smart meters' data such that when the fog node receives the encrypted smart meters' data, it can homomorphically perform different computations, which differ in terms of the required number of multiplications on the encrypted data. The considered services include billing, variance, and non-linear SVM classification that require 1, 2, and 5 multiplications, respectively. We measured the ciphertext size for these statistical operations and based on analysis suggested the smart meter data report frequencies that optimally fill most of the slots of the CKKS ciphertext. Furthermore, we reported the communication delays of transmitting the considered ciphertext sizes from the smart meters to the eNB. We have also shown that the proposed HE method significantly outperforms Paillier and BGN cryptosystems in terms of computational overhead for homomorphic multiplication and addition operations. Our future work includes the implementation of MPC and VC methods and measuring their practical trade-offs.

References

[1] Chen, L., Lu, R., Cao, Z., Alharbi, K. N., and Lin, X. MuDA: Multifunctional data aggregation in privacy-preserving smart grid communications. *Peer-to-Peer Netw. Appl.*, **8**, 777–792.

[2] Meng, W., Ma, R., and Chen, H.-H. Smart grid neighborhood area networks: a survey. *IEEE Netw.*, **28**, 24–32.

[3] Barbosa, P., Brito, A., and de Almeida, H. O. A technique to provide differential privacy for appliance usage in smart metering. *Inf. Sci.*, **370-371**, 355–367.

[4] Ebeid, E., Heick, R., and Jacobsen, R. H. Deducing energy consumer behavior from smart meter data. *Future Internet*, **9**, 29.

[5] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv. (CSUR)*, **51**, 1–35.

[6] Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. *EUROCRYPT*, jan, LNCS, **1592**, pp. 223–238. Springer.

[7] Boneh, D., Goh, E., and Nissim, K. Evaluating 2-dnf formulas on ciphertexts. *TCC*, dec, LNCS, **3378**, pp. 325–341. Springer.

[8] Zhang, L., Zhang, J., and Hu, Y. H. A privacy-preserving distributed smart metering temporal and spatial aggregation scheme. *IEEE Access*, **7**, 28372–28382.

[9] Cheon, J. H., Han, K., Kim, A., Kim, M., and Song, Y. A full RNS variant of approximate homomorphic encryption. *SAC*, jan, LNCS, **11349**, pp. 347–368. Springer.

[10] Lyubashevsky, V., Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings. *EUROCRYPT*, may, LNCS, **6110**, pp. 1–23. Springer.

[11] Zhao, S., Li, F., Li, H., Lu, R., Ren, S., Bao, H., Lin, J., and Han, S. Smart and Practical Privacy-Preserving Data Aggregation for Fog-Based Smart Grids. *IEEE Trans. Inf. Forensics Secur.*, **16**, 521–536.

[12] Garau, M., Celli, G., Ghiani, E., Pilo, F., and Corti, S. Evaluation of smart grid communication technologies with a co-simulation platform. *IEEE Wirel. Commun.*, **24**, 42–49.

[13] Gentry, C. Fully homomorphic encryption using ideal lattices. *ACM STOC*, may, pp. 169–178.

[14] Jokar, P., Arianpoo, N., and Leung, V. C. Electricity theft detection in AMI using customers' consumption patterns. *IEEE Trans. Smart Grid*, **7**, 216–226.

[15] Rahulamathavan, Y., Phan, R. C., Veluru, S., Cumanan, K., and Rajarajan, M. Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. *IEEE Trans. Dependable Secur. Comput.*, **11**, 467–479.

[16] Bajard, J., Martins, P., Sousa, L., and Zucca, V. Improving the efficiency of SVM classification with FHE. *IEEE Trans. Inf. Forensics Secur.*, **15**, 1709–1722.

[17] Benaissa, A., Retiat, B., Cebere, B., and Belfedhal, A. E. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. *CoRR*, **abs/2104.03152**.

[18] Bois, A., Cascudo, I., Fiore, D., and Kim, D. Flexible and efficient verifiable computation on encrypted data. *Public Key Cryptography (2)*, mar, LNCS, **12711**, pp. 528–558. Springer.

[19] Fiore, D., Gennaro, R., and Pastro, V. Efficiently verifiable computation on encrypted data. *ACM CCS*, nov, pp. 844–855.

[20] Brakerski, Z. and Vaikuntanathan, V. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. *CRYPTO*, aug, LNCS, **6841**, pp. 505–524. Springer.

[21] Ganesh, C., Nitulescu, A., and Soria-Vazquez, E. Rinocchio: SNARKs for Ring Arithmetic. *IACR Cryptol. ePrint Arch.* , **?**, 322.

[22] Fan, J. and Vercauteren, F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, **2012**, 144.

[23] Chillotti, I., Gama, N., Georgieva, M., and Izabachène, M. TFHE: Fast Fully Homomorphic Encryption Over the Torus. *J. Cryptol.*, **33**, 34–91.

[24] Mouchet, C., Troncoso-Pastoriza, J. R., and Hubaux, J. Multiparty Homomorphic Encryption: From Theory to Practice. *IACR Cryptol. ePrint Arch.*, **2020**, 304.

[25] NIST Glossary - Data integrity. `https://csrc.nist.gov/glossary/term/data_integrity/`.

[26] Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. On data banks and privacy homomorphisms. *Found. Secur. Comput.*, **4**, 169–180.

[27] Shor, P. W. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. *ANTS*, LNCS, **877** 289. Springer.

[28] Freeman, D. M. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. *EUROCRYPT*, LNCS, **6110**, pp. 44–61. Springer.

[29] Boneh, D., Rubin, K., and Silverberg, A. Finding composite order ordinary elliptic curves using the Cocks–Pinch method. *J. of Number Theory*, **131**, 832–841.

[30] Bailey, D. H. FFTs in external or hierarchical memory. *J. Supercomput.*, **4**, 23–35.

[31] ns-3 Network Simulator. `https://www.nsnam.org/`.

[32] Albrecht, M. R., Player, R., and Scott, S. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, **9**, 169–203.

[33] Cheon, J. H., Coron, J., Kim, J., Lee, M. S., Lepoint, T., Tibouchi, M., and Yun, A. Batch fully homomorphic encryption over the integers. *EUROCRYPT*, LNCS, **7881**, pp. 315–335. Springer.

[34] Doröz, Y., Çetin, G. S., and Sunar, B. On-the-fly homomorphic batching/unbatching. *Financial Cryptography Workshops*, LNCS, **9604**, pp. 288–301. Springer.

[35] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, **6**, 13:1–13:36.

[36] Paillier ciphertext size. `https://www.keylength.com/en/4/`.

[37] Cheng, P., Wang, L., Zhen, B., and Wang, S. Feasibility study of applying lte to smart grid. *IEEE SGMS*, pp. 108–113.

[38] Microsoft SEAL (release 3.7). `https://github.com/Microsoft/SEAL`. Microsoft Research, Redmond, WA.

[39] RELIC is an Efficient Library for Cryptography. `https://github.com/relic-toolkit/relic`.

Chapter 3

Discussion

The papers presented in the previous Chapter contemplate mainly two directions: protocol proposal and implementation techniques on GPUs. Together, they present contributions toward different aspects of developing privacy-preserving algorithms.

We propose directives to model a searchable encrypted database by defining the needed relational algebra and looking for the underlying support of functional encryption schemes. In the same way, we investigate how this class of cryptographic schemes can assist in instantiating a privacy-oriented smart meter network, in which data can be handled encrypted during its entire existence, and a verifiable computation scheme can assert its integrity. In both cases, we aim to develop high-level solutions that consider decryption a sensitive operation that shall be avoided as much as possible. Thus, classical techniques common in the literature, such as online computation involving iterations between nodes in a network composed of decryption-encryption cycles, are never considered by our works.

Our solutions are evaluated mainly by developing simulations based on real-world scenarios. For example, in the work presented in Section 2.1, we explore the case of the famous Netflix Grand Prize. At the time, Netflix was looking for solutions to improve their recommendation algorithm to infer user predilection by analyzing their history in the platform. In 2007, 2008, and 2009, contests were proposed to the community, and the solutions compared regarding performance, latency, and the quality of the outcome when evaluated on a public test set. In our work, we select the winner solution proposed by BellKor's Pragmatic Chaos team, and we build the main SQL queries needed to implement it using our algebra. On the smart meter network work, we propose a communication protocol that can be used so that a grid of devices may execute smart functionalities while never manipulating plaintext data. To evaluate our proposal, we implement it on an NS3 simulation that measures such setup's bandwidth, latency, and processing implication.

Those two works observe the complexity of these solutions from a high-level point of view and consider that the heavy computational tasks are done by powerful devices like GPUs. Thus, a different facet of the problem approached by this thesis is the efficient design for implementing a cryptographic scheme that provides the required functionality on a high level and fits the processing hardware. Two other works discuss this matter. In the first, presented in Section 2.2, we present efficient techniques for implementing BFV, a homomorphic encryption scheme, on CUDA. We show how data structure must be designed to better fit the architecture challenges, considering both computation and

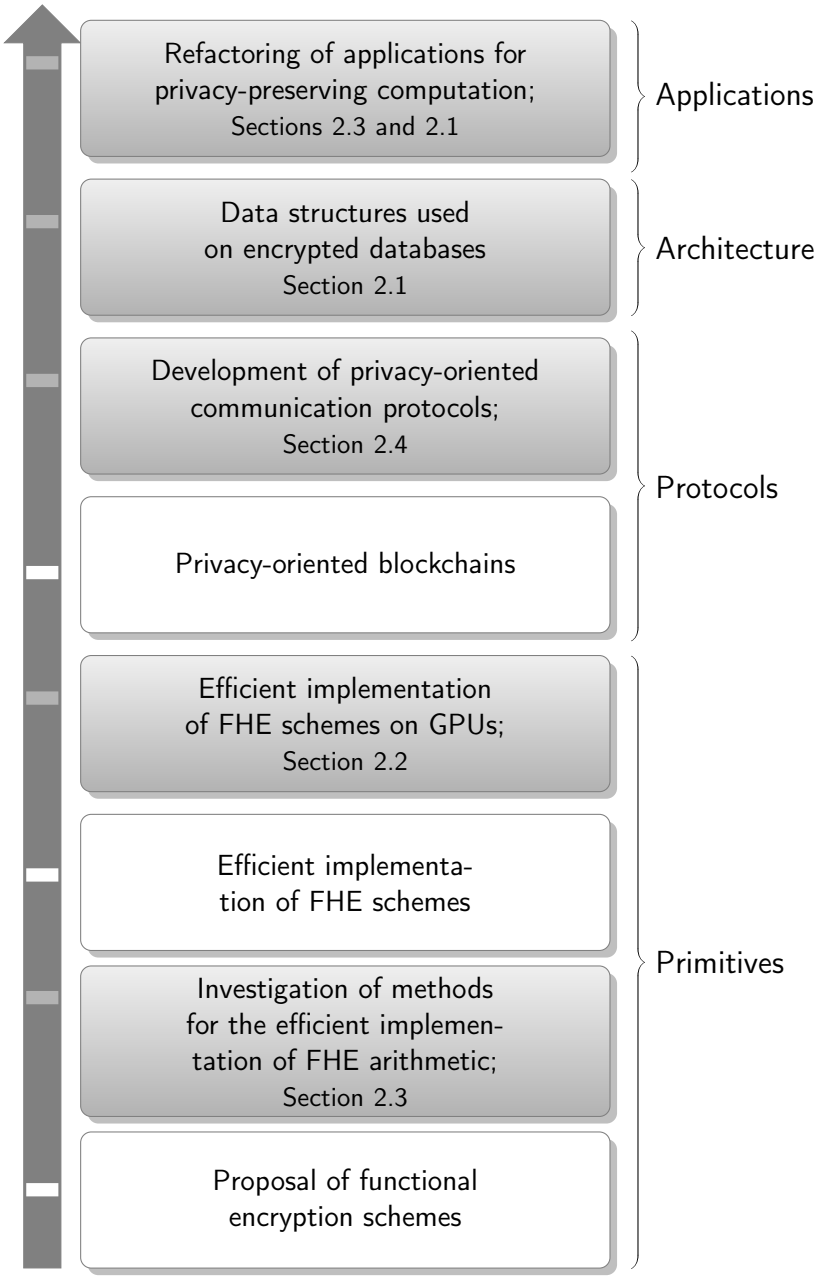


Figure 3.1: Contributions to research on privacy-preserving computing. The gray boxes relate to areas that were benefited from our contributions.

memory paradigms. We also revisit a classical method proposed by Bailey [10] to implement FFT-based techniques on memory constrained devices. We show how it can be used to improve the suitability of DGT to the limitations of a CUDA block, the basic thread structure used to guide the parallelism of a GPU, on an algorithm refereed as Hierarchical DGT (HDGT). This work demonstrates that we achieved a significant performance improvement compared to other literature works. However, a weakness of the research is that it does not isolate each implementation technique, making it impossible to conclude how better, if better at all, is the DGT over the commonly used NTT to accelerate polynomial multiplication.

The following work, presented in Section 2.3 tries to address that question. We de-

veloped two mirror CUDA-based implementations of the CKKS cryptosystem, differing only in the underlying transform: one runs polynomial multiplication through NTT, as commonly observed in the literature, and the other does that through the DGT. Different experiments were executed to highlight the differences each transform causes in the library on different levels. We compare execution latencies for various layers, from the direct comparison of the transforms to their effect on a more complex algorithm, as the logistic regression inference executed homomorphically. We observed that the DGT is a solution that better fits the GPU since its associated data structure implies higher arithmetic density and enhances the rate of dual-issued instructions.

These works, combined, present important contributions to different levels of the implementation of privacy-preserving solutions on real-world cases. With our proposals, legacy software can be adapted to process complex queries over encrypted data without additional hardware or software investment. Furthermore, these proposals connect with solutions developed for more constrained devices, like IoT, which may be able to encrypt data by running architecture-specific libraries, such as SEAL Embedded [26], submit HE-enabled ciphertexts to fog nodes that run powerful GPUs and, by implementing the solutions discussed in Sections 2.2 and 2.3, process them keeping its secrecy, and relay the outcome to powerful central stations where more it can be evaluated on complex machine learning algorithms, as described in Sections 2.1 and 2.4.

Still, our works don't consider an important problem of employing HE in the real world, as how to assert data integrity on homomorphic encrypted data and achieve non-repudiation and integrate it with privacy-preserving algorithms. These are still open problems, and the impact of available solutions on such protocols must be investigated. The malleability of HE ciphertexts imposes the need for proof that a message was indeed produced and manipulated as expected. Without that, there is no way, for instance, for someone to assert if a received ciphertext is a result of malicious behavior on a protocol. A verifier must exist and must be able to ensure that an energy bill was calculated as a sequence of additions of the measures provided by the user meter.

Chapter 4

Conclusion

This thesis contains significant contributions to the field of privacy-preserving computing. Our research covers a wide range of topics, from database encryption to the implementation of fully homomorphic encryption schemes on GPUs. In addition, our findings have underscored the importance of open science practices to enhance the quality of scientific research.

The presented papers offer contributions to many areas, from techniques for low-level implementation on GPUs to investigating how the proposals available in the literature can assist with complex protocols. For example, we propose a framework for database encryption that preserves the search capability, we investigate techniques for the efficient implementation of FHE schemes on CUDA, and we discuss the problem of building a privacy-oriented network of smart electricity meters.

During the execution of this work, we produced theoretical material and released several open-source libraries to be used as proof of concept by the scientific community. This thesis' development made clear to me the importance of that. Code releasing assists and encourages reproducibility and may enhance the quality of the following research. Furthermore, it frequently may guide an interested reader, clarify obscure parts of the paper, and offer a practical insight into the solution's applicability. By avoiding the need to re-implement code, one may more easily *stand on the shoulders of Giants* [1] and produce greater contributions to privacy-preserving computers. Thus, we encourage future researchers to adopt open science ideas and publicly disclosure source codes, datasets, and research papers.

We believe that the results presented in Section 2.3, regarding the suitability of DGT as the preferred solution to accelerate polynomial multiplication, considerably improve the discussion on low-level implementation decisions. However, its scope is limited to the CKKS and does not consider other LWE and RLWE-based schemes. One of them is TFHE, a promising HE scheme that has presented low-latency homomorphic operations and a novel and impressive programmable bootstrap procedure, which allows the evaluation of univariate functions. At the same time, the ciphertext noise is reset, allowing further homomorphic operations [13]. Future works should investigate how the different parameter settings may affect the performance of the transforms and their fitness to the

[1] As stated by Sir Isaac Newton in a letter to Robert Hooke, in 1675, regarding how he succeeded in achieving his scientific contributions to so many fields.

GPU. Moreover, schemes not dependent on the RNS representation, and consequently basis extension methods, will also avoid difficulties common to CKKS, such as the frequent data switch in and out of the transform domain.

In the same way, the work in Section 2.1 targets mainly read-only use cases. In databases with volatile records, which must be frequently updated, we would have to handle major performance bottlenecks to update records and rebuild the database index. In that case, the database index requires a key refreshment algorithm to avoid the risk of persistent passive attacks, as discussed in the paper. Moreover, it does not deeply explore NoSQL DBMSs, as graph databases. These do not necessarily correlate support operations to a relational algebra, thus future work must adapt the described techniques.

Lastly, the reason for anyone to explore HE and functional encryption schemes is to develop real-world solutions capable of protecting data secrecy. However, data malleability is an inherent problem for these schemes. Thus, verifiable computation schemes must be further developed to support the evaluation of more complex arithmetic circuits. In our research, the works available in the literature offered limited computation capability, as discussed in Section 2.4.

Bibliography

[1] Ahmad Al Badawi, Bharadwaj Veeravalli, Jie Lin, Nan Xiao, Matsumura Kazuaki, and Aung Khin Mi Mi. Multi-gpu design and performance evaluation of homomorphic encryption on gpu clusters. *IEEE Transactions on Parallel and Distributed Systems*, 32(2):379–391, 2021.

[2] Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched ntru assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 153–178, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[3] Pedro G. M. R. Alves and Diego F. Aranha. Efficient GPGPU implementation of the Leveled Fully Homomorphic Encryption scheme YASHE. Master's thesis, Institute of Computing, University of Campinas, Brazil, 2016. (In Portuguese).

[4] Pedro G. M. R. Alves and Diego F. Aranha. A framework for searching encrypted databases. In *Brazilian Symposium on Information and Computational Systems Security in*, 2016.

[5] Pedro G. M. R. Alves and Diego F. Aranha. A framework for searching encrypted databases. *Journal of Internet Services and Applications*, 9(1):1–18, Jan 2018.

[6] Pedro G. M. R. Alves, Jheyne N. Ortiz, and Diego F. Aranha. Faster homomorphic encryption over gpgpus via hierarchical DGT. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 520–540. Springer, 2021.

[7] Pedro G. M. R. Alves, Jheyne N. Ortiz, and Diego F. Aranha. Performance of hierarchical transforms in homomorphic encryption: A case study on logistic regression inference. *IACR Cryptol. ePrint Arch.*, page 99, 2022.

[8] Diego De Freitas Aranha. Renais: Residue number systems for cryptography. Online: `https://cs.au.dk/news-events/pages/2021/dff-grant-to-diego-de-freitas-aranha/`, August 2021.

[9] Ahmad Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun, and Khin Mi Mi Aung. High-performance FV somewhat homomorphic encryption on gpus: An implementation using CUDA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):70–95, 2018.

[10] David H. Bailey. FFTs in external or hierarchical memory. *J. Supercomput.*, 4(1):23–35, 1990.

[11] JoppeW. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer Berlin Heidelberg, 2013.

[12] 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, *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*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.

[13] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption library, August 2016. https://tfhe.github.io/tfhe/.

[14] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Concrete: Concrete operates on ciphertexts rapidly by extending tfhe. In *WAHC 2020–8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, volume 15, 2020.

[15] Wei Dai and Berk Sunar. cuHE: A Homomorphic Encryption Accelerator Library. Cryptology ePrint Archive, Report 2015/818, 2015.

[16] Wei Dai and Berk Sunar. cuhe: A homomorphic encryption accelerator library. In *Cryptography and Information Security in the Balkans: Second International Conference, BalkanCryptSec 2015, Koper, Slovenia, September 3-4, 2015, Revised Selected Papers 2*, pages 169–186. Springer, 2016.

[17] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for Using Homomorphic Encryption for Bioinformatics, 2015.

[18] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[19] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 144, 2012.

[20] Gemalto. Poor internal security practices take a toll, findings from the first half of 2017. http://breachlevelindex.com/assets/Breach-Level-Index-Report-H1-2017-Gemalto.pdf, USA, 2017.

[21] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[22] Wonkyung Jung, Sangpyo Kim, Jung Ho Ahn, Jung Hee Cheon, and Younho Lee. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):114–148, Aug. 2021.

[23] Huaxin Li, Qingrong Chen, Haojin Zhu, Di Ma, Hong Wen, and Xuemin Sherman Shen. Privacy leakage via de-anonymization and aggregation in heterogeneous social networks. *IEEE Transactions on Dependable and Secure Computing*, 17(2):350–362, 2020.

[24] Arvind Narayanan and Edward W Felten. No silver bullet: De-identification still doesn't work. *White Paper*, 2014.

[25] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 111–125. IEEE Computer Society, 2008.

[26] Deepika Natarajan and Wei Dai. SEAL-embedded: A homomorphic encryption library for the internet of things. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):756–779, July 2021. `https://tches.iacr.org/index.php/TCHES/article/view/8991`.

[27] Federal Senate of Brazil. Lei nº 13.709. Lei Geral de Proteção de Dados Pessoais (LGPD). `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`, August 2018. Wording given by law nº 13.853, from 2019.

[28] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[29] Al Pascual. 2017 Data Breach Fraud Impact Report: Going Undercover and Recovering Data. Technical report, Javelin Advisory Services, 2017.

[30] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[31] Yingxia Shao, Jialin Liu, Shuyang Shi, Yuemei Zhang, and Bin Cui. Fast de-anonymization of social networks with structural information. *Data Sci. Eng.*, 4(1):76–92, 2019.

[32] Lattigo's team. Lattigo v4. Online: `https://github.com/tuneinsight/lattigo`, August 2022. EPFL-LDS, Tune Insight SA.

[33] Thales. 2019 Thales Data Threat Report. `https://go.thalesesecurity.com/rs/480-LWA-970/images/2019-DTR-Global-USL-Web.pdf`, USA, 2019.

[34] Thales. 2021 Data Threat Report. https://cpl.thalesgroup.com/resources/encryption/2021/data-threat-report, USA, 2021.

[35] Wei Wang, Zhilu Chen, and Xinming Huang. Accelerating leveled fully homomorphic encryption using gpu. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2800–2803, 2014.

Appendix A

Publishers' permissions

This collection thesis contains two published papers, presented at Chapter 2. Those are:

- A work published at the Journal of Internet Services and Applications and presented at Section 2.1. The authors of papers published in that journal retain the copyright of their articles and are free to reproduce and disseminate their work and grant the right to use, reproduce or disseminate their article to third-parties in accordance with the terms of the Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0).
- A work published at the Financial Cryptography and Data Security 2021 and presented at Section 2.2. The conference uses a IFCA license which allows authors to reproduce or authorize others to reproduce the work provided that the source and the IFCA copyright notice are indicated, that the copies are not used in any way that implies IFCA endorsement of a product or service of an employer, and that the copies themselves are not offered for sale.

Following we include both licenses.

Creative Commons Legal Code

Attribution-NonCommercial 4.0 International

Official translations of this license are available in other languages.
Creative Commons Corporation (“Creative Commons”) is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an “as-is” basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

Considerations for licensors: Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright.

Considerations for the public: By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor’s permission is not necessary for any reason–for example, because of any applicable exception or limitation to copyright–then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable.

Creative Commons Attribution-NonCommercial 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.
b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.
c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.
d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.
e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.
f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.
g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.
h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.
i. **NonCommercial** means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange.
j. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public

including in ways that members of the public may access the material from a place and at a time individually chosen by them.

k. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

l. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. **License grant**.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:
   A. reproduce and Share the Licensed Material, in whole or in part, for NonCommercial purposes only; and
   B. produce, reproduce, and Share Adapted Material for NonCommercial purposes only.
2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.
3. Term. The term of this Public License is specified in Section 6(a).
4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.
5. Downstream recipients.
   A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.
   B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.
6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. **Other rights**.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
2. Patent and trademark rights are not licensed under this Public License.
3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties, including when the Licensed Material is used other than for NonCommercial purposes.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. **Attribution**.

1. If You Share the Licensed Material (including in modified form), You must:

   A. retain the following if it is supplied by the Licensor with the Licensed Material:
      i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
      ii. a copyright notice;
      iii. a notice that refers to this Public License;
      iv. a notice that refers to the disclaimer of warranties;
      v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
   B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
   C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.
2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.
4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database for NonCommercial purposes only;
b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and
c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.
b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.
d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.
b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.
b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.
c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.
d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the “Licensor.” The text of the Creative Commons public licenses is dedicated to the public domain under the CC0 Public Domain Dedication. Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark “Creative Commons” or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does

not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.

Additional languages available: العربية, čeština, Dansk, Deutsch, Ελληνικά, Español, euskara, suomeksi, français, Frysk, hrvatski, Bahasa Indonesia, italiano, 日本語, 한국어, Lietuvių, latviski, te reo Māori, Nederlands, norsk, polski, português, română, русский, Slovenščina, svenska, Türkçe, українська, 中文, 華語. Please read the FAQ for more information about official translations.

International Financial Cryptography Association
Copyright Form

Policy on copyrights and publication

In connection with its publishing activities, it is the policy of the International Financial Cryptography Association (hereinafter referred to as "IFCA") to own the copyrights to all copyrightable material in its technical publications and to the individual contributions contained therein, in order to promote research in Financial Cryptography, to protect the interests of the IFCA, its authors and their employers, and, at the same time, to facilitate the appropriate archiving and distribution of this material by others. The IFCA currently contracts with a commercial publisher to distribute its technical publications throughout the world, using various means such as traditional paper printing, Internet distribution, and CD-ROM media. IFCA may also abstract and translate its publications, and articles contained therein, for inclusion in various compendiums and similar publications, etc. When an article is submitted to the IFCA for publication, the author implicitly consents that the IFCA has the rights to do all of these things.

Policy on Public Dissemination

This policy applies to all material submitted to IFCA: The IFCA must of necessity assume that material presented at its meetings or submitted to its publications is properly available for general dissemination to the world. It is the responsibility of the authors, not the IFCA, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it.

Furthermore, the IFCA must assume, if authors use within their article material that has been previously published and/or is copyrighted by another party, that permission has been obtained for such use and that any required credit lines, copyright notices, etc., are duly noted.

IFCA Obligations

In exercising its rights under this agreement, the IFCA will make all reasonable efforts to act in the interests of the authors and employers as well as in its own interest. In handling third-party republication requests for an IFCA work, the IFCA requires that the consent of the first-named author be sought as a condition in granting republication (of a full paper) to others.; and 2) the consent of the employer be obtained as a condition in granting permission to others to reuse all or portions of a paper for promotion or marketing purposes.

Author/Company Rights

If you are employed and you prepared your paper as a part of your job, the rights to your work may rest initially with your employer. In that case, when you sign the copyright and consent to publish agreement, IFCA assumes you are authorized to do so by your employer and that your employer has consented to all the terms and

conditions of this form. If not, it should be signed by someone so authorized. (See also the Public Dissemination policy above.)

Joint Authorship

For jointly authored works, all of the joint authors should sign, or one of the authors should sign as an authorized agent for the others. In the case of multiple authorship where one or more authors are signatories under Part II of this copyright transfer form, but at least one author is not, the non-signatory of that Part should sign Part I of this copyright transfer form.

Copyright Agreement

Name of paper (hereinafter referred to as "the Work"): Faster Homomorphic Encryption over GPGPUs via Hierarchical DGT

Name of publication:

Author(s): Pedro Geraldo Morelli Rodrigues Alves, Jheyne Nayara Ortiz, Diego de Freitas Aranha

Name and address of corresponding author: Pedro Geraldo Morelli Rodrigues Alves, pedro.alves@ic.unicamp.br

PART I

(Government employees whose work is not subject to copyright should so certify by signing Part II below.)

The undersigned hereby assigns all copyright rights in and to the above work to The International Financial Cryptography Association (hereinafter referred to as "IFCA"). The undersigned also represents and warrants that the work is original and that the undersigned is the author of the work, except possibly for material such as text passages, figures, and data that clearly identify the original source, with permission notices from the copyright owners where required. The undersigned also represents possession of the power and authority to make and execute this assignment.

In return for these rights, the IFCA recognizes the retained rights noted in Items 1 and 4 below, and grants to the above authors and employers for whom the work may have been performed a royalty-free license to use the material as noted in Items 2, 3, and 4. Item 6 stipulates that authors and employers must seek permission to republish in cases not covered by Items 2, 3, 4, and 5.

1. Employers (or authors) retain all proprietary rights in any process, procedure, or article of manufacture described in the work.

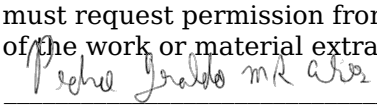
2. Authors/employers may reproduce or authorize others to reproduce the above work, material extracted verbatim from the above work, or derivative works for the author's personal use or for company use provided that the source and the IFCA copyright notice are indicated, that the copies are not used in any way that implies IFCA endorsement of a product or service of an employer, and that the copies themselves are not offered for sale.

3. Authors may publish their contributions on their respective personal Web pages after the conclusion of the Conference to which their papers have been accepted, subject to the restriction that it should carry a prominent copyright notice of the form "© IFCA" to indicate that the copyright for this contribution is held by IFCA. In addition, because IFCA uses a commercial publisher to distribute its work, it is suggested that authors include a link to the primary source of publication, which at this time is http://www.springer.de/comp/lncs/index.html.

4. Authors/employers may make limited distribution of all or portions of the above work prior to publication provided they inform the IFCA of the nature and extent of such limited distribution and gain the consent of IFCA prior thereto.

5. IFCA recognizes that work performed under a Government contract or grant may require that the Government retain royalty-free permission to reproduce all or portions of the above work, and to authorize others to do so, for Official Government purposes only. IFCA further recognizes that certain non-Government contracts or grants may have similar requirements. In either case, appropriate documentation may be attached, but IFCA's Copyright Form MUST BE SIGNED.

6. For all circumstances not covered by Items 2, 3, 4, and 5, authors/employers must request permission from the IFCA to reproduce or authorize the reproduction of the work or material extracted verbatim from the work.



Jan 1st, 2023

Authorized Signature **Date**

PART II to be completed in case an author is a Government employee

Authors who are Government employees in jurisdictions which preclude the copyright of Official work are not required to sign Part I of the IFCA Copyright

Form, but any non-Government coauthors are required to sign Part I (see Joint authorship above).

Authors whose work was performed under a Government contract or grant, but who are not Government employees, are required to sign Part I of this form. (Note: If your work was performed under Government contract but you are not a Government employee, sign Part I of this form and see item 5).

This will certify that all authors of the above work are employees of the Government and performed this work as part of their Official duties and that the work is therefore not subject to copyright protection.

_____ _____

Authorized Signature **Date**

_____

Country

Please direct all questions about IFCA copyright or this form to the President or Vice President of the IFCA.

(IFCA copyright form, 2001-08-23)