DTU Compute
Department of Applied Mathematics and Computer Science

Master Thesis

Benchmarking Post-Quantum Secure Biometric Resilient Authenticated Key Exchange

Matej Poljuha

Benchmarking Post-Quantum Secure Biometric Resilient Authenticated Key Exchange

Master Thesis
February, 2023

By
Matej Poljuha

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark
https://www.compute.dtu.dk/english
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science of the Technical University of Denmark and the Department of Information Security and Communication Technology at the Norwegian University of Science and Technology, in fulfillment of the requirements for acquiring the Master of Science degree in science and engineering at the Department of Applied Mathematics and Computer Science of the Technical University of Denmark. It was supervised by Professor Christian D. Jensen, Pia Bauspieß and Professor Tjerand Silde.

I declare that the work described in this thesis is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Matej Poljuha - s202482

.....

Signature

28.02.2023.

.....

Date

Abstract

In contrast to traditional authentication methods such as passwords or cryptographic keys, user authentication based on biometric characteristics has been shown to be feasible and secure. However biometric data are a non-replenishable unique resource for the user and present a persistent target for attacks. Therefore, biometric authentication protocols need to be constructed using post-quantum secure cryptographic primitives.

This work presents a post-quantum secure implementation of a protocol for Biometric Resilient Authenticated Key Exchange (PQ-BRAKE). It is based on the previously proposed classically secure BRAKE protocol which builds on the security of discrete logarithms. In this thesis, the cryptographic primitives, namely the Oblivious Pseudo-Random Function (OPRF) and the Key Encapsulation Mechanism (KEM), are replaced by lattice-based constructions.

The work presents benchmarking results both for communication and computation cost of the designed protocol and compares these results against the classically secure protocol. The experimental evaluation shows that the computational overhead of PQ-BRAKE is bound by 10% in the optimal parameter setting. However, the communication cost increase is found to be more significant but can still be considered reasonable for real-time applications.

Contents

Preface ii
Abstract iii

1 Introduction **1**
1.1 Contribution 2
1.2 Related Work 3
1.3 Outline 4

2 Background **5**
2.1 Lattices 5
2.2 Lattice-Based Cryptography 6
2.3 Oblivious Pseudo-Random Function (OPRF) 10
2.4 Biometric Resilient Authenticated Key Exchange 13

3 Post-Quantum Secure Biometric Resilient Authenticated Key Exchange (PQ-BRAKE) **14**
3.1 Lattice-Based Oblivious Pseudo-Random Function 15
3.2 Adaptation of the Lattice-based Oblivious Pseudo-Random Function (OPRF) 18
3.3 Key Encapsulation Mechanism (KEM) 21
3.4 Lattice-Based BRAKE 22

4 Implementation **26**
4.1 System Setup 26
4.2 NTL 27
4.3 Lattice-OPRF 27
4.4 Lattice-based KEM 31

5 Performance **32**
5.1 Passive Security 32
5.2 Benchmarking Results 34
5.3 Comparison With Classically Secure Implementation 37

6 Conclusion **39**

List of Figures

2.1 The Short Integer Solution Problem visualized. 7
2.2 q-ary integer lattices in 2D geometric representation (illustration from lectures based on [Pei16]). 8
2.3 Blinded exponentiation for evaluating the Hashed Diffie-Hellman PRF [CHL22]. 12

3.1 Lattice-Based Verifiable OPRF (VOPRF) as presented in [Alb+21b]. 17
3.2 Modified OPRF protocol using the truncated PRF. 19
3.3 PQ-BRAKE enrolment protocol. 23
3.4 PQ-BRAKE verification protocol. 25

List of Tables

5.1 Kyber768 Key Sizes. 34
5.2 Computational Performance of PQ-BRAKE Scenario 1. 35
5.3 Computational Performance of PQ-BRAKE Scenario 2. 35
5.4 Computational Performance of PQ-BRAKE Scenario 3. 35
5.5 Kyber768 Parameter Set [Ava+21]. 36
5.6 PQ-BRAKE Verification Performance in Milliseconds (Extending Upon [Bau+22]). 37
5.7 PQ-BRAKE Verification Performance Penalty in Milliseconds, Relative to Classically Secure BRAKE [Bau+22]. 38
5.8 PQ-BRAKE Communication Costs. 38

Chapter 1

Introduction

User authentication is an essential concept in today's realm of digital communication. It is most commonly accomplished through the use of passwords or cryptographic keys. However, these methods have significant and thoroughly explored drawbacks. For instance, such risks arise from the use of short passwords, passwords composed of common and easily guessable words, reusing passwords across different services, passwords derived from the user's interests or publicly available information like the user's favourite book or date of their child's birth. Therefore, passwords are often considered the weakest link in a password-based authentication system [THB14].

Using cryptographic keys instead of passwords alleviates some of these challenges, but at the cost of usability from the average user's perspective. If the keys are stored as files, the user needs to be actively aware and careful of where and how to store and use these keys. Furthermore, it can be considered impossible for the average user to memorise even a single key as they are usually significantly longer than a password and drawn from a random distribution.

An alternative to these methods is to base authentication on the biometric characteristics of the user. Immediately, it can be observed that this approach solves the issue of remembering a password or cryptographic key, as biometric characteristics are inherent to the user and always present. Additionaly, a significant advantage is that it can also improve upon traditional key-based authentication by having the user's secret key always be derived from a fresh biometric capture and thus removes the need for it to be stored and secured on the client's machine.

However, using biometric characteristics presents its own set of challenges and potential security risks. First among these is the importance and vulnerability of the biometric data itself. They are a non-replenishable unique resource for the user, as a compromised biometric template can identify a person for, potentially, the duration of their lifetime [KHB21]. If a biometric template is compromised and the underlying biometric features are deduced, a user cannot use that biometric instance anymore and is faced with the need to revoke it. The European Union's General Data Protection Regulation (GDPR) [Eur16] recognizes this issue and classifies biometric data as sensitive personal data.

Additionally, standardised requirements for biometric systems have been introduced to guard this sensitive data in the ISO/IEC 24745 International Standard on Biometric Information Protection [ISO22]. This standard emphasizes three requirements that a secure biometric system must fulfill in order to be compliant:

1. For two or more biometric templates: *unlinkability* and *renewability* - meaning these templates cannot be linked to each other or to the subjects from whom they were derived [ISO22].
2. For every biometric template: *irreversibility* - meaning that no information can be learned from a biometric sample about the protected biometric reference it was generated from [ISO22].
3. For the system in general: *performance preservation* - meaning that the computational performance and recognition accuracy of the biometric system must not be significantly affected by protecting the generative biometric data [ISO22; Bau+22].

An additional challenge that needs to be addressed is the low entropy of biometric data. Any protocol that aims to construct cryptographic keys based on such data needs to guarantee a reliable level of security.

As a solution to the aforementioned challenges, a protocol for cryptographic key exchange based on successful biometric authentication which is in compliance with the ISO/IEC 24745 standard has been recently proposed by Bauspieß et al. [Bau+22]. It provides biometric authentication with a key exchange mechanism in which the client does not have to remember or handle any cryptographic key material or password and is built on established and currently trusted cryptographic components. Yet, the key material is not derived from the biometric features directly, but is truly random and only issued to the user in the case of a positive comparison outcome with the stored reference. It accomplishes this with real-time efficiency while preserving high biometric performance.

1.1 Contribution

The goal of this thesis is to adapt this protocol proposed in [Bau+22] and evaluate if it still retains its performance and security features in a post-quantum computing setting. Such an exploration is necessary due to the risk of attacks on the cryptographic components of the classically secure communication protocol [Bau+22] when a large-scale quantum computer becomes available. Due to the lifespan of biometric templates [KHB21], this level of protection is required for biometric authentication systems as of today.

An outline of such potential modifications to the BRAKE protocol to achieve post-quantum security has been suggested in the original work by [Bau+22] and the implementation in this thesis has largely followed these suggestions. Specifically, the adaption consists of two main elements of the protocol whose security needs to be upgraded to post-quantum security using lattice-based cryptography. These two components are the Oblivious Pseudo-Random Function (OPRF) and Key Encapsulation Mechanism (KEM).

The role of the OPRF element is to provide key material in cooperation with another entity while retaining the protection of the biometric data through the oblivious property of an OPRF. At the same time, the client requesting this key material is not able to obtain any information about the other entity's secret evaluation key. The OPRF construction applied in this thesis is based on a work by Albrecht et al. [Alb+21b], which is the only lattice-based OPRF protocol known at the time of writing. An implementation of this proposed construction does currently not exist due to the expected large communication overhead. Therefore, this thesis presents a modification on the original protocol to improve upon the communication and computation performance, while operating under a weaker security assumption of passive instead of active security. This modification aims to bring the communication performance to levels practical for use in an authentication protocol such as the one in [Bau+22].

The KEM aims to be a direct replacement for the classically secure Diffie-Hellman key exchange which is implemented in [Bau+22]. Specifically used is the recently standardised CRYSTALS-Kyber [Ava+21], for which a reference implementation already exists.

1.2 Related Work

Biometric verification is based on similarity scores, scores that quantify how similar one biometric template is to another. This score can be used in biometric authentication protocols in such a way that the correct key is only given if it the similarity score high enough, i.e., if the biometric probe is similar enough to the biometric reference. This is considered a successful biometric authentication.

Generally, in order to use biometric data in cryptographic schemes, a scheme which tolerates high variance due to outside factors (such as aging and variable image quality of captured samples) must be used. Such a scheme was originally proposed in [JS06] which introduced the idea of fuzzy vaults, based on a fuzzy commitment scheme previously introduced in [JW99]. This idea is the basis of many protocols which require storage of biometric reference data, including the BRAKE protocol [Bau+22] whose fuzzy vault scheme is defined in [Tam16] and can be instantiated with fingerprint, face and iris fuzzy vault schemes. This scheme specifically addresses a long-standing vulnerability of fuzzy vaults, the insecurity against offline attacks between multiple obtained vaults.

Other works concerning key exchange protocols using biometric authentication exist, such as the paper [Wan+21] where the authors considered a different approach, through an existing proposal for a Fuzzy Asymmetric Password-Authenticated Key Exchange (fuzzy aPAKE) [Erw+20]. They describe multiple issues when considering it for effective biometric authentication in their considered use case of secure messaging applications. The identified issues include low performance due to needing to run oblivious transfers on a per-bit basis [Bau+22] and the requirement for the biometric representation to be rotation-invariant. To resolve these issues, [Wan+21] propose their own mechanism. However, its implementation has flaws in its biometric comparison systems and vulnerability of derived public keys to offline-attacks as noted in [Bau+22].

The authors of [Bau+22] solve the biometric comparison issue by using more appropriate and better tested algorithms and the offline-attack issue by replacing the hashing element of fuzzy vaults as presented in [Tam16] with an interactive OPRF evaluation for each guess attempt, followed by a Key Encapsulation Mechanism (KEM). Worth noting are existing alternatives which aim to accomplish the same offline-attack resistance without an interactive protocol [QCC18; SS20], mostly based on fuzzy extractors described in [DRS04].

OPRFs have been extensively studied and many applications have been constructed using them. Due to their recent relevance, their properties have been systematized and described in the work by [CHL22]. In [Bau+22], the default instantiation of the protocol uses discrete logarithms for security specifically in the form of a hashed Diffie-Hellman OPRF [FK00]. An alternative instantiation approach is to use lattice-based OPRFs which are assumed to be post-quantum secure [Alb+21a]. The detailed constructions of these lattices have been presented in [BPR12].

In the post-quantum domain, OPRF constructions have not been as widely explored. Currently, the work by Albrecht et al. [Alb+21b] is the only construction that has not been shown to be insecure.

A possible exception is the work by Boneh et al. [BKW20] in which a post-quantum

secure OPRF based on isogenies was proposed. Specifically, an attempt was made to adapt a Diffie-Hellman OPRF to the setting of isogenies of supersingular elliptic curves and provide active security. However, an attack was found by Basso et al. in their work [Bas+21] that compromises the pseudorandomness after a few OPRF evaluations and some offline computation. In a very recent development, a new construction was proposed by Basso in [Bas23] that is based on the original construction by Boneh et al. [BKW20] and provides countermeasures for the vulnerabilities found in [Bas+21].

1.3 Outline

This thesis will first introduce the theoretical principles of lattices and lattice-based cryptography in Chapter 2, with a particular focus on the specific mathematical problems which are the foundation of the security of the OPRF and KEM elements of the protocol.

Next, Chapter 3 will be dedicated to the modifications made to these structures and their security properties. These modifications constitute the main contribution of this thesis, along with the implementation.

A general overview of the implementation will be provided in Chapter 4, including and justifying specific design decisions.

Following the implementation details, Chapter 5 is dedicated to the the performance evaluation of the modified protocol, a comparison with the classically secure version from [Bau+22], and the exploration of the parameter choices which have a significant impact on the performance and security of the post-quantum secure protocol components.

Lastly, a conclusion is presented in Chapter 6 which summarises the work and its performance and gives indication for potential further improvements.

Chapter 2

Background

In order to construct a post-quantum secure BRAKE protocol, it is necessary to introduce the concept of lattice-based cryptography, as the mathematical problems used to secure this modification of the BRAKE protocol are effectively lattice problems.

2.1 Lattices

We begin by introducing the mathematical objects the cryptographic problems used in this thesis build upon, which are lattices.

Definition 1. [Pei16] A lattice is the set of all integer linear combinations of linearly independent basis vectors $\boldsymbol{B} = \{b_1, b_2, ..., b_n\} \subset \mathbb{R}^n$:

$$\mathcal{L} = \mathcal{L}(\boldsymbol{B}) = \{\sum_{i=1}^{n} x_i \cdot \boldsymbol{b_i} : x_i \in \mathbb{Z}\} = \{\boldsymbol{Bx} : \boldsymbol{x} \in \mathbb{Z}^n\}. \tag{2.1}$$

Furthermore, $\boldsymbol{B}$ is not unique, i.e., there is not only one basis that generates this lattice $\mathcal{L}$, there are others that exist as per the following theorem:

Theorem 1 (Multiple bases [MR09])**.** Let **B** and **B'** be two bases of a lattice L. Then $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B'})$ if and only if there exists a unimodular matrix **U** (i.e., a square matrix with integer entries and determinant ± 1) such that $\boldsymbol{B} = \boldsymbol{B'U}$.

Each such basis forms a *fundamental region*, in the case of a two-dimensional lattice it is called a *fundamental parallelepiped*. All of these regions share the same volume, which is equal to the determinant of the lattice in question. The volume of the fundamental region is defined as [MR09]:

$$P(\boldsymbol{B}) = \{\sum_{i=1}^{n} x_i \cdot \boldsymbol{b_i} : 0 \leq x_i < 1\}. \tag{2.2}$$

In geometric terms, the value of the determinant is the inverse of lattice point density [MR09]. This determinant can be efficiently computed from any basis in polynomial time [MR09].

An important notion is the minimum distance of a lattice ($\Lambda = \mathcal{L}(\boldsymbol{B})$), which is the smallest distance between any two lattice points:

$$\lambda(\Lambda) = \inf\{||x - y|| : x, y \in \Lambda, x \neq y\} \tag{2.3}$$

or equivalently, the length of the shortest non-zero lattice vector:

$$\lambda(\Lambda) = \inf\{||v|| : v \in \Lambda \setminus \{0\}\}. \tag{2.4}$$

Especially important in the context of cryptography are so-called q-ary lattices.

Definition 2 (q-ary lattices, [Lyu20])**.** A lattice Λ is considered q-ary if it fulfils the condition $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$. If we take a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ then the q-ary lattice Λ defined by $\boldsymbol{A}$ is:

$$\Lambda = \mathcal{L}_q^{\perp}(\boldsymbol{A}) = \{v \in \mathbb{Z}^m | \boldsymbol{A}v \equiv 0 (\text{mod } q)\}. \tag{2.5}$$

In this context, $\boldsymbol{A}$ can be a parity check matrix for a linear code.

Another term that needs to be introduced for the purposes of the ring variations of problems given later, are *ideal lattices*.

Definition 3 (Ideal lattice, [Pei16])**.** An ideal lattice is a lattice that corresponds to an ideal (a specific subset of elements) in a ring R under a fixed choice of geometric embedding.

2.2 Lattice-Based Cryptography

Lattice-based cryptography builds upon the idea that there exist certain problems which are considered hard on point lattices in $\mathbb{R}^n$ and can be used as the basis for designing cryptographic systems [Pei16]. The idea that formed the foundation of this area of cryptography is the work by Ajtai [Ajt96] where a reduction from worst-case to average case instantiations of lattice problems are introduced. More concretely, the author of [Ajt96] showed that if some related lattice problems are hard in the worst case, they will also be hard for in the average case in cryptographically useful distributions, which is essential for cryptography where average-case intractability is needed by definition [Pei16].

One of the problems used in [Ajt96] is the commonly studied *Shortest Vector Problem* (**SVP**). It is defined as follows:

Definition 4. (Shortest Vector Problem - SVP), (Definition 2.1 in [Pei16]). Given an arbitrary basis $\mathbf{B}$ of some lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$, find a shortest non-zero lattice vector, i.e., a vector $v \in \mathcal{L}$, for which $||v|| = \lambda(\mathcal{L})$.

A variant of this problem includes an approximation factor $\gamma \geq 1$ and is called the *Approximate Shortest Vector Problem* (SVP_γ)[Pei16]:

Definition 5. (Approximate Shortest Vector Problem - $\mathbf{SVP}_\gamma$), (Definition 2.2 in [Pei16]). Given a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$, find a non-zero vector $v \in \mathcal{L}$ for which $||v|| \leq \gamma(n) \cdot \lambda(\mathcal{L})$.

This problem can be further extended to find n linearly independent shortest vectors with some approximation (Approximate Shortest Independent Vectors Problem - $\mathbf{SIVP}_\gamma$) or modified into a decisional problem of determining if the length of the shortest vector $\lambda(\mathcal{L}) \leq 1$ or $\lambda(\mathcal{L}) > \gamma(n)$ (Decisional Approximate SVP - $\mathbf{GapSVP}_\gamma$).

However, in the context of this project, the most important problems are *Short Integer Solution* (**SIS**) and *Learning With Errors* (**LWE**), which will be explained in more detail in the following sections.

2.2.1 Short Integer Solution (SIS) Problem

The Short Integer Solution (**SIS**) problem has its beginnings with Ajtai in the same work as the worst-to-average case reduction [Ajt96], but it was officially introduced in [MR07]. Generally speaking, the goal of an attacker trying to solve this problem is to find a non-trivial small integer combination of vectors $z_1, z_2, ..., z_m \in \{0, \pm 1\}$ such that they result in a zero vector.

$$\underbrace{\begin{pmatrix} .... & A & .... \end{pmatrix}}_{m} \begin{pmatrix} z \end{pmatrix} = 0 \in \mathbb{Z}_q^n \tag{2.6}$$

Figure 2.1: The Short Integer Solution Problem visualized.

The full definition is given below and a visual representation is shown on Figure 2.1:

Definition 6. (Short Integer Solution Problem - SIS$_{n,q,\beta,m}$), (Definition 4.1 in [Pei16]). Given m uniformly random vectors $a_i \in \mathbb{Z}_q^n$, forming the columns of a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero integer vector $z \in \mathbb{Z}^m$ of norm $||z|| \leq \beta$ such that

$$f_{\boldsymbol{A}}(z) := \boldsymbol{A}z = \sum_i a_i \cdot z_i = 0 \in \mathbb{Z}_q^n. \tag{2.7}$$

Here, the parameters n and q are positive integers, β a positive real number much smaller than q, and lastly m, is the number of elements from $\mathbb{Z}_q^n$. These parameters are directly related to the security of the SIS problem, specifically n which makes the problem more difficult as it increases. Increasing m on the other hand makes the problem easier as any solution calculated for the matrix $\mathbf{A}$ will also be a solution for the matrix $[\boldsymbol{A}|\boldsymbol{A'}]$ due to the fact that adding zeroes to it will also provide a valid solution to the expanded matrix [Pei16].

From Definition 6, it can be seen that the SIS problem is not directly related to lattices. However, it can be transformed to a lattice problem in the following way. Let us define a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and let it define a *q-ary* lattice $\mathcal{L}_q^{\perp}(\boldsymbol{A})$.

To relate this with the Ajtai [Ajt96] worst-case-to-average-case reduction, if an attacker is capable of finding a short ($||z|| \leq \beta \ll q$) non-zero vector $z \in \mathcal{L}_q^{\perp}(\boldsymbol{A})$ for a uniformly random $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, they can then solve **GapSVP** and **SIVP** for any n-dimensional lattice [Pei16].

Figure 2.2 shows a *q-ary* lattice and the circle (or sphere in higher dimensions) which encompasses the solution(s) to the SIS problem.

To better showcase the LWE problem, one more lattice problem will be introduced, the *Bounded Distance Decoding* (**BDD**$_\gamma$) problem. It poses the question of finding a lattice vector closest to a given target point $t \in \mathbb{R}^n$ which is guaranteed to be within a certain distance to a lattice point [Pei16]. A formal definition is also given in:

Definition 7. (Bounded Distance Decoding Problem - BDD$_\gamma$), (Definition 2.5 in [Pei16]). Given a basis $\boldsymbol{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ and a target point $t \in \mathbb{R}^n$ with the guarantee that $\text{dist}(t, \mathcal{L}) < d = \lambda(\mathcal{L})/(2\gamma(n))$, find the unique lattice vector $v \in \mathcal{L}$ such that $||t - v|| < d$.

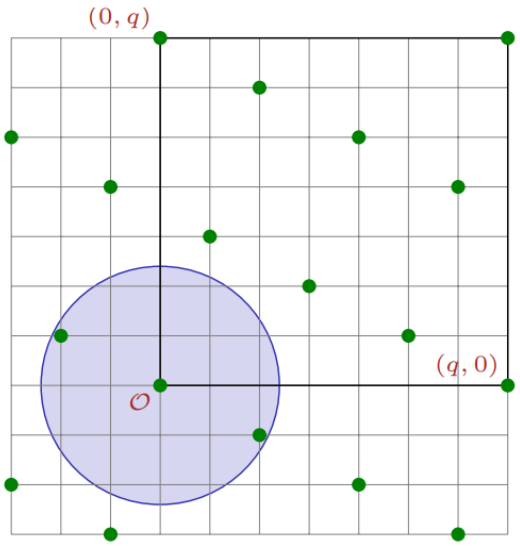

Figure 2.2: q-ary integer lattices in 2D geometric representation (illustration from lectures based on [Pei16]).

2.2.2 Learning With Errors (LWE)

The lattice problem that is sometimes referred to as a dual problem to SIS [Reg10], specifically the one whose variation this thesis is concerned with, is the Learning With Errors (**LWE**) problem introduced by Regev in [Reg09].

A formal definition of the basic problem is given as follows:

Definition 8. (LWE Distribution), (Definition 4.2 in [Pei16]). For a vector $s \in \mathbb{Z}_q^n$ called the *secret*, the LWE distribution $A_{s,\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $a \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow \chi$, and outputting the pair $(a, b = \langle s, a \rangle + e \bmod q)$.

As seen in the definition, the problem has three important parameters: positive integers n and q and an error distribution χ over $\mathbb{Z}$ [Pei16] which is often a discrete Gaussian distribution. Relative to each other, they generally have the following relationships: $\sqrt{n} \leq error \ll q$ [Pei16].

LWE can be split into two different versions, *search* and *decision*. From the perspective of an attacker, the *search*-LWE problem is the problem when an attacker is given m independent LWE samples (a, b) and are trying to find the secret s. Notice that if $m = n$, this appears to be solvable in polynomial time by applying Gaussian elimination as it is just a system of equations. However this is not the case as the errors in each equation compound and thus hide the resulting information effectively [Reg10]. The *decision*-LWE problem consists of the task to distinguish (a, b) from uniformly random (a, b). More formal definitions of both problems are given in the following [Pei16].

Definition 9. (**Search-LWE**$_{n,q,\chi,m}$), (Definition 4.3 in [Pei16]). Given m independent samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ drawn from $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$ (which is fixed for all samples), find s.

Definition 10. (**Decision-LWE**$_{n,q,\chi,m}$), (Definition 4.4 in [Pei16]). Given m independent samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where every sample is distributed according to either: (1) $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$ (which is fixed for all samples), or (2) the uniform distribution, distinguish which is the case (with non-negligible advantage).

Since both of these variations use m independent LWE samples, a more convenient way to display them is available by using matrices. This can be done in the following way: first we combine the samples into the matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ where the columns are vectors $a_i \in \mathbb{Z}_q^n$ then we combine $b_i \in \mathbb{Z}_q$ into a vector $b \in \mathbb{Z}_q^m$ with the b_i as entries. Then we can present all the sample pairs with a simple statement [Pei16]:

$$\boldsymbol{b}^t = s^t \boldsymbol{A} + e^t \pmod{q} \ , e \leftarrow \chi^m. \tag{2.8}$$

To connect this problem with lattices, we can equate it with an average-case of the $\mathbf{BDD}_\gamma$ problem on a family of q-ary m-dimensional integer lattices [Pei16; Reg10]. Essentially, the vector b is relatively close to a single vector in the LWE lattice $\mathcal{L}(\boldsymbol{A}) := \{\boldsymbol{A}^t s : s \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m$ [Pei16]. The goal is to find this vector as per the $\mathbf{BDD}_\gamma$ problem.

Briefly touching on the hardness of these problems and the reductions that establish these statements, *decision*-LWE is at least as hard as *search*-LWE [Reg09] which is in turn at least as hard as solving worst-case approximate lattice problems with approximation factor $\gamma = (n/\alpha)$ (where α is the error rate, the width of the error relative to q).

2.2.3 Ring-LWE

Ring-LWE or **R-LWE** is a variant of the **LWE** problem that tries to improve upon its computational efficiency. To illustrate the issue with regular **LWE**: each calculation of a pseudorandom scalar $b_i \in \mathbb{Z}_q$ from an **LWE** sample requires an n-dimensional inner product which is an amount of work on the order of $\mathcal{O}(n)$. This causes cryptographic systems based on regular **LWE** to have large keys [Reg10] with relatively large values for n on the order of hundreds to thousands.

The desired efficiency improvement is achieved through batching these calculations in such a way that we only need to produce n such pseudorandom scalars in one calculation instead of n^2. First, we need to say that n is a power-of-two, then that vectors a are used in blocks of n samples $a_1, a_2, ..., a_n \in \mathbb{Z}_q^n$ where $a_1 = (a_1, a_2, ..., a_n)$ is chosen uniformly at random and the remaining vectors are given by $a_i = (a_i, ..., a_n, -a_1, ... - a_{i-1})$ [Reg10].

This technique can be viewed as essentially replacing the group $\mathbb{Z}_q^n$ with a cyclotomic polynomial ring $R = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ where q is an integer and n is a power of two [Reg10]. Elements of this ring are polynomials of deg $< n$ with q-mod integer coefficients. So effectively, a single Ring-LWE sample replaces n standard LWE samples [Pei16].

This leads into the formal R-LWE distribution definition [Pei16]:

Definition 11. (**Ring-LWE distribution)**), (Definition 4.6 in [Pei16]). For an $s \in R_q$ called the secret, the Ring-LWE distribution $A_{s,\chi}$ over $R_q \times R_q$ is sampled by choosing $a \in R_q$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, b = s \cdot a + e \bmod q)$.

From this, we can derive similar problems as standard LWE, therefore *search*-R-LWE becomes the problem of finding the secret ring element $s(X) \in R_q$, given as:

$$a_i \leftarrow R_q \ , \ b_i = s \cdot a_i + e_i \in R_q \ , \ i = 1, 2, ..., m. \tag{2.9}$$

and *decision*-R-LWE the problem of distinguishing (a_i, b_i) from uniformly random $(a_i, b_i) \in R_q \times R_q$ (with non-negligible advantage). Formally this is shown in [Pei16]:

Definition 12. (**Decision-R-LWE**$_{q,\chi,m}$), (Definition 4.7 in [Pei16]). Given m independent samples $(a_i, b_i) \in R_q \times R_q$ where every sample is distributed according to either: (1)

$A_{s,\chi}$ for a uniformly random $s \in R_q$ (fixed for all samples), or (2) the uniform distribution, distinguish which is the case (with non-negligible advantage).

The hardness of this problem is also guaranteed. Given that the parameter choices (q, error rate-α and error distribution-χ) are correct, then solving R-LWE is at least as hard as quantumly solving SVP_γ on arbitrary ideal lattices in R, for some $\gamma = poly(n)/\alpha$ [LPR10].

2.2.4 Module-LWE

One last LWE-based problem that is used within this thesis (in the Key Encapsulation Mechanism) is **Module-LWE** or **M-LWE**. It combines features from both LWE and R-LWE [AD17] with regards to efficiency and security. On a high level, M-LWE allows to require calculation of $n \cdot d$ pseudorandom scalars to represent n LWE samples, so more than R-LWE's n but still less than regular LWE's n^2. The scalar d is a module rank, in the context of an M-LWE sample $b = a \cdot s + e$ this simply means that the a coefficient is a matrix with dimensions $n \times n \cdot d$. Considering this, we can also say that R-LWE is essentially M-LWE with module rank 1.

More formally, the distribution used in M-LWE is as follows:

Definition 13. (**Module-LWE Distribution**). For an $s \in R_q^d$ called the secret, the M-LWE distribution $A_{d,s,\chi}$ over $R_q^d \times R_q$ is sampled by choosing $a \in R_q^d$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, b = \langle s \cdot a \rangle + e \bmod q)$.

Once again, from the defined distribution we can derive the definition for the search and decision problems:

Definition 14. (**Decision and Search M-LWE**). Given m independent samples $(a_i, b_i) \in R_q \times R_q$ where every sample is distributed according to either: (1) $A_{d,s,\chi}$ for a uniformly random $s \in R_q^d$ (fixed for all samples), or (2) the uniform distribution, distinguish which is the case (with non-negligible advantage). *Search*-M-LWE is then the problem of finding the secret ring elements $s(X) \in R_q^d$, given as:

$$a_i \leftarrow R_q^d \ , \ b_i = \langle s \cdot a_i \rangle + e_i \in R_q \ , \ i = 1, 2, ..., m. \tag{2.10}$$

The hardness guarantee for M-LWE is based on a different class of lattices when compared with R-LWE. Specifically, M-LWE is based on module lattices (lattices that generalize arbitrary and ideal lattices [LS15]) unlike R-LWE's ideal lattices. Furthermore, the guarantee itself is stronger, as MLWE is equivalent to natural hard problems over module lattices [AD17].

2.3 Oblivious Pseudo-Random Function (OPRF)

The idea of an Oblivious Pseudo-Random Function (OPRF) is built upon the foundations laid with the introduction of pseudo-random functions (PRFs)[GGM86]. However, in order to arrive at the definition of an OPRF, we need to combine the concept of PRFs with the idea of an Oblivious Transfer (OT).

2.3.1 Pseudo-Random Functions (PRF)

Pseudo-random functions are efficiently constructed deterministic functions whose output looks like it is chosen at random [CHL22]. More specifically a family of functions is pseudo-random if an attacker with oracle access for a randomly chosen function from the family cannot differentiate it from a uniformly random function [BPR12]. A formal definition is:

Definition 15 (Pseudo-Random Function, definition 6 in [CHL22])**.** A family of functions $f_k : \{0,1\}^m \rightarrow \{0,1\}^n$ with key $k \in \{0,1\}^\lambda$ is called pseudorandom if the following holds:

- $f_k(x)$ is efficiently computable from k and x.
- It is not efficiently decidable whether one has access to a computation oracle for $F_s(\cdot)$ or to an oracle producing random bitstrings of length n.

2.3.2 Oblivious Transfer (OT)

The generic definition of Oblivious Transfer protocols is that they allow one party, a *sender*, to send a part of its inputs to another party, the *chooser*, in such a way that the sender is assured that the chooser does not receive more information than it is meant to while the chooser is assured that the sender will not learn which part of the inputs it sent to them[NP01]. Furthermore, the sender is not even necessarily aware if the chooser received any information[Rab05].

In more practical terms this can be seen from the simplest basic form of OT, which is 1-out-of-2 Oblivious Transfer (OT_1^2). It is defined as the *sender* having an input composed from two strings (M_0, M_1) and the *chooser* having an input composed of one bit σ. The *chooser* then only learns M_σ, without learning anything about $M_{1-\sigma}$, while the *sender* gets no information about σ[NP01]. This OT can then be generalised to 1-out-of-N oblivious transfer where the *sender* has N messages instead of only two.

2.3.3 OPRF Definition and Application

Combining the concepts of pseudo-random functions and oblivious transfer, oblivious pseudo-random functions (OPRFs) can be defined. This definition then describes an OPRF as: a secure 2-party protocol with the functionality $(k, x) \rightarrow (\perp, f_k(x))$, where $f_k(x)$ is the PRF and with $\perp$ denoting empty output [CHL22; Fre+05].

A formal definition for an OPRF is given as:

Definition 16. (**Strongly-private Oblivious Pseudo-Random Function)**, (Definition 5 in [Fre+05]). A 2-party protocol π between a client (*chooser*) and a server (*sender*) is a strongly-private OPRF if there exists some PRF family f_k, such that π privately realizes the functionality:

- Client has input x; Server has key k.
- Client outputs $f_k(x)$; Server outputs nothing.

This definition specifically refers to a so-called Strongly-private OPRF, however Freedman also defines a Relaxed-OPRF where the client (*chooser* in previous chapters) is allowed to learn partial information about the key [CHL22; Fre+05]. This variant is sometimes used in settings with slightly relaxed security requirements for which otherwise efficient Relaxed-OPRFs are sufficient [CHL22].

A connection between OT (specifically 1-out-of-2 OT) with and an execution of a Strongly-private OPRF is given in [CHL22] as:

Definition 17. (**Connection between OT and OPRF**), (Definition in [Pei16]). f - PRF function, K - key space, k - key, domain 2^k

- Alice, on input m_0, m_1 chooses $k \leftarrow K$, computes $c_0 \leftarrow m_0 \oplus f_k(0), c_1 \leftarrow m_1 \oplus f_k(1)$ and sends c_0, c_1 to Bob.
- Alice and Bob then engage in one execution of the OPRF protocol with Alice playing the role of the server with input k, and Bob in the role of the client with input $b \in \{0, 1\}$. Bob learns $f_k(b)$, Alice learns nothing.
- Bob decrypts $m_b \leftarrow c_b \oplus f_k(b)$ and outputs m_b.

This statement can then be generalized to an m-out-of-n OT with m subsequent strongly-private OPRF evaluations[JL09].

This leads into the topic of real-world applications of OPRFs. One relatively straightforward example is a simple keyword search operation on a database, as explored in [Fre+05].

In this scenario, there is a server that holds a database of records associated with some keywords and a client who sends queries consisting of keywords and wants to receive the records associated with those keywords. On first glance, this seems like a setting that is appropriate for an extended form of 1-out-of-n Oblivious Transfer (for example a k-out-of-n OT shown in [IK97]). However, this is not the case, as in this situation there is an additional step that necessitates a stronger tool.

The client (chooser) sends a search-word and the server needs to check if that input corresponds to any keyword it has stored, as opposed to the client needing to send input from a limited set of possibilities (like the binary choice in a 1-out-of-2 OT). The server then uses a PRF to assign random pseudo-identities to mask the real keywords and their corresponding records $(\hat{x}_i, \hat{p}_i)$ as (x'_i, p'_i) with $x'_i = \hat{x}_i$ and $p'_i = p_i \oplus \hat{p}_i$. Then, the OPRF protocol is invoked with the server providing the random key k for the PRF and the client providing the search-word and a Secure Function Evaluation (SFE) $f_k(w)$ is performed. Afterwards, the client receives an obscured search-word $\hat{w}$ and a piece for unlocking the record $\hat{p}$. The search-word $\hat{w}$ can then be used by the server to provide the client with the record p'_i which the client then uses to output $p'_i \oplus \hat{p}$. If the search-word was found to have a match, otherwise the client outputs $\perp$ [Fre+05].

2.3.4 OPRF Constructions

OPRFs can be constructed in many ways. A recent survey by Casacuberta et al. [CHL22] groups these constructions into categories based on the underlying PRF and method of oblivious evaluation. One of these categories describes OPRFs based on the Hashed Diffie-Hellman PRF and these are of particular interest for this thesis as the current classically-secure implementation of the improved BRAKE protocol [Bau+22] uses such an OPRF. Thus it will be the one used as an example to illustrate the construction of an OPRF.

2.3.5 Hashed Diffie-Hellman

To assert that the function $f_k^H(x) := H(x)^k$ is a PRF an assumption that the hash function H produces uniformly random elements from a group $\langle g \rangle$ [NPR99] is needed. This implies that the group $\langle g \rangle$ must be cyclic and of prime order, meaning every element of the group is a generator. Then, the protocol for evaluating the PRF function itself is shown in Figure 2.3.

C(x)		**S(k)**
$r \leftarrow_R \mathbb{Z}_q$		
$a \leftarrow H(x)^r$	$\xrightarrow{\quad a \quad}$	$b \leftarrow a^k$
Output $b^{1/r}$	$\xleftarrow{\quad b \quad}$	

Figure 2.3: Blinded exponentiation for evaluating the Hashed Diffie-Hellman PRF [CHL22].

In this protocol, the client's input x is hashed into being a group element, then exponen-

tiated with a random r value, therefore mapping it to another group element and then sending it to the server to be exponentiated again with the server's key k. This is effectively a standard Diffie-Hellman procedure, where the client can take the received value from the server and exponentiate it with $1/r$ to obtain the $H(x)^k$ value which is equal to the OPRF evaluation for his input x.

The result of this fulfils the stated goal of an OPRF, namely the client outputs the evaluated function without learning anything about the server's key k and without the server learning anything about the client's original input x.

2.4 Biometric Resilient Authenticated Key Exchange

Finally, the power of the OPRF cryptographic concept can be harnessed in authentication protocols. Such a protocol is considered in this thesis, the Biometric Resilient Authenticated Key Exchange (BRAKE) protocol [Bau+22].

It has three participants, a Client, Server and Evaluator and two functionalities:

- Enrollment
- Verification

The improved fuzzy vault scheme, namely the one presented in [Tam16], is used to secure biometric data. Following the given example of fingerprint data, the use of this scheme in the context of the BRAKE protocol is to create two polynomials, a randomly sampled one f and a corresponding vault polynomial $V(X) = f(X) + \prod_{a \in A}(X - a)$ [Tam16] which obscures the actual biometric fingerprint template by adding the randomly sampled f.

In the enrollment phase, the randomly sampled polynomial f is used as input to the OPRF and the vault $V(X)$ is stored at the server, alongside a unique identifier for the Client and the public key produced by the OPRF. In the verification use case, the Client requests the stored fuzzy vault corresponding to its identifier and creates a polynomial f' by interpolating a fresh biometric probe feature set as $U = \{(b, V(b)) | b \in \mathbf{B}\}$. Then the f' is used in the OPRF in a similar way as in the enrollment.

The purpose of the OPRF mechanism is to produce key material for key pairs from the random polynomials f and f', in the enrollment and verification functionalities respectively, while the biometric features remain hidden. Also important to point out is that the OPRF essentially ensures that an interaction between the participants is mandatory for every attempt at verification, thus hindering brute-force attacks.

The final step is the Key Encapsulation Mechanism (KEM), as part of the verification functionality. Here, the goal of which is to produce a session pre-key encapsulated by the reference public key (previously stored during enrollment) that can only be recovered (decapsulated) by the Client if the public key produced from the freshly captured fingerprint (through the OPRF mechanism) matches the one stored at the Server. In this way, The formal workflow of this protocol will be detailed in Chapter 3.

Chapter 3

Post-Quantum Secure Biometric Resilient Authenticated Key Exchange (PQ-BRAKE)

The goal of this work is to adapt the BRAKE [Bau+22] protocol in such a way that post-quantum security is achieved. The existing implementation [Bau+22] mentioned in Chapter 2 is based on discrete logarithms, which are not considered secure in a quantum setting.

The primitives proposed here as a method of achieving post-quantum security are lattice based primitives, namely those based on the Ring Learning-With-Errors (R-LWE) problem. First, this chapter will give an overview of the underlying structures of the primitives followed by presenting the adapted PQ-BRAKE protocol founded on those primitives. Thereby, this Chapter constitutes the main contribution of this thesis, alongside the implementation described in Chapter 4.

We briefly reiterate the three main components of the classically secure BRAKE protocol described in Chapter 2:

1. The improved fuzzy vault scheme [Tam16].
2. An Oblivious Pseudo Random Function (OPRF).
3. A Key Encapsulation Mechanism (KEM).

The adaptation performed in this work concerns the OPRF and the KEM mechanisms, as implementing them with quantum-resistant primitives should result in the protocol being post-quantum secure. No changes are required with regard to the improved fuzzy vault scheme [Tam16].

Specifically, adapting the OPRF was the main focus of this thesis as no implementation currently exists for the chosen OPRF construction, which is the only lattice-based OPRF construction currently known. This is the previously introduced VOPRF by Albrecht et al. [Alb+21b], which is based on the R-LWE problem.

The KEM adaptation consists of replacing the Diffie-Hellman key exchange used in the classically secure implementation [Bau+22] with a lattice-based alternative. The one chosen was the recently standardised algorithm CRYSTALS-Kyber [Ava+21], which is based on the Module-LWE problem.

3.1 Lattice-Based Oblivious Pseudo-Random Function

Constructions for OPRFs with lattice-based cryptographic primitives are a relatively new development. Lattice-based PRF constructions have been previously proposed in [BPR12] but a full lattice-based OPRF construction has only recently been shown in [Alb+21b].

This construction bases the OPRF on the hardness of the LWE problem and is as such assumed to be post-quantum secure [Reg09]. Additionally this specific construction has the additional property of being *verifiable* (making it a VOPRF), that is, it means the client has a guarantee that the output received from the OPRF evaluation is truly correct and calculated with the server's publicly committed key k [Alb+21b; CHL22].

The specific PRF function implemented in [Alb+21b] is based on the hardness of the Ring-LWE problem and is defined as:

$$F_k(x) = \left\lfloor \frac{p}{q} \cdot a^F(x) \cdot k \right\rceil \tag{3.1}$$

where $a^F : \{0,1\}^L \to R_q^{1\times\ell}$, the ring used is $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, $k \in R_q$ is the key with small coefficients in $\{-q/2, ..., q/2\}$ representation and the $\lfloor\cdot\rceil$ operation is rounding to the nearest natural number [Alb+21b]. A note about the function a^F, in [Alb+21b], this function is intentionally constructed in such a way as to allow the guarantees necessary for the aforementioned verifiability feature. This function will be replaced with a simpler variant in the implementation part of the thesis but this will be expanded upon in a following chapter.

The basic functionality of this VOPRF (with the original a^F function) is as follows [Alb+21b]:

1. The server publishes a commitment $c := a \cdot k + e$ to a small key $k \in R_q$ (c is a R-LWE sample).
2. The client chooses a small $s \in R_q$, small $e_1 \in R_q^{1\times\ell}$, computes and sends $c_x = a \cdot s + e_1 + a^F(x)$, where x is the client's input.
3. The server calculates and sends $d_x = c_x \cdot k + e'$ for a small $e' \in R_q^{1\times\ell}$.
4. The client outputs $y = \lfloor \frac{p}{q} \cdot (d_x - c \cdot s) \rceil$.

A basic idea of security is given from both the perspective of the server and the client [Alb+21b]:

- The server exposes to the client the value d_x which can be expanded into the formulation: $d_x = a \cdot s \cdot k + a^F(x) \cdot k + e_1 \cdot k + e'$. If e' is chosen in an appropriate distribution it can hide the addition of $e_1 \cdot k$, $e \cdot s$ and an e_x value (from another narrow distribution). To the client, this value can then be $d_x = (a \cdot k + e) \cdot s + e' + (a^F(x) \cdot k + e_x) = c \cdot s + (a^F(x) \cdot k + e_x) + e'$ which is essentially just a R-LWE sample if the client cannot differentiate $a^F(x) \cdot k + e_x$ from uniform random (which he indeed cannot do if e_x is picked from a correct distribution).
- The client exposes to the server the value $c_x = a \cdot s + e_1 + a^F(x)$. By the definition of the decisional-R-LWE (Definition 12) problem, this is indistinguishable from random and the secret value s and $a^F(x)$ remain hidden from the server.

The guarantee that the client actually receives the correct OPRF evaluation y is given through the fact that if the values $e_1 \cdot k$, e' and $e \cdot s$ are correctly chosen, they are small

enough so that the value of $d_x - c \cdot s$ is essentially equal to $a^F(x) \cdot k$ which corresponds to the PRF.

In order to show the actual construction of the VOPRF, it is useful to first give a definition of the discrete Gaussian distributions used.

First, the Gaussian function on $\mathbb{R}^n$ centred at $c \in \mathbb{R}^n$ with parameter σ is defined in [Alb+21b] as:

$$\rho_{\sigma,c}(x) = e^{-\pi \cdot ||x-c||^2/\sigma^2}, \forall x \in \mathbb{R}^n. \tag{3.2}$$

Then the discrete Gaussian distribution over $\mathbb{Z}$ (denoted as χ_σ in the protocol) assigns probability $\rho_\sigma(i)/\rho_\sigma(\mathbb{Z})$ to each $i \in \mathbb{Z}$ and 0 to all non-integer points [Alb+21b]. The discrete Gaussian distribution over $\mathbb{R}$ (denoted as $R(\chi_\sigma)$ in the protocol) is the distribution over R where each coefficient is distributed according to χ_σ [Alb+21b].

Now the full VOPRF protocol construction as given in [Alb+21b] can be shown in a protocol diagram on Figure 3.1.

This diagram includes the basic functionality described above but also the zero-knowledge proof systems denoted by $\mathbb{P}_0, \mathbb{P}_1, \mathbb{P}_2$ and additional elements needed to facilitate them (b and common random strings $crs_i, i = 0, 1, 2$). In short, these systems serve to give assurances to both the client and the server that their calculations are honest and truly computed from legitimately known information.

The three proofs are as follows [Alb+21b]:

- The client proof $\mathbb{P}_1(x, s, e_1 : crs_1, c_x, a, a_0, a_1)$ proves that the client legitimately knows x, s and e_1 such that $c_x = a \cdot s + e_1 + a_x \bmod q$.
- The server proof $\mathbb{P}_0(k, e : crs_0, c)$ proves that the server has legitimate knowledge of k and e such that $c = a \cdot k + e \bmod q$, while crs_0 contains a.
- The server proof $\mathbb{P}_2(k, e', e : crs_2, c, d_x, c_x, a)$ proves that the server legitimately knows k, e and e' such that $c = a \cdot k + e$ and $d_x = c_x \cdot k + e' \bmod q$.

VOPRF construction by Albrecht et al. [Alb+21b]

Setup	
$a_0, a_1 \leftarrow R_q^{1\times\ell}$	
$a \leftarrow R_q^{1\times\ell}$	
sample string crs_0	
$crs_0 := (crs_0, a)$	
sample string crs_1, crs_2	
Client	**Server**
	$k \leftarrow R(\chi_\sigma)$
	$e \leftarrow R(\chi_\sigma)^{1\times\ell}$
	$c := a \cdot k + e$
	$\pi_0 := \mathbb{P}_0(k, e : crs_0, c)$
$\xleftarrow{c, \pi_0}$	
$b := \mathbb{V}_0(crs_0, c, \pi_0)$	
if$(b == 0) \Rightarrow$ *abort output*	
store(c)	
$s \leftarrow R(\chi_\sigma)$	
$e_1 \leftarrow R(\chi_\sigma)^{1\times\ell}$	
$a_x := a_{x_1} \cdot G^{-1}(...(a_{x_{L-1}} \cdot G^{-1}(a_{x_L}))...)$	
$c_x := a \cdot s + e_1 + a_x$	
$\pi_1 := \mathbb{P}_1(x, s, e_1 : crs_1, c_x, a, a_0, a_1)$	
$\xrightarrow{c_x, \pi_1}$	
	$b := \mathbb{V}_1(crs_1, c_x, a_0, a_1, \pi_1)$
	if$(b == 0) \Rightarrow$ *abort output*
	$e' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$
	$d_x := c_x \cdot k + e'$
	$\pi_2 := \mathbb{P}_2(k, e', e : crs_2, c, d_x, c_x, a)$
$\xleftarrow{d_x, \pi_2}$	
	output$(\perp)$
$b := \mathbb{V}_2(crs_0, crs_2, c, d_x, c_x, \pi_2)$	
if$(b == 0) \Rightarrow$ *abort output*	
$y_x := \lfloor d_x - c \cdot s \rceil_p$	
output(y_x)	

Figure 3.1: Lattice-Based Verifiable OPRF (VOPRF) as presented in [Alb+21b].

3.2 Adaptation of the Lattice-based Oblivious Pseudo-Random Function (OPRF)

The OPRF used in this work is an adaptation of the VOPRF construction by Albrecht et al. [Alb+21b] for the purposes of a practical PQ-BRAKE protocol implementation.

Originally, as shown in Section 3.1., the presented OPRF has the property of being verifiable (thus being a VOPRF). This property is accomplished by the aforementioned zero-knowledge proofs, which guarantee that the participants of the protocol perform their respective calculations correctly and honestly.

However, the use of these proofs presents a problem of practicality if we wish to implement this exact construction. The amount of data generated by the proofs, which are then transmitted between the participating parties of the protocol, is extremely large. The authors of [Alb+21b] give a rough indication of the amounts in question at approximately 2^{40} bits or around 128 GB of communication data for realistic parameter choices of $log_2(q) \approx 256$ and $N = 16,384$.

Due to this significant communication overhead, a modification to a non-verifiable OPRF and only passive security was necessary. This is achieved by excluding the zero-knowledge proofs and truncating the underlying PRF accordingly. Consequently, the PQ-BRAKE protocol is then considered only in an honest-but-curious setting or semi-honest adversarial model. Essentially, all participants are not allowed to deviate from the protocol but may try to learn as much information as possible during this honest execution of the given protocol [Tre+19].

3.2.1 Truncating the Pseudo-Random Function (PRF)

An option that is made possible by removing the zero-knowledge proofs is the ability to heavily reduce the computation time and communication cost generated by the PRF.

Originally, the PRF is evaluated as $F_k(x) := \lfloor a_x \cdot k \rceil_p \in R_p^{1\times\ell}$ ([Alb+21b]) where a_x is a lattice PRF. This evaluation can be replaced with the PRF $F'_k(x) := \lfloor a_x \cdot k \rceil_p$ where a_x is a hash that is mapped to a ring element so that $F' : R \to R_q$ holds. This truncation shrinks the calculations from using a matrix of polynomials in R_p to just single polynomials in R_p. Consequentially, other values in the OPRF are similarly reduced, including c, c_x, a_x, d_x, y_x.

In practical terms, the input a_x we wish to evaluate the OPRF on, is the random polynomial f generated by the fuzzy vault scheme. Therefore, the element f needs to be mapped to a ring element in a deterministic fashion. The procedure is described in the following steps:

1. Concatenate every coefficient of f into a string cf.
2. Create a hash $h := H(cf)$ using a cryptographic hash function.
3. Produce $N+1$ coefficients of the polynomial a_x by creating a hash of the form $h_i := H(i\|h)$ for $i = 0, ..., N-1$ using the same hash function as before and converting the resulting hashes into integers. Here, $\|$ denotes concatenation.
4. Reduce the coefficients of a_x mod q.

This procedure results in a polynomial a_x which is an element of the ring $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ and can subsequently be used to compute an R-LWE sample.

3.2.2 Modified Lattice-Based Oblivious Pseudo-Random Function

Using the truncated PRF described above, the lattice-based OPRF construction by Albrecht et al. [Alb+21b] can be modified as will be described in the following Section.

Figure 3.2. shows the functioning of the modified OPRF, using the truncated PRF, in more detail.

Modified OPRF protocol, based on [Alb+21b]

	Client		**Evaluator**
2 :		(*public*)	
3 :		$a \leftarrow\!\!\$\ R_q$	
4 :			$k \leftarrow D_\sigma(R_q)$
5 :			$e \leftarrow D_\sigma(R_q)$
6 :			$c := a \cdot k + e$
7 :		$\xleftarrow{c}$	
8 :	store(c)		
9 :			
10 :	$s \leftarrow D_\sigma(R_q)$		
11 :	$e' \leftarrow D_\sigma(R_q)$		
12 :	$a_x := H(x)$		
13 :	$c_x := a \cdot s + e' + a_x$		
14 :		$\xrightarrow{c_x}$	
15 :			$E \leftarrow D_{\sigma'}(R_q)$
16 :			$d_x := c_x \cdot k + E$
17 :		$\xleftarrow{d_x}$	
18 :	$y_x := \lfloor d_x - c \cdot s \rceil_p$		
19 :	output(y_x)		

Figure 3.2: Modified OPRF protocol using the truncated PRF.

First, to define the R-LWE setting, the ring used is $R_q = \mathbb{Z}_q[X]/\langle X^N+1\rangle$, where q is a large positive prime and serves as modulus while N is a power-of-two integer and represents the dimension of the polynomials in R-LWE computations. The quotient polynomial is the cyclotomic polynomial $\langle X^N + 1\rangle$, which has been widely studied and is most commonly used due to its representation simplicity [CP16].

The distributions used for random sampling are:

- Uniform distribution D_σ over R_q which produces ternary values.
- Uniform distribution $D_{\sigma'}$ over R_q which produces values in a range $[-B, B]$, where B is a large power of two smaller than q.

In this setting, the modified OPRF is then executed as follows: first, a uniformly random ring element a with integer coefficients is sampled uniformly from a distribution which encompasses the entire ring R_q and published for all participants.

Then, the Evaluator samples a small ternary key k and a small ternary error polynomial e uniformly from D_σ. Using these ring elements, the Evaluator commits to the key k by constructing an R-LWE sample $c = a \cdot k + e$. The sample c is then sent to the Client and stored there, with the actual value of the key k being indistinguishable from a uniformly random ring element to the Client by the search-LWE assumption (Definition 9).

The Client now blinds its input, the random polynomial f generated by the fuzzy vault

scheme. To accomplish this, the Client hashes the input to a ring element as per the truncated PRF computation outlined in section 3.2.1. Next, a small polynomial s and small error polynomial e' are sampled uniformly from D_σ. Then, an R-LWE sample $c_x := a \cdot s + e' + a_x$ is constructed by adding the a_x value as another error polynomial.

The polynomial c_x is then sent to the Evaluator, who evaluates it by multiplying it with its key k and adding a large error polynomial E, uniformly sampled from $D_{\sigma'}$. This large error term drowns the value of the key k so that it remains hidden from the Client [Alb+21b]. The result is another R-LWE sample $d_x = c_x \cdot k + E$. This part of the OPRF protocol is also known as the blinding operation of the OPRF.

Now the evaluated polynomial a_x is sent back to the Client in the form of the R-LWE sample d_x, with it being secured in transit by the decision-RLWE assumption.

On receiving d_x, the Client computes a polynomial $y_x = \left\lfloor \frac{p}{q} \cdot (d_x - c \cdot s) \right\rceil$ by subtracting the value of the product $c \cdot s$ and then performing a rounding procedure coefficient-wise to obtain an evaluation of its input and fulfilling the OPRF functionality as defined in Definition 16.

The conclusion of the protocol using this polynomial y_x is described in more detail in the following Section.

3.2.3 Rounding

The rounding in the final step produces the Client's output, which is the polynomial y_x. If the rounding is implemented correctly and the protocol has been successfully executed, this rounded value will be equal to the rounded value $\lfloor a_x \cdot k \rceil_p$. This is known as the unblinding operation, which allows the Client to receive the computation of $a_x \cdot k$ without learning the Evaluator's key k, while the Evaluator does not learn the value of a_x.

The principle behind the validity of the rounding mechanism is shown in the following equations [Alb+21b], which depict the total amount of noise that is accrued through the protocol. Firstly, we introduce the R-LWE samples c, d_x and c_x, which form the total noise value. These are elements of the ring $R_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$ and are transmitted between the Client and Evaluator during the execution of the protocol. We recall their definitions as given in Figure 3.2:

$$c = a \cdot k + e \tag{3.3}$$
$$d_x = c_x \cdot k + E \tag{3.4}$$
$$c_x = a \cdot s + e' + a_x. \tag{3.5}$$

Next, we recall the computation of the polynomial y on the Client's side, which includes the values d_x, c and s before they are summed and rounded in y_x:

$$\begin{aligned} y &= d_x - c \cdot s && (3.6)\\ &= c_x \cdot k + E - (a \cdot k + e) \cdot s && (3.7)\\ &= (a \cdot s + e' + a_x) \cdot k + E - a \cdot k \cdot s + e \cdot s && (3.8)\\ &= e' \cdot k + a_x \cdot k + E - e \cdot s. && (3.9) \end{aligned}$$

Then, as the polynomial y_x can be obtained from y as:

$$y_x = \left\lfloor \frac{p}{q} \cdot (d_x - c \cdot s) \right\rceil = \left\lfloor \frac{p}{q} \cdot a_x \cdot k \right\rceil. \tag{3.10}$$

In the expanded equation for y, we notice that it contains the polynomial $a_x \cdot k$ and and a noise polynomial $e' \cdot k - e \cdot s + E$. Therefore, the last equation, showing the value of y_x, is correct with all but a negligible probability $negl(\kappa)$ if the noise polynomial $\left|\frac{p}{q} \cdot (e' \cdot k - e \cdot s + E)\right|$ for is small enough for each coefficient to achieve acceptable correctness after rounding, in other words:

$$\left|\frac{p}{q} \cdot (e' \cdot k - e \cdot s + E)\right|_\infty \ll \mathbb{Z} + \frac{1}{2}. \tag{3.11}$$

Additionally, before actually performing the rounding, it is necessary to represent the values that are to be rounded in $-q/2, ..., q/2$ form [Alb+21b].

The $negl(\kappa)$ probability is present due to the fact that there is always a chance, due to the aforementioned addition of noise, that an overflow can occur where the noise value is just large enough to cause rounding to the wrong integer.

3.3 Key Encapsulation Mechanism (KEM)

A simple definition of a KEM is that it is a mechanism which is used between two parties to securely communicate symmetric key material by using public-key cryptography. In the case of PQ-BRAKE this is used to generate ephemeral key pairs and an additional element γ that participates in the derivation of the symmetric session key ρ, along with the [Bau+22]: stored reference public key cpk_t for the Client who wishes to authenticate, Server's ephemeral public key spk_e, Server's pre-generated public key spk and client's ephemeral public key cpk_e.

A KEM is composed of three algorithms [Bau+22]:

- Key generation - for creating a key pair consisting of a public and a secret key.
- Encapsulation - taking as and input a public key and outputting a ciphertext, which is an encapsulated value that contains within it the secret shared value (γ in BRAKE and PQ-BRAKE) which can be used as key material.
- Decapsulation - taking as input the received ciphertext and a secret key, outputting the secret shared value which can then be safely used by the decapsulating entity.

3.3.1 CRYSTALS-Kyber

The specific post-quantum-secure KEM used in this thesis is the recently standardised CRYSTALS-Kyber [Ava+21], due to its high performance and existing reference implementation.

Kyber is based on the Module Learning With Errors (M-LWE) problem described in Section 2.2.4 and provides IND-CCA2 security [Ava+21]. There exist three versions of Kyber: Kyber512, Kyber768 and Kyber1024. The versions differ primarily in the key and ciphertext sizes and error distributions. They all share some main parameters though, namely $N = 256$ and $q = 3329$, which were specifically chosen for the ability to use the number-theoretic-transform (NTT) providing a very efficient way to perform multiplications in R_q [Ava+21].

In this thesis Kyber768 was chosen as it is recommended by the authors due to its optimal performance while still providing more than `128` bits of security [Ava+21].

3.4 Lattice-Based BRAKE

Having introduced all the necessary components, we can now assemble the full PQ-BRAKE protocol. First, a description of the enrolment functionality's most important elements, as shown in the protocol Figure 3.3:

- (lines 2-4) - Pre-existing knowledge each party has access to, including the assumption that the Evaluator has already committed to the evaluation key k by producing the R-LWE sample c.
- (line 5) - The fuzzy vault scheme output, consists of the random polynomial f and the locked vault V.
- (lines 6-10): The initial step of the OPRF, also known as blinding, where the polynomial f' is secured by creating the R-LWE sample c_x.
- (lines 11-13) - The evaluation step of the OPRF, where the blinded polynomial f in the form of the R-LWE sample c_x is evaluated with the evaluating key k and sent to the Client.
- (line 14) - The rounding step of the OPRF, also known as unblinding that if successful, outputs the evaluated polynomial f which can now be used as key material.
- (line 15) - Expansion of the key material into a suitable secret key.
- (line 16) - Generation of the public key from the secret key.
- (lines 17-19) - Storing the vault V, Client's public key and client's identification `id` at the Server for future authentication.

PQ-BRAKE Enrolment

	Client		**Server**		**Evaluator**
	t reference template		$\texttt{ssk} \in \mathcal{K}$		$c \in R_q$
	$\texttt{id}$ verified identity		$\texttt{spk} \in \mathcal{P}$		$k \in R_q$
	$c \in R_q$				
	$(f, V) \leftarrow \texttt{lock}(t)$				
	$s \leftarrow D_\sigma(R_q)$				
	$e' \leftarrow D_\sigma(R_q)$				
	$a_x := H(f)$				
	$c_x := a \cdot s + e' + a_x$				
		$\xrightarrow{c_x}$		$\xrightarrow{c_x}$	
					$E \leftarrow D_{\sigma'}(R_q)$
					$d_x := c_x \cdot k + E$
		$\xleftarrow{d_x}$		$\xleftarrow{d_x}$	
	$y_x := \lfloor d_x - c \cdot s \rceil_p$				
	$\texttt{csk}_t \leftarrow \texttt{ExpKDF}(y_x)$				
	$\texttt{cpk}_t \leftarrow \texttt{pkGen}(\texttt{csk}_t)$				
		$\xrightarrow{V, \texttt{cpk}_t, \texttt{id}}$			
			store		
			$(V, \texttt{cpk}_t, \texttt{id})$		

Figure 3.3: PQ-BRAKE enrolment protocol.

Next, a description of the verification functionality, as shown in the protocol Figure 3.4:

- (lines 2-5) - Pre-existing knowledge each party has access to, with the same assumption as in enrollment that the Evaluator has already commited to the evaluation key k by producing the R-LWE sample c.
- (lines 6-8) - The biometric component, where the Client sends a biometric claim id to the Server and receives a locked fuzzy vault V connected to that id on which he then performs the fuzzy vault unlocking operation and generates the random polynomial f.
- (line 9) - Beginning of the KEM, where the Client and the Server each generate an ephemeral key pair in preparation for a key exchange.
- (lines 10-13) - The initial step of the OPRF, also known as blinding, where the polynomial f' is secured by creating the R-LWE sample c_x.
- (line 14) - The Client transmits the R-LWE sample c_x along with its ephemeral public key to the Server, who forwards c_x to the Evaluator.
- (lines 15-17)
  - The Server performs an encapsulation of the stored reference public key tied to the client with the received `id` and derives the symmetric session key ρ from the concatenated: client's reference public key `cpk_t`, both ephemeral public keys `cpk_e,spk_e`, Server's pre-existing public key `spk` and shared secret value γ.
  - The Evaluator performs the evaluation step and produces the R-LWE sample d_x.
- (line 18) - The Evaluator transmits d_x to the Server who forwards it to the Client along with a hash of the session key ρ, its ephemeral public key `spk_e` and the ciphertext generated by the encapsulation procedure.
- (line 19) - Unblinding the evaluation.
- (lines 20-21) - Expanding the key material y_x into a usable secret key $\texttt{csk}_{t'}$ and constructing the corresponding public key $\texttt{cpk}_{t'}$ with `pkGen()`.
- (line 22) - The decapsulation step of the KEM, where the Client obtains a shared secret value γ' from the ciphertext `ctx` and the secret key $\texttt{csk}_{t'}$.
- (lines 23-24) - The Client produces a symmetric key ρ' in the same way that the Server produced ρ but the Client uses its own freshly generated key $\texttt{cpk}_{t'}$.
- (line 25) - The output of the protocol, comparing the hash of the Client's computed ρ' with the previously received hash of ρ. If the result is `true` and the hashes are identical, then the protocol execution was successful and the symmetric key ρ is established (user is authenticated).

PQ-BRAKE - Verification

	Client		**Server**		**Evaluator**
	t' probe feature vector		$\texttt{ssk} \in \mathcal{K}$		$c \in R_q$
	$\texttt{spk} \in \mathcal{P}$		$\texttt{spk} \in \mathcal{P}$		$k \in R_q$
	biometric claim $\texttt{id}$		$(V, \texttt{cpk}_t, \texttt{id})$		
	$c \in R_q$				
		$\xrightarrow{\texttt{id}}$			
		$\xleftarrow{V}$			
	$f' \leftarrow \texttt{unlock}(V, t')$				
	$(\texttt{csk}_e, \texttt{cpk}_e) \leftarrow \texttt{KeyGen}()$		$(\texttt{ssk}_e, \texttt{spk}_e) \leftarrow \texttt{KeyGen}()$		
	$s \leftarrow D_\sigma(R_q)$				
	$e' \leftarrow D_\sigma(R_q)$				
	$a_x := H(f')$				
	$c_x := a \cdot s + e' + a_x$				
		$\xrightarrow{c_x, \texttt{cpk}_e}$		$\xrightarrow{c_x}$	
			$(\texttt{ctx}, \gamma) \leftarrow \texttt{encap}(\texttt{cpk}_t)$		$E \leftarrow D_{\sigma'}(R_q)$
			$\rho \leftarrow \texttt{KDF}(\texttt{cpk}_t, \texttt{cpk}_e,$		$d_x := c_x \cdot k + E$
			$\texttt{spk}, \texttt{spk}_e, \gamma)$		
		$\xleftarrow[\texttt{spk}_e, ctx]{d_x, H(\rho)}$		$\xleftarrow{d_x}$	
	$y_x := \lfloor d_x - c \cdot s \rceil_p$				
	$\texttt{csk}_{t'} \leftarrow \texttt{ExpKDF}(y_x)$				
	$\texttt{cpk}_{t'} \leftarrow \texttt{pkGen}(\texttt{csk}_{t'})$				
	$\gamma' \leftarrow \texttt{decap}(\texttt{ctx}, \texttt{csk}_{t'})$				
	$\rho' \leftarrow \texttt{KDF}(\texttt{cpk}_{t'}, \texttt{cpk}_e,$				
	$\texttt{spk}, \texttt{spk}_e, \gamma')$				
	return $H(\rho') = H(\rho)$				

Figure 3.4: PQ-BRAKE verification protocol.

Chapter 4

Implementation

This chapter describes the specifics of how the PQ-BRAKE protocol was implemented, including the system setup, simulation architecture, abstractions and the details of the most important subroutines. The general idea for this implementation was to create a benchmarkable proof-of-concept implementation suited for simulation on a commodity notebook in order to prove correctness and feasibility of the theoretical concept.

4.1 System Setup

The implementation was done in the C++ programming language (version C++11).

Libraries that were used in the implementation are NTL [Sho+01], for number theory and polynomial arithmetic and OpenSSL [Fou04] for generic cryptographic operations.

A file structure that simulates different participants in the protocol, namely the Client, the Server and the Evaluator was used to represent a real world execution of the protocol. To this end, the functionality is split into two categories: the common cryptographic functions accessible to one or multiple protocol participants (files in the the **operations** folder) and the dedicated functionality accessible/used only by that specific participant (files in the **participants** folder). The latter category is organized in such a way that each protocol participant (seen in Figure 3.4) is represented by a class named after it, for example the *Client* and the functions it performs in the protocol are located in the *Client.cpp* file.

These objects are only a simulation and their functionality is still performed on one machine running a single executable that represents all communication between the protocol participants. In a real-world scenario the *Client* would be run on one machine while the *Server* and *Evaluator* would run on a different one, with the *Evaluator* separated on a hardware level but still on the same machine.

The testing code is located in a dedicated folder called **tests**.

The specifications of the machine used for testing are:

- CPU: AMD Ryzen 9 4900HS (8 core, 16 thread, 3 - 4.3 GHz)
- RAM: 16 GB
- OS: Linux (kernel 6.x)

4.2 NTL

The most frequently used library in the implementation is the aforementioned NTL [Sho+01] library for doing number theory.

Specifically, it was used for 4 main purposes: working with large integers (much larger than default data types like *int* or *long* allow), polynomials, rings and floating point calculations (on very large integers). Especially important was the library's support for very fast polynomial arithmetic which is at the core of the OPRF functionality, which is expanded upon in the following section.

First, to provide a brief explanation of the main NTL classes used in the implementation:

- *ZZ* - represents arbitrary length integers along with support for accompanying arithmetic and conversions to and from standard data types and other NTL classes.
- *ZZX* - polynomials with coefficients of type *ZZ* and polynomial arithmetic, used for storing polynomials outside of the ring setting.
- *ZZ_p* - large integers modulo integer q.
- *ZZ_pX* - polynomials with coefficients of type *ZZ_p* with support for polynomial arithmetic in the modulo q setting.
- *ZZ_pE* - ring extension of polynomials with *ZZ_p* coefficients, in simple terms it represents a *ZZ_pX* polynomial modulo another polynomial P (which is a cyclotomic polynomial $X^n + 1$ in this case).
- *RR* - represents arbitrary-precision floating point numbers, the precision is set to 150 bits though the specific computations performed do not require such precision.

A very useful feature of NTL are the conversion routines for these types. Most of them can be efficiently converted directly to/from standard data types, like *string* and *long*.

Some of the conversions between NTL's classes require more than one step. For example when retrieving a coefficient from a polynomial of type *ZZ_pE*, first the polynomial needs to be converted into *ZZ_pX* form, then the coefficient can be accessed and it itself needs to be turned into *ZZ* form and only then to a *RR* floating point number.

This could certainly be improved in an optimized implementation made for real-world use (without using NTL), but it does not impact the performance of this implementation to the degree that it would offset the practical benefit of using NTL's built-in classes.

Apart from the data type classes mentioned above and the arithmetic using these, some NTL-specific functions were used and they will be described in the following sections.

4.3 Lattice-OPRF

The largest piece of the implementation is the lattice-based OPRF. This constitutes the main contribution to the implementation part of this thesis, as it was constructed based on the theoretical design of the OPRF construction described in the original paper [Alb+21b] and such an implementation is not readily available in a library or other code-base. It features three main components:

- Random sampling (Section 4.3.1.)
- Hashing into the lattice (Section 4.3.2.)
- Rounding (Section 4.3.3.)

These can be seen on the protocol shown on Figure 3.2., where the random sampling is used multiple times to generate values of $a, k, e, ...$, the hashing to determine a_x and the rounding to calculate y. Each of these components will be expanded upon in the following sections.

4.3.1 Random Sampling

The random sampling functionality is required specifically to generate the values of:

- a - publicly available uniformly random polynomial with coefficients from $[0, q-1]$.
- k - secret OPRF key, small polynomial with ternary ($[-1, 0, 1]$) coefficients, uniformly sampled.
- e - RLWE error value for commitment to the OPRF key k, small polynomial with ternary ($[-1, 0, 1]$) coefficients, uniformly sampled.
- s - client's RLWE secret, small polynomial with ternary ($[-1, 0, 1]$) coefficients, uniformly sampled.
- e' - client's RLWE error value, small polynomial with ternary ($[-1, 0, 1]$) coefficients, uniformly sampled.
- E - large error value that obscures k from the client, polynomial with large integers from a different distribution ($[-B, B]$).

Everything is sampled uniformly, as discrete Gaussian sampling is not necessary in this implementation, in [Alb+21b] it was used for the mathematical benefits when performing the proofs. Note, in all of these instances, negative values are represented in the calculations by $q - value$, as we are using a mod q setting.

The function used to sample the small polynomials (k, e, s, e') is called *sampleSmallUniformPolynomial* and is located in operations/*Crypto.cpp*. The operations it performs are the following:

1. Initializes a pseudo-random number generator object based on the Mersenne Twister algorithm [MN98] which is implemented by default in C++.
2. Declares the random distribution to be uniform between two bounds which form the input to the function.
3. Produces a pseudo-random number from the desired distribution using the generator object.

The large noise value E is sampled uniformly from a different error distribution [Alb+21b]. It needs to be sufficiently large to drown the value of the OPRF key k from the RLWE sample in d_x but not large enough to introduce too much noise when combined with the other errors already that are already present. This is done through an NTL provided function, *ZZ.RandomBnd* to output a pseudorandom number between 0 and $2 \cdot B - 1$ and then substracted with B to achieve the desired range $[-B, B]$, where B is a boundary for which $B << q$ is true. The upper limit for the value of B is $||E||_\infty \leq 2N \cdot 2^{40}$.

4.3.2 Hashing into the Lattice

As mentioned in Section 3.1., the original $A^F(x)$ function used in [Alb+21b] is replaced with a SHA-256 hash of the input (the fuzzy vault opening candidate polynomial) which is then placed into a ring element (a polynomial). This is done through the following steps of the function *compute_a_x* in *Client.cpp*:

1. The coefficients of the fuzzy vault opening candidate polynomial are concatenated and a hash digest h is created from the resulting string using SHA-256 (specific implementation of this is from the *openssl* library).

2. To create a polynomial that is in the ring we want, every coefficient $a_x[i], i = 0, ..., N-1$ is produced by concatenating its index with h and hashing the resulting string with SHA-256 as SHA256(0 $\parallel$ h), SHA256(1 $\parallel$ h),...).

3. Those hash digests are then converted from their hexadecimal representations into decimal integer representations through NTL's ZZFromBytes() function. The mechanism of this conversion works by summing the *unsigned char* values of each hex character multiplied by a power of 256 corresponding to the index of the character. In short, $x = sum(\mathrm{p}[i] \cdot 256^i, i = 0..N-1)$ where p[i] is a character from the input string and n is the length of the hash digest. This produces at most a 2^{154} bit integer which can still be represented by NTL's big integer class ZZ.

4. Lastly, each coefficient is converted into *ZZ_p* (integer values modulo q) which form the coefficients of a polynomial from the ring/lattice (*ZZ_pE* data type).

This procedure is deterministic and works only in one direction due to the use of hashing. In this way, the value of the original fuzzy vault opening candidate polynomial is further obscured from the Evaluator, even before the addition of RLWE values (blinding).

4.3.3 Rounding

In the final step of the OPRF, the Client needs to round the value received as an RLWE sample from the Evaluator substracted by $c \cdot s$ in order to compute the evaluated input $\lfloor a_x \cdot k \rceil$. The formulas that describe the reasoning behind this are given in the section 3.2.3.

To achieve this in the implementation, first the coefficients $y[i]$ for $i = 0, ..., N-1$ of the polynomial y that is to be rounded are represented in a balanced manner (shifted by $\frac{q}{2}$ to be in the range $[-\frac{q}{2}, \frac{q}{2}]$). After that, the following operation is performed on the shifted values $y[i]$ for $i = 0, ..., N-1$, using division by $\frac{q}{p}$ instead of multiplication with $\frac{p}{q}$ for a slight optimization due to minimizing the impact of floating point representation (since $p = 2$):

$$y_x[i] = \left\lceil \frac{y[i]}{\frac{q}{p}} - 0.5 \right\rceil \text{ for } i = 0, ..., N-1. \tag{4.1}$$

In this way the values are consistently rounded down on ties, as generally the rounding functions provided by NTL are round-to-even and as such that do not always provide the desired behaviour. The rounding output is a ternary polynomial with coefficients $[0, 1, q-1]$. When checking if the OPRF was successfully executed, if $\lfloor y \rceil = \lfloor a_x \cdot k \rceil$, a modulo of 2 is applied to the values to remove the $q-1$ instances which results in a uniform post-rounding output distribution, with equal chances of coefficients being rounded to 0 and 1.

4.3.4 Oblivious Pseudo-Random Function (OPRF)

The operations that are part of the OPRF procedure are done using polynomial arithmetic in the ring context provided by NTL's *ZZ_pE* class. To do this, every value that is part of these calculations is first converted to *ZZ_pE*, if it was not already in that form. The computations done by the Client are done in the methods of the Client class, same with the Evaluator and the final value, the rounded input evaluated with the Evaluator's key k is given as the output of the $OPRF()$ function.

4.3.5 Parameters

The parameters relevant for the OPRF implementation are defined as constant variables in the *parameters.hpp* header. They are the following:

- q - the ring modulus, is a large prime number, type *ZZ*.
- N - degree of polynomials, dimension for RLWE samples, type *long*.
- B - bound for the drowning value E, type *ZZ*.
- p - rounding modulus, type *long*.

The validity and performance of the parameters used was checked using the sagemath [Theyy] module called lwe-estimator [APS15] which estimates the security of different LWE instances based on parameter choices (N, bit-size of q and standard deviation). N and q are varied but the standard deviation is set to $\sqrt{2/3}$ which emulates a uniform distribution. The aimed-for level of security was to be as close to 2^{128} operations needed to break security as possible, with values generaly above 2^{100} considered acceptable.

As per the requirements described in Section 4.3.3., the parameters which impact the noise need to be carefully chosen in order for the system to work properly (to round correctly reliably and consistently but still provide a good level of security). The upper bound for this noise term can be determined by calculating the infinity-norm for the entire term and assuring that it fulfills the following condition (for $p = 2$):

$$\left|e' \cdot k - e \cdot s + E\right|_\infty \ll \frac{q}{4}. \tag{4.2}$$

First, to define the norms used in these calculations [Bau+18]: $||f||_1 = \sum_i |f_i|$, $||f||_2 = (\sum_i |f_i|^2)^{1/2}$, $||f||_\infty = \max_i |f_i|$.

The value of the infinity-norm for a polynomial $f \in R_q$ is defined as $||f||_\infty = \max_i |f_i|$, for E this is equal to B, as that is how the E polynomial is sampled.

The infinity-norm values for the two remaining products are calculated in a different way. Since these values are all elements of the ring $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, the following two inequalities need to stand [Bau+18]:

$$||f||_1 \leq \sqrt{(N)} \cdot ||f||_2 \leq N \cdot ||f||_\infty \tag{4.3}$$

$$||f||_\infty \leq ||f||_1. \tag{4.4}$$

These do indeed stand for our choice of e, e', s, k and B. Furthermore, due to the choice of the specific ring polynomial $X^N + 1$ [Mic07] we can have the following two bound expressions [Bau+18]:

1. If $||f||_\infty \leq \beta, ||g||_1 \leq \gamma$ then $||f \cdot g||_\infty \leq \beta \cdot \gamma$.
2. If $||f||_2 \leq \beta, ||g||_2 \leq \gamma$ then $||f \cdot g||_\infty \leq \beta \cdot \gamma$.

So the resulting sum of infinity norms for the entire noise term is:

$$|e' \cdot k|_\infty + |e \cdot s|_\infty + |E|_\infty = N + N + B \ll \frac{q}{4}. \tag{4.5}$$

If the value of B is $2N \cdot 2^{40}$, the values of N are mostly inconsequential since $B \ll q$ and this condition is indeed fulfilled.

4.4 Lattice-based KEM

The implementation of the key encapsulation mechanism uses the CRYSTALS-Kyber KEM implementation provided by Open Quantum Safe (OQS) [SM16]. The original implementation is coded in C, but OQS also provides a C++ wrapper which was used here.

As mentioned in Section 3.3, Kyber has three functionalities:

- Key generation
- Encapsulation
- Decapsulation

These functions have corresponding implementations in the `liboqs` library, which are utilised in the Kyber() function located in *Crypto.cpp*.

The underlying cryptographic algorithms used in these functions are, as per the CRYSTALS-Kyber specification [Ava+21]:

- SHA3-256 and SHA3-512 for producing hashes.
- SHAKE-128 for expandable output function.
- SHAKE-256 as the key derivation function.
- SHAKE-256 also as a PRF.

The Kyber() function in this implementation is used for benchmarking purposes and performs one instance of a KEM using a randomly generated keypair.

Chapter 5

Performance

This chapter describes the expected and experimentally evaluated security and performance of the PQ-BRAKE protocol. The experimental evaluation was performed with separate tests for two of the three main components: Lattice-OPRF (based on [Alb+21b]) and CRYSTALS Kyber KEM [Ava+21] while the performance of the improved fuzzy vault scheme [Tam16] was not tested, since it is used in the protocol without modification. The improved fuzzy vault performance details shown in the following sections originate in the work that describes the classically secure BRAKE protocol [Bau+22].

Experiments were run on a consumer grade notebook with an AMD Ryzen 9 4900HS CPU@3-4.3 GHz and 16 GB of RAM. Separate test files are provided in the code which test the Lattice-OPRF and KEM functionalities respectively.

As described in Section 4.1., open-source libraries were used in the implementation of cryptographic functions, namely the `OpenSSL` implementation of the SHA-256 hashing algorithm, Open Quantum Safe's `liboqs` C library [SM16] through its C++ wrapper, `liboqscpp`, for the CRYSTALS Kyber implementation.

5.1 Passive Security

As in the context of this thesis, the implementation of the lattice-based OPRF leaves out the guarantees of active security in pursuit of practicality, thus only passive security can be evaluated. An example of a possible situation which could cause an issue and is not handled in a passive security setting is a Client who sends an R-LWE sample with an error composed entirely of zeroes. To reiterate, the zero-knowledge proofs which would provide active security to the OPRF element of the PQ-BRAKE protocol are available and are described in the work on which this implementation is based on [Alb+21b] but have large space requirements which are incompatible with practical applications.

5.1.1 LWE Estimator

For the OPRF part of the protocol, parameter choice is crucial for both communication and computation complexity along with security, and needs to be carefully evaluated. The relevant parameters are:

- q - Modulus, large prime.
- N - Ring dimension, power-of-two.
- p - Rounding modulus, set to $p = 2$.

Testing of the parameter validity was done through the program mentioned in Section 4.3.5., `lwe-estimator` [APS15]. Inputting the desired parameters into this program results in an estimated number of ring operations required to break the security of R-LWE. The tool supports the following attack algorithms:

- Meet-in-the-middle exhaustive search.
- Coded-BKW [GJS15].
- Dual-lattice attack and sparse secret variant [Alb17].
- Lattice-reduction and enumeration [LP11].
- Primal attack via uSVP [AFG14; BG14].
- Arora-Ge algorithm [AG11] using Gröbner bases [Alb+14].

The resulting number of ring operations (ROP), directly corresponds to the bit security for the given parameter choice as: $\log_2(ROP)$. In practical terms, this means that when choosing parameters, the aim is to keep this security level above `100` bits, while having the:

- Largest possible accumulated error rate - an increase in this corresponds to improved worst-case hardness theorem conclusions [CP16].
- Smallest possible q - that supports the large accumulated error rate (fulfills the condition laid out in Equation 4.5.
- Balanced N - increases security but also communication and computation complexity.

The desired security level of over `100` bits is based on the lattice-sieving class of algorithms [Bec+16] for solving the SVP.

Using these parameters, it is also possible to calculate a probability of the rounding step failing, which would result in a decryption failure in practice, due to noise wrapping the value around $\mathbb{Z} + 1/2$ and causing a rounding to the wrong value. As demonstrated in Equation 4.5, the upper bound on the noise is given as: $2N + B \leq \frac{q}{4}$. We consider the probability of one coefficient of the output polynomial y_x being wrongly decrypted to be: $\frac{2N+B}{q}$, and its complement situation, the probability of no error occurring as: $1 - \frac{2N+B}{q}$. With this in mind, we claim that the probability of at least one decryption error occurring during the rounding of N polynomial coefficients and thus the protocol failing in the OPRF step, to be:

$$1 - (1 - \frac{2N + B}{q})^N. \tag{5.1}$$

Applying this formula, we set the parameters so that the failure rate is significantly smaller than the false-accept security of the biometric component, i.e., the improved fuzzy vault scheme. A success rate of 99.9% was chosen for this benchmark.

5.1.2 KEM

The KEM is evaluated independently of the OPRF functionality, as its performance is effectively separate and not impacted by the amount of data or parameters used in the OPRF. Since the KEM that is being used is the recently standardised CRYSTALS-Kyber, its security level and communication requirements are given by the authors [Ava+21].

Specifically, the variant used in this thesis was the Kyber768 variant with the estimated security of at least 128 bits and the following sizes as shown in Table 5.1.:

Secret key	2400 bytes
Public key	1184 bytes
Ciphertext	1088 bytes

Table 5.1: Kyber768 Key Sizes.

In the two functionalities of the protocol, the following amounts of data are stored and sent:

- Enrollment - shown in Figure 3.3, one key pair generated (Client's), one public key is sent (1184 B).

- Verification - shown in Figure 3.4, two key pairs generated, two public keys are communicated to another participant (1184 bytes each), along with a ciphertext (1088 B).

5.2 Benchmarking Results

5.2.1 Modified Lattice-OPRF

With the aforementioned conditions for parameter choices, the following representative sets of parameters were benchmarked. All of them are parameterised in such a way that they are as close as possible to the desired 99.9% decryption success rate.

1. Scenario A - an impractical but highly secure option - $N = 8192$, $q \approx \log_2(2^{77})$, $B = 2^{54}$ with estimated security 408 bits.

2. Scenario B - a faster but less secure option - $N = 2048$, $q \approx \log_2(2^{73})$, $B = 2^{52}$ with estimated security 95 bits.

3. Scenario C - a reasonable middle-ground option - $N = 4096$, $q \approx \log_2(2^{75})$, $B = 2^{53}$ with estimated security 188 bits.

The following results were obtained by running the OPRF, for each set of parameters, 10,000 times and noting the average execution times for each part of the protocol. Each test (for every parameter set) used the same randomly generated fuzzy vault polynomial f in all 10,000 iterations, but every value besides f is randomly sampled in each iteration. Strictly speaking, the publicly available polynomial a would not need to be freshly sampled if a user were to try to authenticate multiple times in a row, but the experiment was set up in this way for a better representation of the worst case performance.

$N = 8192$	$q \approx \log_2(2^{77})$	$B = 2^{54}$
Sampling a	1.96 ms	
Sampling k	1.46 ms	
Sampling e	1.46 ms	
Compute c	5.6 ms	
Sampling s	1.49 ms	
Sampling e'	1.46 ms	
Compute a_x	24.17 ms	
Compute c_x	5.71 ms	
Sampling E	3.14 ms	
Compute d_x	5.47 ms	
Compute y	5.47 ms	
Rounding y	5.99 ms	
Total OPRF runtime	63.64 ms	
Security level	408 bits	

Table 5.2: Computational Performance of PQ-BRAKE Scenario 1.

$N = 2048$	$q \approx \log_2(2^{73})$	$B = 2^{52}$
Sampling a	0.52 ms	
Sampling k	0.40 ms	
Sampling e	0.39 ms	
Compute c	1.13 ms	
Sampling s	0.39 ms	
Sampling e'	0.39 ms	
Compute a_x	6.25 ms	
Compute c_x	1.15 ms	
Sampling E	0.81 ms	
Compute d_x	1.09 ms	
Compute y	1.09 ms	
Rounding y	1.52 ms	
Total OPRF runtime	15.24 ms	
Security level	95 bits	

Table 5.3: Computational Performance of PQ-BRAKE Scenario 2.

$N = 4096$	$q \approx \log_2(2^{75})$	$B = 2^{53}$
Sampling a	1.01 ms	
Sampling k	0.74 ms	
Sampling e	0.73 ms	
Compute c	2.75 ms	
Sampling s	0.74 ms	
Sampling e'	0.73 ms	
Compute a_x	12.24 ms	
Compute c_x	2.78 ms	
Sampling E	1.58 ms	
Compute d_x	2.66 ms	
Compute y	2.65 ms	
Rounding y	3.01 ms	
Total OPRF runtime	31.81 ms	
Security level	188 bits	

Table 5.4: Computational Performance of PQ-BRAKE Scenario 3.

As is visible from the results shown in Tables 5.2, 5.3 and 5.4, the computation of a_x is the single most time consuming step of the OPRF procedure in each scenario. We see that the amount of time it takes is directly dependent on the ring dimension N, so the number of coefficients that need to be hashed in order to hash the randomly generated fuzzy vault opening candidate polynomial f into a ring element. In terms of implementation, this depends on the hash rate which is 0.003 milliseconds per SHA256 hash operation on the test machine. The degree of the f polynomial itself does not have a significant impact as it only changes the length of the input for the first hash operation and thus can be considered negligible in terms of performance.

Furthermore, as expected, the sampling generally slows down when using larger distributions, however an additional performance penalty is present when sampling the drowning error E, due to the implementation invoking an additional arithmetical operation in the sampling process to compute a value in the range $[-B, B]$. This could be optimized through the use of a specifically adapted sampling function which supports sampling numbers centered around zero.

5.2.2 CRYSTALS-Kyber KEM

The KEM performance was benchmarked with a larger number of 1,000,000 iterations due to the high efficiency of the OQS implementation [SM16].

Notably, the parameter set of Kyber768 is using much smaller parameters than the ones used in the modified lattice-OPRF. Additionally, Kyber is based on a modification of the R-LWE problem called Module-LWE. In short this turns the a value in an R-LWE sample from a polynomial into a matrix over a constant-size polynomial ring and the s and e variables in an R-LWE sample into vectors over the same ring [Ava+21]. Using such a variation of R-LWE results in better scalability, reduced communication cost and possibly performance on par with R-LWE (when encrypting messages of a fixed size of 256 bits) [Ava+21].

The benchmark results and parameter set are shown in the following Table 5.5:

$N = 256$	$q = 3329$	$p = 3$
KeyGen	0.018 ms	
Encap	0.021 ms	
Decap	0.015 ms	
Total KEM runtime	0.063 ms	
Security level	$\sim$ 128 bits	

Table 5.5: Kyber768 Parameter Set [Ava+21].

5.2.3 PQ-BRAKE

Combining the results of the previous benchmarks we can show an estimate of the performance of PQ-BRAKE's verification functionality. Notably, this benchmark does not include the generation, hashing and comparison of the symmetric session keys ρ and ρ' as these are not included in the performance evaluation of clasically secure BRAKE [Bau+22] and are dependent on the choice of key derivation function and hashing algorithm. The results are shown in the Table 5.6:

	Polynomial degree					
	6	8	10	12	14	16
Feature extraction and preprocessing			200.59			
unlock	112.24	185.99	276.37	385.26	511.91	694.87
OPRF (1)			63.64			
OPRF (2)			15.25			
OPRF (3)			31.81			
KeyGen Client ephemeral			0.018			
KeyGen Client reference			0.018			
KeyGen Server ephemeral			0.018			
encap			0.021			
decap			0.015			
verification (1)	376.56	451.31	542.69	652.58	780.23	964.19
verification (2)	328.17	402.92	494.30	604.19	731.84	915.80
verification (3)	344.73	419.48	510.86	620.75	748.40	932.36
FMR (%)	1.04%	0.04%	0.00%	0.00%	0.04%	0.09%
1-FNMR (%) (1)	92.88%	88.79%	81.97%	73.18%	60.45%	44.09%
Estimated security in bits	17	23	29	36	44	-

Table 5.6: PQ-BRAKE Verification Performance in Milliseconds (Extending Upon [Bau+22]).

In PQ-BRAKE, as in classically-secure BRAKE [Bau+22], the biometric feature extraction and preprocessing along with the fuzzy vault unlocking have the largest impact on the execution time. These elements are dependent on the polynomial degree $\tau - 1$ of the fuzzy vault random polynomial f where τ is the threshold for a biometric decision, with higher values indicating a lower tolerance for biometric template differences. Additionally, the unlocking procedure run time scales with the degree of the random polynomial f but all other procedures are independent of it.

5.3 Comparison With Classically Secure Implementation

In Table 5.7 the performance penalties are shown for the three different scenarios of PQ-BRAKE's verification functionality. The performance is represented relative to the verification performance of classically-secure BRAKE [Bau+22].

For classically-secure BRAKE, the communication costs are calculated to be [Bau+22]:

- `99` bytes - for locked fuzzy vault of degree at most 43 and coefficients in $\mathbb{F}_{2^{18}}$
- `32` bytes per elliptic curve point - for keys and blinded and evaluated OPRF input
- `32` bytes - for the hash digest

In total, this brings the communication cost of the BRAKE verification to `0.3` KB.

For PQ-BRAKE the communication costs, differing for each scenario, are the following:

	Polynomial degree					
	6	8	10	12	14	16
Classical security						
verification	313.40	387.15	477.53	586.42	713.07	896.03
Post-quantum security						
verification (1)	+20.15%	+16.57%	+13.65%	+11.28%	+9.42%	+7.61%
verification (2)	+4.71%	+4.07%	+3.51%	+3.03%	+2.63%	+2.21%
verification (3)	+10.00%	+8.35%	+6.98%	+5.85%	+4.95%	+4.05%

Table 5.7: PQ-BRAKE Verification Performance Penalty in Milliseconds, Relative to Classically Secure BRAKE [Bau+22].

Scenario	(1)	(2)	(3)	Albrecht et al. [Alb+21b]
Locked fuzzy vault		99 B		
OPRF (3 RLWE samples)	234 KB	55.5 KB	114 KB	1536 KB
Kyber keys		4672 B		
Hash digest		32 B		
Total communication cost	238.7 KB	60.2 KB	118.7 KB	1540.7 KB

Table 5.8: PQ-BRAKE Communication Costs.

Evident from Table 5.8 is the fact that PQ-BRAKE introduces a large communication cost increase even in the optimistic, faster situation (2). This increase is dependent mostly on the ring dimension N, as increasing q by one only increases the size of a R-LWE sample by N bits.

Chapter 6

Conclusion

This thesis presented a post-quantum secure modification of the Biometric Resilient Authenticated Key Exchange (BRAKE) protocol [Bau+22] using lattice-based cryptographic components. As the main contribution, this included the modification of an OPRF mechanism and the usage of the recently standardised CRYSTALS-Kyber KEM.

Furthermore, these components have been successfully implemented in the C++ programming language using open-source cryptographic implementations and libraries and performance tested on a commodity notebook.

An exploration of possible parameter choices for the OPRF has also been performed since the choice of these directly and majorly impacts the computation and communication performance of the protocol, along with the security itself. This is especially relevant as the topic is currently an open problem and is widely discussed in the scientific community.

The obtained results of the performance testing suggest that the real-time efficiency of the classically-secure BRAKE [Bau+22] is also accomplished by the PQ-BRAKE protocol implemented in this thesis.

The biggest variable impacting the computational performance remains the biometric part of the protocol. While the post-quantum-security-providing elements were found to be be slower than their classically-secure counterparts as expected, relative to the biometric part, they still contribute less than 10% to the overall execution time for the best parameter choice.

The experiments also demonstrate the unavoidable communication cost increase, going from requiring 0.3 KB in BRAKE to requiring 60.2 KB, 118.7 KB or 238.7 KB in PQ-BRAKE, depending on the parameters chosen.

Further research can be directed at improving the security of the proposed PQ-BRAKE protocol. For instance, a modification to the PRF was made in the name of practicality which requires an honest-but-curious security model. The original work by Albrecht et al. [Alb+21b] does give an active security guarantee. If an improvement to its OPRF implementation manages to retain this guarantee while achieving the performance comparable to performance reported in this thesis, it would be a step closer to being ready to use in practice.

Bibliography

[AD17] Martin R Albrecht and Amit Deo. "Large modulus ring-LWE ¿= module-LWE". In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*. Springer. 2017, pp. 267–296.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. "On the Efficacy of Solving LWE by Reduction to Unique-SVP". In: *ICISC 13: 16th International Conference on Information Security and Cryptology*. Ed. by Hyang-Sook Lee and Dong-Guk Han. Vol. 8565. Lecture Notes in Computer Science. Seoul, Korea: Springer, Heidelberg, Germany, 2014, pp. 293–310. DOI: 10.1007/978-3-319-12160-4_18.

[AG11] Sanjeev Arora and Rong Ge. "New Algorithms for Learning in Presence of Errors". In: *ICALP 2011: 38th International Colloquium on Automata, Languages and Programming, Part I*. Ed. by Luca Aceto, Monika Henzinger, and Jiri Sgall. Vol. 6755. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, Heidelberg, Germany, 2011, pp. 403–415. DOI: 10.1007/978-3-642-22006-7_34.

[Ajt96] Miklós Ajtai. "Generating hard instances of lattice problems". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108.

[Alb+14] Martin R. Albrecht et al. *Algebraic Algorithms for LWE*. Cryptology ePrint Archive, Report 2014/1018. https://eprint.iacr.org/2014/1018. 2014.

[Alb17] Martin R. Albrecht. "On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL". In: *Advances in Cryptology – EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, 2017, pp. 103–129. DOI: 10.1007/978-3-319-56614-6_4.

[Alb+21a] Martin Albrecht et al. "Homomorphic encryption standard". In: *Protecting Privacy through Homomorphic Encryption*. Springer, 2021, pp. 31–62.

[Alb+21b] Martin R Albrecht et al. "Round-optimal verifiable oblivious pseudorandom functions from ideal lattices". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2021, pp. 261–289.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. *On the concrete hardness of Learning with Errors*. Cryptology ePrint Archive, Paper 2015/046. https://eprint.iacr.org/2015/046. 2015. URL: https://eprint.iacr.org/2015/046.

[Ava+21] Roberto Avanzi et al. "CRYSTALS-Kyber algorithm specifications and supporting documentation". In: *NIST PQC Round* 3 (2021), pp. 1–43.

[Bas+21] Andrea Basso et al. *Cryptanalysis of an oblivious PRF from supersingular isogenies*. Cryptology ePrint Archive, Paper 2021/706. https://eprint.iacr.org/2021/706. 2021. URL: https://eprint.iacr.org/2021/706.

[Bas23] Andrea Basso. *A Post-Quantum Round-Optimal Oblivious PRF from Isogenies*. Cryptology ePrint Archive, Paper 2023/225. https://eprint.iacr.org/2023/225. 2023. URL: https://eprint.iacr.org/2023/225.

[Bau+18] Carsten Baum et al. "More efficient commitments from structured lattice assumptions". In: *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings*. Springer. 2018, pp. 368–385.

[Bau+22] Pia Bauspieß et al. *BRAKE: Biometric Resilient Authenticated Key Exchange*. Cryptology ePrint Archive, Paper 2022/1408. https://eprint.iacr.org/2022/1408. 2022. URL: https://eprint.iacr.org/2022/1408.

[Bec+16] Anja Becker et al. "New directions in nearest neighbor searching with applications to lattice sieving". In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2016, pp. 10–24.

[BG14] Shi Bai and Steven D. Galbraith. "Lattice Decoding Attacks on Binary LWE". In: *ACISP 14: 19th Australasian Conference on Information Security and Privacy*. Ed. by Willy Susilo and Yi Mu. Vol. 8544. Lecture Notes in Computer Science. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, 2014, pp. 322–337. DOI: 10.1007/978-3-319-08344-5_21.

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. *Oblivious Pseudorandom Functions from Isogenies*. Cryptology ePrint Archive, Paper 2020/1532. https://eprint.iacr.org/2020/1532. 2020. DOI: 10.1007/978-3-030-64834-3_18. URL: https://eprint.iacr.org/2020/1532.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. "Pseudorandom functions and lattices". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 719–737.

[CHL22] Silvia Casacuberta, Julia Hesse, and Anja Lehmann. "SoK: Oblivious Pseudorandom Functions". In: *Cryptology ePrint Archive* (2022).

[CP16] Eric Crockett and Chris Peikert. "Challenges for ring-LWE". In: *Cryptology ePrint Archive* (2016).

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2004, pp. 523–540.

[Erw+20] Andreas Erwig et al. "Fuzzy asymmetric password-authenticated key exchange". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 761–784.

[Eur16] European Parliament. *EU Regulation 2016/679 of the European Parliament and of the Council (General Data Protection Regulation)*. 2016.

[FK00] Warwick Ford and Burton S Kaliski. "Server-assisted generation of a strong secret from a password". In: *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*. IEEE. 2000, pp. 176–180.

[Fou04] OpenSSL Software Foundation. *OpenSSL: A toolkit for general-purpose cryptography and secure communication*. 2004.

[Fre+05] Michael J Freedman et al. "Keyword search and oblivious pseudorandom functions". In: *Theory of Cryptography Conference*. Springer. 2005, pp. 303–324.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *Journal of the ACM (JACM)* 33.4 (1986), pp. 792–807.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. "Coded-BKW: Solving LWE Using Lattice Codes". In: *Advances in Cryptology – CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2015, pp. 23–42. DOI: 10.1007/978-3-662-47989-6_2.

[IK97] Yuval Ishai and Eyal Kushilevitz. "Private simultaneous messages protocols with applications". In: *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*. IEEE. 1997, pp. 174–183.

[ISO22] ISO/IEC JTC1 SC27 Security Techniques. *ISO/IEC 24745:2022. Information Technology - Security Techniques - Biometric Information Protection*. International Organization for Standardization. 2022.

[JL09] Stanisław Jarecki and Xiaomin Liu. "Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection". In: *Theory of Cryptography Conference*. Springer. 2009, pp. 577–594.

[JS06] Ari Juels and Madhu Sudan. "A fuzzy vault scheme". In: *Designs, Codes and Cryptography* 38.2 (2006), pp. 237–257.

[JW99] Ari Juels and Martin Wattenberg. "A fuzzy commitment scheme". In: *Proceedings of the 6th ACM conference on Computer and communications security*. 1999, pp. 28–36.

[KHB21] Roman Kessler, Olaf Henniger, and Christoph Busch. "Fingerprints, forever young?" In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 8647–8654.

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *Topics in Cryptology – CT-RSA 2011*. Ed. by Aggelos Kiayias. Vol. 6558. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, 2011, pp. 319–339. DOI: 10.1007/978-3-642-19074-2_21.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2010, pp. 1–23.

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *Designs, Codes and Cryptography* 75.3 (2015), pp. 565–599.

[Lyu20] V Lyubashevsky. *Basic lattice cryptography: encryption and Fiat-Shamir signatures*. 2020.

[Mic07] Daniele Micciancio. "Generalized compact knapsacks, cyclic lattices, and efficient one-way functions". In: *computational complexity* 16 (2007), pp. 365–411.

[MN98] Makoto Matsumoto and Takuji Nishimura. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[MR07] Daniele Micciancio and Oded Regev. "Worst-case to average-case reductions based on Gaussian measures". In: *SIAM Journal on Computing* 37.1 (2007), pp. 267–302.

[MR09] Daniele Micciancio and Oded Regev. "Lattice-based Cryptography". In: *Post-Quantum Cryptography*. Ed. by Daniel J. Bernstein, Johannes Buchmann, and

Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191. ISBN: 978-3-540-88702-7. DOI: 10.1007/978-3-540-88702-7_5. URL: https://doi.org/10.1007/978-3-540-88702-7_5.

[NP01] Moni Naor and Benny Pinkas. "Efficient oblivious transfer protocols". In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. 2001, pp. 448–457.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. "Distributed pseudo-random functions and KDCs". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 327–346.

[Pei16] Chris Peikert. "A decade of lattice cryptography". In: *Foundations and trends® in theoretical computer science* 10.4 (2016), pp. 283–424.

[QCC18] Mingping Qi, Jianhua Chen, and Yitao Chen. "A secure biometrics-based authentication key exchange protocol for multi-server TMIS using ECC". In: *Computer methods and programs in biomedicine* 164 (2018), pp. 101–109.

[Rab05] Michael O Rabin. "How to exchange secrets with oblivious transfer". In: *Cryptology ePrint Archive* (2005).

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40.

[Reg10] Oded Regev. "The learning with errors problem". In: *Invited survey in CCC* 7.30 (2010), p. 11.

[Sho+01] Victor Shoup et al. "NTL: A library for doing number theory". In: (2001).

[SM16] Douglas Stebila and Michele Mosca. *Post-Quantum Key Exchange for the Internet and the Open Quantum Safe Project*. Cryptology ePrint Archive, Paper 2016/1017. https://eprint.iacr.org/2016/1017. 2016. URL: https://eprint.iacr.org/2016/1017.

[SS20] Arpita Sarkar and Binod Kumar Singh. "A Novel Session Key Generation and Secure Communication Establishment Protocol Using Fingerprint Biometrics". In: *Handbook of Computer Networks and Cyber Security*. Springer, 2020, pp. 777–805.

[Tam16] Benjamin Tams. "Unlinkable minutiae-based fuzzy vault for multiple fingerprints". In: *Iet Biometrics* 5.3 (2016), pp. 170–180.

[THB14] Viktor Taneski, Marjan Heričko, and Boštjan Brumen. "Password security—No change in 35 years?" In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2014, pp. 1360–1365.

[Theyy] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.8)*. `https://www.sagemath.org`. YYYY.

[Tre+19] Amos Treiber et al. "Privacy-preserving PLDA speaker verification using outsourced secure computation". In: *Speech Communication* 114 (2019), pp. 60–71.

[Wan+21] Mei Wang et al. "Biometrics-Authenticated Key Exchange for Secure Messaging". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 2618–2631.

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

https://www.compute.dtu.dk/english