KU LEUVEN

FACULTY OF
ENGINEERING SCIENCE

PIRANA implementation using GBFV

Antoine Janssens van der Maelen

Thesis submitted for the degree of
Master of Science in
Electrical Engineering, option
Electronics and Chip Design

Supervisor
Prof. dr. F. Vercauteren

Assessors
Ir. Kn. Owsmuch
K. Nowsrest

Assistant-supervisors
Ir. An Assistent
A. Friend

Academic year 2025 – 2026

Preface

I would like to thank XXX.

Antoine Janssens van der Maelen

Contents

Preface **i**

Abstract **iii**

List of Figures and Tables **iv**

List of Abbreviations and Symbols **v**

1 Introduction **1**
- 1.1 Lorem Ipsum 4–5 1
- 1.2 Lorem Ipsum 6–7 1

2 Theoretical background **3**
- 2.1 Homomorphic encryption 3
- 2.2 Cyclotomic rings and (R)LWE 5
- 2.3 BFV and CLPX 6
- 2.4 GBFV 7
- 2.5 SIMD 9
- 2.6 Private information retrieval 10
- 2.7 PIRANA 12
- 2.8 Fheanor 14

3 PIR implementation **17**
- 3.1 GBFV-PIRANA, single-query small payload 17
- 3.2 GBFV-PIRANA, single-query large payload 18

4 The Final Chapter **21**
- 4.1 The First Topic of this Chapter 21
- 4.2 The Second Topic 22
- 4.3 Conclusion 23

A The First Appendix **27**
- A.1 More Lorem 27
- A.2 Lorem 51 28

B The Last Appendix **29**
- B.1 Lorem 20-24 29
- B.2 Lorem 25-27 30

Bibliography **31**

Abstract

The `abstract` environment contains a more extensive overview of the work. But it should be limited to one page.

List of Figures and Tables

List of Figures

2.1 Ciphertext HE [13] 3
2.2 Gen09 bootstrapping [13] 4

List of Tables

List of Abbreviations and Symbols

Abbreviations

LoG	Laplacian-of-Gaussian
MSE	Mean Square error
PSNR	Peak Signal-to-Noise ratio

Symbols

c	Speed of light
E	Energy
m	Mass
π	The number pi

Chapter 1

Introduction

The first contains a general introduction to the work. The goals are defined and the modus operandi is explained.

1.1 Lorem Ipsum 4–5

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

1.2 Lorem Ipsum 6–7

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 2

Theoretical background

2.1 Homomorphic encryption

So far, encryption technologies are shown to be effective to protect stored and in transit data. However, when data is used for computation, preserving privacy becomes more complex. To achieve this, several privacy-enhancing technologies (PETs) are available. One type of PET is homomorphic encryption (HE), which allows computation on encrypted data without decrypting. No party performing computations has access to the plaintext data, these data remains encrypted. Partially homomorphic encryption (PHE) is a type of HE that only supports homomorphic multiplication or addition, but not both. Full HE (FHE) and Somewhat HE (SHE) support both multiplications and additions, but with SHE only up to a limited computation depth. To enhance security, noise is added to the encrypted data when using HE, in the least significant bits as illustrated in figure 2.1. However, when performing computations the noise can grow beyond the noise padding bits, eventually corrupting the data in SHE. To mitigate this, bootstrapping can be performed in FHE to reduce the amount of noise, thus allowing more computations to be done whilst maintaining data integrity.

FHE offers strong advantages when compared to other PETs. For instance, less communication is needed during computation when compared to MPC (multi-party computation) and it has a better track record in terms of security vulnerability when compared to TEE (trusted execution environment). [14]

On the other hand, there are some disadvantages too. FHE requires requires specialized expertise to implement. But, most importantly, FHE is computationally intensive (thus slow) for large and unstructured data. According to Ulf Mattsson,

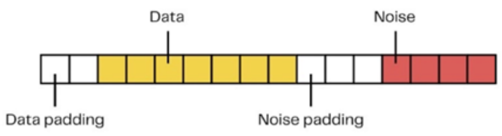

FIGURE 2.1: Ciphertext HE [13]

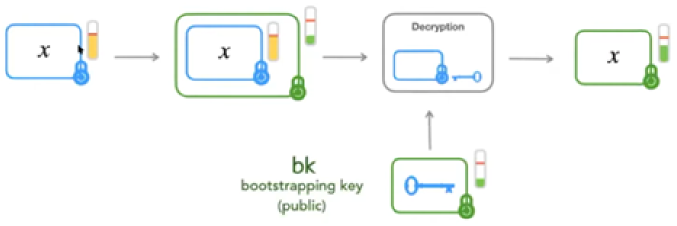

Figure 2.2: Gen09 bootstrapping [13]

general FHE processing is 1.000 to 1.000.000 times slower than equivalent plaintext operations. [20]

Whilst performing a large number of (complex) operations, noise added to the HE cyphertext will grow and overwrite data. To avoid this, two methods are used: using big integers and bootstrapping. By using big integers, enough space is provided for the noise to grow for the full computation - the so called leveled schemes. [1] To have more computation depth possible, bootstrapping operations are performed to reduce the amount of noise in between chains of computations. One should note bootstrapping is computationally and memory-intensive.

FHE schemes can be divided into multiple generations, depending on the type of bootstrapping techniques. [9] First generation schemes include schemes like the Gen09 bootstrapping technique, described in 2009, which is illustrated in figure 2.2. FHE-encrypted data are FHE-encrypted a second time, with a lower level of noise compared to the initial encryption. Then, a bootstrapping key is sent to the computing node, which is the secret key of the initial encryption, encrypted with the public key of the second encryption. Decryption with this bootstrapping key removes the first encryption layer, and one ends up with data solely encrypted via the second FHE-scheme, with a lower level of noise. This type of scheme is no longer used in practical implementations.

Second generation schemes are defined by having a slow and complex bootstrapping. However, bootstrapping cost is compensated by the use of SIMD (Single Instruction Multiple Data) operations, which will distribute this cost over many slot, so many parallel computations. Examples are BGV, BFV and CKKS. Third generation schemes are characterized by a very simple and fast bootstrapping procedure. These exhibit lower circuit complexity, faster execution times, and less noise growth when compared to second generation. On the contrary, they will not offer SIMD slots for parallel processing. Examples include torus FHE (TFHE).

Next to these generations, some leveled homomorphic encryption schemes where no efficient bootstrapping technique is known for the moment. The CLPX scheme fro Chen et al. [7] is an example, where the parameters of the scheme are set as such level to allow deep circuit evaluation before noise corrupts the result.

[1] BFV and CKKS are often implemented without bootstrapping, as a leveled scheme, but are bootstrappable.

2.2 Cyclotomic rings and (R)LWE

2.2.1 Cyclotomic rings

BFV (Brakerski-Fan-Vercauteren) is built on the RLWE problem (ring learning with errors), which is a hardness problem used in cryptography. We define the m-th cyclotomic polynomial as follows:

$$\Phi_m(x) = \prod_{j \in \mathbb{Z}_m^\times} (x - \omega_m^j)$$

- w_m is the primitive m-th root of unity, $\in \mathbb{C}$ where $m \geq 1$
- $\mathbb{Z}_m^\times$ is the unity group of integer modulo m.

The degree of the cyclotomic polynomial is equal to $\varphi(m)$, the result of the Euler's totient function of m. Although the cyclotomic polynomial have complex roots, it has been proven that the coefficients are integer numbers and the polynomials are monic (leading coefficient is 1) and irreducible. The RLWE problem is then defined over the ring $\mathcal{R} = Z[x]/(\phi_m(x)$. This ring is a subring of the cyclotomic number field $\mathbb{Q}[x]/(\Phi_m(x))$.

2.2.2 LWE

The ciphertext is constituted of two parts: uniform random numbers a_i and b, where b is the sum of the multiplication of the uniform random numbers a_i with the secret key s_i, some Gaussian distributed noise e and the message m, normalized by a delta coefficient. The corresponding ciphertext can be represented on a ring, all the possible values of m are put on a ring, at a spacing delta from each other. To decrypt, the decryption formula (2.2) is used, which is equivalent to rounding the value on the ring (which corresponds to delta times the message plus the error) to the closest possible value for m. If the error becomes too large, the value will round to the wrong message value, so returning a faulty message.

$$\text{Encryption:} \quad ct = (a_0, \ldots, a_{n-1}, b) \quad where \quad b = \sum_{i=0}^{n-1} a_i s_i + e + \Delta m \tag{2.1}$$

$$\text{Decryption:} \quad m \approx \frac{b - \mathbf{a} \cdot \mathbf{s}}{\Delta} \tag{2.2}$$

$a_i \in \mathbb{Z}_q$ are chosen uniformly at random
$s_i \in \mathbb{Z}_q$ are the secret key coefficients
$e \in \mathbb{Z}_q$ is a small error term (typically Gaussian)
m is the message encoded in the ring
Δ is the message scaling factor
$b \in \mathbb{Z}_q$ is the second component of the ciphertext

This scheme already allows to do some operations on the ciphertext: we can add two ciphertexts and perform multiplications of the ciphertext with non-encrypted integers.[2]

2.2.3 RLWE

Ring LWE is similar to LWE but it will, when constructing the ciphertext, use polynomial modulo's instead (for the message, secret key, uniform random numbers and error). When encrypting, we will normalize the message and add a Gaussian error, like in LWE. For every coefficient of the polynomial, the value of delta m plus the error will again be represented on a ring. Rounding will give the polynomial coefficients of the message m. The RLWE scheme allows to perform additions between ciphertexts and multiplication with non-encrypted constant polynomial functions.

The RLWE problem is based on the RLWE distribution for integers $q \geq 2$ and a secret s sampled from χ_{key}. The decision RLWE problem is, given many plaintext-ciphertext samples, to decide whether the samples come from a uniform random distribution or from the RLWE distribution. When solving the search RLWE problem, many plaintext-ciphertexts are given from the RLWE distribution, and one needs to find the secret s. Both variants are supposed to be hard.

2.3 BFV and CLPX

In BFV by Kim et al.[16], a subring $\mathcal{R}_t$ of $\mathcal{R}$ is ceated by taking modulo t of rhe ring $\mathcal{R}$. In the case of BFV, we fix this t to the prime integer p. The ciphertext also has a modulus q and the message is scaled by a factor $\delta = q/t$. The plaintext space corresponds to $R_t = \mathbb{Z}[x]/(\Phi_m(x), p)$. Encryption is then done via following formula:

$$\text{Ciphertext:} \quad \mathbf{ct} = \left(\left[\lfloor \Delta \cdot m \rceil + \mathbf{a} \cdot \mathbf{s} + e \right]_q, -\mathbf{a} \right) \tag{2.3}$$

$$\text{Decryption:} \quad m = \left\lfloor \frac{c_0 + c_1 \cdot \mathbf{s}}{\Delta} \right\rceil \tag{2.4}$$

The scheme can be implemented as a leveled scheme (SHE) or can be bootstrapped to a fully homomorphic encryption scheme.In BFV, one can perform addition, multiplication and automorphism over the plaintext space.

In CLPX, the idea is to use a plaintext ring modulo t, with $t = x - b$ instead of an integer p, as in BFV. The plaintext space is now defined as

$$\mathcal{R}_t = \mathbb{Z}[x]/(\Phi_m(x), x - b) = \mathbb{Z}[x]/(x - b, p) \cong \mathbb{Z}_p \tag{2.5}$$

In this CLPX-scheme, m is a k-th power of 2. When encrypting first a hat encoding is performed on the message m, by taking the modulus quotient ring of

[2]An operation on ciphertexts will, in FHE, correspond to an operation on plaintexts.

R modulo t. We get $\hat{m}$ which only has small coefficients. Encryption is done as follows:

$$\mathbf{c} = \left(\left[\Delta \cdot \hat{m} + \mathbf{a} \cdot \mathbf{s} + e\right]_q, -\mathbf{a}\right) \tag{2.6}$$

Decryption is performed using the secret key $\mathbf{s}$:

$$\hat{m} = \left\lfloor \frac{t}{q} \cdot \left(c_0 + c_1 \cdot \mathbf{s}\right)\right\rceil \tag{2.7}$$

In CLPX, addition and multiplication can be performed homomorphically.[7].

CLPX can encrypt a single huge integer modulo $\phi_m(b)$ and has much lower noise growth when compared to BFV - the growth is sublinear in b (instead of $\phi_m(b)$). This makes CLPX suitable for high-precision arithmetic HE operations. However, in CLPX Only a single element is encrypted, so no SIMD operations can be performed. Also, since the size of p is exponential in m, there are no known efficient bootstrapping techniques for CLPX.

2.4 GBFV

CLPX has much lower noise growth when compared to BFV, but does not support SIMD operations and is not known to be efficiently bootstrappable for cryptographically secure parameters. Geelen and Vercauteren propose the GBFV scheme which combines the SIMD and bootstrapping capabilities of BFV with the lower noise growth of CLPX, by tuning the parameters m and $t(x)$. Combing both properties would either yield a scheme capable of evaluating deeper circuits or would yield a scheme capable of working with smaller ring dimensions.

GBFV operates over the cyclotomic ring $\mathcal{R} = \mathbb{Z}[x]/\phi_m(x)$. The plaintext space is defined modulo an arbitrary non-zero principal ideal generated by a polynomial $t = t(x)$. This ring is $R_t = R/tR$. A plaintext $m \in \mathcal{R}_\sqcup$ is encrypted into a ciphertext $ct \in \mathcal{R}_\amalg^\in$ with RLWE:

$$ct = \left(\left[\lfloor \Delta \cdot m \rceil + a \cdot s + e\right]_q,\ -a\right)^{3} \tag{2.8}$$

The ciphertext space will be a ring $\mathcal{R}_q^2$ with $q \geq 2$. The scaling factor Δ is defined by q/t, with q the ciphertext modulus. This scaling factor is not rounded to $\mathcal{R}$, resulting in a conceptually simples scheme definition when compared to BFV and CLPX.

For correct decryption, the canonical infinity norm of the plaintext modulus must be much smaller then the ciphertext modulus. This ensures that the decryption correctly recovers plaintexts without modular wrap-around or rounding errors, all contributions from $t(x)$ (i.e. $t(x) * m(x)$) and the noise much be much smaller than q.

[3] $\lfloor x \rceil$ is rounding to the nearest integer

2.4.1 Scheme functions

The GBFV scheme has several functions which it can perform:

- Secret key generation: Samples a secret key s from a key distributon χ_{key}, $s \in \mathcal{R}$, returns s.
- Relinearization key: after multiplication of ciphertexts, one gets a higher order polynomial in s, which can not be decrypted since the scheme only knows s and not s to a higher power. The relinearization key approximates the ciphertext back to a linear equation in s. Returns the evaluation key.
- Decryption: A ciphertext is decrypted using $m = \lfloor \frac{c_0+c_1\cdot s}{\Delta} \rceil$. Returns m.

GBFV supports standard homomorphic operations on ciphertexts, using following functions:

- Ciphertext-ciphertext addition: ciphertext addition is done component-wise modulo q and returns ct_{add}.
- Plaintext-ciphertext addition: the plaintext is encrypted as follows:

$$ct' = \Big(\big[\lfloor \Delta \cdot m \rceil\big]_q,\ 0 \Big)$$

  After this stage, add the original ciphertext with ct' (ciphertext-ciphertext addition).
- Key switching: reduces the result of the ciphertext-ciphertext multiplication back to two components.
- Ciphertext-ciphertext multiplication: two ciphertexts $ct(c_0, c_1)$ and $ct' = (c_0', c_1')$ are multiplied as follows:

$$\mathbf{c}'' = \Big(\big[\lfloor \frac{c_0 \cdot c_0'}{\Delta} \rceil\big]_q,\ \big[\lfloor \frac{c_0 \cdot c_1' + c_1 \cdot c_0'}{\Delta} \rceil\big]_q \Big), \tag{2.9}$$

$$c_2'' = \Big[\lfloor \frac{c_1 \cdot c_1'}{\Delta} \rceil\Big]_q \tag{2.10}$$

  Since the c_2'' contains a second-order term in s, a relinearization is performed using the relinearization key, creating ct'''. This ciphertext is then added to ct''.
- Ciphertext-plaintext multiplication: takes the ciphertext and multiplies both parts with the flattened message m [4]

[4]Flattening involves reducing the coefficients from, ensuring the coefficients are reduced modulo t but expressed in $\mathcal{R}$. Following formula is used:

$$\text{Flatten} : \mathcal{R}_t \to \mathcal{R} : \quad m \mapsto t \cdot \big[\frac{m}{t}\big]_1$$

- Automorphism. Applying an automorphism to the ciphertext polynomials, this permutes slots in packed plaintexts after decryption. Then, the plaintext moduli are corrected if they changed under the automorphism. A key switching is performed to bring back the secret key to s. Finally, the adjusted ciphertexts are combined to form the final output.

2.5 SIMD

Smart and Vercauteren [24] noticed that it was possible to encode multiple elements in one plaintext, using the Chinese Remainder Theorem. This splitting in slots can allow SIMD operation - performing a single operation on multiple data.

BFV can support packing in slots and thus SIMD operations. However, doing this puts a restriction on the use of BFV. If p is the modulus of the plaintext $(\phi_m(b))$, the upper bound of the output noise will grow proportional to to the product of this factor and the sum of the upper bounds on the input noise. When one wants to have a high precision arithmetic, one chooses a high p, which will result in more output noise. This makes BFV SIMD-schemes impractical when performing precise arithmetic calculations, often needed in HE-applications. For instance, privacy-preserving machine learning uses moduli up to 80 bits.[12] Also, a higher value of p enables a higher packing density. The packing density, which is equal to the number of slots divided by the ring dimension, is equal to $1/d$. d is the multiplicative order of p modulo the cyclotomix index m. To achieve full packing, p needs to be larger then m. When using power-of-two cyclotomics, the number of slots will be upper bounded by $(p+1)/2$. To conclude, while a large value of p allows for greater packing density and more precise arithmetic operations, it simultaneously exacerbates noise growth.

GBFV works modulo a plaintext modulus polynomial $t(x)$. To enable SIMD computation we want this field to be isomorphic to a product of fields. To do this, we equal p to the smallest positive integer in $t\mathcal{R}$ where p is prime and $p \equiv 1 \mod 2n$. When p is prime, then $\mathbb{F}_p[x]$ is a principal ideal domain. Since $p \equiv 1 \mod 2n$, the multiplicative group $\mathbb{F}_p^\times$ contains an element of order $2n$, which guarantees that there exists a τ such that the $2n$-th root of τ is equal to 1 and the n-th root of τ is equal to -1. Choosing p in such way gives us base field encoding which is most commonly used to maximize the number of slots. We can then factor the plaintext ring $\mod p$: the ideal $(x_{n+1}, t(x))$ becomes $(\tau(x), p)$ where $\tau(x)$ is equal to the greatest common divisor of $(x_{n+1}, t(x))$. Thus the plaintext space becomes isomorphic to the product of fields, over a restricted set of roots of $\tau(x)$. We obtain SIMD slots where each slot corresponds to one of these roots of $\tau(x)$:

$$\mathbb{F}_p[x]/(\tau(x)) \;\cong\; \prod_{s \in S} \mathbb{F}_p. \tag{2.11}$$

Homomorphic operations will be performed component-wise on these slots. Rotations of the slots can be performed via automorphisms. One can also use p^e instead of p, which can increase plaintext correctness or reduce wraparound.

In GBFV, a non-standard slot ordering is applied when compared to the conventional BFV scheme which usually switched the roles of the generators that determine the position of the slots. In GBFV only a part of the slots is used, so that operations are handled more naturally. This slot orderings maps noise with one full bit reversal, similmar to a plaintext NTT (number theoretic transform) algorithm. Increasing the modulus or doubling the dimension of the ring will add slots or duplicate slot-vectors respectively, which is more convenient to describe compared to BFV. Finally, slot permutations during conversion reduce to circular bit shifts on the slot indices, easier than complex reversals in BFV [10].

2.6 Private information retrieval

When retrieving information from a remote server, the database holder will know which elements are queried. To protect the user, one wants to hide which elements are queried from the server. Private information retrieval or PIR is often considered to achieve this goal.

The goal of PIR is to ensure the server does not learn anything about the index from the user query. This will enhance the privacy of the user, since no information will be leaked to the (remote) server(s), potentially causing serious privacy issues. PIR finds application in multiple scenarios where sensitive data are used. For example, a doctor querying a database with patient's medical data will get back the requested medical information, without the server learning which patient or which patient record was requested. PIR also finds application in technology. Apple uses PIR to provide caller ID information of an incoming phone call, without them learning who is calling who [19].

PIR protocols can be categorized in two groups: single-server PIR and multi-server PIR. The single-server PIR is the most straightforward setting: one server holds the full dataset, and the client queries the server to get the data of interest. In multi-server PIR, there are multiple servers holding a copy of the full dataset, and the client queries multiple servers to obtain the data of interest. The core idea when using multi-server PIR is that, although the dataset is replicated on multiple servers, the query is split into parts. In this way, none of the servers learn which bit is requested but the requested bit can be recovered from the results of the different servers.

A multi-server PIR scheme is demonstrated in the following example. Let D be the database with 4 bits, and the client wants to retrieve bit b_3 privately. There are two non-colluding servers holding a copy of the database.

$$\textbf{Database: } D = [\, b_1,\, b_2,\, b_3,\, b_4 \,]$$

Step 1: Query generation

The client samples a random binary vector:

$$q_1 = [\, q_{11},\, q_{12},\, q_{13},\, q_{14} \,]$$

and constructs

$$q_2 = q_1 \oplus e_3$$

where e_3 is the unit vector with a one in the third position.

$$e_3 = [\,0,\,0,\,1,\,0\,]$$

The client sends q_1 to Server 1,

$$q_2 \text{ to Server 2.}$$

Step 2: Servers compute answers

Each server computes the XOR of all b's at positions where the query vector has a one:

$$a_1 = q_1 \cdot D = \bigoplus_{j=1}^{4} (q_{1j} \cdot b_j)$$

$$a_2 = q_2 \cdot D = \bigoplus_{j=1}^{4} (q_{2j} \cdot b_j)$$

Step 3: Client combines responses

$$a_1 \oplus a_2 = \left(\bigoplus_{j=1}^{4} (q_{1j} \cdot b_j) \right) \oplus \left(\bigoplus_{j=1}^{4} (q_{2j} \cdot b_j) \right) = b_3$$

In this way, the client learns b_3, and neither server learns which b_i was requested.

Security holds as long as the servers do not collude. This assumption is difficult to achieve, because the database is usally under one authority and distributing the database on multiple servers with different authorities is mostly not feasible in practice. Therefore, single-server PIR schemes are often preferred since they rely on cryptographic hardness assumptions, but at the cost of incurring a performance overhead. In this thesis, we will further focus on single-server PIR.

A scheme is information-theoretic secure when the queries asked by the user give no information whatsoever about the requested bit to the server. Multi-server PIR schemes can achieve information-theoretic security, such as the PIR-protocol proposed by Ghoshal et al. [11]. Single-server PIR schemes can not achieve information-theoretic security. One exception, when the single server sends the full database to the client. The client can then query the database and the server will not learn anything about the requested bit. However, this trivial scheme has a communication cost of $\mathcal{O}(n)$, with n the size of the database. Single-server PIR schemes with smaller communication cost are only computationally secure; computationally secure schemes only guarantee that the server can not compute the requested bit in a reasonable amount of time, given the queries. Kushilevitz and Ostrovsky made use of the number-theoretic assumption to deduce a single-server computationally secure PIR with subpolynomial communication cost [17]. The scheme has a communication complexity of $\mathcal{O}(n^\epsilon)$ for any $\epsilon > 0$. This scheme however requires n big integer multiplications. Cachin et al. proposed a two-round computationally secure PIR using the ϕ-hiding assumption where communication complexity is polylogarithmic in n. This scheme requires n modular exponentiations, with large moduli, which makes multiplication slower then in Kuhilevitz's scheme [8]. Chang subsequently proposed a scheme with logarithmic communication complexity, using Paillier's cryptosystem [22, 6]. In 2007, Sion and Carbunar pointed out that these single-server PIR protocols are mostly orders of magnitude slower than the trivial transfer of the entire database to the client [23]. Later work by Aguilar-Melchor et al. however showed that this argument is incorrect: single-server PIR can be faster than downloading the entire database, when using lattice-based cryptographic methods. These methods have smaller per-bit computation cost when used in a batched fashion [1].

More recent PIR protocols make use of fully homomorphic encryption. FHE typically incurs significant communication overhead due to the ciphertext expansion factor. However, keeping the query size as low as possible while maintaining computation cost reasonable is the objective in these protocols. Arranging the database as a hypercube will increase the computation efficiency, as used in Respire [5]. Furthermore, transciphering can be used to further lower the query size. In transciphering, the client will use a symmetric encryption scheme to encrypt the query, which is then homomorphically evaluated on the server side. The server will homomorphically decrypt the query and evaluate it on the database, returning the encrypted result to the client. The client will then decrypt the result symmetrically. This method reduces communication cost since FHE ciphertexts are only used for the query and result, while symmetric encryption is used for the database. Kang proposed a novel transciphering method to further reduce the communication cost when compared to (T-)Respire [4, 3]. In this scheme, the client transmits only one part of the LWE ciphertext. The full LWE ciphertext is reconstructed using a pseudo random generator seed shared between the client and server. By sending only a single LWE component and a short seed, Pirouette succesfully achieves query compression while keeping computational cost associated with transciphering reasonable. [15].

2.7 PIRANA

PIRANA is a single-server protocol developed at Zhejiang University [18]. The protocol is based on constant-weight codes, which is a way to encode the queries. In constant-weight codes, all codewords have a length m and have the same Hamming weight, meaning they have the same number of ones. In later steps, it will be clear how the database is structured and how information is retrieved. There should at least be the same amount of codewords as there are columns in the database. To estimate the length of the codeword in bits, we need to know the number of columns and the Hamming weight k. The number of codewords is equal to the binomial coefficient $\binom{m}{k}$. To estimate the code length m, knowing the Hamming weight k and the number of columns n, we can then use following formula:

$$m \in O\left(\sqrt[k]{k!\,n} + k\right) \tag{2.12}$$

According to the Mahdavi-Kerschbaum mapping method, every index i of a column is mapped to the i-th codeword.

PIRANA can be used in single-query and multi-query set-up. In the following part, single-query PIRANA will be discussed for small and large payloads, as well as a comparison of PIRANA with constant-weight PIR (cwPIR).

2.7.1 Single-query PIRANA for small payloads

In single-query PIRANA for small payloads, the client wants to retrieve a single element from the database. Small payloads means the elements in the database are smaller then the plaintext modulo p. So multiple elements can be packed in one ciphertext. The database of n elements is structured as a 2D-matrix with r rows and c columns, where $n = r \cdot c$ and r is the number of slots in a ciphertext. Every column is represented by one codeword of length m and Hamming weight k. To determine m and k, one wants to find the smallest m such that $\binom{m}{k} \geq c$. To retrieve an element in position (i, j), the client constructs a query made out of m ciphertexts, with r ciphertext slots, thereby constructing a matrix of size $m \times r$. This is a all-zero matrix, except for the i-th row, which contains the codeword corresponding to column j. This query is then sent to the server. The sever will receive this query and, for every column, will take the corresponding codeword. For every bit equal in the codeword, the server will take the corresponding column from the query and perform $k - 1$ homomorpic multiplications. The server will do this for all columns, thus $k - 1 * n$ ciphertext-ciphertext multiplications. Hereby creating a selection matrix of size $r \times c$. All values in these matrix are zero, except for the position (i, j), which contains the one. Every column is a ciphertext. The server will then perform a ciphertext-plaintext multiplication between the columns of the selection matrix and the columns of the database. This can be done because the dimensions of both matrices match. Finally, the server will sum up all columns, returning a ciphertext with r slots, where all slots are zero, except for slot i, which contains the requested element. This ciphertext is sent back to the client, who will decrypt and get the requested element.

This protocol has some flaws. First of all, if ns is large, the amount of ciphertext-ciphertext multiplications becomes huge. Secondly, the communication cost to retrieve one element is high, since the client needs to send m ciphertexts to the server, and the server will return one ciphertext with r slots. Lastly, when choosing a large k, the amount of ciphertext-ciphertext multiplications increase and m increases too, which will increase the communication cost.

2.7.2 Single-query PIRANA for large payloads

PIRANA can also be used for large payloads, i.e. elements that are bigger than the plaintext modulus p. Every element is split into multiple chunks, each chunk smaller than p. The database is now a 3D-matrix of size $r \times c \times l$, where l is the number of layers, equal to the number of chunks per element. The selection matrix is created in the same way as for small payloads, resulting in a matrix of size $r \times c$. In the next step every column of the selection matrix is multiplied not with one column but with l columns of the database. The results of this multiplication give a 2D-matrix of size $r \times l$. Every column is a ciphertext with all zeros, except at position i, which contains one chunk of the requested element. To reduce the amount of ciphertexts to send back to the client, the algorithm will perform a rotate-and-sum operation. In this operation, the first column/ciphertext is rotated by one position, and the second column is added to it. This sum is then rotated by one position again, and added with the third column. This is repeated untill all columns are added. If there are more chunks then slots in a ciphertext, a new ciphertexts are needed to retrieve the remaining chunks. The final ciphertexts are sent back to the client, which will decrypt and can reconstruct the requested element, by combining the chunks.

For small database size n, one could pre-compute the rotations in the set-up time of the database. This will reduce the computation time when performing a query to the database.

2.7.3 PIRANA performance comparison

Liu et al. compared the performance of PIRANA with constant-weight PIR. PIRANA was implemented in C++ based on Microsoft SEAL HE library [5], and the BFV scheme was used with $N \in \{4096, 8192\}$. Tests are performed on an Intel Xeon Cooper Lake with a base frequency of 3.4 GHz and turbo frequency of 3.8 GHz. The server was running on Ubuntu 20.04. This set-up is done similar to the set-up of cwPIR [2]. PIRANA outperforms cwPIR in terms of selection vector generation time. As expected: in PIRANA $c \cdot (k-1)$ ciphertext-ciphertext multiplications are needed, while in cwPIR $n \cdot (k-1)$ multiplications are needed. When the database size n increases, the difference in performance becomes larger. Inner product calculation is also faster in PIRANA when compared to cwPIR. In cwPIR, every ciphertext needs to be transformed using NTT (number theoretic transform) before multiplying with the database. In PIRANA, there are only m ciphertexts to transform, which is r times smaller than n. The query size of PIRANA is up to 2.5 times larger when compared to cwPIR, and the response size is equivalent. Thus, commmunication cost is higher in PIRANA. But, one can query $\lfloor \frac{N}{1.5} \rfloor$ elements for the same communication cost in PIRANA (multi-query).

PIRANA was also compared to some state-of-the-art PIR schemes by Liu et al. [18]. To answer a single query, PIRANA is mostly slower than other PIR schemes. However, PIRANA becomes more competitive when the number of queries increases.

2.8 Fheanor

In this thesis, we will implement GBFV in the Fheanor library[6]. Feanor is a Rust library containing building blocks for homomorphic encryption, implementing sev-

[5] https://github.com/microsoft/SEAL

[6] https://github.com/FeanorTheElf/fheanor

eral FHE schemes, including BFV and CLPX[7]. Fheanor is build on feanor-math[8], a Rust library for number theory and algebra. Both libraries are open-source and can be found on GitHubu.

The library supports implementations over both power-of-two and genaral cyclotomics [21]. This is interesting for FHE implementations, in particular because the use of non-power-of-two cyclotomics can allow greater SIMD capabilities by having a larger number of slots with small plaintext moduli. Feanor also explicitly models arithmetic circuits, providing tools for their computation. The Fheanor library is close in performance to the HELib and SEAL libraries, which are state-of-the-art.

Spiessens created a wrapper called easy-GBFV, which will use the Fheanor library to create a GBFV-scheme. This wrapper offers some easy-to-use functions. Some functions that will be used in this GBFV implementation are:

- get_gbfv_(16/32/64)bit(): creates a GBFV scheme with plaintext modulo 16, 32, or 64 bits.
- pack(): to get an amount of slots in a ciphertext.
- slot_ring(): hands the canonical slot ring instance that the hypercube uses for plaintext packing.
- gen_sk(): generates a secret key for the GBFV scheme.
- gen_pk(&SecretKey): generates a public key for the GBFV scheme.
- enc_slots(): will take the output of slot_ring() and encrypt it.
- dec_slots(): will decrypt a ciphertext and return the slots.
- clone_ct(): clones an element of the ciphertext.
- hom_mul(): performs homomorphic multiplication between two ciphertexts.
- hom_rotate(): will rotate the slots in the ciphertext.
- hom_matmaul(): performs homomorphic matrix multiplication between a ciphertext and a plaintext matrix.
- hom_add(): performs homomorphic addition between two ciphertexts.

[7]CLPX is not implemented in any other major library [21]

[8]https://github.com/FeanorTheElf/feanor-math

Chapter 3

PIR implementation

3.1 GBFV-PIRANA, single-query small payload

In this section, the implementation of the PIRANA protocol using the easy-GBFV library is presented. The implementation is a single-query implementation for small payloads. This means that the elements of the database are smaller when compared to the plaintext modulus p. To test the implementation, client and server are created in the same file. Therefore, there is no need to send the public key from client to server. The single-query small payload implementation can be found in the `examples\5_GBFV_PIRANA_Spayload` folder of the thesis github [1].

First, a database/matrix is created with size $r \cdot c$, where r is equal to the number of slots in the ciphertext and c is equal to the amount of elements divided by the number of slots in the ciphertext [2]. All indices of the columns of the matrix are substituted with a constant weight codeword. To achieve this, m and k have to be chosen properly. In this implementation, k will be set to 2, meaning that every codeword has a Hamming weight of 2. This will keep the amount of ciphertext-ciphertext multiplications low. Knowing k, m can be calculated as $c \leq \binom{m}{2}$, with c the amount of columns in the matrix. Later, every column will be multiplied with a ciphertext. Therefore, the plaintext elements of one column are set into a plaintext ring.

Subsequently, an instance of GBFV is created. When creating this instance, one has to set several attributes. One has to choose $m1$ in such a way that m, the cyclotomic order, can be divided by $m1$, to create the number of slots $m2$. A second attribute is $log2(N)$, which will determine the cyclotomic order m as $m = 2^{log2(N)+1}$. A prime number p_{mod} is a prime which will determine the plaintext modulus. Attribute t, which is the plaintext modulus, also has to be set. In our case, we use BGFV, so t is a polynomial. After creating the GBFV instance, the slot_ring() function will create a slot representation.

Having a database and having created a GBFV instance, the PIRANA set-up is

[1]github implementation of antoine

[2]When working with small payloads, all elements in the database are of maximal size i32 or plaintext modulo.

finished. The client can now create a query for an element in the database. Imagine the client wants to retrieve element (i, j) from the database. First, the client will look up which codeword corresponds to column j. The client will create the query matrix, which is a matrix of size $r \cdot m$. This matrix is an all-zero matrix, except for the i-th row, which is subsituted with the codeword.

Before sending the query, the client has to encrypt the query. Therefore, he generates a secret key and a public key. Every column of length r (amount of slots in a ciphertext) will be encrypted. The client sends m ciphertexts to the server.

The server will, for each column, take the codewords of length m and look at which position the codeword has a 1. In our case, there are only two one's (remember, the Hamming weight equals 2). The server will take the corresponding ciphertexts of these two positions in the query and multiply them with each other. This new ciphertext is one column of the selection matrix. This process is repeated for all c columns of the database. After creating the selection matrix, the server will perform a homomorphic plaintext-ciphertext multiplication between every column of the selection matrix and the corresponding column of the database. Finally, all the columns of the resulting matrix are summed together, by going through all the columns and adding them via an accumulator. The result is then sent back as one ciphertext to the client.

The client will receive the ciphertext from the server and will decrypt using his secret key. Every ciphertext is decrypted and will return a vector of slot ring elements. All elements are equal to zero, except for the i-th element, which is equal to the desired element in the database. The client can now format this element and retrieve the desired value.

3.2 GBFV-PIRANA, single-query large payload

In this section, the implementation of a single-query PIR protocol for large payloads using the easy-GBFV library is presented. The implementation can be found in the `examples\5_GBFV_PIR_Lpayload` folder of the thesis github [3].

- d3_finder(element_size_bit: usize, p_mod: &str): This function will determine in how many chunks a large payload can be split. It will return the amount of chunks. To achieve this, the function needs to know the plaintext modulus p_mod and the maximal size of an element in the database (in bits).

  Equation 3.1 shows how the number of chunks are calculated.

$$numberofchunks = \left\lceil \frac{element\_size\_bit}{\lfloor log_2(p\_mod) \rfloor} \right\rceil \tag{3.1}$$

- base_p_decompose(n,p, chunks): This function will do a base-p decomposition of a big integer n into *chunks* amount of chunks. Euclidian division of the integer n by the plaintext modulus p is used. The division will be done *chunks*

[3] github implementation of antoine

amount of times, and every chunk will keep the remainder of the division. This will create a vector of size *chunks*, containing numbers smaller than p.

- recompose_base_p_to_str(digits, p): This function will recompose the chunks back into one large integer. It will take the vector of chunks and the plaintext modulus p. The recomposition is done by multiplying every chunk with p^i, with i the index of the chunk in the vector. The results are summed together to create one large integer, which is then converted to a string and returned.

$$n = \sum_{i=0}^{chunks-1} digits[i] \cdot p^i \tag{3.2}$$

- `get_rand_matrix(nr_elements, element_size_bits, nr_slots, p)`: To create the database, this function is used. It will create a 3D-matrix with size $r \cdot c \cdot d3$, with r the number of slots in a ciphertext, c the number of elements divided by the number of slots, and $d3$ the amount of chunks needed to split one large element. Every element in the database is a large integer with size equal to *element_size_bits*. Every large integer is split into $d3$ chunks via base-p decomposition. This function will return the 3D-matrix.

Chapter 4

The Final Chapter

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

4.1 The First Topic of this Chapter

4.1.1 Item 1

Sub-item 1

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

Sub-item 2

Proin non sem. Donec nec erat. Proin libero. Aliquam viverra arcu. Donec vitae purus. Donec felis mi, semper id, scelerisque porta, sollicitudin sed, turpis. Nulla in urna. Integer varius wisi non elit. Etiam nec sem. Mauris consequat, risus nec congue condimentum, ligula ligula suscipit urna, vitae porta odio erat quis sapien. Proin luctus leo id erat. Etiam massa metus, accumsan pellentesque, sagittis sit amet, venenatis nec, mauris. Praesent urna eros, ornare nec, vulputate eget, cursus sed, justo. Phasellus nec lorem. Nullam ligula ligula, mollis sit amet, faucibus vel, eleifend ac, dui. Aliquam erat volutpat.

4.1.2 Item 2

Fusce vehicula, tortor et gravida porttitor, metus nibh congue lorem, ut tempus purus mauris a pede. Integer tincidunt orci sit amet turpis. Aenean a metus. Aliquam vestibulum lobortis felis. Donec gravida. Sed sed urna. Mauris et orci. Integer ultrices feugiat ligula. Sed dignissim nibh a massa. Donec orci dui, tempor sed, tincidunt nonummy, viverra sit amet, turpis. Quisque lobortis. Proin venenatis tortor nec wisi. Vestibulum placerat. In hac habitasse platea dictumst. Aliquam porta mi quis risus. Donec sagittis luctus diam. Nam ipsum elit, imperdiet vitae, faucibus nec, fringilla eget, leo. Etiam quis dolor in sapien porttitor imperdiet.

4.2 The Second Topic

Cras pretium. Nulla malesuada ipsum ut libero. Suspendisse gravida hendrerit tellus. Maecenas quis lacus. Morbi fringilla. Vestibulum odio turpis, tempor vitae, scelerisque a, dictum non, massa. Praesent erat felis, porta sit amet, condimentum sit amet, placerat et, turpis. Praesent placerat lacus a enim. Vestibulum non eros. Ut congue. Donec tristique varius tortor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nam dictum dictum urna.

Phasellus vestibulum orci vel mauris. Fusce quam leo, adipiscing ac, pulvinar eget, molestie sit amet, erat. Sed diam. Suspendisse eros leo, tempus eget, dapibus sit amet, tempus eu, arcu. Vestibulum wisi metus, dapibus vel, luctus sit amet, condimentum quis, leo. Suspendisse molestie. Duis in ante. Ut sodales sem sit amet mauris. Suspendisse ornare pretium orci. Fusce tristique enim eget mi. Vestibulum eros elit, gravida ac, pharetra sed, lobortis in, massa. Proin at dolor. Duis accumsan accumsan pede. Nullam blandit elit in magna lacinia hendrerit. Ut nonummy luctus eros. Fusce eget tortor.

Ut sit amet magna. Cras a ligula eu urna dignissim viverra. Nullam tempor leo porta ipsum. Praesent purus. Nullam consequat. Mauris dictum sagittis dui. Vestibulum sollicitudin consectetuer wisi. In sit amet diam. Nullam malesuada pharetra risus. Proin lacus arcu, eleifend sed, vehicula at, congue sit amet, sem. Sed sagittis pede a nisl. Sed tincidunt odio a pede. Sed dui. Nam eu enim. Aliquam sagittis lacus eget libero. Pellentesque diam sem, sagittis molestie, tristique et, fermentum ornare, nibh. Nulla et tellus non felis imperdiet mattis. Aliquam erat volutpat.

4.3 Conclusion

Vestibulum sodales ipsum id augue. Integer ipsum pede, convallis sit amet, tristique vitae, tempor ut, nunc. Nam non ligula non lorem convallis hendrerit. Maecenas hendrerit. Sed magna odio, aliquam imperdiet, porta ac, aliquet eget, mi. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vestibulum nisl sem, dignissim vel, euismod quis, egestas ut, orci. Nunc vitae risus vel metus euismod laoreet. Cras sit amet neque a turpis lobortis auctor. Sed aliquam sem ac elit. Cras velit lectus, facilisis id, dictum sed, porta rutrum, nisl. Nam hendrerit ipsum sed augue. Nullam scelerisque hendrerit wisi. Vivamus egestas arcu sed purus. Ut ornare lectus sed eros. Suspendisse potenti. Mauris sollicitudin pede vel velit. In hac habitasse platea dictumst.

Suspendisse erat mauris, nonummy eget, pretium eget, consequat vel, justo. Pellentesque consectetuer erat sed lacus. Nullam egestas nulla ac dui. Donec cursus rhoncus ipsum. Nunc et sem eu magna egestas malesuada. Vivamus dictum massa at dolor. Morbi est nulla, faucibus ac, posuere in, interdum ut, sapien. Proin consectetuer pretium urna. Donec sit amet nibh nec purus dignissim mattis. Phasellus vehicula elit at lacus. Nulla facilisi. Cras ut arcu. Sed consectetuer. Integer tristique elit quis felis consectetuer eleifend. Cras et lectus.

Ut congue malesuada justo. Curabitur congue, felis at hendrerit faucibus, mauris lacus porttitor pede, nec aliquam turpis diam feugiat arcu. Nullam rhoncus ipsum at risus. Vestibulum a dolor sed dolor fermentum vulputate. Sed nec ipsum dapibus urna bibendum lobortis. Vestibulum elit. Nam ligula arcu, volutpat eget, lacinia eu, lobortis ac, urna. Nam mollis ultrices nulla. Cras vulputate. Suspendisse at risus at metus pulvinar malesuada. Nullam lacus. Aliquam tempus magna. Aliquam ut purus. Proin tellus.

Appendices

Appendix A

The First Appendix

Appendices hold useful data which is not essential to understand the work done in the master's thesis. An example is a (program) source. An appendix can also have sections as well as figures and references.

A.1 More Lorem

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

A.1.1 Lorem 15–17

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi.

In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

A.1.2 Lorem 18–19

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

A.2 Lorem 51

Maecenas dui. Aliquam volutpat auctor lorem. Cras placerat est vitae lectus. Curabitur massa lectus, rutrum euismod, dignissim ut, dapibus a, odio. Ut eros erat, vulputate ut, interdum non, porta eu, erat. Cras fermentum, felis in porta congue, velit leo facilisis odio, vitae consectetuer lorem quam vitae orci. Sed ultrices, pede eu placerat auctor, ante ligula rutrum tellus, vel posuere nibh lacus nec nibh. Maecenas laoreet dolor at enim. Donec molestie dolor nec metus. Vestibulum libero. Sed quis erat. Sed tristique. Duis pede leo, fermentum quis, consectetuer eget, vulputate sit amet, erat.

Appendix B

The Last Appendix

Appendices are numbered with letters, but the sections and subsections use arabic numerals, as can be seen below.

B.1 Lorem 20-24

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

B.2 Lorem 25-27

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

Duis aliquet dui in est. Donec eget est. Nunc lectus odio, varius at, fermentum in, accumsan non, enim. Aliquam erat volutpat. Proin sit amet nulla ut eros consectetuer cursus. Phasellus dapibus aliquam justo. Nunc laoreet. Donec consequat placerat magna. Duis pretium tincidunt justo. Sed sollicitudin vestibulum quam. Nam quis ligula. Vivamus at metus. Etiam imperdiet imperdiet pede. Aenean turpis. Fusce augue velit, scelerisque sollicitudin, dictum vitae, tempor et, pede. Donec wisi sapien, feugiat in, fermentum ut, sollicitudin adipiscing, metus.

Donec vel nibh ut felis consectetuer laoreet. Donec pede. Sed id quam id wisi laoreet suscipit. Nulla lectus dolor, aliquam ac, fringilla eget, mollis ut, orci. In pellentesque justo in ligula. Maecenas turpis. Donec eleifend leo at felis tincidunt consequat. Aenean turpis metus, malesuada sed, condimentum sit amet, auctor a, wisi. Pellentesque sapien elit, bibendum ac, posuere et, congue eu, felis. Vestibulum mattis libero quis metus scelerisque ultrices. Sed purus.

Bibliography

[1] C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian. XPIR: Private information retrieval for everyone. Cryptology ePrint Archive, Paper 2014/1025, 2014.

[2] R. Akhavan Mahdavi and F. Kerschbaum. Constant-weight pir: Single-round keyword pir via constant-weight equality operators. In *Proceedings of the 31st USENIX Security Symposium*, page 1723–1740. USENIX Association, 2022.

[3] S. Belaïd, N. Bon, A. Boudguiga, R. Sirdey, D. Trama, and N. Ye. Further improvements in AES execution over TFHE: Towards breaking the 1 sec barrier. Cryptology ePrint Archive, Paper 2025/075, 2025.

[4] N. Bon, D. Pointcheval, and M. Rivain. Optimized homomorphic evaluation of boolean functions. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(3):302–341, Jul. 2024.

[5] A. Burton, S. J. Menon, and D. J. Wu. Respire: High-rate PIR for databases with small records. Cryptology ePrint Archive, Paper 2024/1165, 2024.

[6] Y.-C. Chang. Single database private information retrieval with logarithmic communication. In *Information Security and Privacy*, pages 50–61, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[7] H. Chen, K. Laine, R. Player, and Y. Xia. High-precision arithmetic in homomorphic encryption. In *Topics in Cryptology – CT-RSA 2018*, Lecture Notes in Computer Science, pages 116–136. Springer, Mar. 2018. International Conference on Cryptographers Track at the RSA Conference on Topics in Cryptology, CT-RSA 2018 ; Conference date: 16-04-2018 Through 20-04-2018.

[8] C. Christian, M. Silviio, and S. Markus. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 202–214, Prague, Czech Republic, 1999. Springer.

[9] R. Geelen and F. Vercauteren. Bootstrapping for bgv and bfv revisited. *Journal of Cryptology*, 36(12), 2024.

[10] R. Geelen and F. Vercauteren. Better GBFV bootstrapping and faster encrypted edit distance computation. Cryptology ePrint Archive, Paper 2025/1104, 2025.

[11] A. Ghoshal, B. Li, Y. Ma, C. Dai, and E. Shi. Scalable multi-server private information retrieval. Cryptology ePrint Archive, Paper 2024/765, 2024.

[12] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210, New York, USA, 20–22 Jun 2016. PMLR.

[13] Ilaria Chillotti. TFHE Deep Dive - Ilaria Chillotti, FHE.org, Aug. 2022. YouTube video.

[14] N. J. Bouman. Comparison of Privacy Enhancing Technologies and MPC, Aug. 2024.

[15] J. Kang and L. Schild. Pirouette: Query efficient single-server PIR. Cryptology ePrint Archive, Paper 2025/680, 2025.

[16] A. Kim, Y. Polyakov, and V. Zucca. Revisiting homomorphic encryption schemes for finite fields. In *Lecture Notes in Computer Science*, volume 1309, pages 608–639. Springer, 2021.

[17] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 364–373, 1997.

[18] J. Liu, J. Li, D. Wu, and K. Ren. PIRANA: Faster multi-query PIR via constant-weight codes. Cryptology ePrint Archive, Paper 2022/1401, 2022.

[19] Machine Learning Research. Combining machine learning and homomorphic encryption in the apple ecosystem. URL: `https://machinelearning.apple.com/research/homomorphic-encryption`, last checked on 2025-27-10.

[20] U. Mattsson. Security and Performance of Homomorphic Encryption, Apr. 2025.

[21] H. Okada, R. Player, and S. Pohmann. Fheanor: a new, modular FHE library for designing and optimising schemes. Cryptology ePrint Archive, Paper 2025/864, 2025.

[22] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[23] R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, pages 364–373, San Diego, California, USA, 2007. Internet Society (ISOC).

[24] N. P. Smart and F. Vercauteren. Fully homomorphic simd operations. *Designs, Codes and Cryptography*, 71:57 – 81, 2012.