ABSTRACT

PARAMETER SELECTION IN FULLY HOMOMORPHIC ENCRYPTION SCHEMES AND FHE APPLICATIONS

by
Cavidan Yakupoglu

Quantum computing has been gaining momentum as a result of recent technological advances. Existing cryptographic systems rely on the difficult problems that can be solved by sufficiently powerful quantum computers. As the quantum age approaches, the desire to discover new difficult problems that cannot be solved by quantum systems has increased. Lattice-based cryptography is a prominent tool for the post-quantum era that facilitates the implementation of encryption systems for practical applications.

The Learning with Error (LWE) and Ring-LWE problems introduce new lattice hardness assumptions that have been incorporated into public-key cryptosystems to facilitate the implementation of numerous privacy-enhancing applications. Fully homomorphic encryption (FHE) is the crown jewel of lattice-based cryptography under the hardness assumptions of LWE and RLWE. FHE provides computation on encrypted data without revealing the actual data and private key. FHE is a versatile tool for privacy-preserving technologies, also known as the "Holy Grail of data privacy". Existing FHE schemes have a number of issues, including parameter selection, noise growth, and performance.

Parameter selection is an open problem for the FHE schemes. The initial step of this thesis focuses on the parameter selection problem on FHE. To provide security, accuracy, and efficacy, FHE parameters must be carefully chosen. Otherwise, it results in security failure, incorrect results, or inefficient computations. The introduction of a method for parameter selection will increase the pace of FHE schemes in real-world applications and make it easier for non-experts to apply FHE in their own

applications. This work employs BFVrns, a prominent lattice-based FHE scheme, as the primary and building block scheme for FHE applications. This study introduces a novel parameter selection model for the BFVrns scheme that is based on a hybrid principles-based approach that combines theoretical and experimental analyses.

With the advent of new technologies such as cloud computing and the Internet of Things (IoT), the demand for privacy-enhancing technologies has increased. Private information retrieval (PIR) and secure multiparty computation (MPC) are fundamental tools for new technologies that ensure the security and confidentiality of cloud-based data. PIR enables a user to retrieve data privately from a public database. This signifies that neither the database operator nor any other third party knows which entry the user is querying. PIR is an innovative solution for a wide range of applications, including online patents, real-time stock data, and search engines. FHE is referred to as the "Swiss army knife of cryptography" because it is utilized as a building block in numerous privacy-preserving technologies, such as computational PIR (cPIR) schemes. This study introduces three FHE-based cPIR protocols that are efficient in terms of communication and computation.

Secure MPC is another cryptographic tool that enables a number of distinct parties to perform a joint computation of a function while maintaining the confidentiality and accuracy of each party's data. Due to the sensitivity of the data in certain applications, such as medical and financial data, it is essential for the systems that users encrypt their data before sending it to a third party (e.g., cloud). In this scenario, the cloud must perform computations on encrypted data encrypted with different keys. Secure MPC is a competent solution for such systems, which are utilized in a wide variety of commercial applications, including blockchain, mobile computing, and cryptographic voting schemes. This work proposes three distinct FHE-based secure MPC protocols that improve the number of rounds and the performance and communication efficiency.

PARAMETER SELECTION IN FULLY HOMOMORPHIC ENCRYPTION SCHEMES AND FHE APPLICATIONS

by
Cavidan Yakupoglu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology – Newark
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

August 2022

Copyright © 2022 by Cavidan Yakupoglu
ALL RIGHTS RESERVED

APPROVAL PAGE

PARAMETER SELECTION IN FULLY HOMOMORPHIC ENCRYPTION SCHEMES AND FHE APPLICATIONS

Cavidan Yakupoglu

Kurt Rohloff, Dissertation Advisor — Date
Associate Professor of Computer Science, NJIT

Ali Mili, Committee Member — Date
Professor of Computer Science, NJIT

Yiannis Koutis, Committee Member — Date
Associate Professor of Computer Science, NJIT

Qiang Tang, Committee Member — Date
Associate Professor of School of Computer Science, The University of Sydney, Australia

---

Hai Phan, Committee Member — Date
Assistant Professor of Data Science, NJIT

BIOGRAPHICAL SKETCH

Author: Cavidan Yakupoglu

Degree: Doctor of Philosophy

Date: August 2022

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2022
- Master of Science in Computer Engineering,
  Ankara Yildirim Beyazit University, Ankara, Turkey, 2016
- Bachelor of Science in Computer Engineering,
  TOBB University of Economics and Technology, Ankara, Turkey, 2013
- Bachelor of Science in Mathematics,
  TOBB University of Economics and Technology, Ankara, Turkey, 2012

Major: Computer Science

Presentations and Publications:

Cavidan Yakupoglu, "A Comparative Study of Bitcoin and Alternative Crytocurrencies," *Turkey Higher Education Academic Publication Repository*, 120, 2016.

Cavidan Yakupoglu and Kurt Rohloff, "Parameter Selection for Computationally Efficient Use of the Bfvrns Fully Homomorphic Encryption Scheme," *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering*, Vol:16, No:07, 2022.

Cavidan Yakupoglu and Kurt Rohloff, "Towards Efficient FHE Based cPIR Schemes and Their Parameter Selection," *4th Workshop on Security, Privacy, and Identity Management in the Cloud (SECPID 2022).*

Cavidan Yakupoglu and Kurt Rohloff, "PREFHE, PREFHE-AES and PREFHE-SGX: Secure Multiparty Computation Protocols from Fully Homomorphic Encryption and Proxy ReEncryption With AES and SGX," *under peer review*, 2022.

To my loved ones...

ACKNOWLEDGMENT

In the name of Allah, God, the Most Gracious, the Most Merciful.

First and foremost, I would like to praise and thank Allah, the Almighty, for His blessings and giving me strength and courage during my study. May Allahs blessing goes to His Prophet Muhammad (peace be up on him), his family and his companions.

I would like to thank my advisor Prof Kurt Rohloff for his aspiring guidance and continued encouragement. I would like to express my gratitude to him for supporting me financially from his grants.

I would also like to thank my committee members; Professors Ali Mili, Yiannis Koutis, Qiang Tang and Hai Phan for their valuable comments and time.

My sincere gratitude goes to Republic of Turkey Ministry of National Education scholarship program for supporting me financially during my education and giving me this valuable opportunity to pursue my PhD study.

I would like to thank the Department of Computer Science for giving me financial support. Also, I express my gratitude to Defense Advanced Research Projects Agency and the US Navy Space and Naval Warfare Systems Center Pacific (SSCPAC) (under Contract Number N66001-17-1-4043) for funding my research.

Many thanks must also go to Dr. Yuriy Polyokov for answering my many questions patiently and clearly. I would like to thank my late team friend, Jerry Ryan for his help with my endless questions. He was a great soul. He will keep smiling in my memories.

I am so lucky to have many amazing friends. I would like to express my thanks to my friends in Turkey and here for their unconditional support, prayers and love during my difficult times. A huge thank you to my lab bestie, Sangat Vaidya at NJIT Cyber Security Lab!

I would like to mention my love for my family and gratitude for their precious prayers for me.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Dissertation Outline 5

2 BACKGROUND 6

2.1 Notation 6

2.2 Lattice Background 6

2.2.1 The Learning With Errors Problem 9

2.2.2 Ring Learning With Errors 10

2.2.3 Hardness of The LWE and RLWE Problems 10

2.3 Relation Between Hermite Factor and Security Parameter in LWE . . 11

2.4 The BFVrns Scheme 12

2.5 The Ciphertext Noise 14

3 PARAMETER SELECTION FOR COMPUTATIONALLY EFFICIENT USE OF THE BFVRNS FHE SCHEME 15

3.1 Introduction 15

3.1.1 Contributions 18

3.1.2 Related Works 18

3.2 Methodology 19

3.2.1 The Parameters 21

3.3 Relation Between LWE Parameters 22

3.4 Parameter Selection for the BFVrns Scheme 24

3.4.1 Analysis of the Parameters for EvalMult Runtime and Noise . 24

3.4.2 Analysis of the Parameters for Encryption Noise 30

3.4.3 Multi-objective Optimization 32

3.5 Results 34

TABLE OF CONTENTS
(Continued)

Chapter **Page**

3.5.1 Security 35

3.5.2 Comparison with Other Suggested Parameters 36

3.5.3 Trade-off Between Parameters 38

3.6 Discussion 39

3.7 Conclusions 40

4 TOWARDS EFFICIENT FHE-BASED CPIR PROTOCOLS AND THEIR PARAMETER SELECTION 42

4.1 Introduction 42

4.1.1 Contributions 45

4.1.2 Related Works 46

4.2 Background 47

4.2.1 Packed Encoding 47

4.3 Proposed cPIR Protocols 47

4.3.1 P-cPIR Protocol 47

4.3.2 Recursive P-cPIR Version I 51

4.3.3 Recursive P-CPIR Version II 55

4.3.4 Security and Correctness 56

4.4 Parameter Selection for Recursive P-cPIR Version II 57

4.4.1 Supporting Analysis 57

4.4.2 Parameter Selection Algorithm 57

4.5 Results 64

4.5.1 Performance Results 64

4.5.2 Parameter Selection Results 69

4.6 Conclusion 71

5 PREFHE, PREFHE-AES AND PREFHE-SGX: SECURE MULTIPARTY COMPUTATION FROM FULLY HOMOMORPHIC ENCRYPTION AND PROXY REENCRYPTION WITH AES AND SGX 72

TABLE OF CONTENTS
(Continued)

Chapter **Page**

5.1 Introduction 72

5.1.1 Our Contributions 75

5.1.2 Related Works 77

5.2 Background 78

5.2.1 Intel SGX 78

5.3 Secure MPC Protocols 79

5.3.1 PREFHE: Secure MPC from Multikey FHE and PRE 79

5.3.2 PREFHE-AES: Secure MPC from Multikey FHE and PRE with AES 82

5.3.3 PREFHE-SGX: Secure MPC from Multikey FHE and PRE with SGX 84

5.3.4 Correctness 86

5.3.5 Security Analysis 87

5.4 Results 88

5.4.1 Implementation on PALISADE 88

5.4.2 Performance Results 88

5.5 Conclusion 91

6 CONCLUSION 92

REFERENCES 93

LIST OF TABLES

Table		Page
3.1	Execution Times for Each BFVrns Operation	20
3.2	Comparison of the Parameter Set in [54] and Our Parameter Set for $\lambda = 80$ (lg Stands for log_2.)	24
3.3	Regression Analyses For BFVrns	27
3.4	Regression Results for Encryption Noise	31
3.5	Recommended Parameter Sets for the BFVrns Scheme at Least $\lambda = 128$	35
3.6	Recommended Parameter Sets for the BFVrns Scheme for Different λ and $L = 5$	35
3.7	SageMath LWE-Estimator Results from [8]	36
3.8	EvalMult Performance and CEF Comparison with [61] Parameters	37
3.9	EvalMult Performance Comparison with [61] Parameters on the Workstation for the Given Parameters	38
4.1	Communication Cost Comparison of Proposed cPIR Protocols	64
4.2	Communication Cost Comparison of the Protocols	65
4.3	Comparison of Communication Cost Results in Number of Ciphertexts [# ctxt]	66
4.4	Comparison of Computation Costs of All cPIR Protocols in Terms of Number of Polynomial Multiplication	67
4.5	Comparison of Server Computation Costs of All cPIR Protocols in Terms of Number of Polynomial Multiplication	67
4.6	Running Time per Polynomial Multiplication of BFVrns and BFV (Time Unit is ms)	69
4.7	Recommended Parameter Sets for Recursive P-cPIR Version II $d = 2$	70
4.8	Security of the Recommended Parameter Sets Given in Table 4.7	70
5.1	Performance Comparison of Main and Our Protocols	90
5.2	Performance Results of the Protocols	90

LIST OF FIGURES

Figure **Page**

3.1 Y axis represents $\log_2$ of BKZ 2.0 reduction running time and x axis is for relation of runtime in security parameter δ. This nonlinear approach is taken from [7] as LP model is considered not appropriate for lattice reduction. 23

3.2 BFVrns runtime vs. cyclotomic order m for *pbits* and L on a laptop and a workstation (ws). 25

3.3 BFVrns noise growth vs. cyctomic order m in log-log *model*. 29

5.1 Key generation and exchange phases of the PREFHE protocol. 81

5.2 Reencryption phase of PREFHE and PREFHE-SGX protocol. 81

5.3 Second reencryption and decryption of the PREFHE protocol. 81

5.4 Key generation and distribution of the PREFHE-AES protocol. 83

5.5 The second reencryption and decryption phase of the PREFHE-AES protocol. 83

5.6 The first reencryption and evaluation of function F phase of the PREFHE-AES protocol. 84

5.7 Key exchange phase of the PREFHE-SGX protocol. (Grey box represents the SGX enclave.) 85

5.8 Second reencryption and decryption of the PREFHE-SGX protocol.(Grey box represents the SGX enclave.) 85

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the improvements of technology we need privacy and security in every part of our lives such as personal computers, cloud applications and smart home devices. At the same time, advanced attack mechanisms for the powerful computers have been developed recently. Current encryption mechanisms rely on well-known problems such as factoring, discrete logarithm (DL), and elliptic-curve discrete logarithm, which are vulnerable to quantum computers with sufficient computational power. Quantum-safe algorithms, which cannot be broken by adversaries with sufficient computing power or a quantum computer, can be used to restore the robustness of current encryption systems. Current public key cryptosystems rely on factoring or DL problems that can be solved in a post-quantum world [107]. When public key encryption systems transition to quantum-resistant systems such as lattice-based, hash-based, code-based, and multivariate cryptography, adversaries will be unable to launch successful cryptanalytic attacks.

This dissertation focuses on lattice-based cryptography which is a pioneering tool for post-quantum world. In 1996, Ajtai proposed the first lattice-based cryptographic construction [4] that is based on hardness of well-known lattice problems. Ajtai and Dwork showed Short Integer Solution (SIS) problem which is an average-case lattice problem and proved that it is at least as hard as worst-case lattice problem [5]. Learning with errors (LWE) is another computational hard problem in lattice-based cryptography proposed by Regev in 2005 [99]. LWE proposes many work areas such as studies in [101, 21, 47, 91, 87, 102, 79, 80]. Ring-LWE (RLWE) is

the improved version of LWE on rings by Lyubashevsky et al. [83]. RLWE introduces shorter key sizes and computational efficiency over LWE.

RLWE provides an innovative basis for homomorphic encryption (HE) which is first proposed by Rivest et al. [100] in 1978. HE allows making some kind of computation on the encrypted data, whereas fully HE (FHE) enables evaluation of arbitrary computation on encrypted data. Users store their data on the cloud without encrypting them that results in privacy violations in the presence of curious cloud service. In this case, FHE helps users to outsource their data by keeping privacy of the data. Another important use case of FHE is secure data sharing and collaboration. Organizations collaboratively make computation on their encrypted data without seeing others' data. For example, data about health, money, and national security should be shared with third parties using systems like FHE that protect privacy [10].

In 2009, Gentry proposed the first FHE scheme based on ideal lattices [50]. FHE branches into three generations. The first generation includes Gentry's work [50], the study presented by Dijk et al. [119], but these schemes have rapid noise growth problem. The second generation encompasses the works proposed by Brakerski and Vaikuntanathan and Brakerski et al. [22, 20, 19] with the improved noise control. The third generation includes the work proposed by Gentry et al. [56] with asymmetric multiplication approach where the noise growth is mainly influenced by the first multiplicand.

This dissertion uses the BFVrns FHE scheme [61] as a building block in the parameter selection and constructing the FHE applications. Security of BFVrns depends on the hardness of the RLWE problem. BFVrns is a Residue Number System (RNS) variant of the BFV scheme [19, 46]. We use the version proposed by Halevi et al. [61] due to its efficiency over Bajard variant [13]. BFVrns also takes advantages of packing of the multiple plaintext entries into one with the Single Instruction-Multiple Data approach that is proposed in [111, 53]. The parameter selection algorithms and

FHE applications proposed in this dissertation are developed on PALISADE lattice cryptography library [97], which includes many FHE implementations such as BGV [20], BFV [19, 46], CKKS [30], FHEW [44], Stehle-Steinfeld [113], RNS variants of BGV and BFV, Bajard variant of BFV [13]. PALISADE also introduces efficient improvement techniques such as [111, 53] and proxy-reencryption and multikey capabilities. SEAL proposes the implementations of BGV, CKKS and BFVrns [71] and HElib has BGV and CKKS with packing optimization [62].

Lattice-based cryptography is criticized by its difficult parameter selection process, which has interrelated parameters that affect security, correctness, computational performance and ciphertext expansion factor (CEF). In the literature, there have been very few studies on the parameter selection problem of FHE. Some of them propose paramaters for particular parameters while they have no explanation for the other parameters of the FHE schemes. The HE standardization [6], for example, provides parameter sets that only include security parameter λ, ciphertext modulus q and ring dimension N. [96] provides noise growth estimation for FV scheme [46] on SEAL library [71]. The parameter selection directly affects the security and performance of the systems. When an inappropriate parameter set is chosen, adversaries may break the system or users may not get the optimum performance from the system. Both results are not acceptable in real-life applications. This study proposes a parameter selection algorithm for BFVrns to make quantum-resistant schemes more common and useful in real-life use cases. This approach also sheds light on other schemes' parameter selection as well.

Data creation and processing have been increased excessively, which requires new privacy-preserving technologies to share, store, and compute on the data. These systems can be implemented with FHE efficiently. Private information retrieval (PIR) and secure multiparty computation (MPC) systems achieve privacy-preserving computation on shared encrypted data. PIR allows retrieving data from a public

database without disclosing any information about the data to the server. PIR is a good fit for many applications, such as online patents, stock-market, financial, and medical databases. PIR was proposed by Chor et al. in [33] in 1995. They constructed a set of schemes to build PIR with replicated databases, which provide information-theoretic security. Another type of PIR is the computational PIR (cPIR) scheme that is based on a single-server setting. Kushilevitz and Ostrovsky proposed the first cPIR which is based on the quadratic residuosity assumption [68]. After them, many variants of cPIR were studied, such as [24, 55, 77, 59]. There have been many cPIR schemes, such as [1, 88, 76, 92, 26, 9] recently, but they cannot provide enough communication and computational efficiency. This dissertation proposes three different FHE-based cPIR constructions that outperform many of the recent cPIR schemes. It also introduces a parameter selection algorithm for these schemes that removes the burden of parameter selection from cPIR users.

Another popular use case of FHE is the secure MPC protocol. It allows many parties to conduct collaborative computation on their shared encrypted data without revealing their data to other parties. Secure MPC was first proposed by Yao in 1986 [122] for two-party case and multiparty case by Goldreich, Micali and Wigderson [58]. Some FHE-based secure MPC protocols have been proposed recently [81, 12, 82]. Some of these works have communication complexity which depends on the function to be evaluated, and they require all parties to be online during the computation [15, 43, 41]. Since these kinds of conditions cannot be provided in many use cases, secure MPC protocols should provide enough practicality to accelerate their use in real-world applications. This study introduces three different FHE-based secure MPC protocols by using its proxy-reencryption-based (PRE) feature, such that round and communication complexities are independent of the computed function. In addition, these secure MPC protocols do not impose online availability during the computation

of the function. These secure MPC protocols introduce efficiency for different use cases for real-life applications.

1.2 Dissertation Outline

The rest of the dissertation is organized as follows.

Chapter 2 gives the notation and background that are used in the dissertation and provides descriptions of Learning with Error (LWE) and Ring-LWE (RLWE). It also includes details about lattices.

Chapter 2.5 presents a parameter selection algorithm for BFVrns, which provides optimal security, computational performance and improved ciphertext extension factor.

Chapter 3.7 introduces three different computational Private Information Retrieval (cPIR) protocols, which are based on the BFVrns scheme. It also gives a parameter selection algorithm for one of the cPIR protocols as an example which is easily adjustable to other protocols. This chapter provides a comprehensive comparison of communication and computational complexity with other studies.

Chapter 5 gives three different constructions of secure Multiparty Computation (MPC) protocols from BFVrns using its Proxy Re-Encryption (PRE) capability. This chapter describes the three protocols and compares them to other state-of-the-art secure MPC protocols.

Chapter 6 summarizes the overall findings of this dissertation.

CHAPTER 2

BACKGROUND

In this section, we provide the necessary background information for this study.

2.1 Notation

We use bold lower-case letters to donate vectors and bold upper-case letters to show a matrix. $\langle ., . \rangle_q$ refers to dot product of two vectors modulo q. For a ring element $x \in \mathcal{R}$, $[x]_q$ refers to $x \mod q$ with the coefficients reduced into the range $(-q/2, q/2]$. We use $\lfloor . \rfloor$, $\lceil . \rceil$ and $\lfloor . \rceil$ to donate rounding down, rounding up and rounding to the nearest integer, respectively. $\|.\|$ refers to infinity norm and $\|.\|_2$ refers to the Euclidean norm.

2.2 Lattice Background

Definition 2.2.1 *Lattice: A lattice is a discrete additive subgroup of* $\mathbb{R}^m$ *with basis* $\boldsymbol{B} = \{\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}\}$*, a set of linearly independent vectors where* $n \leq m$*.*

$$\mathcal{L}(\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}) = \Big\{ \sum_{i=1}^{n} x_i \boldsymbol{b_i} : x_i \in \mathbb{Z} \text{ for } 1 \leq i \leq n \Big\}$$

A lattice can be also represented by it basis matrix $\boldsymbol{B} \in \mathbb{R}^{n \times m}$ *when we consider the basis vectors as rows of the matrix. When* $n = m$ *satisfies, it is considered a full-rank lattice. Since different basis can produce the same lattice, the basis of a lattice is not considered unique. If* $\boldsymbol{B'}$ *is another basis for* $\mathcal{L}$*, there is a relation such that* $\boldsymbol{B'} = \boldsymbol{BU}$ *where* $\boldsymbol{U}$ *is a unimodular matrix (integer matrix of determinant* ± 1*).*

Definition 2.2.2 *Volume of a lattice: It is determinant of a lattice which is the absolute value of the determinant of the basis matrix.*

$$vol(\mathcal{L}) = \det(\mathcal{L}(\boldsymbol{B})) = |det(\boldsymbol{B})|$$

Definition 2.2.3 *q-ary lattice: Lattices $\mathcal{L}$ are satisfying $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$ for possibly prime integer q are called as q-ary lattice. There is in one-to-one correspondence with linear codes in $\mathbb{Z}_q^n$. Lattice-based cryptographic constructions mostly use q-ary lattices. For matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and for integers q, m, n, two q-ary lattices can be showed as below and they are dual to each other showed as $\Lambda_q^{\perp} = q \cdot \Lambda_q(\boldsymbol{A})^*$. Every q-ary lattice is a full-rank lattice.*

$$\Lambda_q(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} = \boldsymbol{A^T s} \mod q, \boldsymbol{s} \in \mathbb{Z}^n\}$$
$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{Ay = 0} \mod q\}.$$

For a full-rank lattice $\mathcal{L}$, dual lattice is showed as follows.

$$\mathcal{L}^* = \{\boldsymbol{y} \in \mathbb{R}^m \mid \forall \boldsymbol{z} \in \mathcal{L}, \langle \boldsymbol{y}, \boldsymbol{z} \rangle \in \mathbb{Z}\}.$$

There have been some well-known hard problems based on lattices as given in the following definitions.

Definition 2.2.4 *Shortest Integer Solution Problem (SIS) [4]: Vectors are given as $\boldsymbol{v_i} \in \mathbb{Z}_q^n$, it aims to find non-trivial small $z_1, \ldots, z_m \in \{-1, 0, 1\}$ such that $z_1\boldsymbol{v_1} + z_2\boldsymbol{v_2} + \ldots + z_m\boldsymbol{v_m} = 0$ in $\mathbb{Z}_q^n$.*

Definition 2.2.5 *Shortest Vector Problem (SVP): For given a lattice basis $\boldsymbol{B}$, its goal is to find shortest nonzero vector in $\mathcal{L}(\boldsymbol{B})$.*

Definition 2.2.6 *Closest Vector Problem (CVP): For given a lattice basis $\boldsymbol{B}$ and a target vector $\boldsymbol{t}$, its aim is to find the closest point $v \in \mathcal{L}(\boldsymbol{B})$ to $\boldsymbol{t}$.*

Definition 2.2.7 *Shortest Independent Vectors Problem (SIVP): For given a lattice basis $\boldsymbol{B} \in \mathbb{Z}^{n \times n}$, its aim to find n linearly independent lattice vectors $\boldsymbol{S} = [\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_n}]$ in lattice that minimize the $\|\boldsymbol{S}\| = max_i \|\boldsymbol{s_i}\|$.*

Definition 2.2.8 *Bounded Distance Decoding (BDD): For a given lattice basis* $\boldsymbol{B}$*, a target vector* $\boldsymbol{t}$ *and a bound on the distance from the target to the lattice, its aim is to find a lattice vector within this bound of the target vector.*

Definition 2.2.9 *Lattice Reduction: For a given lattice, lattice reduction aims to find a basis with short, nearly orthogonal vectors. LLL (due to Lenstra, Lenstra and Lovász) [72, 90], BKZ (Block Korkine-Zolotarev) [105] and BKW (Blum, Kalai and Wasserman) [17] are well-known lattice reduction algorithms implemented in [108]. BKZ 2.0 is the new version of BKZ with recent optimization methods [29, 28].*

Definition 2.2.10 *Root-Hermite factor* δ *determines quality of the reduced basis and runtime of the reduction [48].* δ *depends on lattice reduction algorithm. Small root-Hermite factor implies a short basis vector pointing to a good quality basis. For a basis* $\boldsymbol{B} = \{\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}\}$ *which given as output of a lattice reduction algorithm, the Hermite factor* δ^n *and the shortest nonzero vector* $\boldsymbol{b_1}$ *in the basis satisfies the following equation.*

$$\|b_1\|_2 = \delta^n det(\mathcal{L})^{1/n}.$$

For q-ary lattices, a vector of length q can be found. Along with this, the length of a short vector $\boldsymbol{v}$ *that is found by a lattice reduction is approximately as follows.*

$$\|\boldsymbol{v}\| = min\{q, \delta^n \det(\mathcal{L})^{1/n}\}.$$

Lemma 2.2.1 *Geometric Series Assumption: Schnorr [104] proposed that the vectors of BKZ-reduced basis* $\boldsymbol{B}$ *has the following behavior for* $0 < \alpha < 1$*.*

$$\left\|\tilde{\boldsymbol{b}_i}\right\| = \|\boldsymbol{b_1}\| \, \alpha^{i-1}.$$

[75] confirmed this lemma with their experiences. Factor α can be computed from root-Hermite factor as follows.

$$\begin{aligned} det(\mathcal{L}) &= \prod_{i=1}^{m} \left\| \tilde{\boldsymbol{b}_i} \right\| \\ &= \|\boldsymbol{b_1}\|^m \prod_{i=1}^{m} \alpha^{i-1} \\ &= \|\boldsymbol{b_1}\|^m \, \alpha^{\sum_{i=1}^{m} i-1} \\ &= \delta^{m^2} det(\mathcal{L}) \alpha^{m(m-1)/2} \\ \Rightarrow \alpha &= \delta^{-2m/(m-1)} \approx \delta^{-2}. \end{aligned} \tag{2.1}$$

2.2.1 The Learning With Errors Problem

Regev proposed Learning with errors (LWE) problem [99] which is a generalization of Learning with Parity (LPN) problem with a larger modulus. LWE is provably as hard as some worst-case problems. It has parameters like m dimension, q ciphertext modulus, error distribution σ over $\mathbb{Z}$.

Definition 2.2.11 *LWE [99]: For $\boldsymbol{s} \in \mathbb{Z}_q^n$, an error term is $e \leftarrow \chi$, LWE distribution $A_{s,\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing uniformly random $\boldsymbol{a} \in \mathbb{Z}_q^n$ and outputting this pair as $(\boldsymbol{a}, t = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Search version of LWE is finding $\boldsymbol{s}$ from number of independent samples $(\boldsymbol{a_i}, t_i) \leftarrow A_{s,\chi}$. The decision version of LWE is to distinguish with non-negligible advantage between any independent samples $(\boldsymbol{a_i}, t_i) \leftarrow A_{s,\chi}$ and same number of samples from uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Also, we can show it with a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and $e_i \in \mathbb{Z}$, $t_i \in \mathbb{Z}_q$, vectors $\boldsymbol{e} \in \mathbb{Z}^m$, $\boldsymbol{t} \in \mathbb{Z}_q^m$ and we get $\boldsymbol{A}$, $\boldsymbol{t} = \boldsymbol{A}^T \times \boldsymbol{s} + e \mod q$. Decision and search LWE problems can use this notation as well.*

LWE problem may also be seen as an average-case bounded-distance decoding (BDD) problem for a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ where the lattice is

$$\Lambda(\boldsymbol{A}^T) = \{\boldsymbol{z} \in \mathbb{Z}^m : \exists \boldsymbol{s} \in \mathbb{Z}_q^n \text{ such that } \boldsymbol{z} = \boldsymbol{A}^T \boldsymbol{s} \mod q\}.$$

Also, SIS (Shortest Integer Solution) problem can be considered for the same matrix $\mathbf{A}$ where the lattice is

$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y}\boldsymbol{A}^T = 0 \mod q\}.$$

2.2.2 Ring Learning With Errors

Definition 2.2.12 *RLWE [83]: For a security parameter λ, a monic polynomial $f(x) = x^n + 1$ which is irreducible over $\mathbb{Z}$ and a cyclotomic polynomial with n is a power of 2. Let $\mathcal{R} = \mathbb{Z}[x]/f(x)$ be a polynomial ring. Let $q \in \mathbb{Z}$ be a sufficiently large integer modulus where $q \geq 2$. $\mathcal{R}_q = \mathcal{R}/q = \mathbb{Z}[x]_q/f(x)$. χ is a distribution over $\mathcal{R}$. $RLWE_{n,q,\chi}$ problem is to distinguishing two distributions as follows. The first distribution is uniformly generated samples $(a_i, b_i) \in \mathcal{R}_q^2$. In the second distribution, firstly one samples s from $\mathcal{R}_q$ uniformly and samples $(a_i, b_i) \in \mathcal{R}_q^2$ where $a_i \leftarrow \mathcal{R}_q$ uniformly and $e_i \leftarrow \chi$, $b_i = a_i s + e_i$. Since SVP problem can be reduced to RLWE [83], RLWE problem is considered hard for polynomial-time quantum algorithms. RLWE is a more efficient to use due to its lower key sizes and efficient operations.*

2.2.3 Hardness of The LWE and RLWE Problems

Since the best known algorithms to solve LWE run in exponential time and quantum algorithms do not work for LWE setting, LWE problem is known as a hard problem. Also, it is an extension of learning parity with noise problem and it is also known as hard. For error distribution $\sigma = D_{\mathbb{Z},\alpha q}$ and $\alpha q \geq 2\sqrt{n}$, Regev [99] showed a quantum reduction from a worst case lattice problem to LWE showing that LWE is provably as hard as some lattice problems in the worst case and requires time exponential in the main security parameter to solve. LWE is hard by making some assumptions and using the worst-case hardness of standard lattice problems such as GAPSVP (Gap-Shortest Vector Problem), SIVP [99, 93]. Peikert gave a classical reduction

from gapSVP to BDD [93] and Regev provided a quantum reduction that reduces from search SIVP to BDD [99]. BDD and LWE are very closely related problems.

Similar to the LWE problem, the RLWE problem is considered a hard problem since it is related to some well-known lattice problems [93, 94, 84]. Lyubashevsky et al. [83] showed a reduction from approximate SIVP on ideal lattices to decision Ring-LWE problem. Peikert et al. showed another reduction from worstcase lattice problems to decision Ring-LWE for any number field [94].

Theorem 2.2.2 *[83]. For d is a power of 2, let ring $\mathcal{R} = \mathbb{Z}[x]/(x^d+1)$, q is a prime integer $q = 1 \mod d$ and $B = \omega\sqrt{d \log d}$. $\chi \leftarrow \mathcal{R}$ with with length B. If there exists an efficient algorithm solving $RLWE_{d,q,\chi}$, there is an efficient quantum algorithm to solve $d^{\omega(1)}$. q/B- approximate worst-case SVP for ideal lattices over $\mathcal{R}$.*

2.3 Relation Between Hermite Factor and Security Parameter in LWE

In this section, we analyse relation between root-Hermite factor and security parameter. This relation is used in the next chapters. Hermite factor was defined in [48] as quality of reduced basis by reduced a lattice reduction algorithm. The most prominent lattice reduction algorithms are LLL, BKZ, BKW. δ is called as root-Hermite factor and δ^n is called as Hermite factor.

A basis $\boldsymbol{B}$ of an n-dimensional lattice Λ has Hermite factor δ^n for $\delta \geq 1$ satisfies $\|b_1\| = \delta^n det(\Lambda)^{1/n}$ where $\boldsymbol{b_0}$ is the shortest non-zero vector in the output basis.

We will show Hermite factor on q-ary lattices. Suppose we have a matrix $\boldsymbol{A} \in \mathbb{Z}^{n \times d}$ with $d \geq n$ and following two lattices are d-dimensional q-ary lattices:

$$\Lambda_q(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^d : \boldsymbol{y} = \boldsymbol{z} \cdot \boldsymbol{A} \mod q \text{ for some } \boldsymbol{z} \in \mathbb{Z}^n\}$$

$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^d : \boldsymbol{y} \cdot \boldsymbol{A}^T = 0 \mod q\}$$

[75] gives important results on δ and security level. They show running time of BKZ experiments on random q-ary lattices with different paremeters and $m = \sqrt{n \log_2(q)/\log_2(\delta_0)}$ which is optimal subdimension that provides root-Hermite factor δ_0. They use least-square regression in these experiments and they get $t_{BKZ}(\delta) = 1.086/\log_2(\delta) - 91$. Also, they accept $\log_2(t_{BKZ})(\delta) = 1.8/\log_2(\delta) - 110$ for conservative lower bound. This result says that there is a relation between δ and security level $\log_2(t_{BKZ})$. [118] accepts $\log_2(t_{BKZ})$ as security level. They give the equation below where Q is largest modulus and σ is standard deviation and λ is security level;

$$n \geq \frac{\log_2(Q/\sigma)(\lambda + 100)}{7.2} \tag{2.2}$$

The work from [54] uses the same security analysis from [75] to get ring dimension n. Also, [54] derives estimation for Q (largest modulus) for L-leveled homomorphic encryption scheme:

$$Q \approx 2^{22.5.L - 3.6.\sigma.d^L}$$

We give LP model here to show the relation between Hermite factor and security level. But, we will show the correct relation equation in the next chapter. Because LP model was argued not to be compatible in the context of lattice reduction.

2.4 The BFVrns Scheme

We give a brief explanation for the BFVrns scheme referenced from [19, 46, 61], but mainly from [61]. Fan and Vercauteren [46] give RLWE version of work proposed by Brakerski in [19]. Residue Number System (RNS) variant of the BFV scheme is presented by Halevi et al. [61] for more efficient procedures for BFV. Its implementation is publicly accessible in PALISADE library. BFVrns provides some improvements on decryption and homomorphic multiplication by using Chinese

Remainder Theorem (CRT) representation. We take m-th cyclotomic polynomial and we choose m to be power of 2 for the experiments. Also, we use RLWE mode in PALISADE library [97] which means s is sampled from error distribution. In this study, error distribution χ refers to discrete Gaussian distribution.

BFVrns includes some parameters such as t, $q \in \mathbb{Z}$, t stands for plaintext modulus, n stands for $\phi(m) = n$, ciphertext modulus is $Q = \prod_{i=1}^{k}, q_i$ for the same size q_i, σ is the standard deviation of error distribution χ. rw stands for the size of the relinearization window. Let rings be $\mathcal{R} = \mathbb{Z}[x]/\Phi_m(x)$, $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$, $\mathcal{R}_t = \mathcal{R}/t\mathcal{R}$. We choose a uniform $\alpha_i \in \mathcal{R}_Q$ and $e_i \leftarrow \chi$, ${q_i}^* = Q/q_i$, $q_i' = [{q_i}^{*-1}]_{q_i}$, $\beta_i = [q_i' {q_i}^* s^2 - \alpha_i s + e_i]_Q$ for $i = 0, 1, \ldots, k$. The public key consists of $\boldsymbol{pk}$ and $\boldsymbol{W_i} \coloneqq (\beta_i, \alpha_i)$.

- **KeyGeneration:** Secret Key: Sample $s \leftarrow \chi$, set $\boldsymbol{sk} = (1, s) \in \mathcal{R}^2$.
  Public Key: Sample $a \leftarrow \mathcal{R}_Q$, $e \leftarrow \chi$, set $\boldsymbol{pk} = ([-(as + e)]_Q, a) \in \mathcal{R}_Q^2$.
- **Encryption**(m, pk)**:** $m \in \mathcal{R}_t$, $u \leftarrow \chi$, $e_0, e_1 \leftarrow \chi$, $\boldsymbol{ct} = [u \cdot \boldsymbol{pk} + (e_0, e_1) + (\Delta m, 0)]$ where $\Delta = Q/t$. Output $\boldsymbol{ct}$.
- **Decryption**$(\boldsymbol{ct}, \boldsymbol{sk})$**:** $\boldsymbol{ct} = (ct[0], ct[1])$, $x = [\langle \boldsymbol{sk}, \boldsymbol{ct} \rangle]_Q = [c[0] + c[1]s]_Q$ and output $m \coloneqq [\lceil x \cdot t/Q \rfloor]_t$.
- **Add**$(\boldsymbol{ct_0}, \boldsymbol{ct_1})$**:** Output $[\boldsymbol{ct_0} + \boldsymbol{ct_1}]_Q$.
- **Mult**$(\boldsymbol{ct_0}, \boldsymbol{ct_1})$**:** For $\boldsymbol{ct_0}, \boldsymbol{ct_1}$, tensoring and relinearization are computed as follows:
  - **Tensoring:** $c[0] = ct_0[0]ct_1[0]$, $c[1] = ([ct_0[0]ct_1[1] + [ct_0[1]ct_1[0]]_Q)$, $c[2] = ct_0[1]ct_1[1]$ and $c = (c[0], c[1], c[2])$. Output $c'[i] = [\lceil t/Q \cdot c[i] \rfloor]_q$ for i =0, 1, 2.
  - **Relinearization:** If $rw > 0$, decompose $c'[2]$ into base of rw bits, if $rw = 0$, decompose $c'[2]$ into its CRT components for $i = 0, 1, \ldots, k$, $c'[2][i] = [c'[2]]_{q_i}$, set $c''[0] = [\sum_{i=1}^{k} \beta_i c'[2][i]]_Q$, $c''[1] = [\sum_{i=1}^{k} \alpha_i c'[2][i]]_Q$, output $\boldsymbol{ct_{mult}} = [(c'[0] + c''[0], c'[1] + c''[1])]_Q$.
- **EvalAtIndexKeyGen**$(sk,$ $indexList)$**:** $indexlist$ is the indices that the function is evaluated at. It returns evaluation key map that helps to reach the indices at $indexList$.
- **EvalAtIndex**(c,ix)**:** It returns the element at the index ix as a ciphertext c_{ix}.

- **MultiPartyKeyGen(pk):** $pk = (p_0, p_1)$, $a \leftarrow p_1$, $s \leftarrow \chi$, $b = -(e + (a \cdot s)) + p_0$, set new key $\boldsymbol{pk'} = (b, a)$ and $sk = s$.
- **ReKeyGen($\boldsymbol{newpk}$, $\boldsymbol{oldsk}$):** For $rw = 0$, $\boldsymbol{newpk} = (p_0, p_1)$, $e1_i, e2_i \leftarrow \chi$, $u_i \leftarrow \chi$. For each element in $oldsk$; $c0_i = p_0 \cdot u_i + e1_i + f_i$ where f_i are elements at index i in $\boldsymbol{oldsk}$. $c1_i = p_1 \cdot u_i + e2_i$. Set $\boldsymbol{evalKey} = (\{c0_k\}, \{c1_k)\}$ for $0 \leq k < v$ where v is the number of elements in $\boldsymbol{oldsk}$. Output $\boldsymbol{evalKey}$.
- **ReEncrypt($\boldsymbol{evalKey}$, $\boldsymbol{ciphertext}$):** Apply KeySwitch on $\boldsymbol{ciphertext}$ with $\boldsymbol{evalKey}$.
- **KeySwitch($\boldsymbol{evalKey}$, $\boldsymbol{ciphertext}$):** $\boldsymbol{evalKey} = (b, a)$. Set $digitsC1$ as CRT Decompose of $ciphertext[1]$ on base rw, $c_1 = digitsC1[0] \cdot a[0]$, $c_0 = c_0 + digitsC1[0] \cdot b[0]$. Set $c_0 = c_0 + \sum_{i=1}^{k-1}(digitsC1[i] \cdot b[i])$, $c_1 = \sum_{i=1}^{k-1}(digitsC1[i] \cdot a[i])$ where k is size of $digitsC1$. Output $\boldsymbol{newCiphertext} = (c_0, c_1)$.

2.5 The Ciphertext Noise

In this section, we define what the noise means in the BFVrns scheme. A ciphertext $\boldsymbol{c}$ and secret key $\boldsymbol{s}$ are vectors over a polynomial ring $\mathbb{A}$. $\boldsymbol{c}$ is decrypted using secret key s reducing to ciphertext modulus q, later reducing the result to plaintext modulus t. The noise in the ciphertext is the result of the first step of this process as follows:

$$noise \leftarrow [\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q. \tag{2.3}$$

The size of a polynomial $a \in \mathbb{A}$ is the l_∞ norm of its canonical embedding as used in [54]. Canonical embedding of $a \in \mathbb{A}$ into $\mathbb{C}^{\phi(m)}$ is the $\phi(m)$-vector of complex numbers $\sigma(a) = (a(\zeta_m^i))_i$ where ζ_m is a complex primitive m-th root of unity with index i range over all of $(\mathbb{Z}/m\mathbb{Z})^*$. We use the canonical embedding norm to evaluate the norm of the noise. Canonical embedding norm of a is the norm of $\sigma(a)$ and showed as follows:

$$\|a\|_\infty^{can} = \|\sigma(a)\|_\infty. \tag{2.4}$$

In the experiments, we use the `Norm()` function in PALISADE which returns the infinity norm (basically the largest value in the ring element).

CHAPTER 3

PARAMETER SELECTION FOR COMPUTATIONALLY EFFICIENT USE OF THE BFVRNS FHE SCHEME

3.1 Introduction

Fully homomorphic encryption (FHE) refers to a family of encryption methods that allows evaluation of arbitrary computation on encrypted data. FHE was first proposed by Rivest et al. [100] in 1978. It was an open problem until Gentry proved that FHE is possible by using lattices in his work in 2009 [50]. After Gentry's breakthrough, many variants of FHE were proposed in the literature [113, 119, 110, 51].

The Learning with Errors problem (LWE) was introduced in 2005 by Regev [99] that is regarded a hard problem based on lattice-based cryptography. Ring-LWE [114], the more efficient variant of LWE, allows constructing some FHE schemes such as BGV [20], BFV [46], BFVrns [61] and CKKS [30]. BFVrns stands for Residue Number System (RNS) variant of the BFV scheme.

Due to the complexity of FHE schemes, we have the following challenges. The first one is rapid noise growth in the ciphertext. Every ciphertext contains some amount of noise, which grows with each evaluation. When the noise level surpasses a threshold, decryption fails. To reduce the noise in the ciphertext, a time-consuming procedure called bootstrapping is required. We may adjust the parameters properly to keep the noise small and so we run bootstrapping fewer times. This results in an increase in the performance of FHE schemes. We offer an approach for optimizing the noise in the ciphertext without reducing security. The greatest increase in noise occurs following a multiplication (EvalMult) operation with $B \rightarrow B \cdot poly(n)$ in the ciphertext where n is the depth of the circuit. Once we have mastered and optimized noise growth for multiplication, it becomes much easier to manage noise growth for circuits assessed using the FHE scheme. That is why we concentrate on multiplication

operation and run performance tests and optimize the parameters for this specific operation.

The second one is high Ciphertext Expansion Factor (CEF). CEF is the ratio of the ciphertext modulus size and plaintext modulus size. CEF has a significant influence on the communication cost of the FHE schemes. When transferring ciphertexts one point to another, a small CEF reduces network costs. The transfer of ciphertexts is frequently observed when computing is outsourced to a third party or when privacy-enhancing technologies such as private information retrieval (PIR), multiparty computation (MPC), or other cloud applications are used. CEF has the potential to make a significant practical impact on communication cost for such scenerios. For instance, the approach described in [36] entails transferring approximately 73 TB of encrypted data across the network for 4MB of plaintext data. This issue can be resolved if we have an efficient CEF.

The final problem is parameter selection for FHE schemes. This issue is regarded particularly challenging owing to the interrelation of its parameters. Lattice-based schemes require large lattice dimensions and modulus that makes parameter selection more complicated. Inefficient parameter selection can result in significantly longer FHE operation times. Additionally, each parameter has a unique impact on the correctness, performance, and security, complicating the procedure. For example, increasing the ciphertext modulus Q allows for more noise but decreases the security level. Because noise increase is proportional to the plaintext modulus t, t should be small. However, decreasing the value of t leads in an increase in the amount of plaintext used to represent data, resulting in a drop in performance. Increased ring dimension n values are chosen for increased security, but at the expense of performance. In this situation, users face considerable difficulty in selecting parameters.

We are familiar with the behaviour of certain parameters that are defined by the HE standardization [6]. The HE Standard specifies parameter settings such as lattice dimension n and modulus size $\log Q$ for various security levels $lambda$. Additionally, we must choose additional parameters such as the plaintext modulus t, standard deviation σ of the error distribution, relinearization window size rw, as correctness, security, and performance are all dependent on these factors. The issue is that there is no feasible way to understand the parameters when all of them are taken together.

To solve these problems optimally, we propose a novel parameter selection approach. We combine the theoretical analysis with the practical experiments. For the practical experiment analyses, we run many BFVrns multiplication experiments with various parameter sets and create the dataset for runtime and noise growth, then we analyze the effect of each parameter on this data set with regression analysis. Regression analysis is a well-known statistical method for estimating the relationships between a dependent variable and independent variables. It is widely used to provide data analytics in many areas such as business, economics and health care industry. In cryptography, to determine behavior of procedures or parameters it is becoming an important tool.

In the multi-objective optimization step, we use estimated results from these regression analysis as the objective functions and constraints along with the theoretical noise bounds of BFVrns. This enables finding the optimum set of parameters for security, performance and CEF.

In this study, we provide a concrete understanding of all parameters on the security, performance. As a result of these steps, our techniques identify more optimal parameter sets that provide at least 128-bit security, improved performance and CEF. We propose parameter sets for the security levels that are proposed in the HE Standard.

3.1.1 Contributions

We propose a method for the parameter selection based on statistical approach. In this study, we present these contributions as follows:

- A parameter selection is proposed for optimizing performance, noise growth and CEF.
- A new hybrid model for the parameter selection problem is presented.
- The main parameters of the FHE scheme are included in the parameter selection while other studies mainly focus on ciphertext modulus and lattice dimension.
- The new smaller ring dimensions are proposed with the method of extrapolating BKZ 2.0 runtime.
- The influence of the different parameters on performance and security are identified approximately.
- Different sets of parameters for BFVrns are suggested for PALISADE library users.

For the future work, we aim to create a background for the automatic parameter selection scheme that enables FHE schemes easier to use and more practical for real-world use cases. Since any parameter selection study on the BFVrns scheme has not been studied so far, this work will be the first one in this area.

3.1.2 Related Works

There have been few parameter selection studies for the FHE schemes so far. The most well-known study has been proposed by Player in her PhD dissertation [96]. She presents some important noise analyses for SEAL library [71], but she provides recommendation only for limited number of parameters. Albrecht et al. provide LWE Estimator[1] for estimating security for the LWE problem [8]. Lindner and Peikert propose a model to determining the relation between the security parameter and other parameters [75] for LWE. A similar approach to the LP model has been proposed

[1]https://bitbucket.org/malb/lwe-estimator/src/master/ Retrieved on 5 May 2022

by Albrecht et al. [7] with a quadratic approach. Costache et al. contribute to BGV and FV noise growth concept by presenting new notions of noise growth for BGV [37]. Viand and Shafagh propose a framework that automatically reports FHE overhead and selects the necessary parameters [120]. Herbert proposes a parameter selection module, CinguParam, for the BFV scheme [64] for different libraries such as Cingulata, FV-NFLlib and Microsoft SEAL, but provides only ciphertext modulus and ring dimension like Homomorphic Encryption standard [6]. Note that these prior works are all about security and not about performance. Our work will be the first one for improving performance and CEF in the BFVrns scheme.

3.2 Methodology

In this section, we explain the methodology that we follow in this parameter selection approach. Firstly, we run KeyGeneration, Encryption, Decryption, Evaluation of Addition (EvalAdd), Evaluation of Multiplication (EvalMult) functions for BFVrns and we see that EvalMult takes the most of the running time. Also, we can see this result from the complexities of the operations where EvalMult has the highest computational complexity according to [61]. We provide an example runtime experiment result for BFVrns in Table 3.1 for $L = 1$ and CEF $= 2.13$. Also, the largest noise growth develops after the EvalMult operation as shown in BFVrns [61]. Thus, we have decided to work on EvalMult to improve the performance and control the noise growth. In the rest of this chapter, when we mention runtime and noise growth, they refer to the EvalMult operation. We analyze the runtime and noise behavior of the BFVrns scheme on a PC running MacOS Catalina on Intel Core i5 with two cores, 2.6 GHz CPU and 8 GB RAM with single thread mode.

Regression Analysis: Our aim is to develop two approaches for predicting runtime and noise growth. This step helps us to understand the influence of parameters on the noise growth and performance. To create the dataset we run

Table 3.1. Execution Times for Each BFVrns Operation

Operation	**Time(ms)**
KeyGen	5.258
Encryption	5.88
EvalAdd	0.203
EvalMult	20.625
Decryption	0.935

the experiments for all combinations of given parameter values and measure runtime and norm of noise after one EvalMult operation. We apply mean normalization on the runtime dataset to get more accurate results. We use STATA [112] Ordinary Least Squares (OLS) method that estimates the parameters in a regression model by minimizing the sum of the squared residuals. We provide different statistical measures to analyze the regression result. In statistic, *R-squared* value is the relative measure of the percentage of dependent variable variance that the model explains, but not sufficient by itself. We also provide *standard error* for each coefficient which is a stronger goodness-of-fit measure for regression analysis. The lower values of *standard error* signify that observations are closer to the fitted line. In statistic, statistical significance is often referred as *p-value* short for "probability value". A *p-value* is used in the hypothesis testing to help you support or reject the *null hypothesis*. *Null hypothesis* is the opposite of the alternate hypothesis that we offer. A *p-value* less than *0.05* means that less than a 5% probability the *null hypothesis* is correct. It gives strong evidence against the *null hypothesis* and shows statistical significance of our hypothesis for parameters on the runtime and noise. *p-value* is a suitable measure to confirm that the independent variable really belongs to the model. That is why we mostly take *p-value* into consideration while deciding on the parameters and coefficients. Even though some regression models have small *R-squared* value, when

p-value is small enough, regression model is considered valid. Another important point in the regression analysis is the decision on the parameters that we use in the multi-objective optimization step. First of all, we conduct a regression analysis on all parameters by accepting all parameters that have some influence on the runtime and noise. From these statistical indicators mentioned above, we take *p-value* into consideration while determining which parameters to include in the second regression. After deciding the parameters that are statistically significant at 1 percent level (where *p-value* is < 0.01), a second regression analysis is applied to these parameters to obtain the final coefficients of the estimated equation. We observe during these steps that the coefficients of meaningful parameters do not change. In the tables of regression results, we provide the results of the first regression analysis to present the impacts of all parameters. All data for the regression analysis is generated by single-threaded process.

Multi-objective optimization: The parameters of the scheme show different behaviors in the noise growth, security and runtime. For example, a large rw shows a decreasing effect on runtime but an increasing behavior on the noise growth. Our aim is to find an optimal parameter set for both the security and performance without compromising security. We merge the regression results with multi-objective optimization (MOO) algorithms. We include the theoretical noise bounds from [61] as the nonlinear inequalities in the MOO process.

3.2.1 The Parameters

In the experiments, we analyze the main parameters of BFVrns as follows; m (m-th cyclotomic polynomial), L (multiplicative depth of the function), σ (standard deviation of error distribution), $pbits$ (size of ciphertext modulus (q_i) which refers to $dcrtbits$ variable in PALISADE), rw relinwindow size, t plaintext modulus. $\log Q$ is

the joint ciphertext modulus of *pbits* values. We add rw, t, L to the experiments so that we can recommend all FHE parameters for non-expert users.

Security of the parameter values for the experiments. We determine the value range for each parameter that ensures correctness and at least 128-bit security level that complies with HE Standard. This means that we provide zero decryption failure rate for the value ranges of the parameters and proposed parameters. Also, when we test every possible combinations of the parameters in these ranges, all of them give correct decryption result. For t, we use a random range of values that does not affect the security. For σ values, we choose the most known values like 3.2 as in HE standard, 4 as in [54], 3 as mentioned in many research studies. We also add 3.6 which is safe to use due to being in the range of valid σ values given by HE standard. We choose the range of rw values from 1 to 30 and *pbits* values from 30 to 60 due to the implementation of PALISADE. We want to analyze the behavior of rw by choosing $rw > 0$. We test all combinations of parameters on [103] and each gives at least 128-bit security.

We run the experiments on the PALISADE homomorphic encryption library [97] release version 1.10.6 [2]. We choose BFVrns for the parameter selection model which is one of the most efficient FHE schemes.

3.3 Relation Between LWE Parameters

In this section, we create a runtime estimation model on BKZ 2.0 [29] to get a ring dimension estimation. BKZ is a lattice reduction algorithm for solving LWE problem [106]. We get another relation between runtime and security for LWE that also applies to RLWE. We use a similar method given in Lindner-Peikert model [75], but we use BKZ 2.0 that provides the current improvements such as pruned enumeration, pre-processing of local blocks and early termination. We use the same approach as

[2]https://gitlab.com/palisade/palisade-release Retrieved on 5 May 2022

in [7] that is a non-linear approximation. The linear approach in [75] is mentioned as less intiuitive than non-linear approach in [7]. Also, we do not use the result of this study [7] that uses data points from [78] which are not enough to make regression analysis accurately. We have more data points than [7], so it provides better approximation. Thus, we create our model with similar approach as given in the work by Albrecth et al. [7]. We create our data points with SageMath [103] using BKZ 2.0 to get more accurate result. We get the estimated relation as Equation (3.1) where δ is Root Hermite Factor which is a security parameter in LWE. Figure 3.1 shows our estimation and the estimation mentioned in [7]. Our estimation fits the data better than the estimation in [7]. Also, we provide a comparison of the result of our model and the result in Appendix C.3 [54] as in Table 3.2 below.

Our estimation proposes smaller lattice dimension while giving very close values for ciphertext moduli. We present Table 3.2 to show that Equation (3.1) provides smaller values of ring dimension with the same security condition.

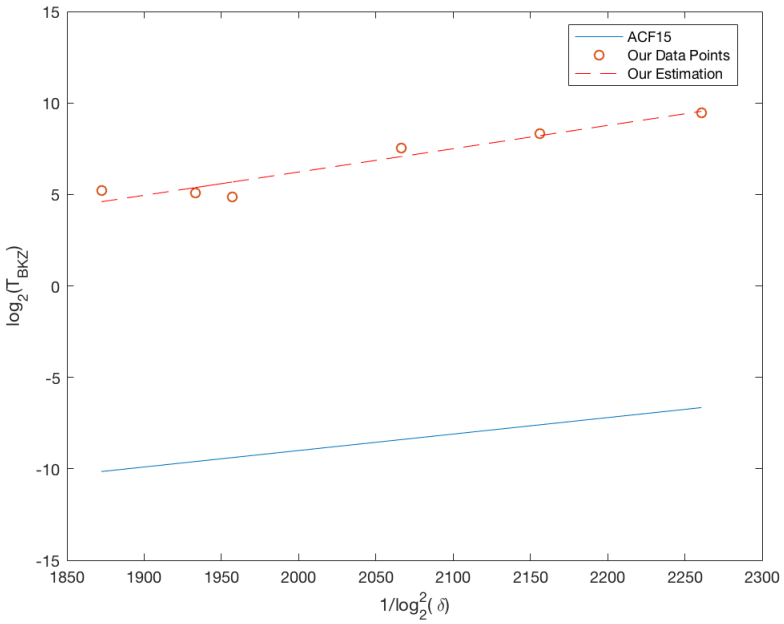


Figure 3.1. Y axis represents $\log_2$ of BKZ 2.0 reduction running time and x axis is for relation of runtime in security parameter δ. This nonlinear approach is taken from [7] as LP model is considered not appropriate for lattice reduction.

$$\widehat{\log_2(T_{BKZ2.0})} = \frac{0.01273}{log_2^2(\delta)} - 19. \tag{3.1}$$

Table 3.2. Comparison of the Parameter Set in [54] and Our Parameter Set for $\lambda = 80$ (lg Stands for log_2.)

	[54] Estimates				Our Estimates			
L	n	$lg(q_0)$	$lg(q_i)$	$lg(Q)$	n	$lg(q_0)$	$lg(q_i)$	$\lg(Q)$
10	9326	37.1	17.9	177.3	7740	36.8	17.7	173.9
20	19434	38.1	18.4	368.8	16010	37.8	18.2	361.6
30	29749	38.7	18.7	564.2	24500	38.4	18.5	553.3
40	40199	39.2	18.9	762.2	33020	38.9	18.8	747.5
50	50748	39.5	19.1	962.1	41660	39.2	18.9	943.6
60	61376	39.8	19.2	1163.5	50440	39.5	19.1	1141.3
70	72071	40.0	19.3	1366.1	59150	39.7	19.2	1340.1
80	82823	40.2	19.4	1569.8	67950	39.9	19.3	1540.1
90	93623	40.4	19.5	1774.5	76800	40.1	19.4	1740.9

3.4 Parameter Selection for the BFVrns Scheme

In this section, we provide our parameter selection approach and parameter sets for different levels L and security levels λ for the BFVrns scheme.

3.4.1 Analysis of the Parameters for EvalMult Runtime and Noise

In this section, we present two regression analyses to create the models for the BFVrns runtime and noise growth based on the parameters. We create the BFVrns runtime and noise datasets with t; 20, 30, 40, 50, m; 4096, 8192, 16384, 32768, L; 1, 2, 3, 4, σ; 3, 3.2, 3.6, 4, $pbits$; 30, 38, 46, 52, rw; 1, 10, 20, 30. $\hat{rt}$ and $\hat{noi}$ stand for the estimated runtime and estimated noise, respectively. We choose smaller values of L due to hardware constraints. Besides that, regression analysis helps projecting behavior of the parameters to larger values. Thus, we can provide reasonable estimation for larger values of L. We create the noise dataset in $\log - \log$ scale as explained below.

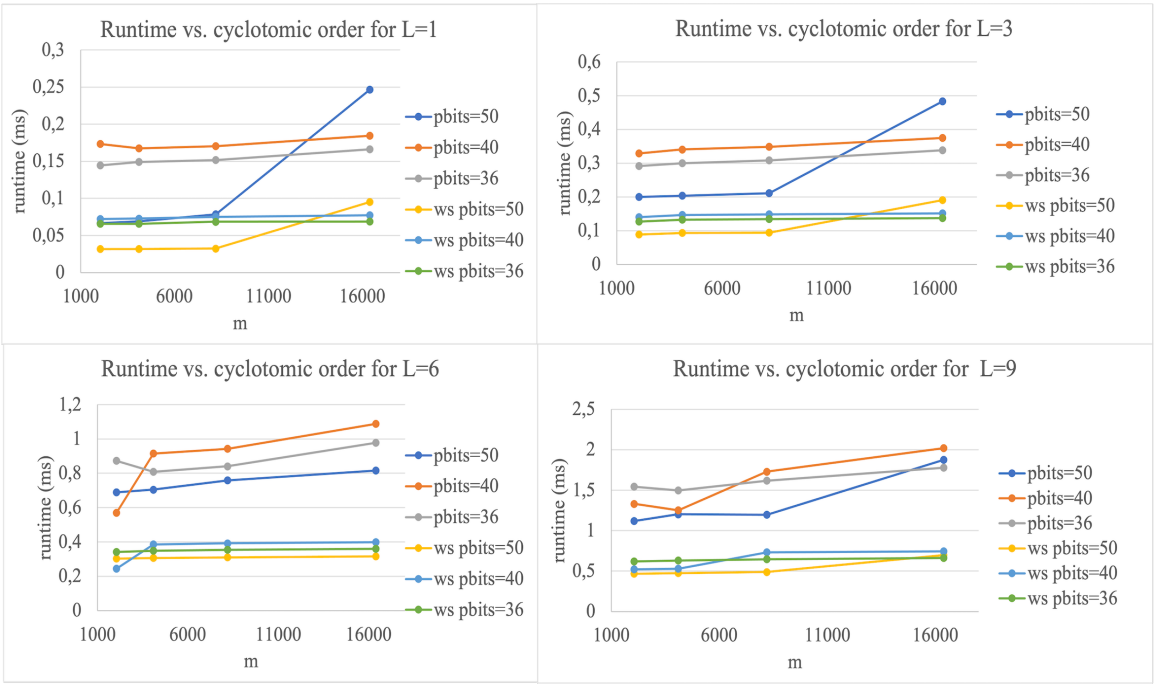


Figure 3.2. BFVrns runtime vs. cyclotomic order m for *pbits* and L on a laptop and a workstation (ws).

Why do we choose linear regression for the runtime? We prefer linear regression analysis for the runtime because when we increase m and L, the ciphertext size increases linearly. Then, the number of operations and runtime increase linearly. t is used mostly in the linear operations that affect the runtime almost linearly. Based on the experiments, σ has a negligible effect on the runtime. Thus, it is considered linear relation to the runtime. rw is the number of bits in the relinearization window. The number of operations in the ciphertext changes almost linear to rw. Also, as seen from the Figure 3.2, m, L and *pbits* increase linearly in runtime in a laptop (with two cores, 2.6 GHz CPU and 8 GB RAM) and a workstation (with 4 cores, 2.9 GHz CPU and 64 GB RAM) with larger CPU and memory. That is why we choose linear regression for the runtime. This rule also applies to other high performance environments. This means that our approach is also valid for other type of hardware. According to Table 3.3a, the runtime mainly depends on L, rw, m and *pbits* with statistical significance at 1 percent level. Also, *standard errors* of these parameters

are very small. *Standard error* represents the average distance that the observed values fall from the regression line. The smaller values of *standard error* show that observations are closer to the fitted line. We get the following estimated runtime equation as in Equation (3.2). The coefficients of L and m are positive and give increasing effect on the runtime. This means that if L and m increase, the runtime increases. The coefficients of rw and $pbits$ show decreasing effect. For this regression result, according to *p-value* and *standard error* measures, this regression is considered an efficient model despite *R-squared* is relatively small. However, in statistics if the first two measures are satisfied, *R-squared* can be ignored. $1 \leq rw \leq 30$, $4096 \leq m$, $30 \leq pbits < 60$ and $3 \leq \sigma \leq 4$ are the bounds of parameters to prevent any negative result from the equations.

$$\hat{rt}(BFVrns) = 0.0784162L - 0.1753079rw + 0.023195m - 0.0284464pbits. \quad (3.2)$$

Table 3.3. Regression Analyses For BFVrns

(a) Regression Results for BFVrns Runtime

Parameters	Runtime
t	0.0044932^{*}
	(0.0031289)
L	0.0784162^{***}
	(0.0031289)
σ	0.0057511^{*}
	(0.0030362)
rw	-0.1753079^{***}
	(0.0031171)
m	0.023195^{***}
	(0.0030446)
$pbits$	-0.0284464^{***}
	(0.0030939)
const	$4.08e-08^{*}$
	(0.0011661)
Obs.	4096
R-squared	0.4907

(b) Regression Results for BFVrns Noise

Parameters	Noise
t	3.97373^{***}
	(0.4491426)
L	36.08466^{***}
	(0.31536)
σ	5.041768^{***}
	(1.389387)
rw	0.2879736^{**}
	(0.1166101)
m	1.616277^{***}
	(0.19712)
$pbits$	8.807353^{***}
	(0.6541515)
const	-47.15998^{***}
	(5.55294)
Obs.	4096
R-squared	0.7891

Notes: Asterisks show *p-value* of coefficient as follows; *** $p < 0.01$, ** $p < 0.05$,* $p < 0.1$. const stands for constant coefficient of the regression. The values in the parenthesis show *standard error*. Obs. stands for number of observations.

Why do we choose log-log ***model* for the noise growth?** Firstly, we draw the graphs for the noise growth with normal scale that did not show any specific clear relation between parameters and the noise. Later, when we use log-log *model*, it shows a linear relation between log of the noise and log of the values of the parameters as seen in Figure 3.3. From Figure 3.3, it is seen that log of m, L and $pbits$ have linear relation to log of the noise growth. Since this rule also applies to rw and σ, we do not include their figures here to avoid redundancy. We formalize this model to understand how we can utilize this property of the noise growth behavior. By giving the following method, the analysis model becomes more clear. For instance, when we have a relation as Equation (3.3) with n different parameters and apply log to both sides as Equation (3.4) and rename the parameters, it becomes a linear model as in Equation (3.5). On this linear relation, we apply linear regression and find the A, c_i values and the relation. Since this transform is similar to the noise growth, we apply this idea on the noise growth. Thanks to this step, the noise relation turns to a linear relation. Then, we apply the linear regression analysis on the noise data, it gives the coefficients of the relation of the noise as in Equation (3.7) where we define $1 \leq rw \leq 30$, $4096 \leq m$, $30 \leq pbits < 60$ and $3 \leq \sigma \leq 4$ to prevent negative values that may occur. This equation says all parameters have increasing effects in log scale. That is why the model needs to keep them small as much as possible while keeping the security condition and correctness.

$$y = aX_0^{c_0} X_1^{c_1} X_2^{c_2} .. X_n^{c_n}. \tag{3.3}$$

$$\begin{aligned} \log_2 y &= \log_2(a x_0^{c_0} x_1^{c_1} x_2^{c_2} \dots x_{n-1}^{c_{n-1}}) \\ &= \log_2(a) + c_0 \log_2(x_0) + c_1 \log_2(x_1) + \dots c_{n-1} \log_2(x_{n-1}). \end{aligned} \tag{3.4}$$

$$Y = A + c_0 X_0 + c_1 X_1 + c_2 X_2 + \dots c_{n-1} X_{n-1}. \tag{3.5}$$

From the theoretical EvalMult noise bound of the BFVrns scheme given in Equation (3.6) from [61], we can see that our approach is verified as almost accurate. It shows that the noise has a log relation to the parameters. In the equation, δ stands for polynomial multiplication expansion factor. $l_{rw,Q}$ is the number of base-rw digits in ciphertext modulus Q, ϵ_2 is error rate. $\|s\|_\infty$ stands for canonical embedding norm of secret key s. k is the number of $logq_i$ in BFVrns. B_e is the effective (low-probability) upper bound for Gaussian errors.

$$\begin{aligned} \|c_{mult}\| &< C_1^L + LC_1^{L-1}C_2, \\ C_1 &= (1+\epsilon_2)\delta^2 t\|s\|_\infty, \\ \epsilon_2 &= 5(\delta\|s\|_\infty)^{-1}, \\ C_2 &= \delta^2\|s\|_\infty((1+2\|\epsilon_s\|_\infty)\|s\|_\infty + t^2) + \delta l_{rw,2^v} rwkB_e. \end{aligned} \tag{3.6}$$

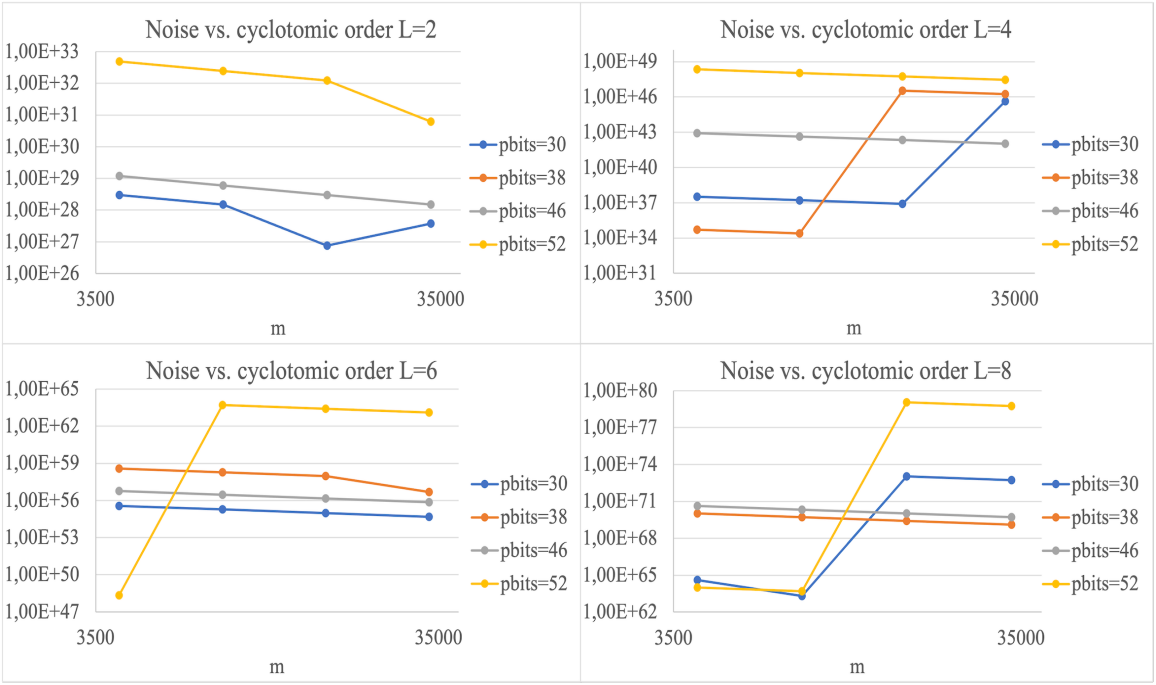


Figure 3.3. BFVrns noise growth vs. cyctomic order m in log-log *model*.

As a result of regression analysis in Table 3.3b, we provide the following estimated noise as Equation (3.7). This table provides coefficients of log of the

parameters. All parameters except rw, are statistically significant at 1 percent level that means this model can explain the data in terms of these parameters. Since *p-value* is the main measure of regression analysis, it is clear that this model fits the data well. In addition, the smaller *standard errors* support this results. Also, *R-squared* is close to 1 that means this model is a good fit. All statistical measures show that estimation Equation (3.7) can represent the noise growth in BFVrns. The noise model provides an estimation for the noise growth without compromising the correctness because our value bounds.

$$\hat{noi} = 3.9\log_2(t) + 36.1\log_2(L) + 5\log_2(\sigma) + 1.6\log_2(m) + 8.8\log_2(pbits). \quad (3.7)$$

3.4.2 Analysis of the Parameters for Encryption Noise

For a correct decryption result, it is essential to control noise amount after the encryption. We conduct regression analysis on the encryption noise to see how the parameters affect the noise growth. Firstly, we create a dataset for the noise in the ciphertext after one encryption with the given values in Section 3.4.1. Secondly, we run the regression analysis on this dataset similar to the noise analysis in Section 3.4.1. As seen in Table 3.4, we get the estimation result in Equation (3.8). $1 \leq rw \leq 30$, $4096 \leq m$, $30 \leq pbits < 60$ and $3 \leq \sigma \leq 4$ are the bounds of parameters to prevent any negative result from the equations. The same explanation applies to this regression analysis as in previous section since the parameters show the same behavior. Equation (3.8) should be less than encryption noise bound in Equation (3.12) to keep the control on the noise growth.

Table 3.4. Regression Results for Encryption Noise

Parameters	Noise
t	2.959277^{***}
	(0.4491375)
L	36.08471^{***}
	(0.3153531)
σ	5.041774^{***}
	(1.389376)
rw	0.2879741^{**}
	(0.1166073)
m	1.616275^{***}
	(0.1971081)
$pbits$	8.80732^{***}
	(0.654132)
const	-43.74002^{***}
	(5.552507)
Obs.	4096
R-squared	0.7887

Notes: Asterisks show *p-value* of coefficient as follows; *** $p < 0.01$, ** $p < 0.05$. const stands for constant coefficient of the regression. The values in the parenthesis show *standard error*.

$$\hat{EncNoi} = 2.95\log_2(t) + 36.08\log_2(L) + 5.041\log_2(\sigma) + 1.61\log_2(m) + 8.8\log_2(pbits). \tag{3.8}$$

3.4.3 Multi-objective Optimization

In the practical experiments, we observe that some parameters show opposite behaviors on the performance and noise growth. Finding an optimum set of parameters is a trade-off between the noise and performance. In this case, we need the concept of multi-objective optimization that involves more than one objective functions to be minimized. In this study, we have three different objective functions to be minimized: the runtime, noise growth and CEF. But minimizing noise does not mean that we compromise correctness or security. We provide at least 128-level security and give enough room for the noise to get correctness with overwhelming probability.

MOO helps us to find the best trade-off between objective functions. If dominance of a set of solutions cannot be found, we try to find the closest solution with Pareto optimality. The non-dominated set is called as Pareto front set and mapping of this set has a boundary which is called as Pareto-optimal front. Our aim is to find the closest possible solutions to Pareto-optimal front. With Pareto optimality approach, we try to find optimum set of parameters with the help of MATLAB functions [85]. MATLAB offers two different solvers to find Pareto optimal solution. We use `gamultobj` that uses a genetic algorithm to find Pareto front of multiple fitness functions. Genetic algorithm was introduced by Holland in the early 1970s [63]. Genetic algorithm is a probabilistic search algorithm that depends on the idea of natural selection. It represents the computational model of evolution theory by natural selection. Genetic algorithm has been applied on various kind of optimization problems such as graph coloring, traveling salesman problem, vehicle

routing problem and many more problems. Genetic algorithm is commonly used in complex multi-objective optimization problems in engineering to investigate a solution set that satisfies its objective functions and nonlinear conditions.

MOO solvers require objective functions, lower and upper bounds for each parameter and nonlinear inequality functions. For the given L and λ, we apply MOO operation with the following setup.

Objective Functions: We have three objective functions, one is the runtime, another one is the noise growth that come from regression analyses in the Section 3.4.1 as in Equations (3.9) and (3.10), respectively. The final one is CEF $2\log Q/\log t$. The smaller values of CEF imply less network cost in case of ciphertext transferring from one server to another one or any client to a server.

$$\hat{rt} = 0.07841L - 0.1753rw + 0.0231m - 0.02844pbits. \tag{3.9}$$

$$\hat{noi} = 3.9\log_2(t) + 36.1\log_2(L) + 5\log_2(\sigma) + 1.6\log_2(m) + 8.8\log_2(pbits). \tag{3.10}$$

Nonlinear Functions: We have two nonlinear equalities; the first one is the dimension equality from Equation (3.11) where λ is the security parameter. We substitute $lg(\delta) = lg(Q)/4N$ in Equation (3.1) as in [54] and create the constraint for the parameters. The second one is the encryption noise bound. The noise of the ciphertext is given as $B_{clean} = B_{err}(1 + 2\delta B_{key})/2$ in [61] where B_{err} is effective upper bound for Gaussian errors, δ is the polynomial multiplication expansion factor and B_{key} is bound for the secret key. $B_{err} = \sigma\sqrt{\alpha}$ where α is assurance measure and given as 36 and $\delta = 2\sqrt{\phi(m)}$ in PALISADE. We use secret key distribution mode as $RLWE$ which is Gaussian noise distribution in PALISADE. For $RLWE$ mode, B_{key} is given as $B_{key} = \sigma\sqrt{\alpha}$. We adjust the encryption noise bound for the given PALISADE construction as in Equation (3.12). Equation (3.8) should be smaller than Equation (3.12) as in Equation (3.13) while providing correctness by keeping the security level

> 128.

$$pbits - 2m\sqrt{0.01273/(\lambda + 19)} \leq 0 \tag{3.11}$$

$$\begin{aligned} B_{clean} &= \frac{B_{err}(1 + 2\delta B_{key})}{2} \\ &= 3\sigma(1 + 4\sqrt{\phi(m)}6\sigma) \\ &= 3\sigma(1 + 24\sqrt{m/2}\sigma). \end{aligned} \tag{3.12}$$

$$\hat{EncNoi} < B_{clean} \tag{3.13}$$

Boundaries: We use the following bounds that guarantee at least 128-bit security. $1 \leq rw \leq 30$, $4096 \leq m \leq 65536$, $30 \leq pbits < 60$ and $3 \leq \sigma \leq 4$, $20 \leq t \leq 50$.

3.5 Results

Experimental Setup. We run our performance experiments on a personal computer running MacOS Catalina on Intel Core i5 with two cores, 2.6 GHz CPU with 8 GB RAM and a workstation with Intel(R) Xeon(R) W-2102 with 4 cores 2.9 GHz CPU and 64 GB RAM. All experiments run single-threaded mode.

We run `gamultobj` solver for the objective functions, constraints and bounds given in Section 3.4.3. According to this MOO setting, we provide the following tables for different L and λ values. λ values are taken from the HE Standard to comply with the recent HE standardization.

As seen in Table 3.5, for larger L values, m and $\log Q$ increase. The optimization model sets rw to an average value to balance the runtime and noise growth because rw has a decreasing effect on the runtime and increasing effect on the noise growth. The optimization model offers $\sigma = 4$ for all conditions that is a common value used by many studies. For larger L values, $pbits$ decreases while t increases because the model balances the value of CEF and the noise at the same time.

The main benefit of regression analysis is estimating the result for the values other than the dataset values. This helps us to project the model for the larger values of L. We offer the recommended parameter set for $L = 5$, 10, 20 as in Table 3.5.

Table 3.5. Recommended Parameter Sets for the BFVrns Scheme at Least $\lambda = 128$

L	t	σ	rw	m	$pbits$	$\log Q$	LSB
5	41	4	15	16384	56	168	$2^{174.4}$
10	44	4	14	32768	51	306	$2^{192.9}$
20	48	4	14	65536	49	588	$2^{202.1}$

Note: LSB Stands for the Least Security Bit Result from Three Attacks on LWE Estimator.

Table 3.6. Recommended Parameter Sets for the BFVrns Scheme for Different λ and $L = 5$

Param. Set	λ	t	σ	rw	m	$pbits$	$\log Q$
Param Set 1	128	41	4	15	16384	56	168
Param Set 2	192	49	4	15	16384	52	156
Param Set 3	256	47	4	9	32768	57	168

3.5.1 Security

For the given security level λ, the generated output provides a higher security according to LWE Estimator. This result means that there is a security guarantee that the system always provides at least the given λ security, not smaller than this input λ. We provide 3 different parameter sets as in Table 3.6 for the users who prefer different security levels. We run LWE Estimator [8] to find the lowest security levels of the parameter sets with the uSVP, decoding and dual attacks. Table 3.7 demonstrates that the least value of the number of security bits for all three LWE attacks is very close or higher than our expected security level. For example, Parameter Set 1, we expect 128-bit security, but LWE Estimator shows this parameter set provides at least 174-bit security. This result also applies to Parameter set 2 and 3. These proposed

parameter sets provide improved BFVrns performance, CEF and noise growth without compromising security and correctness. Any set of parameters provides successful decryption thanks to the optimization model that keeps noise growth under the control. One may argue that there is a gap between expected security level and their LWE Estimator results. This gap does not imply any failure of the approach. On the contrary, it shows that this approach supports better security level with the given parameter sets. In addition, as hardware technology improves, higher security levels will be required in the future.

Table 3.7. SageMath LWE-Estimator Results from [8]

Param Set	**uSVP**	**dec**	**dual**
Param Set 1	$2^{174.4}$	$2^{178.8}$	$2^{207.9}$
Param Set 2	$2^{189.9}$	$2^{195.5}$	$2^{203.2}$
Param Set 3	$2^{390.0}$	$2^{403.4}$	$2^{414.0}$

3.5.2 Comparison with Other Suggested Parameters

Table 3.8 demonstrates the performance and CEF comparison between given parameters in [61] and this work. For the first comparison for $L = 5$, our parameter set demonstrates 20% better performance and $5x$ smaller CEF than [61] parameters. This means that our parameter set results in a lower communication cost while sending ciphertexts. For $L = 10$, our parameter set results in 17% more performance than [61] parameter set. CEF is 275 for [61] and 50.9 for our work. This comparison shows that our model balances these factors by keeping the same security level and correctness. In the last comparison with $rw = 0$ (DCRT decomposition case where ciphertext decompositon occurs on q [61]), our approach demonstrates 3.5% less performance than another work, but it provides $4.7x$ better CEF. In all cases, our approach yields better outcomes. There are very few research on parameter selection for FHE schemes,

particularly BFVrns. As a result, we could make a comparison to a single study in this area.

We also provide performance comparison on a workstation which has a higher computing power than the previous environment. The performance comparison shown in Table 3.9 on the workstation shows that our parameter set provides improved performance mostly with the same CEF as in Table 3.8. This shows that our approach can be applicable to other environments as well.

Table 3.8. EvalMult Performance and CEF Comparison with [61] Parameters

	L	m	σ	$pbits$	rw	t	Runtime	RT Impr. Ratio	CEF	CEF Impr. Ratio
[61]	5	8192	3.2	55	30	2	20.32	0.2	165	4.95
This Work	5	8192	3	56	20	33	**16.22**		**33.3**	
[61]	10	16384	3.2	55	30	2	93.96	0.17	275	5.4
This Work	10	16384	4	56	20	45	**78.3**		**50.9**	
[61]	20	32768	3.2	55	0	2	**211.4**	-0.035	495	4.68
This Work	20	32768	4	59	0	48	219.2		**105.6**	

Notes: Runtime Improvement Ratio means the performance improvement percantage of our work vs. [61]. CEF Improvement Ratio is the ratio of CEF of [61] / CEF of this work. (PALISADE determines k based on security constraints mentioned in [61]. (To get a fair result, we set $2 \leq t \leq 50$ for this comparison.)

Table 3.9. EvalMult Performance Comparison with [61] Parameters on the Workstation for the Given Parameters

	L	m	σ	$pbits$	rw	t	Runtime	Runtime Improv. Ratio
[61]	5	8192	3.2	55	30	2	5.91	0.03
This Work	5	8192	3	56	20	33	**5.74**	
[61]	10	16384	3.2	55	30	2	23.46	0.005
This Work	10	16384	4	56	20	45	**23.34**	
[61]	20	32768	3.2	55	0	2	**71.45**	-0.12
This Work	20	32768	4	59	0	48	81.41	

Notes: Runtime Improvement Ratio means the performance improvement percantage of our work vs. [61]. CEF values are the same as Table 3.8.

3.5.3 Trade-off Between Parameters

One may argue that why our approach does not always result in superior performance. When we compromise from performance, it ends up with a larger CEF which is undesirable. In the computational PIR schemes, CEF is an important factor to determine communication cost between client and server. When CEF increases, it means that amount of data increases and transmission of this data takes longer time or requires larger bandwidth. When the algorithm selects parameter sets for a system where transmission of the encrypted data is not necessary, it selects the parameter sets that results in better performance.

Even while our approach results in less performance in some cases, it is extremely close to another result, such as 3.5% less performance. When the algorithm picks a larger $pbits$, it increases CEF and reduces security. On the other hand, it increases noise room in the ciphertext, allowing more multiplication before bootstrapping. When t increases, k increases and indirectly $\log Q$ increases, resulting in an increase in runtime but a decrease in CEF. This trade-off might take several forms depending on the requirements of application, with certain factors being overlooked. We regard

performance and CEF to be critical aspects in our context, and they are also commonly used as crucial indicators in real-world applications.

When CEF is more important than performance, our algorithm chooses the parameter set with better CEF factors. Because it selects smaller *pbits* resulting less noise room and more bootstrapping call. In this case, smaller *pbits* results in higher security than the given level that means our algorithm still satisfies the security condition.

For other special cases, we can add specific objective functions and nonlinear constraints to get the other versions of this parameter selection approach. For example, for circuits having more than 30 times EvalAdd operations than EvalMult (Recall that EvalMult is ~ 30 times slower than EvalAdd.), we can replace EvalMult with EvalAdd regression analyses and noise bounds in the MOO process. In the specific instance when CEF is irrelevant, such as for local computations, we can exclude CEF's objective function from the model. It is a flexible model, which enables us to simply adapt it to the requirements of FHE applications and create new parameter selection approaches for them.

In summary, our statistical and optimizational approach to the parameter selection for the BFVrns scheme shows that parameter selection is possible with a reasonable security, correctness and strength against given LWE attacks. We can apply this methodology on other FHE schemes to reduce the user's responsibility. Because of the intricacy of parameter selection process, non-expert users avoid utilizing FHE in their applications. Non-professionals can now use the FHE schemes without burden of parameter decision.

3.6 Discussion

We discuss some important points in this section as follows.

Hardware constraints. FHE schemes require a large amount of memory and time for larger values of L. That is why we create the datasets with smaller values of L. With the help of the regression analysis, we can project the behavior of the parameters for larger values of L. For the users who may use larger L values, this study can ease their parameter selection decision process. Due to the hardware resource constraints, we select four different values for each parameter. We still have good numbers of observations like 4096 for the regression models.

Estimated equations. In this study, we aim to give approximate recommended parameter sets. As we know that regression method cannot give the perfect fit for dependent variable, but it gives very close approximation on the relations of the parameters. To present more accurate results, we use a statistical software, STATA that provides trusted results. Also, we justify the regression results with the three different statistical measures mentioned above.

Library specific analysis. We conduct our analyses on PALISADE and give the recommended parameters for PALISADE library specifically. PALISADE includes many FHE scheme implementations and it is one the popular FHE libraries. We believe that our approach can shed light on the parameter selection for other libraries, too.

3.7 Conclusions

In this study, we construct a novel approach for the parameter selection problem for BFVrns. The parameter selection problem is regarded a non-trivial task in lattice-based cryptography. Our study serves as a solution for BFVrns and an example for other FHE schemes as well. We propose a clear understanding for the parameters on CEF which is another problem in the FHE schemes. This study also provides an extensive analysis on the behaviors of the parameters in the performance. Based on this approach, we provide optimized parameter sets for different L and λ that

facilitate the usage of FHE. We also demonstrate the security of the approach and the proposed parameter sets to present secure and robust parameters against some well-known LWE attacks. In addition, we provide smaller ring dimension values with a new approach with the help of BKZ 2.0. In future work, we will implement a MOO solver in PALISADE to create a fully automated parameter selection tool. This will allow FHE schemes to be more accessible by practical applications without the trouble of parameter selection process.

CHAPTER 4

TOWARDS EFFICIENT FHE-BASED CPIR PROTOCOLS AND THEIR PARAMETER SELECTION

4.1 Introduction

When accessing a public database, it is possible to keep the searches private from any third party, but keeping the request hidden from the data operator is sometimes needed for sensitive applications. Private Information Retrieval (PIR) was proposed as a novel solution to this problem. PIR enables the confidentiality of queries in many applications, such as online patents, real-time stock data, location-based services, medical conditions, search engines, and many more use cases. Recent, Fully homomorphic encryption (FHE) was introduced to enable computation on encrypted data. The combination of these two approaches empowers FHE-based computational PIR (cPIR) protocols to solve this problem more effectively.

PIR was first proposed by Chor et al. in [33] in 1995. They constructed a set of schemes to build PIR with replicated databases which provide information-theoretic security. cPIR enables PIR protocols to run on a single database. Although Sion and Carbunar showed that cPIR is not practical [109], lattice-based cryptography changed this opinion about cPIR [3]. Kiayias et al. proposed a number-theoretic homomorphic encryption based cPIR sheme [66] which has a good communication performance, but it has a computational problem.

FHE refers to a family of encryption methods that allows evaluation of arbitrary computation on encrypted data. FHE was first proposed by Rivest et al. [100] in 1978. It was an open problem until Gentry proved that FHE is possible by using lattices in his work [50] in 2009. After Gentry's breakthrough, many variants of FHE were proposed in the literature [113, 119]. FHE offers privacy for data and efficiency on distributed computing on encrypted data and enables powerful privacy-preserving

applications. One of these important applications is the family of PIR protocols. FHE provides creating cPIR schemes with a single database in an efficient way. Lattice-based cryptography is known for its quantum resistance, so our cPIR schemes are quantum-resistant as well. We propose three cPIR protocols with BFVrns [19, 46, 61] that is a ring-learning with error (RLWE) based FHE scheme. We offer efficient cPIR implementations based on PALISADE[1] [97].

cPIR was a computationally expensive concept that was not efficient for use in real-life applications. After cPIR is considered practical with the help of RLWE, many studies proposed either communication or computationally efficient protocols. XPIR, SealPIR and OnionPIR are the most recent cPIR protocols that utilize FHE schemes such as BV [23], FV [46], FV and RGSW [31], respectively. Three performance metrics are used to evaluate cPIR schemes: request/query cost, response cost, and server computation cost. The term "query cost" refers to the quantity of data transferred from the client to the server in order to obtain the desired database entry. The cost of a response is defined as the quantity of data transmitted from the server to the client. The term "server computation cost" refers to the amount of computation performed by the server in order to open the request and prepare the response. XPIR presents a query cost with $\mathcal{O}(n)$ in the normal version and SealPIR has $\mathcal{O}(d\lceil \sqrt[d]{n}/N \rceil)$ complexity where n is the number of data indices in the database, d is the dimension in database and N is the ring dimension. SealPIR uses query expanding techniques and Probabilistic batch codes (PBC) to reduce communication and computation complexity. OnionPIR utilizes external product to enable slow noise growth. We mainly focus on improving request and response size at improved server computation cost with BFVrns and the packing method.

In our study, we propose three cPIR protocols using a more efficient FHE scheme, BFVrns, and *Packed Encoding* method. *Packed Encoding* method helps

[1]https://gitlab.com/palisade/palisade-release Retrieved on 5 May 2022

to reduce communication complexity by packing multiple plaintext into one. We decrease the computation cost with *Packed Encoding* which offers performance advantages over scalar operations [111]. We provide $\lceil n/N \rceil$, $\lceil \sqrt[d]{n}/N \rceil + (d-1)(\sqrt[d]{n}\lceil \sqrt[d]{n}/N \rceil)$ and $d\lceil \sqrt[d]{n}/N \rceil$ query cost in P-cPIR, Recursive P-cPIR version I and II, respectively. Another efficiency measure in cPIR protocols is computation cost. P-cPIR and P-cPIR version I introduce at least $\sim 2^{14}\times$ performance improvement over the recent cPIR protocols for the server side computation.

While FHE schemes provide efficiency in cPIR protocols, it comes at a price, now that we have two problems; managing noise growth and parameter selection. Every ciphertext contains some amount of noise, which increases with every computation on the ciphertext. Another one is the parameter selection problem which is considered difficult due to its interrelated parameters. In the case of wrong parameter selection, decryption may result in a wrong plaintext or allow attackers to break the encryption scheme due to lower security. Another possible consequence is a drop in performance for FHE procedures. In addition, each parameter has a different influence on correctness, performance, and noise growth that makes this process more complex. For example, larger values of ciphertext modulus q support more noise but provide lower security. Noise growth depends on plaintext modulus t, so t should be small. However, smaller values of t result in more plaintexts to represent data that causes a decrease in performance. A larger value of ring dimension N is needed for the security, but it results in a decrease in performance. Towards a solution to these problems, we present a parameter selection algorithm that controls noise growth, correctness of the operations, and security of the system. Since Recursive P-cPIR version II is an efficient PIR protocol, we present the version of this algorithm for Recursive P-cPIR version II. This algorithm is easily applicable to our all protocols as well.

Current cPIR protocols attempt to optimize query, response and computation cost. However, there is one more critical factor to consider that has been ignored in

previous studies. This factor is the security of the underlying scheme in the cPIR protocols. Their parameter selection has a significant impact on security as well as performance. Proposed LWE-based cPIR protocols use parameter sets that suggest less than 128-bit security. For example, OnionPIR [88] uses a parameter set that results in 111-bit security. Since parameter selection affects performance, parameter sets that provide $>$ 128-bit security level may result in worse performance. This means that proposed communication and computation performance results may change with higher security levels that are suggested in the homomorphic encryption standard. In cPIR protocols, another important factor is Ciphertext Expansion Factor (CEF) that is the ratio of the ciphertext modulus size and plaintext modulus size. CEF has a significant influence on the communication cost of the cPIR protocols. A small CEF reduces request and response costs. The parameter of the cPIR schemes should be chosen optimally so that communication(request and response) and computation cost should be balanced along with enough security of the protocol. That is why we integrate a parameter selection approach satisfying these conditions in the protocols. Our parameter selection method provides the parameters sets for different security levels that comply with the homomorphic encryption standard[6].

4.1.1 Contributions

In this study, we provide three different cPIR schemes based on BFVrns. P-cPIR represents a cPIR scheme based on BFVrns with *Packing encoding* with the best server computation performance. Recursive P-cPIR version I provides very close computational cost to P-cPIR with taking the database as a d-dimensional matrix. Recursive P-cPIR version II introduces query and response efficiency over our other two protocols and other suggested cPIR protocols. Our protocols enable computational and communication efficient cPIR and we do not need to have any redundant copies of the database. For a database with $n = 2^{20}$ entries, P-cPIR

computation improvement is $\sim 2^{18}\times$, $\sim 2^{17}\times$ over XPIR and SealPIR, SHECS-PIR, respectively. It reduces the computation cost by $\sim 2^{15}\times$ over OnionPIR. Recursive P-cPIR version I shows $\sim 2^{17}\times$ performance improvement over XPIR and SealPIR, $\sim 2^{14}\times$ over OnionPIR. P-cPIR version II provides the best query and response cost in recent state-of-the-art protocols. Depending on the requirements of the application, we propose different protocols that are efficient for specific cases. We also provide a generic parameter selection algorithm for our protocols and present results for Recursive P-cPIR version II with $d = 2$ to show that parameter selection can be feasible for FHE-based cPIR protocols. This approach is also applicable to all of our protocols. These contributions make it possible for cPIR to be utilized in real-world applications.

4.1.2 Related Works

Kushilevitz and Ostrovsky proposed the first cPIR with single database PIR which is based on quadratic residuosity assumption [68]. After that, they proposed another approach for single database PIR with one way trapdoor permutations [69]. Cachin et al. proposed the first single PIR with polylogarithmic communication under Φ-hiding number-theoretic assumption. After the introduction of lattice-based cryptography, cPIR became more practical. A lattice-based scheme was proposed by Aguilar-Melchor and Gaborit [3]. XPIR was proposed by Melchor et al. [1] which is based on Ring-LWE problem and uses the BV scheme from [23]. Also, [86, 117, 2, 67] present alternatives to cPIR schemes based on non-standard problems, but some of them were broken [35, 11, 16, 73]. Some related works like [9, 52] used other SHE primitives to improve efficiency in terms of communication or computation cost by using compressing techniques. Park and Tibouchi proposed a PIR scheme based on deeper circuits and GSW-style homomorphic encryption [92] with better response size than SealPIR. Mughees et al. present external product approach on BFV and RGSW

ciphertexts for OnionPIR and homomorphic evaluation of copy networks method for stateful OnionPIR [88]. They propose a better response size than SealPIR at a close computation cost.

4.2 Background

4.2.1 Packed Encoding

Packed Encoding is an implementation of an efficient encoding for packing multiple integers into a single plaintext polynomial (single ciphertext). It enables SIMD (Single Instruction, Multiple Data) operations on these integers as proposed by Smart and Vercauteren [111]. SIMD refers to a computing method that enables processing of multiple data with a single instruction. PALISADE supports addition, multiplication, and rotation capabilities for packed ciphertexts. For *Packed Encoding*, m has to divide $t-1$, where t is the plaintext modulus.

4.3 Proposed cPIR Protocols

We construct the cPIR protocols with the BFVrns scheme due to its efficient construction [61]. We also use *Packed Encoding* in the cPIR protocols that provides an efficient encoding for packing multiple integers into a single plaintext polynomial (single ciphertext). This approach provides efficiency and improvement in the computation performance.

4.3.1 P-cPIR Protocol

In this section, we introduce a cPIR scheme called as P-cPIR (due to *Packed Encoding*). P-cPIR uses a similar approach to the works in [115] and [1] but a more efficient version with *Packed Encoding*. We assume that the database has n elements $\{db_1, db_2 \dots db_n\}$ and the client aims to retrieve the record/records at the ith index of the database. The client creates an array where ith index is set to 1, others are set to 0. This array is divided into N-size groups where N is the ring dimension

($N = m/2$ where m is power of 2. The client operates *Packed Encoding* on them to pack each group into each plaintext. The client encrypts the packed plaintexts with his pk and sends them to the server. On the database side, database is also divided into N-size groups. The server operates *Packed Encoding* on every group. After that, the server can run EvalMult (evaluation of multiplication) on every group of the array and database with their corresponding order. The server runs EvalAdd (evaluation of addition) on these ciphertexts and obtains a final ciphertext and sends it to the client. The client decrypts this ciphertext with his private key sk and reveals the data at the index i. We refer the reader to Protocol 1 for more details of the operations. In this protocol, packing one array/vector into one plaintext improves the performance by decreasing the computational cost. It has an efficiency advantage over running the operations one by one on the each entry of the database.

Protocol 1 P-cPIR Protocol

Setup (Client): Create a crypto context and encryption keys (public key pk, secret key sk) for given λ security and publish the public key pk.

Query Generation by Client:

1. Create a n-size array where ith index is set to 1 and other indices are set to 0.
2. Divide the array into N-size groups, apply *Packed Encoding* to every group.
3. Encode each group of this query with *Packed Encoding* in PALISADE.
4. Encrypt each encoded element with client's public key pk and it returns the result as $(e_0, e_1 \dots e_{\lceil n/N \rceil - 1})$.
5. Send $(e_0, e_1 \dots e_{\lceil n/N \rceil - 1})$ to the database operator.

Reply Generation by Server:

1. Divide the data into N-size groups and encode each element of the database with *Packed Encoding* as $(p_0, p_1, \dots p_{\lceil n/N \rceil - 1})$.
2. For $i = 0$ to $i = \lceil n/N \rceil - 1$, Operate $c_i \leftarrow EvalMult(p_i, e_i)$ and operate $EvalAdd$ on all outputs c_i and get the final ciphertext as r.
3. Send r to the client.

Reply Extraction by Client:

1. Operate $Decrypt(r, sk)$ and get the record at ith index of the database.

Protocol 2 Recursive cPIR Protocol Version I for d

Setup (Client): Create a crypto context and encryption keys (public key pk, secret key sk) for given λ security and publish the public key pk. We assume the client aims to retrieve $[i_1][i_2]..[i_d]$th index of the database.

Query Generation by Client:

1. Create an array which has $\sqrt[d]{n}$ elements. $[i_1]$th index is set to 1, other indices are set to 0.

2. Divide this array into N-size groups and encode these groups column based with *Packed Encoding* in PALISADE.

3. Encrypt this encoded output with client's public key and return result as e_i where $i = 1$ to $i = \lceil \sqrt[d]{n}/N \rceil$.

4. Do the following process $d - 1$ times. Set $j = 2$.

   - Create a $\sqrt[d]{n} \times \sqrt[d]{n}$ matrix. Set $[i_j]$th column to ones, other columns are set to zeros.
   - Divide these each dimension into N-size groups and encode them column based with *Packed Encoding*.
   - Encrypt each of them with client's public key pk and return result as $(ed_{j_1}, ed_{j_2} \dots ed_{j_{\lceil \sqrt[d]{n}/N \rceil}})$.
   - Set $j = j + 1$. Go to the first operation of Step 4.

5. Send all e_i and ed_{j_y} (for all j and y values created in the previous steps) to the server.

Protocol 3 Recursive cPIR Protocol Version I for d. (Cont.)

Reply Generation by Server:

1. Transform the database into a d-dimensional matrix with each dimension having each dimension $\sqrt[d]{n}$ size. Divide each column of this matrix into N-size groups and encode them column based with *Packed Encoding* as $p_1, p_2, \ldots p_{(\lceil n/N \rceil)}$ for all d.

2. For each k, Operate $c_i \leftarrow EvalMult(p_i, e_k)$ where $i = 1$ to $i = d\lceil n/N \rceil$ and $k = 1$ to $k = \lceil \sqrt[d]{n}/N \rceil$.

3. The following steps repeats $d - 1$ times. Set $j = 2$.

   - For $i = 1$ to $i = \lceil n/N \rceil$, operate $c_i \leftarrow EvalMult(c_i, ed_{j_t})$ where $1 \leq t \leq \lceil \sqrt[d]{n}/N \rceil$ and operate $EvalAdd$ on all outputs set as c_i.
   - Merge these outputs in ascending index order as $d-1$-dimensional matrix. (In this step, dimension reduces by one. It goes until $d = 1$.) Set every column as c_i.
   - $j = j + 1$. Go to to the first operation of 3.

4. Send the final c_i ($i = 1$ means only one ciphertext) to the client.

Reply Extraction by Client:

1. Operate $Decrypt(c_i, sk)$ and gets the record at $[i_1][i_2]..[i_d]$th index of the database.

4.3.2 Recursive P-cPIR Version I

In this section, we present a recursive version of P-cPIR with dimension $d > 1$. Protocols 2 and 3 present the algorithm of Recursive P-cPIR version I in detail. In

this version, we demonstrate the data as a d-dimensional matrix with $\sqrt[d]{n}$ elements in each dimension. Recursive P-cPIR version I benefit from both *Packed Encoding* and recursion approach. We assume that the client aims to fetch the data at the $i_1 i_2 \dots i_d$th index of the database. We apply similar steps in P-cPIR but with a different approach. Firstly, the client creates an array for the first dimension and sets the i_1th index to one, others to zero. For other dimensions, we take the query as a two-dimensional matrix by setting i_kth ($2 \leq k \leq d$) column to ones, rest of the columns to zeros. All queries are packed into N-size plaintexts and encrypted with the client's public key pk, then sent to the server. The server side takes the database in d dimensions with a similar approach. It takes the client queries and runs EvalMult, EvalAdd operation on query and database elements in order of indices. The next step is merging these ciphertexts in the $d-1$-dimensional matrix. In each step, the dimension reduces by one and it continues these operations until it becomes one. When the dimension results in one, the final result ends up with a ciphertext. The server sends this final ciphertext (one ciphertext) to the client. The client decrypts this with his sk and reveals the data at the $i_1 i_2 \dots i_d$th index of the database. This approach reduces the computation cost in compare to the version II, but it suggests higher communication cost than version II. Since the queries are sent as two-dimensional matrices for the dimensions $2 \leq k \leq d$, it results in higher communication cost. The reason behind this issue is column-based packed ciphertexts cannot be taken as row-packed ciphertexts. Thus, we send queries as two-dimensional matrices that allows running EvalMult and EvalAdd column based without losing the actual data. This problem is solved in the version II with the transformation step, but it brings some amount of computational cost. Despite this extra cost, it provides very close computational cost to other state-of-the-art protocols such as SealPIR, SHECS-PIR. We show the detailed communication costs in Table 4.1 in Section 4.5.

Protocol 4 Recursive P-CPIR Version II for d.

Setup (Client): Create a crypto context and encryption keys (public key pk, secret key sk) for given λ security and publish the public key pk. (The database is transformed into a d-dimensional matrix for the recursion purpose.)

Query Generation by Client:

1. Create d arrays with size of $\sqrt[d]{n}$. Let $i_{k_1} i_{k_2} \dots i_{k_d}$ be the index of the data what the client requests from the server. For each array in d arrays, set the index i_{k_x} of xth array to 1 others 0 where $1 \leq x \leq d$.

2. Divide these d arrays into N-size groups and apply row-based *Packed Encoding* on every group.

3. Encrypt each encoded element with client's public key pk and it returns the result as $(e_1, e_2 \dots e_l)$ where $l = d\lceil \sqrt[d]{n}/N \rceil$.

4. Send $(e_1, e_2 \dots e_l)$ to the server.

Protocol 5 Recursive P-CPIR Version II for d. (Cont.)

Reply Generation by Server:

1. Transform the database into a d-dimensional matrix with each dimension having $\sqrt[d]{n}$ size.

2. Divide these arrays into N-size groups and apply row based *Packed Encoding* on every group and return the result as $(p_0, p_1, \ldots p_l)$ where $l = d\lceil \sqrt[d]{n}/N \rceil$.

3. Set $c_i = e_i$ for $1 \leq i \leq l$.

4. Starting from the last dimension d, for each t where $j = 1$ to $j = d\lceil \sqrt[d]{n}/N \rceil$ operate $c_j \leftarrow EvalMult(p_j, c_t)$ where $1 \leq t \leq d$.

5. Transform row-based packed encoded matrix to column-based encoded one if it is row packed. Transform column-based packed encoded matrix to row-based encoded one if it is column packed.

6. Operate *EvalAdd* on all columns of each d dimensions.

7. Merge these columns in ascending index order into $d-1$-dimensional matrix. Set every column as c_j for $1 \leq j \leq (d-1)\lceil \sqrt[d]{n}/N \rceil$.

8. Set $d = d - 1$. Go to Step 4 of reply generation until $d = 1$.

9. When $d = 1$, return the final result as c.

10. Send c (only one ciphertext) to the client.

Reply Extraction by Client:

1. Operate $Decrypt(c, sk)$ and get the record at $i_1 i_2 \ldots i_d$th index of the database.

4.3.3 Recursive P-CPIR Version II

This version of P-CPIR offers a different query generation step than version I. We assume that the client aims to retrieve the data at the $i_1 i_2 \dots i_d$th index of the database. In this version, the client creates arrays for each dimension with the size $\lceil \sqrt[d]{n}/N \rceil$ and sets the i_kth index to one, others to zero (for $1 \leq k \leq d$). It packs every array into N-size plaintext, encrypts and sends these $d\lceil \sqrt[d]{n}/N \rceil$ ciphertexts to the server. As the first step, the server packs every column of each dimension. It takes the first query and runs EvalMult on the first query and the each packed plaintext of the database. As the second step, the server transforms this results from column-packed to the row-packed and adds these ciphertext in each group of $\lceil \sqrt[d]{n}/N \rceil$th index of the database. Then, it merges these results in another $d-1$-dimensional matrix. The process continues from the first step with column-based packed version. In each dimension row-based and column-based packing is called alternately. This recursion process ends in dimension one and returns the final ciphertext (one ciphertext) to the client. The client reveals the data at the aimed index $[i_1 i_2 \dots i_d]$ by decrypting it with his sk. This transformation step creates computation cost for the server side, but this version provides a substantial improvement in communication cost over version I. The client sends evaluation-at-index keys to the server offline which are required for the transformation of the matrix from column packed to row packed or the other way around. We do not need to send all evaluation keys because transformation to row to column is the reverse version row to column. That is why it is enough to send $\sqrt[d]{n}x\sqrt[d]{n}$ evaluation keys to the server offline. *Packed Encoding* allows shifting of elements in the packed ciphertext. In the transformation step, we shift every element in every row with EvalAtIndex method as required. Then, we apply addition on these rows and get the final result as column packed ciphertexts. Column to row transformation follows the similar process but with the reverse order of evaluation keys. Protocol 4

and 5 present the detailed steps of the protocol. Recursive P-CPIR version II enables better communication cost than previous two protocols.

4.3.4 Security and Correctness

Security. Security of the proposed cPIR protocols mainly depends on the hardness of RLWE problem. This means that no attacker can distinguish queries or replies from a random query or reply, respectively. This means that no attacker or database operator can know which entry is queried from the database or which entry is sent to the client from the database. Also, security of the protocols are provided by the parameter selection algorithm that we present in Section 4.4.

Correctness. The proof of correctness follows idea of the generic PIR that is constructed on FHE. For P-cPIR protocol, the client creates an array with size of the database and sets the index (that is aimed to retrieve from the server) to 1, others are set to 0. He packs them into N-size groups and encrypts with his pk as c_i and sends to the server. The server operates packing on the database as p_i. Then, he operates $EvalMult(c_i, p_i)$ and $EvalAdd$ on these results. It means that the data at the aimed index is multiplied with by 1 and others with 0 and sum all encryption of zeros and the intended data. At the end of the protocol, the result is the encrypted version of the data. The server sends this final result to the client and the client decrypts it with his sk and reveals the intended data. Correctness comes from the homomorphism and parameter selection of the scheme. In the next section, we provide a parameter selection algorithm that checks the correctness of the decryption and security of the parameters. Other two recursive versions of P-cPIR follow a similar correctness proof. The only difference is that they have operations on d dimensions, but they still use homomorphism while reducing the dimensions to one. For Recursive P-cPIR version II, we use transformation of column-based packing to row base, or vice versa. This

step also uses homomorphism that keeps the data integrity at that index even after the transformation.

4.4 Parameter Selection for Recursive P-cPIR Version II

4.4.1 Supporting Analysis

We analyze the main parameters of RLWE schemes such as m (m-th cyclotomic polynomial), σ (standard deviation of error distribution), $\log q$ (size of joint ciphertext modulus), t (plaintext modulus) and L (multiplicative depth of circuit) in the parameter selection algorithm.

BFVrns EvalMult runtime regression analyses. Due to the complexity of EvalMult operation, it takes the longest time in all homomorphic operations. That is why we consider EvalMult as the main operation to minimize its running time. We conduct a regression analysis to create a model for BFVrns EvalMult runtime and noise growth from the parameters. We create BFVrns runtime and noise datasets with t, m, L, σ, $pbits$ (ciphertext modulus ($\log q_i$ in the BFVrns scheme) in PALISADE that later creates $\log q$ as a final ciphertext modulus) and rw. BFVrns runtime mainly depends on L, rw, m and $pbits$ with statistical significance at 1 percent level according to regression analysis results conducted in STATA software [112]. We control the noise with the noise limit which is given in the BFVrns paper [61]. We use Equation (4.1) result in the parameter selection algorithm to get the parameters that provide the best performance.

$$\hat{rt}(BFVrns) = 0.0784L - 0.1753rw + 0.0231m - 0.0284pbits. \tag{4.1}$$

4.4.2 Parameter Selection Algorithm

In this section, we construct a parameter selection algorithm for the Recursive P-cPIR version II for $d = 2$ because d is a parameter typically chosen as 2 or 3. This algorithm is easily adjustable to our other protocols. It only requires the protocol change that

is called as a sub procedure in the algorithm. We also do not consider L due to the fixed number of level in cPIR. We give the ranges of the parameters such as t: 32769, 65537, 131073, 262145, 524289, 1048577; σ: 3, 3.2, 3.6, 4; m: 8192, 16384, 32768, 65536; *pbits*: 33, 38, 46, 52 to give as the starting point for the algorithm. *pbits* is the number of bits in the small ciphertext modulus that creates the joint ciphertext modulus $\log q$. The parameter selection algorithm utilizes Equation (4.1) to minimize the runtime. We also use the BFVrns noise constraint from [61] in Equation (4.2). In the equation, δ stands for polynomial multiplication expansion factor. $l_{rw,q}$ is the number of base rw digits in ciphertext modulus q, ϵ_2 is error rate. k is the number of q_i in the BFVrns setting. B_e is the effective (low-probability) upper bound for Gaussian errors. The parameter selection algorithm returns the selected parameters for the crypto context of the proposed cPIR protocol.

$$\begin{aligned} \|c_{mult}\| &< C_1^L + LC_1^{L-1}C_2, \\ C_1 &= (1+\epsilon_2)\delta^2 t\|s\|_\infty, \\ \epsilon_2 &= 5(\delta\|s\|_\infty)^{-1}, \\ C_2 &= \delta^2\|s\|_\infty((1+2\|\epsilon_s\|_\infty)\|s\|_\infty + t^2) + \delta l_{rw,2^v} wkB_E. \end{aligned} \tag{4.2}$$

The parameter selection algorithm starts with assigning the best values for minimizing the runtime. Then, it checks security of the parameters and updates values of the parameters for the given security. Later, it calculates the noise in the ciphertext and the noise constraint for the given parameters values. If the noise in the ciphertext is smaller than the noise limit, it returns the parameter set. Otherwise, it revises the parameter values for decreasing the noise in the ciphertext by updating the values of parameters as given in the algorithms. Then, it checks the security of these parameters. This step repeats until the noise is smaller than the noise limit. After this step ends, the algorithm returns the parameters. *prms* and *values* arrays represent the parameters in order of t, σ, rw, m, *pbits*. The client side run the

parameter selection algorithm and sends the values to the server so that the server can generate the necessary crypto context.

Algorithm 1 The upper limit for the noise in the ciphertext after EvalMult operation from PALISADE implementation [97].

1: **procedure** EvalMultNoiseLimit($params[](p, \sigma, rw, m, \log q),\ L,\ \lambda\ )$

2: $\delta(N) \leftarrow 2 * \sqrt{N}$ ▷ $N = \phi(m)$

3: $Bkey \leftarrow 6 * params[1]$

4: $eps1(N) \leftarrow 5/(\delta(N) * Bkey)$

5: $Vnorm(N) \leftarrow Bkey * (1 + 2 * \delta(N) * Bkey)$

6: $C1(N) \leftarrow (1 + eps1(N)) * \delta(N)^2$

$*params[0] * Bkey$

7: $C2_t(N, logqPrev) \leftarrow (\lfloor (logqPrev/(\log(2) * pbits)) \rfloor + 1)$

$*params[2] * Bkey$

8: $C2(N, logqPrev) \leftarrow \delta(N)^2 * Bkey$

$*(1.5 * Bkey + params[0]^2) + \delta(N) *$

$C2_t(N, logqPrev)$

9: $EvalMultNoise(N, logqPrev) \leftarrow 4 * params[0] * pow(C1(N), L)$

$*(C1(N) * Vnorm(N)$

$+L * C2(N, logqPrev))$

10: $NoiseLimit \leftarrow EvalMultNoise(N, params[4])$

11: **return** NoiseLimit

12: **end procedure**

Algorithm 2 Minimizes the runtime of EvalMult.

procedure MINIMIZERUNTIME(L, λ)

  $rt_coefs \leftarrow 0, 0.0784, -0.1753, 0.0231, -0.0284$ ▷ Refer Eq 4.1.

  for $i \leftarrow 0, i < rt_coefs.size$ **do**

    if $rt_coefs[i] \geq 0$ **then**

      $values[i] \leftarrow min_element(var_range[i])$ ▷ var_range[][] is a matrix for the values of the parameters as mentioned in 4.4.2.

    else

      $values[i] \leftarrow max_element(var_range[i])$

    end if

  end for

  $\delta \leftarrow 2 * \sqrt{N}$, $\sigma \leftarrow values[1]$, $Bkey \leftarrow 6 * \sigma$

  $p \leftarrow values[0]$, $rw \leftarrow values[2]$, $pbits \leftarrow values[4]$

  $logqPrev \leftarrow 6 * \log(10)$ ▷ Refer PALISADE implementation.

  $nRLWE \leftarrow n$ from HE Standard ▷ Refer here [6]

  $eps1(N) \leftarrow 5/(\delta(N) * Bkey)$

  $Vnorm(N) \leftarrow Bkey * (1 + 2 * \delta(N) * Bkey)$

  $C1(N) \leftarrow (1 + eps1(N)) * \delta(N)^2 * params[0] * Bkey$

  $C2_t(N, logqPrev) \leftarrow (\lfloor (logqPrev/(\log(2) * pbits)) \rfloor + 1)$
  $* params[2] * Bkey$

end procedure

Algorithm 3 Minimizes the runtime of EvalMult. (Cont.)

procedure MINIMIZERUNTIME(L, λ)(Cont.)

$C2(N, logqPrev) \leftarrow \delta(N)^2 * Bkey$

$*(1.5 * Bkey + params[0]^2) + \delta(N) *$

$C2_t(N, logqPrev)$

$logqBFV \leftarrow \log(4 * p) + (L - 1)$

$* \log(C1(N)) + \log(C1(N) * Vnorm(N) + L *$

$C2(N, logqPrev))$

$logq \leftarrow logqBFV(N, logqPrev)$

while $nRLWE(logq) > n$ **do** ▷ Check if n satisfies RLWE security condition

$n = 2 * n$, $logq = logqBFV(N, logqPrev)$, $logqPrev = logq$

end while

$values[3] \leftarrow 2 * N$, $values[4] \leftarrow logqPrev$ ▷ Update new values

return values[]

end procedure

Algorithm 4 Revises parameters when noise in the ciphertext larger than noise limit.

procedure REVISEPARAMS($params[](p, \sigma, rw, m, \log q)$, L, λ)

Increase param[0] ▷ Increase t, this decreases CEF.

Increase param[1] ▷ Increase σ

if CheckSecurity($params[]$, L, λ **then**

Return $params[]$

else

ReviseParams($params[]$, L, λ)

end if

end procedure

Algorithm 5 Selects parameters based on other functions as the final step.

procedure ParameterSelect(L, λ)
    $params[], EvalMultNoise \leftarrow EvalMultNoise(\ L,\ \lambda);$
    $EvalMultNoiseLimit \leftarrow EvalMultNoiseLimit(params[],\ L,\ \lambda);$
    while Decryption is wrong and ($EvalMultNoise > EvalMultNoiseLimit$) **do**
        $params[] \leftarrow ReviseParams(params[],\ L,\ \lambda);$
        $params[], EvalMultNoise \leftarrow EvalMultNoise(\ L,\ \lambda);$
        if Decryption is correct and (EvalMultNoise ¡ EvalMultNoiseLimit) **then**
            return $params[]$;
        else
            $params[] \leftarrow CheckSecurity(params[],\ L,\ \lambda);$
            $params[], EvalMultNoise \leftarrow EvalMultNoise(L,\ \lambda);$
        end if
    end while
    while Decryption is correct and ($EvalMultNoise < EvalMultNoiseLimit$) and $params[4] < 59$ **do** ▷ Find the max ciphertext modulus.
        $params[4] \leftarrow params[4] + 1;$
        $params[], EvalMultNoise \leftarrow EvalMultNoise(\ L,\ \lambda);$
    end while
    while Decryption is correct and ($EvalMultNoise < EvalMultNoiseLimit$) and $params[4] < 59$ **do** ▷ Find the ciphertext modulus for optimum security.
        $params[4] \leftarrow params[4] - 1;$
        $params[], EvalMultNoise \leftarrow EvalMultNoise(\ L,\ \lambda);$
    end while
    return params[]
end procedure

Algorithm 6 Checks the security of the parameters.

procedure CHECKSECURITY($params[](p, \sigma, rw, m, \log q),\ L,\ \lambda$)

  $sec_cond \leftarrow params[4] - 2 * params[3] * \sqrt{0.01273/(\lambda + 19)}$ ▷ Refer to [121].

  if $sec_cond > \lambda$ **then**

    return params[]

  else

    while $sec_cond < \lambda$ **do**

      $params[3] = params[3] * 2$ ▷ Increase m

      $sec_cond = params[4] - 2 * params[3] * sqrt(0.01273/(\lambda + 19));$

    end while

  end if

  return params[]

end procedure

Algorithm 7 Simulates calculating the noise amount in a ciphertext of database reply in cPIR.

procedure EVALMULTNOISE($L,\ \lambda$)

  $params[] \leftarrow MinimizeRuntime(L, \lambda)$

  $params[] \leftarrow CheckSecurity(params, L, \lambda)$

  Create crypto context from $params[]$

  Run cPIR protocol. ▷ For parameter selection for other protocols, you can call them here.

  Measure the EvalMult noise as EvalMultNoise in the ciphertext of cPIR database reply. ▷ Refer Norm method in PALISADE

  return EvalMultNoise and params[]

end procedure

4.5 Results

4.5.1 Performance Results

We analyze the communication and computational costs of our protocols, as well as the trade-offs among them. Additionally, we provide communication and computational comparisons to recently proposed state-of-the-art cPIR protocols.

Table 4.1. Communication Cost Comparison of Proposed cPIR Protocols

	Query	**Response**
XPIR	$d\sqrt[d]{n}$	F^{d-1}
Chang et al.[26]	$\sqrt{n}\log n$	$\log n$
Lipmaa [76]	$\log^2 n$	$\log n$
SealPIR	$d\lceil\sqrt[d]{n}/N\rceil$	F^{d-1}
SHECS-PIR	$l\lceil logn/N\rceil$	1
OnionPIR	$(2N_1 + 4l\lceil log_4(n/N_1)\rceil)/N$	1
P-cPIR	$\lceil n/N\rceil$	1
Recursive P-cPIR Version I	$\lceil\sqrt[d]{n}/N\rceil + (d-1)(\sqrt[d]{n}\lceil\sqrt[d]{n}/N\rceil)$	1
Recursive P-cPIR Version II	$d\lceil\sqrt[d]{n}/N\rceil$	1

Notes: The communication cost (query and response size) is given in the number of ciphertexts. (N_1 is the size of the array in the first step of the query generation of OnionPIR. F stands for CEF. ℓ is the gadget vector length in SHECS-PIR [92] and OnionPIR [88].)

Communication performance: Table 4.1 suggests the communication costs for well-known protocols and our three protocols. This comparison suggests that our cPIR protocols offer better query and response cost than XPIR, the studies by Chang et al. [26] and Lipmaa [76]. Recursive P-cPIR version II proposes better response cost than SealPIR. Recursive P-cPIR version II performs very close query cost to SHECS-PIR and OnionPIR, but it shows almost half response cost of SHECS-PIR for $l = 5$

(which is generally set as 5 in other studies). These final four protocols have relatively similar query and response costs. Table 4.2 demonstrates three different database sizes showing that Recursive P-cPIR version II shows the best communication cost in the given protocols in terms of query and response size. Table 4.3 covers all our three protocols along with the recent protocols such as SealPIR, SHECS-PIR and OnionPIR. OnionPIR shows the best query cost, but when we do modulus switching from $q_i = 59$ to $q_i = 30$, our query size decreases by half that leads to 1 for query size and half for response size for Recursive P-cPIR Version II which is better than OnionPIR as the total communication cost.

Table 4.2. Communication Cost Comparison of the Protocols

	Rec. P-cPIR v.I			**Rec. P-cPIR v.II**			**SealPIR**			**SHECS-PIR**		
DB. size n	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}
Query [# ctxt]	257	513	1025	2	2	2	2	2	2	5	5	5
Response [# ctxt]	1	1	1	1	1	1	8	8	8	1	1	1

Notes: Query and response size is expressed in the number of ciphertexts [# ctxt].
$N = 2^{13}$, $d = 2$, $\log q = 60$, $\log t = 15$, thus $F = 8$. $\ell = 5$ is a parameter in SHECS-PIR [92].

Computation performance: The implementations of the present cPIR protocols (except SealPIR) are not publicly available for performance comparison. That is why we compare all protocols according to their total number of polynomial multiplications on the server side. The comparison in n, l, N, and k is shown in Table 4.4. We present Table 4.5 to give the approximate number of polynomial multiplications for server side computation of all protocols. This result indicates that

Table 4.3. Comparison of Communication Cost Results in Number of Ciphertexts [# ctxt]

	P-cPIR	**R. P-cPIR v.I**	**R. P-cPIR v.II**	**XPIR**	**SealPIR**	**SHECS-PIR**	**OnionPIR**
Query [# ctxt]	128	1025	2	2^{11}	2	5	1
Response [# ctxt]	1	1	1	8	8	1	1

Notes: This experiment is for $n = 2^{20}$, $d = 2$, $N = 2^{13}$, $\log q = 60$, $\log t = 15$, thus $F = 8$. $\ell = 5$ is a parameter in SHECS-PIR [92].

P-cPIR and Recursive P-cPIR version I outperform other protocols. Since they use *Packed Encoding* to pack many ciphertexts into a single one, server computation cost is reduced. The BFVrns scheme makes use of the Double Chinese Representation (DCRT) mode, which helps reduction in computation required for multiplications. Additionally, during the relinearization step, we decompose ciphertexts into CRT components, which leads in a faster relinearization process. When we employ other strategies such as probabilistic batch codes (PBCs) or copy networks, the calculation cost increases. That is why we provide a compact protocol structure with an efficient scheme and packing encoding method. For $n = 2^{20}$, P-cPIR reduces computation $\sim 2^{13}\times$, $\sim 2^{12}\times$ over XPIR and SealPIR, SHECS-PIR, respectively. It increases computation performance $\sim 2^{10}\times$ over OnionPIR. Recursive P-cPIR version I incurs $\sim 2^{20}\times$ performance improvement over XPIR and SealPIR, $\sim 2^{9}\times$ over OnionPIR. For $n = 2^{40}$, P-cPIR and Recursive P-cPIR version I present $\sim 2^{10}\times$ computation improvement in compare to OnionPIR. It incurs $\sim 2^{13}\times$, $\sim 2^{12}\times$ faster computation than SealPIR and SHECS-PIR, respectively.

Table 4.4. Comparison of Computation Costs of All cPIR Protocols in Terms of Number of Polynomial Multiplication

	Server Computation
XPIR	$2(n + F\sqrt{n})(4 + 2k)$
SealPIR	$2(n + F\sqrt{n})(4 + 2k)$
SHECS-PIR	$2\ell(n - 1)$
OnionPIR	$2n + 4\ell(n/N_1 + n/4N_1 + n/16N_1 + \ldots)$
P-cPIR	$\lceil n/N \rceil (4 + 2k)$
Recursive P-cPIR v.I	$2(\lceil n/N \rceil) + (\lceil n/N \rceil + (d-2)(\sqrt[d]{n}\lceil \sqrt[d]{n}/N \rceil))(4 + 2k)$
Recursive P-cPIR v.II	$\sum_{i=1}^{d}(2\lceil \sqrt[d-i+1]{n}/N \rceil + \sqrt[d-i+1]{n}\lceil \sqrt[d]{n}/N \rceil(4 + 2k)) + \lceil \sqrt[d]{n}/N \rceil(4 + 2k)$

Notes: ℓ is the gadget vector length in SHECS-PIR [92] and [88]. k is decomposition factor of the BV and BFVrns schemes.

Table 4.5. Comparison of Server Computation Costs of All cPIR Protocols in Terms of Number of Polynomial Multiplication

	Server Computation	
Database size	$n = 2^{20}$	$n = 2^{40}$
XPIR	$2^{24} + 2^{17}$	$2^{44} + 2^{27}$
SealPIR	$2^{24} + 2^{17}$	$2^{44} + 2^{27}$
SHECS-PIR	$10(2^{20} - 1)$	$10(2^{40} - 1)$
OnionPIR	$\sim 2^{21} + 2^{17} + 2^{15} + 2^{13} + 2^{11} + 2^9 + 2^7 + 2^5$	$\sim 2^{41} + 2^{37} + 2^{35} + 2^{33} + 2^{31} + 2^{29} + 2^{27} + 2^{25} + 2^{23} \ldots$
P-cPIR	2^{11}	2^{31}
Recursive P-cPIR v.I	$2^{11} + 2^9$	$2^{31} + 2^{29}$
Recursive P-cPIR v.II	$2^{23} + 2^{11} + 2^9 + 2^3 + 2$	$2^{51} + 2^{29} + 2^{12} + 2^9$

Note: This experiment is for $N = 2^{12}$, $F = 8$, $N_1 = 2^7$, $d = 2$, $k = 2$ and $\ell = 5$ (from [88, 92]).

Advantages of the BFVrns Scheme: BFVrns proposes efficient methods for Chinese Remainder Theorem (CRT) basis extension and scaling in RNS using floating-point arithmetic for some intermediate computations. It suggests a lower computational complexity along with a lower noise growth. Without bootstrapping, it enables cPIR protocols to have less latencies during computation. For instance, when $n = 4096$ and $\log q_i = 55$ are used, multiplication of two ciphertexts takes 5.03 ms for BFVrns and 198.58 ms for BFV. Additionally, BFVrns has the benefit of decomposing the ciphertext into DCRT bits during the relinearization step while

BFV decomposes ciphertexts into relinearization window rw which leads to slower operations. Due to bit length limit, PALISADE offers $1 \leq rw \leq 30$. When rw is selected as a large number, such as 30, the noise in the ciphertext increases. If it is set to a small value, such as 1, it results in higher number of polynomial multiplication operations increasing computation cost. That is why, for best runtime and noise growth, we should choose a number between 1 and 30 optimally. Due to this trade-off, in the experiment showed in Table 4.6, we set $rw = 15$. As seen in from this table, BFVrns is $\sim 40\times$ faster than BFV per polynomial operation. XPIR, SeaLPIR, and OnionPIR all utilize BFV, which results in a slower multiplication in these schemes. That is why we build our cPIR schemes on BFVrns, which introduce a stronger multiplication efficiency. We refer to Table 2 in Section 5.2 of [61] for more information on the complexity of multiplication operation. It demonstrates the efficacy of the BFVrns scheme in comparison to another RNS variant of BFV proposed by Bajard et al. [13]. When this improvement is considered, total server computation of our protocols decreases by 40 times than the normal comparison in Table 4.5 that results in improving P-cPIR computation cost $\sim 2^{18}\times$, $\sim 2^{17}\times$ over XPIR and SealPIR, SHECS-PIR, respectively. It reduces computation cost $\sim 2^{15}\times$ over OnionPIR. Recursive P-cPIR version I shows $\sim 2^{17}\times$ performance improvement over XPIR and SealPIR, $\sim 2^{14}\times$ over OnionPIR. According to this improvement, for $n = 2^{40}$, P-cPIR and Recursive P-cPIR version I suggest $\sim 2^{15}\times$ computation improvement over OnionPIR. It also presents $\sim 2^{18}\times$, $\sim 2^{17}\times$ faster computation than SealPIR and SHECS-PIR, respectively.

Communication-Computation Trade-off: Due to network constraints, several applications prioritize communication over computation. In this scenario, Recursive P-cPIR version II has one of the best communication performance in all protocols. When computational performance is a primary concern, P-cPIR and Recursive P-cPIR version I outperform all other protocols. Recursive P-cPIR version

Table 4.6. Running Time per Polynomial Multiplication of BFVrns and BFV (Time Unit is ms)

	BFVrns	**BFV**
Time per poly.	0.62	24.82
Schemes	P-cPIR, Rec. P-cPIR v.I and v.II	XPIR, SealPIR, OnionPIR

Note: This experiment is done with $N = 4096$ and $\log q = 55$, $t = 25$, $rw = 15$, $\sigma = 3$ on a Macbook Pro with 2.6 GHz processor and 8 GB RAM.

I and P-cPIR outperform OnionPIR, the most current protocol. Our work reveals that our three cPIR protocols can be used in a variety of ways depending on its intended use case.

4.5.2 Parameter Selection Results

Based on the parameter selection algorithm we present in Section 4.4.2, Table 4.7 shows the recommended parameters for different security levels λ for Recursive P-cPIR version II for $d = 2$. These parameter sets ensure correctness, security and best CEF and performance at the same time. We present three different security levels that comply with HE Standard [6]. For larger λ, the parameter selection algorithm suggests larger m values. When the plaintext modulus t is increased, CEF and noise room of the ciphertext decrease. With a smaller t, the CEF and noise room in the ciphertext are increased. On the other hand, a larger ciphertext modulus q incurs larger CEF and noise room, as well as improved performance. That is why the parameter selection algorithm determines the optimal parameters for the security levels. One may argue that we offer larger m values. The algorithm assesses noise growth and security conditions and then selects the final parameters based on all other parameters. Additionally, the algorithm analyses and re-evaluates the parameters in light of the several EvalMult operations performed in our cPIR protocols.

SealPIR and XPIR have response size as F^{d-1} where F stands for CEF. When we decrease CEF while keeping the performance and security optimal, it helps the

schemes having lower response size and computational cost for the server. Lower CEF values reduce the network cost and hardware consumption that leads more practical applications of cPIR protocols.

Table 4.7. Recommended Parameter Sets for Recursive P-cPIR Version II $d = 2$

	λ	m	$\log q_i$	σ	t	**CEF**
Param Set 1	128	16384	54	3.2	65537	6.75
Param Set 2	192	16384	55	3.2	65537	6.87
Param Set 3	256	32769	55	3.2	65537	6.87

Security: Table 4.8 presents the security of the parameter sets with LWE Estimator [8] in terms of LWE attacks such as usvp, decoding, and dual attacks. It implies that the specified parameter sets provide security levels equal or greater than the given λ that leads a security guarantee for the result parameter set. One may argue that there is a gap between the given λ and LWE Estimator results. This gap does not imply any insecurity or inefficiency of our algorithm. With improvement in technology, even 128-bit security may not provide enough security in the future. In this case, our algorithm still presents a enough security. Our proposed parameter settings ensure protocol security and improved runtime and lower CEF at the same time.

Table 4.8. Security of the Recommended Parameter Sets Given in Table 4.7

	usvp	**dec**	**dual**
Param Set 1	$2^{290.4}$	$2^{304.3}$	$2^{315.7}$
Param Set 2	$2^{284.2}$	$2^{297.3}$	$2^{308.4}$
Param Set 3	$2^{640.6}$	$2^{674.8}$	$2^{692.8}$

4.6 Conclusion

FHE brings practicality and security to cPIR protocols. While FHE is a major helper tool, it brings some problems like parameter selection. In this study, we eliminate this problem by proposing a parameter selection algorithm that optimizes the runtime and noise growth of the cPIR scheme. We present computation and communication efficient cPIR schemes and the parameter selection algorithm that is applicable to our cPIR schemes. We also provide the optimized set of parameters for the Recursive P-cPIR version II scheme for $d = 2$. This study allows anyone to use cPIR schemes even without advanced cryptography knowledge. We have showed that recent FHE can be used as a building block to implement an efficient cPIR protocol that is applicable to real-life applications.

CHAPTER 5

PREFHE, PREFHE-AES AND PREFHE-SGX: SECURE MULTIPARTY COMPUTATION FROM FULLY HOMOMORPHIC ENCRYPTION AND PROXY REENCRYPTION WITH AES AND SGX

5.1 Introduction

With the improvement of technology, new approaches have been proposed such as cloud storage/computing or distributed computing. While these services offer ease and practicality in real life, it comes at a price of privacy. When we store our data in the cloud, the cloud or other cloud users may snoop our confidential data. To protect this sensitive information, storing the data in encrypted way is the first step to solve this problem. This idea leads to another problem of making computations on this data. Homomorphic encryption (HE) enables computation of some functions on the encrypted data while FHE allows any kind of computation. Secure MPC takes this approach to another level with computing on the encrypted data collaboratively. It enables n parties computing a function F on n encrypted inputs, such that each party learns the result of the function but not any others' input.

Secure MPC was first introduced by Yao in 1986 [122] for two-party case and multiparty case by Goldreich, Micali and Wigderson [58]. For constructing secure MPC, two different approaches have been proposed in general. The first method is based on secure secret sharing where there is an honest majority among the parties [14, 27]. The second approach uses binary circuit representation of the function [122, 58]. Secure MPC has progressed in terms of efficiency with the recent studies. Based on HE, many MPC studies were proposed such as [39, 42, 15, 43, 95]. After the introduction of FHE, many MPC studies were proposed based on FHE such as [81] and FHE variants such as threshold FHE [12] and multikey FHE [82].

FHE refers to a family of encryption methods that allows evaluation of arbitrary computation on encrypted data. FHE was first proposed by Rivest et al. [100] in 1978. It was an open problem until Gentry proved that FHE is possible by using lattices in his work [50] in 2009. After Gentry's breakthrough, many variants of FHE were proposed in the literature [113, 119]. FHE offers privacy for data and efficiency on distributed computing on encrypted data and enables powerful privacy-preserving applications. One of these important applications is the family of secure MPC protocols. FHE allows constructing efficient secure MPC schemes. Lattice-based cryptography is known for its quantum resistance, so our secure MPC protocols are quantum-resistant as well. We propose three secure MPC protocols with BFVrns [19, 46, 61] that is a ring-learning with error (RLWE) based FHE scheme. We offer efficient secure MPC implementations based on PALISADE lattice cryptography library[1] [97].

Efficiency of secure MPC is measured based on three metrics such as rounds, computation cost and communication cost. Rounds refers to total number of communication between parties and servers and among servers. Communication cost is used for the amount of data that is transferred during the execution of the protocol. Computation cost is the amount of computation conducted in each party and server to process the protocol. Some studies suggest some secure MPC protocols that computational and communication complexities of all the parties who participate in the protocol depend on the complexity of the function [27, 14, 58, 65]. This type of protocols may not be efficient in real-life application due to the high complexity of the function. Some recent protocols require communication at every gate and all parties to be online [15, 43, 41] which is not very possible all the time. In some protocols, decryption is jointly computed [82] which is not very practical in most of use cases. The protocol of [12] requires the parties to generate a joint public key online which

[1]https://gitlab.com/palisade/palisade-release Retrieved on 5 May 2022

is not always achievable in real life. Halevi et al. suggests a secure MPC protocol for a small class of functions that is not enough for most of the applications [60]. To overcome these challenges, we introduce our FHE and PRE-based secure MPC construction and its two variants with AES and SGX.

We introduce a novel concept of secure MPC as a hybrid of FHE and PRE. Our secure MPC constructions mainly depend on FHE approach. We make use of its proxy reencryption (PRE) capability for constructing the secure MPC protocols. PRE allows us to conduct computation on encrypted data which are encrypted under many different keys while FHE enables secure computation for two parties. The recent Threshold FHE schemes have some pitfalls that a third party should know all secret keys of the parties to create a common multiparty operation key. We overcome this problem by using PRE. PRE allows reencryption of the ciphertext under a common key to enable evaluation of the function on the given ciphertexts by different parties. It also enables reencrytion of the result of the function evaluation under individual keys of each party so that each party can decrypt the result without exchanging any keys or partial decrypted results.

Our protocols depend on honest-but-curious (semi-honest) security model that is a common security model used in practical implementations of secure MPC due to performance reasons [18, 25, 45, 74]. We assume that two untrusted servers in the protocols do not collude. The assumption of existence of two non-colluding servers is widely used in many studies [32, 25, 45, 40]. Furthermore, we propose further improvements to enhance the security model to malicious model with non-interactive zero knowledge (NIZK) proofs proving plaintext knowledge as done in the work [82]. Our protocols can be easily used by the privacy-preserving applications such as auctions, electronic voting or secure machine algorithms can be good use cases for our secure MPC protocols.

In our main protocol PREFHE, we assume that we have two non-colluding but untrusted servers. One of the server is responsible for creating FHE parameters and generating common key pair and reencryption keys for each part. Another one conducts evaluation of the function on the ciphertexts and reencryption of the result. In PREFHE, the parties are involved in the protocol while uploading their encrypted data and decrypting the final result. In the rest of the protocol, any party does not need any further communication with the servers. The communication between the clients or servers does not depend on the complexity of the function to be evaluated. Also, the computation of the parties/clients does not depend on the complexity of the function F. All parties are not expected to be online after sending their encrypted data to the server. In the encryption or decryption phase, we do not require any joint computation by the parties. The parties only send their encrypted data to the server and they get the final result in encrypted at the end of the protocol.

In the AES variation of PREFHE, PREFHE-AES, the parties send their data encrypted under AES-128 that enables less communication cost between the parties and the server (S_1). We adapt the idea of the work in [89] to our secure MPC protocol. This adaption creates an extra computational cost on the server (S_1), but this can be handled with improvements such as parallelization or including special hardware for AES encryption/decryption. We propose SGX variation of PREFHE as PREFHE-SGX that combines PALISADE library (for implementation of the FHE scheme), Intel SGX technology and secure MPC. We use Gramine as a bridge between PALISADE and Intel SGX to avoid adjusting the PALISADE code to SGX [70].

5.1.1 Our Contributions

PREFHE: PREFHE proposes the first combination of FHE and PRE approach for secure MPC protocols. It enables practical secure MPC implementation for real-life use cases. It requires four rounds including key exchange phase which is not included

as a round in many previous works. The parties in the PREFHE protocol are not required to be online after they send their encrypted data to the server. This makes the protocol a good fit for the applications that do not allow online access all the time. The parties send their data encrypted under the BFVrns scheme that may result in kilobytes sometimes megabytes. This may cause some problems in low-bandwidth network or applications that have small memory or network channel. To handle this problem, we propose AES version of PREFHE.

PREFHE-AES: To reduce the communication cost of PREFHE, the parties send their data encrypted in AES-128 which has smaller data size to be sent to the server. We reduce communication cost, but it comes at a price of computation cost on the server side. Computing decryption of AES-128 homomorphically is a costly operation. For applications that require less communication cost, PREFHE-AES is a good fit with the same number of rounds as PREFHE at a reasonable computational time.

PREFHE-SGX: For the applications that allow only one server or have constraints of local computation, PREFHE-SGX is a convenient approach with three rounds that has less than most of the state-of-the-art protocols. It also introduces reasonable computational cost which only depends on the function to be evaluated. We lead adapting SGX and FHE together in secure MPC protocols that leads practical use of secure MPC protocols in real-life applications.

Performance In Practice: Our three protocols provide reasonable amount of running time on the client and server side with milliseconds or seconds that mainly depends on the function. Even if circuit depth is high, we use an efficient FHE scheme,

BFVrns, that takes advantages of Residue Number System (RNS) and packing of plaintext in SIMD manner [111, 53].

5.1.2 Related Works

The idea of using threshold homomorphic encryption in MPC protocols was first presented by Cramer, Damgard, and Nielsen [39]. Somewhat homomorphic encryption was used to boost implementation of MPC protocols in some studies such as [15, 43]. In their protocols, all parties compute proportional to the complexity of the function to be computed and interact at every gate. Choudhury et al. proposed better communication at a computation cost [34]. Their work suggested a kind of interactive bootstrapping protocol to refresh ciphertexts. Cloud server idea came with the work by Kamara et al. [65]. They proposed server-aided MPC idea by assigning large amount of works from the computation to some parties. Halevi et al. suggested the idea of secure computation on the web to minimize communication between parties in the computation [60]. After FHE is proposed by Gentry [50], new approaches based on FHE were suggested by Lopez et al. [81, 82] using multikey FHE approach. Asharov et al. presented an efficient threshold FHE based secure MPC scheme in terms of round, communication and computation costs [12]. TFHE schemes allow to jointly generate a common FHE public key along with a secret key that is shared by them later. For decryption, they conduct a collobarative decryption process on ciphertexts to get the final plaintext without learning others' inputs. Garg et al. achieved 2-round MPC from indistinguishability obfuscation [49]. As an optimization they suggested another 2-round MPC protocol from multikey FHE that is independent of the circuit to be computed.

5.2 Background

5.2.1 Intel SGX

The Trusted Execution Environment (TEE) is an approach for secure computation that enables the processing of sensitive data within the main processor's secure area (enclave). TEE delivers memory in secure enclaves for isolated computation in the presence of a malicious host. Other processes, such as user or kernel level operations, cannot modify the code contained within the enclave. SGX is a hardware-assisted version of TEE that is available on various Intel processors. SGX enables code to execute in a protected enclave that can communicate with other applications through a dedicated channel, but other applications cannot access the enclave itself [38]. Enclave execution takes place in the protected mode (at ring 3) and follows the address translation done by the operating system kernel [38]. SGX has remote attestation to prove the integrity of the code running in the enclave. When a malicious party attacks a system, it cannot access the code running/stored in the enclave. This reduces the attack surface of the system. SGX has also some disadvantages as follow.

- Paging cost: Data is encrypted and decrypted during exchanging the data between the enclave and outer program. This step creates latency during the paging process.
- Memory limit: SGX has physical memory limit of 128 MB while the practical limit is 90 MB [57, 38].
- Lack of library support: Some C++ operations or libraries cannot be used in SGX such as vector from standard library and system calls.

In our study, SGX helps us to execute sensitive data in an untrusted server. In PREFHE, we have to use two servers to prevent disclosing of parties' data. In the PREFHE-SGX version, one server handles the tasks of two servers. When the server in PREFHE-SGX runs the sensitive data that helps decrypting the clients' encrypted data. In the new version, creating the common key pair and reencryption key that

use the secret key of common key pair take place in the SGX enclave which cannot be tampered by the host server or any other malicious third parties.

Gramine: Gramine is an open-source, lightweight Lib OS(Library Operating System) project that supports Intel SGX and allows users to run their existing applications on Intel SGX [70]. Intel Labs initiated Gramine (previously called Graphene) to provide an open-source compatibility layer for Intel SGX. Gramine bridges various kind of applications and Intel SGX without modifying the application code. Gramine supports native Linux binaries on all platforms. We use Gramine to integrate PALISADE homomorphic encryption library and Intel SGX. Since Intel SGX does not allow usage of some libraries, Gramine handles this problem for developers. PALISADE, Gramine and Intel SGX are first used by Takeshita et al. [116] while we pioneer using this system for building secure MPC protocols.

5.3 Secure MPC Protocols

In this section, we introduce our three secure MPC protocols in detail.

5.3.1 PREFHE: Secure MPC from Multikey FHE and PRE

We construct a secure MPC protocol based on the BFVrns scheme which is a prominent lattice-based FHE scheme. Our protocol is constructed on two untrusted servers and n clients who want to run secure MPC on their secret data collabaratively. We utilize PRE approach to enable privacy of the inputs of the clients and the result of secure multiparty computation. PRE allows reencrypting different ciphertext under the same secret key. Indirectly, it also allows decrypting the same input under different secret keys. In the protocol, we utilize the PRE method proposed in [98] as a building block to manage two-untrusted-servers setting. We assume that these two servers do not collude and the clients and S_1 use a secure channel to prevent S_2 to access any ciphertext which is encrypted under reencryption keys generated with the common

key pair. We summarize how this protocol works step by step. Also, we provide the security analysis for semi-malicious model (a.k.a. honest-but-curious model).

1. The untrusted server S_2 decides on the FHE parameter set which is generated by our previous work in Chapter 2.5 and verified by LWE Estimator [8] that is used in Homomorphic Encryption Standard [6]. S_2 decides on the client index j who initiates *Key Generation* and S_2 creates the crytocontext cc and a common key pair $CKPair$ from cc to reencrypt the ciphertext under the same key pairs, publishes FHE parameters, $CKPair.pk$ and j to the clients as in Figure 5.1.

2. The clients can check the security of the parameter set with LWE Estimator and if it does not provide necessary security limit (i.e. at least 128-bit security level), they can ask for S_2 to generate a new parameter set and publish it to all parties.

3. The client j initiates the *Key Generation* process and broadcasts his public key as pk_j.

4. Other clients operate *Multiparty Key Generation* using pk_j. Each party encrypts their data under their individual public key.

5. All clients operate *ReKey Generation* process to generate new encryption keys to enable S_1 to eval the function on the data.

6. The clients reencrypt their ciphertext under their new individual reencryption keys. They send these new ciphertexts $cptxtC_i$ to S_1. S_1 evaluates the function to be computed on these ciphertexts as in Figure 5.2. S_1 does not know the secret key of the common key pair $CKPair$, thus S_1 cannot decrypt the ciphertexts of the clients.

7. As seen in Figure 5.3, S_2 creates reencryption keys to allow the clients to decrypt the result under their individual secret keys. S_2 sends these keys $REKey_i$ to S_1 allowing that noone can see the result in plaintext including S_1 and S_2.

8. S_1 reencrypts the result under corresponding new reencryption keys of the clients and sends all r_i to corresponding public key holder clients where $1 \leq i \leq n$.

9. Each client decrypts their reencrypted ciphertext r_i under their individual secret keys and learns the result of the multiparty computation.

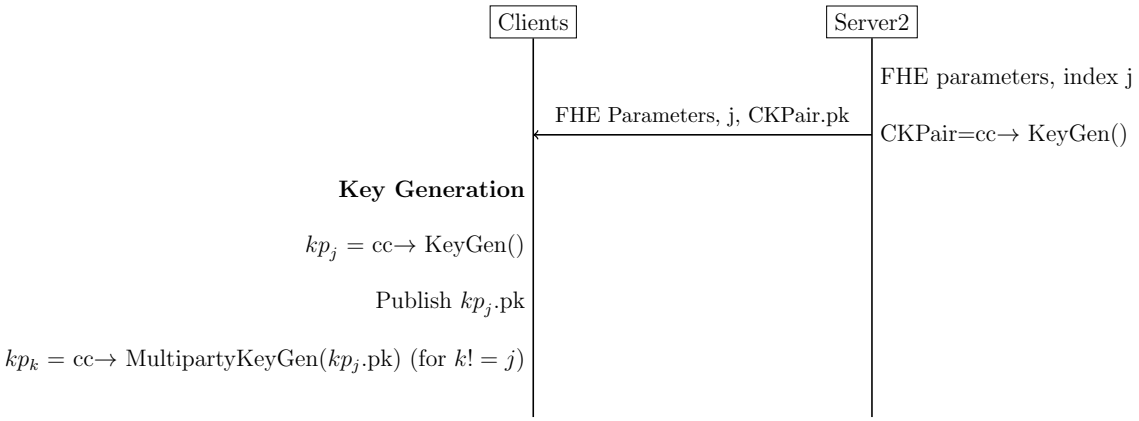


Figure 5.1. Key generation and exchange phases of the PREFHE protocol.

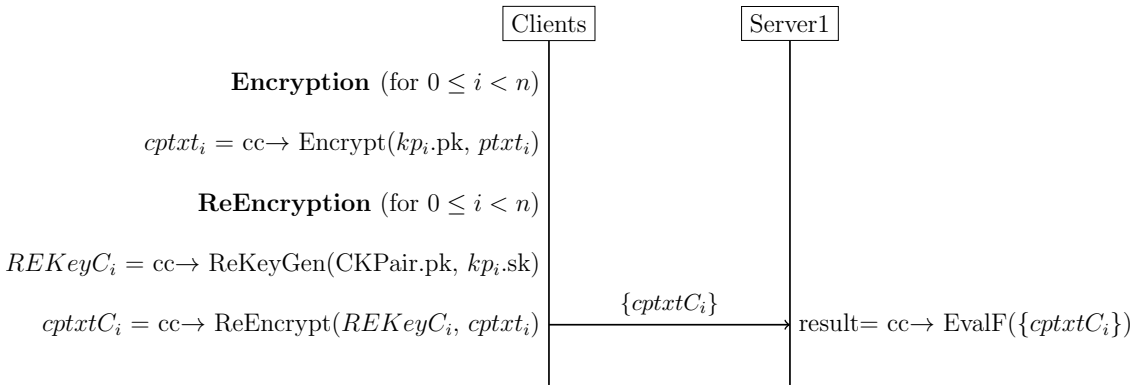


Figure 5.2. Reencryption phase of PREFHE and PREFHE-SGX protocol.

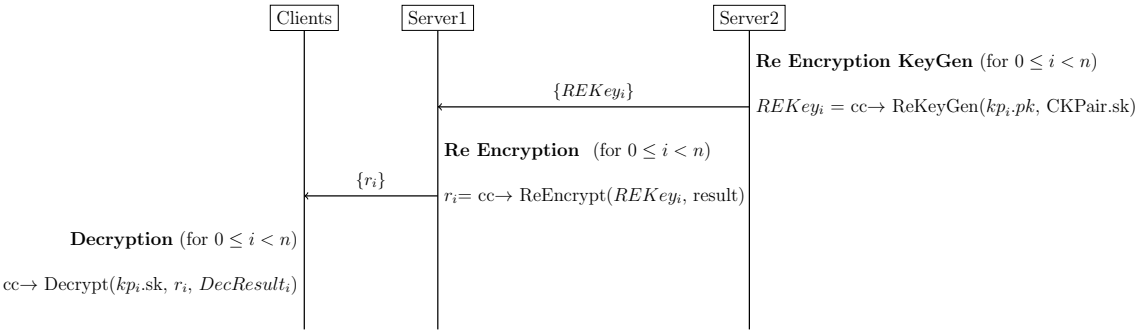


Figure 5.3. Second reencryption and decryption of the PREFHE protocol.

5.3.2 PREFHE-AES: Secure MPC from Multikey FHE and PRE with AES

We combine the first version of our secure MPC protocol with AES-128 encryption method to reduce communication cost between the servers and clients. In the previous scheme, the clients need to send their ciphertexts that are encrypted under the BFVrns scheme. The ciphertext size in the FHE schemes is large such as $2N \log q$ bits where N is the ring dimension, $\log q$ is ciphertext modulus. To overcome this problem, the clients send their data encrypted under AES-128 encryption that results in a far smaller ciphertext sizes than the BFVrns encrypted one. The ciphertext size in AES-128 depends on the plaintext size, mode of operation and padding. The steps of the protocol are explained as follows:

1. The protocol starts with the key generation and key distribution as in Figure 5.4. The FHE parameters are decided by S_2 as in PREFHE and distributed to the clients. At the same time, S_2 creates a cryptocontext cc and generates a common key pair $CKPair$ from cc to let the clients create their reencryption keys and reencrypt their ciphertexts.

2. The clients generate their public, private key pairs and encrypt their data under their AES-128 key as c_i as in Figure 5.4.

3. The clients encrypt their AES key with their FHE public key as ck_i and it can sent offline.

4. They create reencryption key from the common key pair as $REKP_i$ and send $REKP_i$, c_i to S_1.

5. S_1 encrypts the ciphertexts with their FHE public keys and unravels the ciphertexts with homomorphic AES-128 decryption (AES^{-1}) as in Figure 5.6.

6. S_1 reencrypts these ciphertexts again under their corresponding reencryption keys and evaluates the necessary function on the FHE ciphertexts.

7. S_2 generates the second reencryption key to enable the ciphertexts to be decrypted under individual private keys of the clients and sends $REKeyRC_i$ to S_1 as in Figure 5.5.

8. S_1 reencrypts the result with their respective reencryption keys and conducts dimension reduction as in [22] to reduce the size of the ciphertexts, then S_1 sends these ciphertexts to the corresponding clients.

9. The clients decrypt their ciphertext with their private keys and get the result in plaintext.

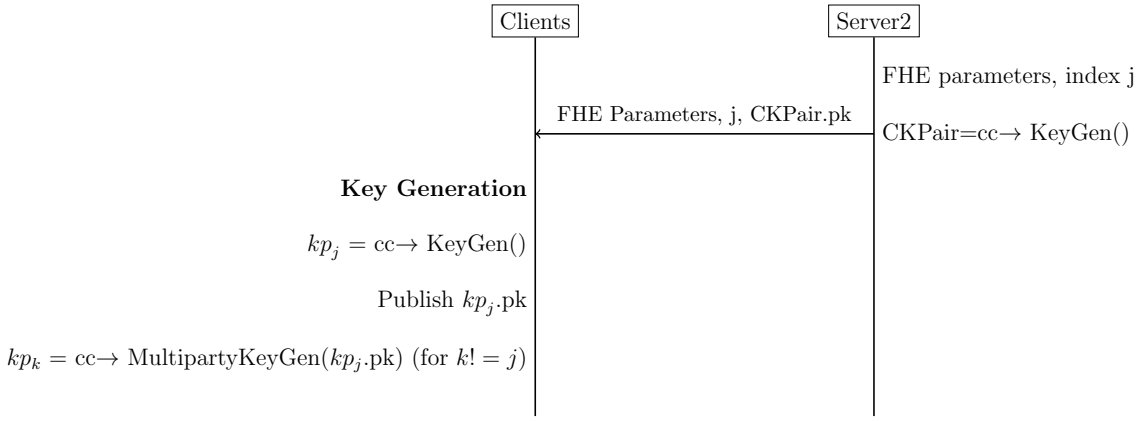


Figure 5.4. Key generation and distribution of the PREFHE-AES protocol.

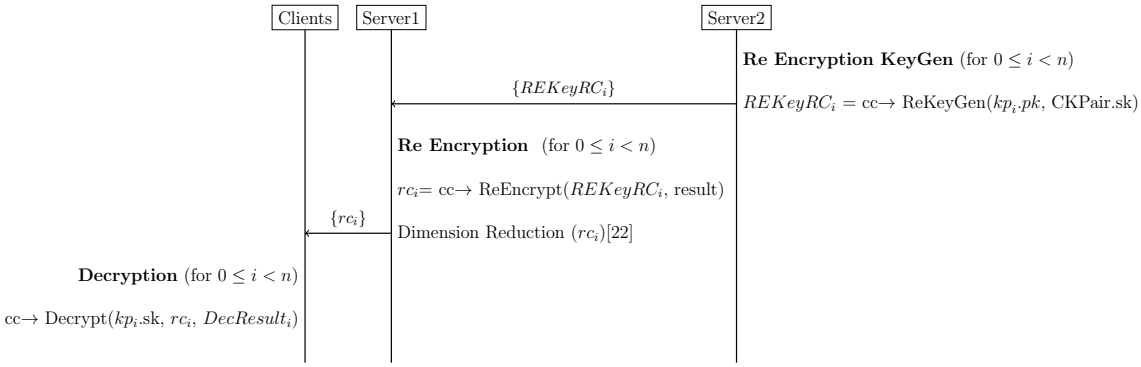


Figure 5.5. The second reencryption and decryption phase of the PREFHE-AES protocol.

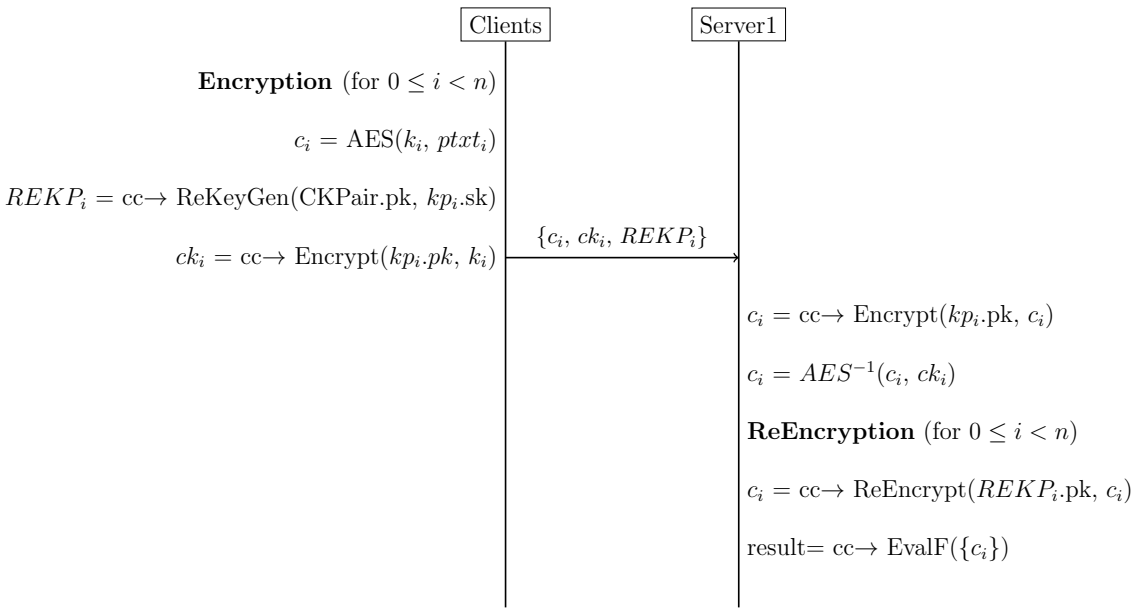


Figure 5.6. The first reencryption and evaluation of function F phase of the PREFHE-AES protocol.

5.3.3 PREFHE-SGX: Secure MPC from Multikey FHE and PRE with SGX

In this section, we present PREFHE-SGX that enables constructing the PREFHE protocol with one-server setting. Intel SGX enables calculation of sensitive data in the SGX enclave by protecting the data from the outer applications or adversaries. The difference between this protocol and PREFHE is calculating common key pair and reencryption keys in the SGX enclave. In PREFHE, S_2 conducts these operations separately to prevent the clients' data to be seen in plaintext by S_1.

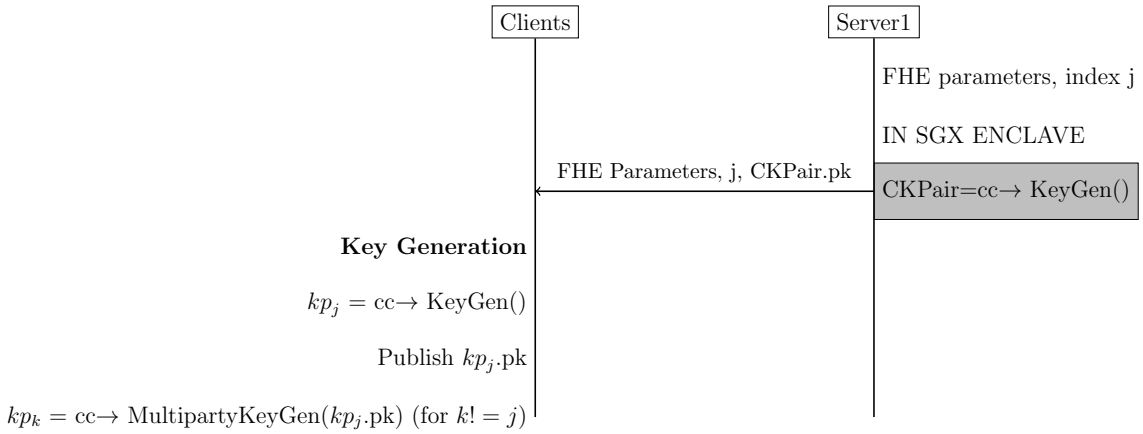


Figure 5.7. Key exchange phase of the PREFHE-SGX protocol. (Grey box represents the SGX enclave.)

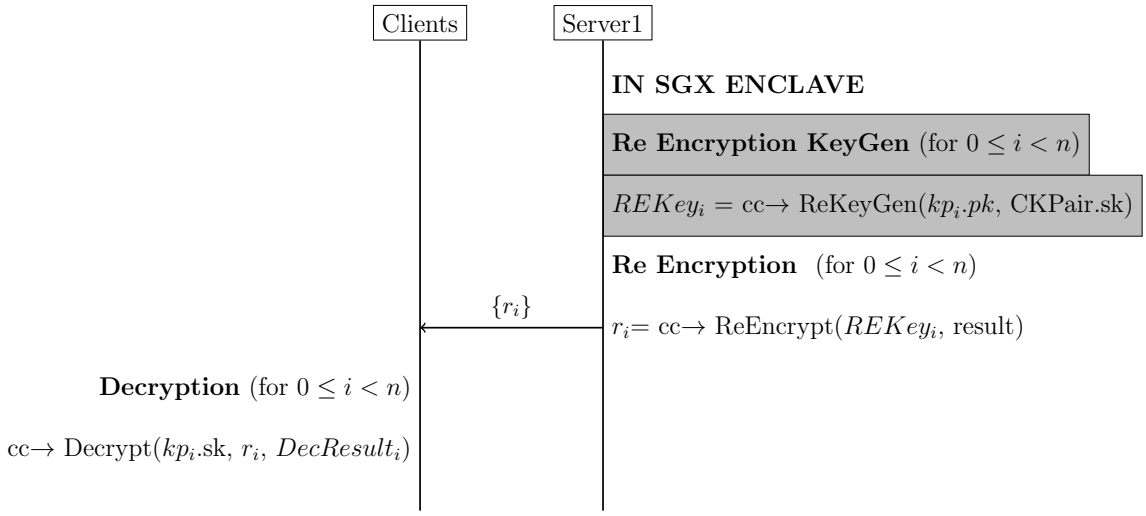


Figure 5.8. Second reencryption and decryption of the PREFHE-SGX protocol.(Grey box represents the SGX enclave.)

With SGX, the secret key of the common key pair is stored in the SGX enclave. The operations which use this secret key also locate in the SGX enclave so that the server cannot decrypt the clients' encrypted data. We refer the reader to the explanation of PREFHE to avoid redundancy. In this protocol, Gramine allows us to run PALISADE code on the SGX enclave without any adjustments on the code. Figures 5.7, 5.2 and 5.8 depict the steps of the protocol in order.

5.3.4 Correctness

Correctness of the protocols mainly depends on the BFVrns scheme given in Chapter 2. Halevi et al. provides correctness proof of the scheme in [61]. We present the correctness of the PREFHE protocol in this section. Other two protocols follow the same correctness proof.

At the beginning of the protocol, *KeyGeneration* step creates the public, private key pair for the encryption step. After this step, we make use of PRE scheme that proposed in [98]. In this step, PRE allows encrypting of all ciphertexts under the same key pair to evaluate a function on them. PRE has *ReKeyGen* routine that creates new *evalKey* for transforming the ciphertext to another one. The second step of PRE is *ReEncryption* step which calls *KeySwitch* routine due to being a PRE operation. *KeySwitch* changes the given input ciphertext to another ciphertext that is encrypted under the new key given in *ReKeyGen* step. *KeySwitch* method uses the algorithm given in [22] as digit decomposition method. Correctness of PRE scheme is proved in [98]. In this step all ciphertexts are transformed to be encrypted under the same public key. The corresponding secret key of this public key is only known by S_2. All ciphertext are sent to S_1 to evaluate the function which clients want to calculate. S_1 evaluates the function on the ciphertexts. S_2 runs *ReKeyGen* routine to create new *evalKey* set for reencrypting the result to be decrypted under each clients' secret key. S_1 runs *ReEncryption* and sends the results to corresponding clients. Correctness of this step also depends on the PRE scheme. Clients decrypt these new individual ciphertexts with their secret key and open the result in plaintext. Correctness of this protocol mainly depends on PRE which is used two times as a subroutine.

The final decryption of ciphertext after reencryption can be represented as follows where $\boldsymbol{pk} = (p0, p1)$ and $b = p0 \cdot u[i] + e1[i] + s[i]$ from ReKeyGen procedure:

$$\begin{aligned} c_0 - s \cdot c1 &= c0 + \sum_{i=1}^{k-1}(digitsC1[i] \cdot b[i]) - s \cdot \sum_{i=1}^{k-1}(digitsC1[i] \cdot a[i]) \\ &= c0 + \sum_{i=1}^{k-1}(digitsC1[i] \cdot (p0 \cdot u[i] + e1[i] + s[i])) - s \cdot \sum_{i=1}^{k-1}(digitsC1[i] \cdot a[i]). \end{aligned} \tag{5.1}$$

After scaling down by t/q;

$$\lceil (c_0 - s * c_1)t/q \rfloor = \lceil t/q(c_0 - s \cdot \sum_{i=1}^{k-1}(digitsC1[i] \cdot a[i]) + \sum_{i=1}^{k-1}(digitsC1[i] \cdot (p0 \cdot u[i] + e1[i] + s[i]))) \rfloor. \tag{5.2}$$

$\lceil \sum_{i=1}^{k-1}(digitsC1[i] \cdot ((-e - as) \cdot u[i] + e1[i] + s[i]))t/q \rfloor$ is small enough due to $q \ll t$. It can be seen that

$$\lceil (c_0 - s \cdot c_1)t/q \rfloor = m \pmod{t}. \tag{5.3}$$

5.3.5 Security Analysis

Security for the semi-honest model: The semi-honest model implies that all parties follow the protocol description, but they still try to gather information about other parties inputs, intermediate results or overall outputs just by looking at the protocol's transcripts.

Security of the PREFHE protocol mainly depends on underlying FHE scheme of the protocol and BFVrns depends on RLWE assumption that is mentioned in Definition 2.2.12 in Chapter 2.

KeyGen/MultipartyKeyGen: This step creates random sk and pk pairs for the clients and its security depends on RLWE assumption.

Encryption: The clients encrypt their inputs under their public key so noone else can decrypt and see their input.

ReKeyGen + ReEncrypt: This step depends on PRE scheme proposed in [98] which is proved as IND-CPA secure in [98]. The clients create new $evalKey$ to reencrypt their encrypted inputs to allow function evaluations on all of the clients' data. This step requires individual sk so that noone else can create other $evalkey$ from their sk. These reencrypted ciphertexts are sent to only S_1 through a secure channel to prevent any decryption by S_2. According to our assumption on two non-colluding servers, S_1 cannot decrypt ciphertexts sent by the clients and result of function evaluation on ciphertexts because it has no access to $CKPair.sk$. Also, S_2 cannot decrypt or manipulate ciphertexts because it cannot see clients' ciphertexts or the final result. S_1 cannot get any additional information from intermediate outputs. *ReKeyGen + ReEncrypt* is used in second, third and forth rounds to prevent leaking any additional information about the ciphertexts and result.

When all rounds come together, the PREFHE protocol ensures security in the semi-honest model. We can improve the security model to malicious model with NIZK protocols to prove plaintext knowledge as in the work proposed in [82].

5.4 Results

5.4.1 Implementation on PALISADE

Experimental Setup. We run the experiments on the Microsoft Azure Standard DC2s v2 virtual machine that has two cores and 8 GB memory running Ubuntu 20.04. We use PALISADE lattice cryptography library version 1.10.6 [97] and Gramine version 1.1 [2].

5.4.2 Performance Results

In this section, we analyze our secure MPC protocols in terms of three metrics such as rounds, communication and computation complexity and compare with the state-

[2]https://github.com/gramineproject/gramine Retrieved on 5 May 2022

of-the-art protocols. Due to the unavailability of their implementations, it is not possible to compare running times with other suggested protocols. We compare the protocols such as [81, 95, 82, 12] which provide semi-malicious security model version to provide a fair comparison. AES step of the PREFHE-AES protocol is simulated on Microsoft Azure Standard DC2s v2 using its homomorphic implementation. In the simulation, we use AES-128 with no key expansion version. It requires 6800 AND, 25124 XOR and 1692 NOT gates. We simulate each gate with BFVrns evaluation functions.

Rounds: PREFHE and PREFHE-AES propose four rounds while PREFHE-SGX has three rounds in total. Each round in our protocols requires less computation than the rounds in [81, 82]. Our protocols do not require the parties to be online after they send their encrypted data to the server. On the other hand, some cutting-edge protocols require communication at every gate and the presence of all parties online [15, 43, 41] which is not feasible in real-life applications. In some protocols, decryption is jointly computed [82] that increases the number of rounds. In the work presented in [81], encryption key is jointly computed while in our protocols, this is not required which means more practicality. In our protocols, Table 5.1 suggests that PREFHE-SGX has the less number of rounds. In PREFHE-SGX, one server handles two servers' jobs, so it reduces the number of total rounds.

Communication Complexity: The communication cost between clients and the server is independent of the function to be computed. In PREFHE and PREFHE-AES, the communication cost between two servers is also independent of the complexity of the function. The works in the [81, 12], communication cost depends on the length of the input and outputs. They generally contain a set of indices, ciphertexts and eval keys which is larger than a ciphertext size.

Computation Complexity: In our protocols, the computation complexity on the server S_1 is linear in the size of the circuit computing F. Since multiplication of

two ciphertexts is considered the most costly operation, the multiplicative depth of the function mainly determines the computation cost.

Table 5.1. Performance Comparison of Main and Our Protocols

	[81]	[95]	[12]	PREFHE	PREFHE-AES	PREFHE-SGX
Rounds	4	4	4	4	4	3
Communication Comp.	$\|I/O\|$	$\|c\|$	$\|I/O\|$	$\|c\|$	$\|c_{AES}\|$	$\|c\|$
Computation Comp. (Server)	$\|F\|$	$\|F\|$	$\|F\|$	$\|F\|$	$\|F\|$	$\|F\|$

Notes: Communication complexity refers to the communication cost between a client/party and server that evaluates the function F. $\|c\|$ represents size of a ciphertext and $\|c_{AES}\|$ stands for AES-128 encrypted ciphertext size.

Table 5.2. Performance Results of the Protocols

	PREFHE	**PREFHE-AES**	**PREFHE-SGX**
Client time	24.931	12.276	26.354
Server 1 time	19.183	45142.833	25836.862
Server 2 time	27.701	29.975	NA

Notes: The experiment is done for $t = 32769$, $m = 16384$, $\log q_i = 55$, $\sigma = 3$, $rw = 0$, $\lambda = 128$. F has one multiplication as an example in this experiment. Time unit is ms. Client time represents the total runtime of one client.

Performance Results: Table 5.2 suggests that PREFHE has the fastest client and servers time with two-untrusted-servers setting. Since homomorphic decryption of AES-128 takes a long time, Server 1 computation time of PREFHE-AES is around 45 seconds which is still practical for real life. For functions having larger number of multiplications, computation cost can be improved with better hardware and

parallelization techniques. The client side of PREFHE-AES has better performance over other two protocols due to outsourcing the encryption of the data to Server 1. PREFHE-SGX performs better than PREFHE-AES for Server 1 but worse than PREFHE. The reason is that Server 1 in PREFHE-SGX handles two servers' tasks and it utilizes SGX that has paging latency.

Trade-off between protocols: According to the needs of the application, the user may consider trade-off between client and server side computations or rounds or communication cost. For applications that have network bandwidth constraints, the user may prefer PREFHE-AES over others. For computational constraints or time sensitivity, PREFHE is the best fit with the short latency. For the systems that allow one server in the secure MPC setting, PREFHE-SGX signifies the best round-efficient one in all secure MPC protocols.

5.5 Conclusion

We propose three distinct secure MPC protocols constructed from FHE, AES-128, and Intel SGX. PREFHE is highly efficient in real-world applications, whereas PREFHE-AES introduces a communication-efficient protocol that is highly efficient in low-bandwidth networks. PREFHE-SGX proposes a single-server setting with three rounds and is a pioneer in the use of FHE and SGX in secure MPC protocols. Our protocols' communication costs are function-independent. Additionally, our protocols do not require parties to be online following the transmission of encrypted data to Server 1. The decryption phase does not require any collaboration on the part of the parties, which increases the protocols' practicality. We present efficient and secure MPC protocols that are applicable to a variety of use cases in real life.

CHAPTER 6

CONCLUSION

The contributions of this dissertation is tri-fold. In Chapter 3, we propose the first parameter selection algorithm for all FHE parameters. This approach for the parameter selection on the FHE scheme, BFVrns, suggests optimizing performance, noise growth and CEF by keeping security level at least 128-bit. It is also easily adjustable to other schemes.

In Chapter 4, we introduce P-cPIR and Recursive P-cPIR version I provide at least $\sim 2^{18}\times$ computational improvement over XPIR, SealPIR, at least $\sim 2^{14}\times$ over SHECS-PIR, OnionPIR. Recursive P-cPIR version II proposes the same query and half response cost as OnionPIR and less than other protocols such as SealPIR, SHECS-PIR, XPIR. Parameter selection algorithm proposing lower CEF values and at least 128-bit security.

In Chapter 5, we provide three constructions of PREFHE protocols that are based on FHE and PRE. We introduce different secure MPC protocols for different trade-offs. The first one is PREFHE which is for computational efficiency. PREFHE-AES suggests a new construction of communication-efficient secure MPC. The last one is PREFHE-SGX that has one-server-setting secure MPC with Intel SGX with less number of rounds than existing protocols. PREFHE-SGX enables the first SGX and FHE-based secure MPC. It also combines FHE, PRE, GRAMINE, and Intel SGX for a secure MPC protocol. We introduce practical secure MPC protocols for real-world applications.

We implemented all protocols and algorithms with PALISADE library except AES part of PREFHE-AES to show their real-life efficiency and practicality.

REFERENCES

[1] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. *Proceedings on Privacy Enhancing Technologies*, April 2016:155-174, April 2016.

[2] Carlos Aguilar Melchor, Benoît Crespin, Philippe Gaborit, Vincent Jolivet, and Pierre Rousseau. High-speed Private Information Retrieval Computation on GPU. In *Second International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'08)*, pages 263-272, Cap Esterel, France, August 2008. IEEE Computer Society Press.

[3] Carlos Aguilar-Melchor and Philippe Gaborit. A lattice-based computationally-efficient private information retrieval protocol. *Cryptology ePrint Archive, Report*, 446, 2007.

[4] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC 1996, pages 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[5] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC 1997, pages 284-293, New York, NY, USA, 1997. Association for Computing Machinery.

[6] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[7] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *International Association for Cryptologic Research Cryptology ePrint Archive*, page 636, 2012.

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

[9] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. In *IEEE Symposium on Security and Privacy (SP)*, pages 962-979. IEEE, 2018

[10] David Archer, Lily Chen, Jung Hee Cheon, Ran Gilad-Bachrach, Roger A Hallman, Zhicong Huang, Xiaoqian Jiang, Ranjit Kumaresan, Bradley A Malin, Heidi Sofia, et al. Applications of homomorphic encryption. In *Crypto Standardization Workshop, Microsoft Research*, page 14, 2017.

[11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 403-415, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 483–501, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] 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 Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 423–442, Cham, 2017. Springer International Publishing.

[14] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 351-371. 2019.

[15] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 169-188, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] Jingguo Bi, Mingjie Liu, and Xiaoyun Wang. Cryptanalysis of a homomorphic encryption scheme from isit 2008. In *IEEE International Symposium on Information Theory Proceedings*, pages 2152-2156. IEEE, 2012.

[17] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of Annual Association for Computing Machinery*, 50(4):506–519, 2003.

[18] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *International Conference on Financial Cryptography and Data Security*, pages 325-343. Springer, 2009 .

[19] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868-886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] 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. Annual Association for Computing Machinery.

[21] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC 2013, pages 575-584, New York, NY, USA, 2013. Association for Computing Machinery.

[22] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 97-106, USA, 2011. IEEE Computer Society.

[23] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 505-524, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[24] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 402-414, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[25] Octavian Catrina and Florian Kerschbaum. Fostering the uptake of secure multiparty computation in e-commerce. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, ARES '08, pages 693-700, USA, 2008. IEEE Computer Society.

[26] Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 50-61, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[27] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC 1988, pages 11-19, New York, NY, USA, 1988. Association for Computing Machinery.

[28] Yuanmi Chen. Lattice reduction and concrete security of fully homomorphic encryption. *Department Informatique, ENS, PhD thesis*, 2013.

[29] Yuanmi Chen and Phong Q. Nguyen. Bkz 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 1-20, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[30] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409-437, Cham, 2017. Springer International Publishing.

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

[32] Seung Geol Choi, Ariel Elbaz, Ari Juels, Tal Malkin, and Moti Yung. Two-party computing with encrypted data. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, pages 298-314, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[33] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of ACM*, 45(6):965-981, November 1998.

[34] Ashish Choudhury, Jake Loftus, Emmanuela Orsini, Arpita Patra, and Nigel P. Smart. Between a rock and a hard place: Interpolating between mpc and fhe. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 221-240, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[35] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. STOC (Symposium on Theory of Computing) 2003, pages 136-142, New York, NY, USA, 2003. Association for Computing Machinery.

[36] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 446-464, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[37] Anamaria Costache, Kim Laine, and Rachel Player. Homomorphic noise growth in practice: comparing bgv and fv. Cryptology ePrint Archive, Report 2019/493, 2019. Accessed: 2022-3-3.

[38] Victor Costan and Srinivas Devadas. Intel sgx explained. Cryptology ePrint Archive, Paper 2016/086, 2016. Accessed: 2022-5-5.

[39] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 280–300, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[40] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 378-394, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[41] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security – ESORICS 2013*, pages 1-18, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[42] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 558-576, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[43] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 643-662, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[44] Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 617-640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[45] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In Ian Goldberg and Mikhail J. Atallah, editors, *Privacy Enhancing Technologies*, pages 235-253, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[46] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. Accessed: 2022-3-3.

[47] Steven D Galbraith. Space-efficient variants of cryptosystems based on learning with errors. *url: https://www. math. auckland. ac. nz/˜ sgal018/compact-LWE. pdf*, 2013. Accessed: 2022-3-3.

[48] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 31-51, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[49] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In Yehuda Lindell, editor, *Theory of Cryptography*, pages 74-94, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

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

[51] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 107-109, USA, 2011. IEEE Computer Society.

[52] Craig Gentry and Shai Halevi. Compressible fhe with applications to pir. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464, Berlin, Heidelberg, 2019. Springer Berlin Heidelberg.

[53] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 465-482, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[54] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 850-867, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[55] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *In Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, pages 803-815, Berlin, Heidelberg, 2005. Springer-Verlag.

[56] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75-92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[57] Anders T. Gjerdrum, Robert Pettersen, Håvard D. Johansen, and Dag Johansen. Performance of trusted computing in cloud infrastructures with intel sgx. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, CLOSER 2017, pages 696-703, Setubal, PRT, 2017. SCITEPRESS - Science and Technology Publications, Lda.

[58] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. STOC (Symposium on Theory of Computing) 87, pages 218-229, New York, NY, USA, 1987. Association for Computing Machinery .

[59] Jens Groth, Aggelos Kiayias, and Helger Lipmaa. Multi-query computationally-private information retrieval with constant communication rate. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography – PKC 2010*, pages 107-123, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[60] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 132-150, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[61] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved rns variant of the bfv homomorphic encryption scheme. In Mitsuru Matsui, editor, *Topics in Cryptology CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 83-105, Germany, 2019. Springer Verlag.

[62] Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481, 2020. Accessed: 2022-3-3.

[63] Frederick Hayes-Roth. Review of "adaptation in natural and artificial systems by john h. holland", the u. of michigan press, 1975. *Association for Computing Machinery Special Interest Group on Artificial Intelligence Bulletin*, (53):15, August 1975.

[64] Vincent Herbert. Automatize parameter tuning in ring-learning-with-errors-based leveled homomorphic cryptosystem implementations. Cryptology ePrint Archive, Paper 2019/1402, 2019. Accessed: 2022-12-6.

[65] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Paper 2011/272, 2011. Accessed: 2022-12-6.

[66] Aggelos Kiayias, Nikos Leonardos, Helger Lipmaa, Kateryna Pavlyk, and Qiang Tang. Optimal rate private information retrieval from homomorphic encryption. *Proceedings Privacy Enhancing Technologies*, 2015(2):222–243, 2015.

[67] Aggelos Kiayias and Moti Yung. Secure games with polynomial expressions. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming*, pages 939-950, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[68] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings Thirty-Eighth Annual Symposium on Foundations of Computer Science*, pages 364-373. IEEE, 1997.

[69] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 104-121, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[70] Dmitrii Kuvaiskii, Gaurav Kumar, and Mona Vij. Computation offloading to hardware accelerators in intel sgx and gramine library os. *arXiv preprint arXiv:2203.01813*, 2022. Accessed: 2022-12-6.

[71] Kim Laine. Simple encrypted arithmetic library 2.3. 1. *Microsoft Research https://www. microsoft. com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1. pdf*, 2017. Accessed: 2022-3-3.

[72] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[73] Tancrède Lepoint and Mehdi Tibouchi. Cryptanalysis of a (somewhat) additively homomorphic encryption scheme used in PIR. In Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2015 International Workshops, BITCOIN, WAHC, and Wearable, San Juan, Puerto Rico, January 30, 2015, Revised Selected Papers*, volume 8976 of *Lecture Notes in Computer Science*, pages 184-193. Springer Berlin, Heidelberg, 2015.

[74] Yehuda Lindell and Benny Pinkas. A proof of security of yaos protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[75] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 319-339, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[76] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *Information Security*, pages 314-328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[77] Helger Lipmaa. First cpir protocol with data-dependent computation. In Donghoon Lee and Seokhie Hong, editors, *Information, Security and Cryptology – ICISC 2009*, pages 193-210, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[78] Mingjie Liu and Phong Q. Nguyen. Solving bdd by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, pages 293-309, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg .

[79] Alex Lombardi, Vinod Vaikuntanathan, and Thuy Duong Vuong. Lattice trapdoors and IBE from middle-product LWE. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 24-54. Springer Berlin Heidelberg, 2019.

[80] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. 2-message publicly verifiable wi from (subexponential) lwe. Cryptology ePrint Archive, Report 2019/808, 2019. Accessed: 2022-3-3.

[81] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. Cloud-assisted multiparty computation from fully homomorphic encryption. *Cryptology ePrint Archive*, 2011. Accessed: 2022-3-3.

[82] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC 2012, pages 1219-1234, New York, NY, USA, 2012. Association for Computing Machinery.

[83] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1-23, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[84] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 35-54, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[85] MATLAB. version 9.6.0.1072779 (r2019a), 2019.

[86] Carlos Aguilar Melchor and Philippe Gaborit. A fast private information retrieval protocol. In *2008 IEEE International Symposium on Information Theory*, pages 1848–1852, 2008.

[87] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 21-39, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg .

[88] Muhammad Haris Mughees, Hao Chen, and Ling Ren. Onionpir: Response efficient single-server pir. In *Proceedings of the 2021 ACM Special Interest Group on Security, Audit and Control (SIGSAC) Conference on Computer and Communications Security*, CCS '21, pages 2292-2306, New York, NY, USA, 2021. Association for Computing Machinery.

[89] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the Third Annual Association for Computing Machinery Workshop on Cloud Computing Security Workshop*, pages 113-124, New York, NY, USA, 2011. Association for Computing Machinery.

[90] Phong Q Nguyen and Brigitte Vallée. The LLL algorithm. In *Advances in Cryptology – CRYPTO 2013*, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[91] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 525-542, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[92] Jeongeun Park and Mehdi Tibouchi. Shecs-pir: Somewhat homomorphic encryption-based compact and scalable private information retrieval. In *Computer Security ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 1418, 2020, Proceedings, Part II*, pages 86-106, Berlin, Heidelberg, 2020. Springer-Verlag.

[93] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC 2009, pages 333-342, New York, NY, USA, 2009. Association for Computing Machinery.

[94] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-lwe for any ring and modulus. STOC (Symposium on Theory of Computing) 2017, pages 461-473, New York, NY, USA, 2017. Association for Computing Machinery.

[95] Andreas Peter, Erik Tews, and Stefan Katzenbeisser. Efficiently outsourcing multiparty computation under multiple keys. *IEEE Transactions on Information Forensics and Security*, 8(12):2046–2058, 2013.

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

[97] Yuriy Polyakov, Kurt Rohloff, and Gerard W Ryan. Palisade lattice cryptography library, 2018. Accessed: 2022-3-3".

[98] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Transactions on Privacy and Security*, 20(4), September 2017.

[99] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual Association for Computing Machinery Symposium on Theory of Computing*, STOC 2005, pages 84-93, New York, NY, USA, 2005. Annual Association for Computing Machinery.

[100] 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.

[101] Miruna Roşca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 283-297, Cham, 2017. Springer International Publishing.

[102] Ron D. Rothblum, Adam Sealfon, and Katerina Sotiraki. Towards non-interactive zero-knowledge for np from lwe. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 472-503, Cham, 2019. Springer International Publishing.

[103] Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.4.0)*, 2019. Accessed: 2022-3-3.

[104] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. STACS '03, pages 145-156, Berlin, Heidelberg, 2003. Springer-Verlag.

[105] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.

[106] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1):181–199, 1994.

[107] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[108] Victor Shoup. Number theory library 5.5. 2 (ntl) for c++. *url: https://libntl.org/*, 1996. Accessed: 2022-3-3.

[109] Radu Sion and Bogdan Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, pages 2006–06. Internet Society, 2007.

[110] Nigel P. Smart and Fan Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography – PKC 2010*, pages 420-443, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[111] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[112] StataCorp. Stata statistical software: Release 13, 2013. Accessed: 2022-3-3.

[113] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 377-394, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[114] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. ASIACRYPT '09, pages 617-635, Berlin, Heidelberg, 2009. Springer-Verlag.

[115] Julien P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology — ASIACRYPT'98*, pages 357-371, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[116] Jonathan Takeshita, Colin McKechney, Justin Pajak, Antonis Papadimitriou, Ryan Karl, and Taeho Jung. Gps: Integration of graphene, palisade, and sgx for large-scale aggregations of distributed data. *Cryptology ePrint Archive*, 2021.

[117] Jonathan Trostle and Andy Parrish. Efficient computationally private information retrieval from anonymity or trapdoor groups. In Mike Burmester, Gene Tsudik, Spyros Magliveras, and Ivana Ilić, editors, *Information Security*, pages 114-128, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[118] Joop van de Pol and Nigel P. Smart. Estimating key sizes for high dimensional lattice-based systems. In Martijn Stam, editor, *Cryptography and Coding*, pages 290-303, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[119] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24-43, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[120] Alexander Viand and Hossein Shafagh. Marble: Making fully homomorphic encryption accessible to all. In *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '18, pages 49-60, New York, NY, USA, 2018. Association for Computing Machinery.

[121] Cavidan Yakupoglu and Kurt Rohloff. Parameter selection for computationally efficient use of the bfvrns fhe scheme. *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering*, 16(07), 2022.

[122] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, pages 162-167. IEEE, 1986.