A non-interactive key exchange based on ring-learning with errors

de Kock, B.B.

Award date:
2018

TU/e Technische Universiteit
Eindhoven
University of Technology

Master's Thesis in Computer Science & Engineering

Information Security Technology

A Non-Interactive Key Exchange based on Ring-Learning With Errors

Bor de Kock

Defense date: May 29th, 2018
Graduation committee: Prof. dr. T. Lange *(Coding Theory & Cryptology)*
Dr. B. Škorić *(Security Group)*
Drs. T. Attema *(TNO)*
Drs. ir. M.P.P. van Heesch *(TNO)*

Student ID 0783422

Abstract

Ongoing developments in the field of quantum computing make it necessary to develop new algorithms for most cryptographic purposes, a research field we call post-quantum cryptography. Ways to perform post-quantum key exchanges have been invented, for instance based on lattices. Ring-Learning With Errors (RLWE)-based exchanges are very promising, but always introduce a small discrepancy between the key material of the two parties. This can be solved using methods like a reconciliation mechanism (as in e.g. NewHope, Peikert) or an encrypt-and-encapsulate-mechanism (as in NewHope-Simple), but that always necessitates more round-trips. It also leads to protocols where the different parties perform different operations. In this thesis we investigate whether an RLWE-based key exchange is possible without these extra round-trips. The goal is to create an exchange which is similar in structure to the Diffie-Hellman key exchange. We do so by severely increasing the modulus of the ring used, thus allowing for large coefficients where the error only has a small influence on the leading bit. This has a strong negative influence on the hardness of the underlying lattice problem which we can compensate for by increasing the lattice dimension as well. In this thesis we explain this new post-quantum non-interactive key exchange protocol based on RLWE, give security and correctness proofs, perform an experimental analysis of the hardness of the RLWE problem for large parameters, and propose a parameter set that makes a secure non-interactive exchange possible. We show that it is possible to perform such an exchange with 256 bits of security and failure rate $\leq 2^{-64}$ with parameters $n = 4001$, $\log(q) = 99$ and $\sigma = 6.384$, resulting in keys that are approximately 48 kB each, making them significantly larger than in existing protocols, but definitely usable for real-life applications where minimizing the number of roundtrips is more important than message size efficiency. We also define a parameter set for 512 bits of security, which is still usable at 74 kB.

Acknowledgements

This thesis is the result of a research internship at TNO's Cyber Security and Robustness department in The Hague. I am very grateful for the opportunity to work in such an inspiring and fun environment and for all the things I got to learn.

The first people to thank for this project are Thomas Attema and Maran van Heesch, who have been the ones to steer me in the right direction, help me keep track of my goals and who provided mathematical and moral support where needed. This thesis would not be here without your help. I also want to thank the other wonderful people of the CSR department, especially my fellow interns, and the great crypto people at TU/e who welcomed me on the days I worked from Eindhoven. Christine and Lorenz deserve a honorable mention for all life advice and abuse of their employee coffee cards.

After starting the security master I was drawn towards the field of cryptology. Many thanks go to Tanja Lange, who supervised this thesis project, but who has also advised me on numerous other occasions in the past few years. Thanks to you I was able to spend four amazing months in the US, to travel to CCC in Hamburg and to the cryptography summer school in Croatia, but most of all you helped me find and explore a field I love. I also thank Boris Škorić for taking the time to be a part of my thesis committee.

This thesis also marks the end of my time at TU/e, where I became a different person and also learned many things not related to computer science or cryptography. For that I need to thank many people and I don't even know where to start. Thanks go out to Cursor, for asking me to write columns, to Kees Huizing for asking me to be a teaching assistant, to all of the people I was in the FR and UR with, to Marloes van Lierop for listening to my complaints, to Fred, Trees and Mirjam for all the coffee and for all others who gave me the opportunity to try new things and learn valuable skills that are not part of a normal university curriculum.

Apart from the (quasi-)serious business, I really need to thank all friends I made along the way. Without the wonderful phenomenon that is GEWIS, I wouldn't be who I am today. Thanks to all the friends I made there, specifically to GELIMBO and my friends in the 33rd GEWIS board, to my orchestra buddies at ESMG Quadrivium, to Glenn and

Jelte and to all the other wonderful friends I made through FSE, WISO, SNiC and other acronym-powered student organizations.

I also thank the friends, classmates and colleagues I had when I was in Amsterdam for my minor and in Philadelphia for my internship. Dropping everything to move to a different country turned out to be an awesome adventure, and the warm welcome made it completely impossible to get homesick or feel lost.

Of course there are many other people who made the past years unforgettable and made it possible to write this thesis. Thank you, many unmentioned friends, family members and others inside and outside TU/e.

I owe the fact that I have been able to reach this goal to my amazing parents, who have supported me unconditionally for my entire life. Thank you for everything you have done to make this possible. I promise to call you more often.

Writing this thesis is the final step towards the wonderful title *Master of Science*, or *Ingenieur*. Achieving this goal would not have been possible without Monique Tonino, my high school mathematics teacher. I am forever grateful for everything you did for me.

Bor de Kock
Eindhoven, May 2018

Contents

List of Definitions and Theorems

Notation Index

Some symbols and other nomenclature as used throughout the thesis.

Remark that we use the period for decimal separation: $5.000 \approx 5$ and $5,000 = 5000$.

Mathematical symbols

$\mathbf{a}$	A polynomial $a(x)$.
$\mathbf{a_A}$	The polynomial $a(x)$ belonging to user A.
$\|\mathbf{a}\|_\infty$	The infinity norm of $\mathbf{a}$.
$\mathbf{p}$	Public key polynomial.
$\mathbf{s}$	Secret key polynomial.
$\mathbf{e}$	Error value polynomial.
A, B, C	Alice, Bob and Charlie, communicating parties.
$G = \langle g \rangle$	Cyclic group G with generator g.
R_q	A ring of polynomials with modulus q.
$\log(x)$	The base-2 logarithm of x ($\log_2(x)$), unless specified otherwise.
c	A ciphertext.
m	A message or plaintext.
$\mathsf{Enc}_k(m)$	Encryption of message m with key k.
$\mathsf{Dec}_k(c)$	Decryption of ciphertext c with key k.
$\mathsf{Sign}_k(m)$	Signing of message m with key k.
$\mathsf{Verify}_k(c)$	Verifying the signature c with key k.

Abbreviations

AES	Advanced Encryption Standard.
DH	Diffie-Hellman Key Exchange.

ECC	Elliptic-Curve Cryptography.
ECDH	Elliptic-Curve based Diffie-Hellman exchange.
IETF	Internet Engineering Task Force.
IKE(v2)	Internet Key Exchange (version 2).
KEM	Key Encapsulation Mechanism.
KEP	Key Exchange Protocol.
KEX	Key Exchange.
KDF	Key Derivation Function.
MAC	Message Authentication Code.
NIKE	Non-Interactive Key Exchange.
NIST	National Institute for Standards in Technology.
NTRU	Cryptosystem based on N-th degree truncated polynomial rings.
(R)LWE	(Ring) Learning With Errors.
RSA	The Rivest–Shamir–Adleman public-key crypto system.
S/MIME	Secure/Multipurpose Internet Mail Extensions.
SSL	Secure Socket Layer (precursor of TLS).
TLS	Transport-Layer Security.
Tor	The Onion Router.

Chapter 1

Introduction

1.1 Motivation

Cryptography has existed for many centuries [32], but the development of modern, computer-based cryptosystems started in the 1970s [21] and really took flight when the internet was invented and subsequently became widespread in the 1990s. Since then, several symmetric and asymmetric cryptosystems have been developed [42] that are strong enough to hold up against modern attackers, even if they are armed with fast supercomputers and enormous budgets. Most traditional cryptographic schemes like RSA encryption [48] and the Diffie-Hellman exchange are now, however, facing the threat of attackers equipped with a universal quantum computer.

In 1994 Shor developed an algorithm [50] that enables a universal quantum computer — as soon as it is built — to break many traditional cryptographic schemes. This is because it is able to efficiently solve problems that are hard to solve with traditional computers. Two years later, Grover's algorithm [28] was published, enabling a quantum computer to gain a speed advantage while breaking almost all symmetric ciphers. Luckily, this last threat can be mitigated by increasing the key size.

A disastrous example of the effect Shor's algorithm has, is decreasing the security of RSA, which is based on the problem of factorization into primes, i.e. computing primes p and q given only $N = p \cdot q$. Currently this prime decomposition is very hard, but a universal quantum computer executing Shor's algorithm is able to perform it efficiently. Similarly a quantum computer can be used to compute discrete logarithms, which is also considered to be too hard to do with current computers for certain groups. The current hardness of this discrete logarithm problem is the basis of for instance elliptic-curve cryptography.

Although a quantum computer large enough to actually execute these algorithms for cryptographic sizes has not been devised yet, the future existence of such a contraption poses a big risk to the current cryptographic landscape today. If an attacker were to store encrypted messages and their related key exchanges now, he would be able to decrypt them as soon as he has access to a large-scale quantum computer. Currently these computers are expected to be ready in the late 2020s [14, 47], which is a problem for a lot of sensitive data (e.g. military tactics or health records).

These developments in the field of quantum computing have led to the conception of a research area called *Post-Quantum Cryptography (PQCrypto)*, aiming to develop cryptosystems that can resist attacks using these computers. In 2006 a first PQCrypto-conference was held in Leuven and in 2008 the book "Post-quantum cryptography" [14] was published, kicking off the search for new mathematical primitives that could be the basis for cryptosystems that are 'quantum-proof', i.e. that can resist attacks by adversaries who have access to quantum computers. That the search for a suitable post-quantum cryptography scheme is important was highlighted furthermore in 2016, when the National Institution for Standards in Technology (NIST) opened a competition [33] to find a suitable candidate for standardization. NIST standardization of an algorithm signifies a further step in the adaptation of post-quantum cryptography in real-world scenarios. At the time of writing, several candidate primitives have been submitted. Currently the main candidates are cryptographic schemes based on codes, isogenies, hashes, lattices and multivariate equations. [35]. However, there is still a lot to be done before one of these primitives is finalized, standardized and ready to be used by the internet community.

In this thesis we focus on *lattice problems*. The idea of using lattices for cryptology predates the large-scale initiatives related to post-quantum cryptography, as it has existed since 1996 [2], the same year the first public-key encryption scheme using lattices was invented [31]. Some lattice problems are considered infeasible to solve efficiently, even when using a quantum computer. The (Ring-)Learning With Errors problem has been a popular lattice problem for cryptology in recent years [4, 9, 10, 46, 52], and is currently mature enough to be used for real-world applications [37].

Although switching from a pre-quantum to a post-quantum algorithm is very important, there are also possible caveats. For the post-quantum algorithms we currently assume that they are resistant to attacks using a quantum computer, but they are all relatively new and they do not have the advantage of all the years of security research that went into the traditional algorithms that are the standard today. RSA, for instance, has existed since 1977 [48] and the hardness of integer factorization has been researched even before that. When it comes to those, we are reasonably sure that they are secure to all kinds of attacks, as long as quantum computers are not involved. Although research into cryptography has expanded significantly in recent years, we still need to consider the

possibility that weaknesses are found in more recent inventions. Therefore it would be good to develop a cryptosystem that has the best of both worlds: one that is secure against quantum computers, but also has a fallback to a traditional scheme in case the primitive turns out to be broken.

1.2 Our contribution

In this thesis we propose a new way of error removal in RLWE-schemes, namely by increasing the modulus of the lattice while keeping the errors small, so we can eliminate the difference between Alice and Bob's versions of the shared secret simply by truncating the least-significant bits. We show that we can then remove the extra round-trips, and therefore end up with a Post-Quantum Non-Interactive Key Exchange based on Ring Learning With Errors, that closely resembles the Diffie-Hellman key exchange, while retaining the security of RLWE schemes.

We look into how hard the RLWE problem is when we use these large moduli and investigate the influence the lattice dimension has on that hardness. Finally we propose a parameter set that makes a key exchange possible with low failure rate and high security, and look at the effects these parameters will have on the practical side of the exchange, for instance on the message size.

1.3 Roadmap

This thesis consists of six chapters, of which this first chapter is an introduction to the topic and brief outline of the document itself. In Chapter 2 of this thesis the current state of cryptology is explained: why we need cryptology at all and we briefly review the mathematical background that is necessary to understand the rest of the thesis. The chapter then continues by introducing the current Ring Learning With Errors-based key exchanges and their history.

In Chapter 3 we propose the new key exchange and explain how it is related to other RLWE exchanges. We prove why this protocol is correct and secure and derive constraints on the parameters

In Chapter 4 we go on to choose parameters for the protocol and the underlying lattice problem that make sure the protocol provides a correct exchange while being secure enough. We explain what these parameter choices mean for the protocol and its usability in practice, for instance in terms of running time and message size.

In Chapter 5 we elaborate on the consequences of using the protocol in real-world scenarios. We revisit the concept of hybrid cryptography and explain how this protocol can be incorporated into a hybrid system.

We end with Chapter 6, which summarizes conclusions and recommendations from this thesis and discuss the future research that might build upon this work.

Chapter 2

Modern-day Cryptography

We exchange data all the time, via all kinds of channels: whether we place a file on a USB drive, transmit a document over a wifi network, or when we make a phone call.

Many of these channels, including the internet, are public to the world: if Alice and Bob exchange data, all other users are able to see it and read it. This is of course a problem: if Alice and Bob want to exchange private data and one of the other users — Eve, the malicious eavesdropper — is listening in, the private data is not private anymore. We call such an attack a *passive attack* (Figure 2.1). It also poses another problem: if Eve sends a message to Bob ending it simply with "Best regards, Alice", how does Bob know whether the message actually came from Alice, or whether it was manipulated by an attacker in what we call an *active attack* (Figure 2.2)?

There are many situations in which we want to be able to exchange confidential information over the internet, or be confident about the identity of our conversational partner. We also know there are people and organizations online who try to compromise our information exchanges.

The research area of *cryptography* has devised many tools to ensure three main properties:

1. *Confidentiality* of data, i.e. that it can only be read by the intended person(s);
2. *integrity* of data, i.e. that the data we read has not been modified in the process;
3. *authenticity* of data, i.e. that we can verify who was the sender of a message.

In this chapter we first explain the basics of cryptography: symmetric cryptography in Section 2.1 and asymmetric cryptography in Section 2.2. Section 2.3 then explains key exchange protocols, providing some background in the mathematics involved in them and the security models we need to describe them. We then build up to explaining

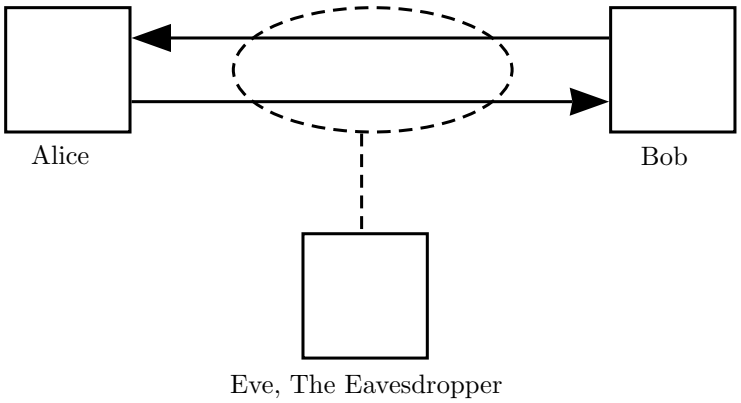

Figure 2.1: A passive attack: The two parties, Alice (A) and Bob (B), are exchanging messages. Eve (E), the eavesdropper, is listening in.

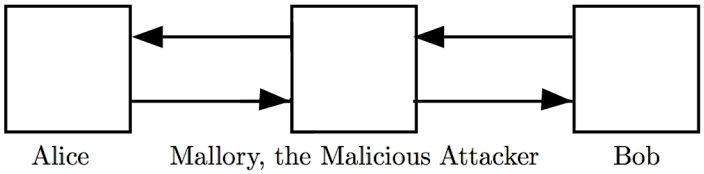

Figure 2.2: An active attack. The two parties, Alice (A) and Bob (B), are exchanging messages. Mallory (M), the malicious adversary, intercepts the message in the conversation, and replaces it with a different one. Can A see the message is not from B?

the post-quantum Ring-Learning With Errors key exchange, which is important to understand the contribution of this thesis.

We also spend some time on explaining the notions of security we need to reason about cryptographic systems (Section 2.4) and about the existing works related to hybrid cryptography (Section 2.5).

2.1 Symmetric cryptography

Symmetric cryptographic protocols are an important subset of the cryptographic toolbox. Using a symmetric algorithm, Alice can *encrypt* her message (or *plaintext*) using a *cryptographic key*, which is essentially a string of bits that functions like the password. Other users can use the same key to *decrypt* the resulting encrypted message (or *ciphertext*). As long as they both make sure adversaries do not get access to the key, the data is safe. After all, without the key it is not possible to decrypt the *ciphertext* into *plaintext*.

Several symmetric algorithms have been invented through the years, first running on dedicated hardware and later, particularly since the introduction of the internet, running on conventional computers.

A cryptosystem is said to encrypt a plaintext (or message) m into ciphertext c under key k, denoted as $c = \mathsf{Enc}_k(m)$. Similarly, decryption is denoted as $m = \mathsf{Dec}_k(c)$.

Definition 2.1 (Correctness of symmetric cryptosystem)**.** A symmetric cryptographic cryptosystem is said to be correct with probability p if decrypting a ciphertext with the same key k yields the original plaintext m with probability at least p, i.e.

$$\Pr(m = \mathsf{Dec}_k(\mathsf{Enc}_k(m))) \geq p.$$

Currently, the leading standard in symmetric cryptography is the Advanced Encryption Standard (AES) as it was standardized by NIST, which is based on the Rijndael cipher invented by the Belgian cryptographers Rijmen and Daemen [20].

2.2 Asymmetric cryptography

Up until the 1970s, symmetric cryptography was prevalent. Unfortunately symmetric cryptography has some inconvenient pre-requirements. The most notable is that the keys have to be exchanged between the parties in some secure way, which could be by courier, or trough a list of ready-to-use keys that would be exchanged beforehand.

Luckily, this pre-requirement has been solved. A few developments took place to make this possible: first the concept of public key distribution was invented by Merkle [30]. Then Diffie and Hellman invented the concept of public-key cryptography and designed a key exchange protocol that became widely used [21].

An asymmetric cryptosystem uses different keys for encryption and decryption. Every user A has a *public key* p_A and a *private key* (or *secret key*) s_A, also denoted as the key pair (p_A, s_A).

Typically, the public key is shared publicly while the private key is stored securely by the owner of the pair. If a user B wants to send a message to A, he encrypts using the publicly available key p_A so he knows only a user possessing s_A can decrypt the message: *confidentiality* is achieved.

Definition 2.2 (Correctness of asymmetric protocols)**.** An asymmetric cryptographic protocol is said to be correct with probability p if, using a key pair (p, s),

$$\Pr\left(m = \mathsf{Dec}_s\left(\mathsf{Enc}_p(m)\right)\right) \geq p.$$

In practice, symmetric cryptography is a lot more efficient than asymmetric cryptography in terms of computational effort. Therefore, real-world applications often use an asymmetric cryptographic system to establish a shared secret which is then used for symmetric encryption. This way, the advantages of both kinds of systems are combined.

It is preferable to let establishing the shared key be separate from the protocol that performs the actual data encryption: we separate the *Key Encapsulation Mechanism (KEM)* from the *Data Encapsulation Mechanism (DEM)* [51].

2.3 Key Exchange Protocols

In 1977, Diffie and Hellman published a paper [21] describing their *ax1x2* system, which is now widely known as the Diffie-Hellman (DH) key exchange, although Hellman later argued that it ought to have been named after Merkle as well [30].

The DH key exchange is based on the *Discrete Logarithm Problem (DLP)* in cyclic groups.

Definition 2.3 (Groups)**.** Let G be a set and let $+$ be an operator on G. G is a *group* if and only if the four group axioms are satisfied:

1. closure: for any $a, b \in G$, $(a + b) \in G$;
2. associativity: for any $a, b, c \in G$, $a + (b + c) = (a + b) + c$;
3. identity: there exists an element $i \in G$ such that for any $a \in G$, $i + a = a + i = a$;
4. invertibility: for any $a \in G$ there exists $b \in G$ such that $a + b = i$.

Definition 2.4 (Cyclic groups)**.** Let G be a group with operator $+$. For $g \in G$, let $2g = g + g, 3g = g + g + g$ and so on. G is a *cyclic group* if and only if there exists a *generator* $g \in G$ such that $G = \{ng | n \in \mathbb{Z}\}$. A group G generated by g can be denoted as $G = \langle g \rangle$.

Definition 2.5 (Discrete logarithm problem)**.** Let g be the generator of a group $G = \langle g \rangle$, and let $a \in \mathbb{N}$. Computing the *discrete logarithm* is recovering a given x and ax.

There is no general method known for computing this discrete logarithm efficiently using present-day computers, although computing it is easy in a lot of known cases. As long as we avoid those cases, it is assumed that the DLP is hard to solve for an attacker and therefore suitable for cryptographic purposes.

The DH protocol, described in Protocol 2.1, uses the hardness assumption of the DLP: Alice and Bob both pick a random secret value (a and b respectively) not larger than $|\langle g \rangle| - 1$. These are the *private keys* of Alice and Bob. Then Alice computes her *public key* g^a and sends it to Bob, and Bob responds with his public key g^b. Now Alice and Bob can

both compute a shared secret $(g^a)^b = g^{ab} = (g^b)^a$. When g is chosen correctly, attackers who intercept g^a and g^b cannot compute g^{ab} efficiently (even if g is known). Note that we are working with an example in $\mathbb{Z}_p^*$ here, therefore we use the multiplicative notation as it is more intuitive.

A big advantage of the DH-exchange is that it is *non-interactive*: Alice just needs g^b to write a message to Bob, and vice versa. This enables her to write a message to Bob even if he is not currently available, as long as she can find his public key online.

Diffie-Hellman key exchange

Select $g \in \mathbb{Z}_p^*$

Alice		**Bob**
Pick $a \in_R [0, \lvert\langle g\rangle\rvert - 1]$		Pick $b \in_R [0, \lvert\langle g\rangle\rvert - 1]$
$A = g^a$	$\xrightarrow{A}$	$B = g^b$
	$\xleftarrow{B}$	
$s = B^a$		$s = A^b$

Protocol 2.1: The Diffie-Hellman key exchange [21].

Elliptic Curve Diffie-Hellman

A popular instance of the traditional DH-exchange is *Elliptic Curve Diffie-Hellman*, or ECDH. While the first implementations of DH used the multiplicative group of integers modulo a prime, ECDH uses points on elliptic curves over finite fields instead, since the discrete logarithm problem (Definition 2.5) is even harder on elliptic curves: if we have a point P on a curve, we can easily compute $2P = P + P$, $3P = P + P + P$ and so on, but given points P and Q it is hard to find a satisfying $Q = aP$.

The efficiency of elliptic curve computations and the security provided make it a suitable alternative for the classic DH exchange based on integers [34, 43]. It is assumed ECDH is even harder than classic DH.

First, the parameters of the protocol are established for all participants, most notably a reference point P and the order n of P. Alice then creates a private key consisting of an integer d_A and computes the public key $Q_A = d_A P$, while Bob similarly computes $Q_B = d_B P$. The respective Q values are exchanged and Alice and Bob can compute a shared secret $d_A Q_B = d_A d_B P = d_B Q_A$, while attackers cannot compute that secret based only on Q_A and Q_B. Protocol 2.2 illustrates the exchange.

Elliptic Curve Diffie-Hellman

Select point P of order n

Alice		**Bob**
Pick $d_A \in [1, n-1]$		Pick $d_A \in [1, n-1]$
$Q_A = d_A P$	$\xrightarrow{Q_A}$	$Q_B = d_B P$
	$\xleftarrow{Q_B}$	
$s = d_A Q_B$		$d_B Q_A$

Protocol 2.2: The Elliptic curve Diffie-Hellman key exchange [34, 43].

2.3.1 Post-quantum key exchanges

As described in the motivation of this thesis (Section 1.1), the development of large-scale universal quantum computers poses a threat to the security of most traditional cryptographic systems. The quantum algorithm by Shor enables an attacker to efficiently solve the discrete logarithm problem, and thus it becomes possible to efficiently find a given g^a and g (in traditional DH), or to find d given dP and P (in ECDH). Therefore a new way to establish a shared key is needed.

In recent years, new hard mathematical problems have been proposed that make a key exchange possible, while maintaining security under the assumption that a universal quantum computer exists. Here, we focus on the class of lattice-based key exchanges, of which (Ring) Learning With Errors (RLWE) is an example.

In this thesis, we will focus on the RLWE-problem as a basis for a key exchange mechanism. In the following section, we describe the basis of this mechanism.

2.3.2 Learning With Errors-based key exchanges

To understand Learning With Errors (LWE) and why we can use it for key exchanges, we define the following mathematical notions.

Definition 2.6 (Lattice)**.** A *lattice* L is a subset of $\mathbb{R}^n$ such that

$$L = \left\{ \sum_{i=1}^{m} a_i v_i \middle| a_i \in \mathbb{Z} \right\},$$

where $\{v_1, v_2, \ldots v_m\}$ are the *basis vectors* of L, $v_i \in \mathbb{R}^n$, $i \in \{1, \ldots, m\}$.

Intuitively, a lattice can be visualized as a repeating pattern of points which is constructed by linearly combining the *basis vectors*. Lattices are interesting for cryptographic purposes, because some problems on lattices are very hard to solve. An example is the *Shortest Vector Problem (SVP)*, and its relaxed variant SVP_γ.

Definition 2.7 (SVP)**.** Let L be a lattice. Solving the *Shortest Vector Problem (SVP)* is finding a shortest non-zero vector in L. The length of this vector is denoted $\lambda(L)$.

Definition 2.8 (SVP_γ)**.** Let L be a lattice and $1 \leq \gamma \in \mathbb{R}$ a number. Solving the *Approximated Shortest Vector Problem (SVP_γ)* is finding a non-zero vector in L of length at most $\gamma \cdot \lambda(L)$.

Several other problems in group theory and discrete mathematics are reducible to the lattice problems described above. This is interesting since there is currently no efficient way to solve these problems, both in the classical way and with a quantum computer. An example is *Learning With Errors*:

Definition 2.9 (Learning With Errors)**.** Let $x, s \in \mathbb{Z}_q^n$ with $n, q \in \mathbb{N}$, $y, e \in \mathbb{Z}_q$. Now, generate n pairs (x_i, y_i), $i \in \{1, 2, \ldots, n\}$ such that $y_i = x_i s + e_i$, where e_i is sampled from a known distribution and xs is the normal dot product.

- the *LWE Search Problem* is finding the value of s, having access to this list;
- the *LWE Decision Problem* is to determine whether a given pair (x, y) is sampled randomly, or is an *LWE sample*.

An encryption method based on the Learning With Errors-problem was first proposed by Regev in 2005 [46]. Both the decision and search related LWE-problems are considered to be hard: in [44] it is shown that they can be reduced to the shortest vector problem in lattices, which is considered hard.

In this thesis, the hardness forms a basis for cryptography, similar to the way the discrete logarithm problem is used as a basis for the Diffie-Hellman key exchange.

Ring-Learning with Errors

In 2010, Lyubashevsky, Peikert and Regev introduced *Ring-Learning With Errors (RLWE)*, a variant of LWE in which the lattice has more structure. RLWE can be used very well for key exchanges, since this algebraic variant of the LWE-problem gives us a "truly efficient" key exchange that can be shown to provide very strong hardness guarantees, reducible to the approximate-shortest vector problem [41]. The efficiency claim relates for instance to the key size: the bitlength of RLWE keys is about the square root of the LWE key bitlength.

Definition 2.10 (Rings)**.** A *ring* is a set R with two binary operators, satisfying three Ring Axioms:

1. R is a commutative *group* (Definition 2.3), $(+ : R \times R \to R)$;
2. R has a second operator, multiplication $(\times)$, which is associative: for all $a, b, c \in R$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ and has an identity element $1 \in R$ such that $a \cdot 1 = a$ for all $a \in R$.
3. Multiplication is distributive: $a \cdot (b + c) = a \cdot b + a \cdot c$.

RLWE uses a ring of polynomials, which is constructed by taking the infinite polynomial ring $\mathbb{Z}_q[x]$ and reducing it modulo a polynomial. We write this as $R_q = \mathbb{Z}_q[x]/f(x)$, where $f(x)$ a polynomial, q an integer and n a power of 2. Most RLWE systems use $f(x) = x^n + 1$.

Now we show how RLWE can be used as a primitive for key exchanges. For readability, we denote polynomials using bold characters when describing protocols, e.g. $\mathbf{a}$ refers to a polynomial $a(x)$ such that

$$\mathbf{a} = a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1},$$

this common notation originated with LWE-schemes, as it used vectors instead of polynomials.

Definition 2.11 (Infinity Norm)**.** The *infinity norm* of $\mathbf{a}$ is $\ell_\infty\left(a(x)\right) = \|\mathbf{a}\|_\infty = \max|a_i|$.

When we describe key exchange mechanisms, we always let $\mathbf{a}$ be the initialization value, we let $\mathbf{p}$ be the public key and we let $(\mathbf{s}, \mathbf{e})$ be the secret key, consisting of two polynomials of which $\mathbf{e}$ is the error. In cases where both participants in the protocol need their own public or private key, we use e.g. $\mathbf{p_A}$ and $\mathbf{p_B}$ for Alice and Bob, respectively. We denote sampling a polynomial $\mathbf{a}$ from a distribution χ as $\mathbf{a} \leftarrow_R \chi$, and sampling it uniformly random from the ring R_q as $\mathbf{a} \leftarrow_R R_q$. Here, χ is n-dimensional.

In a RLWE-exchange, we generate polynomials that are small in the infinity norm (Definition 2.11 below). We do this by randomly generating small coefficients, which are typically sampled from a narrow Gaussian distribution using only integer values, a so-called discrete Gaussian distribution. We denote the resulting distribution on R_q by χ. Taking only the coefficients, χ is an n-dimensional Discrete Gaussian distribution [25].

We define the following elements, all of which are elements of the ring, i.e. $\mathbf{a}, \mathbf{s}, \mathbf{e} \in R_q$:

- $\mathbf{a}$ is public and known to all network users.
- $\mathbf{s}$ and $\mathbf{e}$ together form the secret key.
- The public key is the polynomial $\mathbf{p} = \mathbf{as} + \mathbf{e}$.

Definition 2.12 (The RLWE Problem)**.** Let $\mathbf{s}$ be a secret. Generate i sample pairs $(\mathbf{a_i}, \mathbf{p_i})$, such that $\mathbf{a_i}$ is chosen randomly and $\mathbf{p_i}$ is generated by sampling an error polynomial $\mathbf{e_i}$, and then computing $\mathbf{p_i} = \mathbf{a_i s} + \mathbf{e_i}$.

- the *RLWE Search Problem* is to retrieve $\mathbf{s}$ based on this list;
- the *RLWE Decision Problem* is to determine whether a given pair $(\mathbf{a}, \mathbf{p})$ is sampled randomly from $R_q \times R_q$ or constructed as above, given the list.

Schemes based on RLWE

After the public polynomial $\mathbf{a}$ is established, Alice generates random samples $\mathbf{s_A}, \mathbf{e_A} \leftarrow_R \chi$ and uses them to generate a key $\mathbf{b} = \mathbf{as} + \mathbf{e}$ that she sends to Bob. Bob then generates his own samples $\mathbf{s_B}$, $\mathbf{e_B}$ and $\mathbf{e'_B}$ and uses these values to compute $\mathbf{u}$ and $\mathbf{v}$ as described in Protocol 2.3. He transmits $\mathbf{u}$ and some $\mathbf{r}$ based on $\mathbf{v}$ back to Alice. They can now each combine the random values they generated in the beginning with the values they received from the other party, and they both establish a secret value. Observe in the protocol that both parties have a slightly different version of the secret:

$$\mathbf{ass'} + \mathbf{es'} \approx \mathbf{ass'} + \mathbf{e's}.$$

These secrets are *almost* identical, and to make them exactly identical, a reconciliation mechanism is used. The scheme described here can be seen in Protocol 2.3.

Remark that in Protocols 2.3 and 2.4, we see that some extra functions are present: the former uses Reconciliation and a Helper function, the latter uses an Encode-and-Extract method. Explaining in detail the way these functions work is a bit beyond the scope of this thesis. The important concept is that, in general, the error value(s) introduced in the key exchange make it necessary to apply some form of error correction to establish a shared secret between the participants, be it through reconciliation or through an encryption and extraction step. These two methods both require exchanging one or more values between the two parties.

Several steps have been taken to turn the RLWE-exchange into a serious contestant for a post-quantum key exchange mechanism. Ding, Xie and Lin [23] built a protocol based on RLWE in 2012. In 2014, Peikert took an important step [45], working on making the RLWE-exchange usable for existing internet protocols and making it a passively secure KEM.

In early 2015 Bos, Costello, Naehrig and Stebila [17] created ciphersuites for TLS based on Peikert's version of RLWE, combined with traditional RSA or ECC authentication to provide transitional security as described in section 2.5. Additionally they provided proofs of the security of their suites and showed the results of some tests where the suites were implemented into OpenSSL.

Reconciliation-based KEM

$\mathbf{a} \leftarrow_R R_q$

Alice		**Bob**
$\mathbf{s}, \mathbf{e} \leftarrow_R \chi$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \leftarrow_R \chi$
$\mathbf{b} = \mathbf{as} + \mathbf{e}$	$\xrightarrow{\mathbf{b}}$	$\mathbf{u} = \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} = \mathbf{bs}' + \mathbf{e}''$
	$\xleftarrow{\mathbf{u}, \mathbf{r}}$	$\mathbf{r} = \text{HelpRec}(\mathbf{v})$
$\mathbf{v}' = \mathbf{us}$		$\mu = \text{Rec}(\mathbf{v}, \mathbf{r})$
$\mu = \text{Rec}(\mathbf{v}', \mathbf{r})$		

Protocol 2.3: NewHope: a KEM using reconciliation [10].

Later that year, the implementations of [17] were further improved in several ways, resulting in a key exchange dubbed NewHope by the authors [10]: the parameter set was revisited, the error-values are generated using a different distribution, the error-reconciliation mechanism is redesigned and some other changes are proposed in order to protect the scheme better against backdoors. Compared to [17], the security of the protocol is doubled, the communication overhead is halved and the necessary computations are sped up by more than a factor 8.

In Protocol 2.3, the NewHope key exchange is shown, which is comparable to other exchanges in this class. In 2016, the same group of authors proposed NewHope-Simple [9], a key exchange that uses the hardness of the RLWE-problem as well, but has a different set-up: most notably the reconciliation mechanism that had been present since [23] is replaced by a different way of negotiating the shared value between the two parties: the two parties now exchange an extra encrypted value (as shown in Protocol 2.4) that makes the messages slightly larger. The complexity of the reconciliation step can then be avoided, making this key exchange significantly simpler: it is easier to understand the concepts and more convenient to implement.

2.3.3 Error distributions

All RLWE-based protocols require sampling of errors from some distribution χ. This is traditionally based on the Discrete Gaussian distribution [17] with some width σ which differs per protocol.

The main exception to the rule is NewHope [10], which uses a centered binomial distribution simply because it is computationally easier to sample from and close enough to

Encryption-based KEM

$\mathbf{a} \leftarrow_R R_q$

Alice		**Bob**
$\mathbf{s}, \mathbf{e} \leftarrow_R \chi$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \leftarrow_R \chi$
$\mathbf{b} = \mathbf{as} + \mathbf{e}$	$\xrightarrow{\mathbf{b}}$	$\mathbf{u} = \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} = \mathbf{bs}' + \mathbf{e}''$
		$\nu \in_R \{0,1\}^n$
		$\mathbf{k} = \text{Encode}(\nu)$
	$\xleftarrow{\mathbf{u}, \mathbf{c}}$	$\mathbf{c} = \mathbf{v} + \mathbf{k}$
$\mathbf{v}' = \mathbf{us}$		$\mu = \text{Extract}(\mathbf{k})$
$\mathbf{k}' = \mathbf{c} - \mathbf{v}'$		
$\mu = \text{Extract}(\mathbf{k}')$		

Protocol 2.4: NewHope-Simple: a KEM that avoids the reconciliation step [9].

the Discrete Gaussian to keep the same security properties.

2.4 Notions of security

In Definition 2.1 and 2.2 we defined the notion of correctness for both symmetric and asymmetric cryptographic schemes, respectively. The notion of security is a bit more complex.

Recall the concept of active and passive attackers: in a passive attack the adversary is able to "listen in": she can hear all communications over the public channel and is able to perform computations and analysis on them. An active attacker has more capabilities: she is able to manipulate the messages, for instance by replacing a message from A to B after intercepting the original, or by sending the message later or repeatedly.

In 1998, Bellare, Desai, Pointcheal and Rogaway established the notions of indistinguishability and non-malleability [13]. Specifically they introduced the *indistinguishability under chosen plaintext attack (IND-CPA)* and the *indistinguishability under chosen ciphertext attack (IND-CCA1 and IND-CCA2)* notions. The attacks allow us to describe the security of a cryptographic scheme in more detail.

To describe the attacks we use an attacker and a challenger. The intuition is the following: the attacker claims to know an attack. The challenger challenges that knowledge.

IND-CPA means that the attacker has the benefit of being able to access an *encryption oracle* which is able to encrypt messages chosen by the attacker; this effectively means the attacker has access to the encryption algorithm and to the public key. The attacker then generates two plaintexts of equal length and gives them to the challenger, who randomly selects one of them, encrypts it and returns the ciphertext to the attacker. The attacker then guesses which of the plaintexts was encrypted.

IND-CCA1 is similar, but the attacker is additionally able to access a decryption oracle, before he gets the challenger's ciphertext, i.e. the queries to the oracle cannot depend on the challenge.

IND-CCA2 even allows the attacker to access the decryption oracle after getting the challenge, as long as he does not ask for a decryption of exactly the challenge ciphertext.

For both scenarios we can analyze the *advantage* ϵ the attacker has over a random coin flip, i.e. $\Pr(\text{guess} = m) = \frac{1}{2} + \epsilon$. In the (hypothetical) scenario where perfect security is achieved, $\epsilon = 0$ and the attacker cannot do any better than guessing randomly.

2.5 Hybrid Encryption Schemes

In 2016, Schanck, Whyte and Zhang identified several reasons for the slow migration from currently used schemes to post-quantum cryptographic protocols: the development of new systems leads to low confidence from the industry in both the security of the primitives, as well as the security of the implementations. This is mainly caused by the fact that there has been less research into their security, and the fact that the mathematics behind those schemes is often harder to understand. Additionally, the need to comply with existing standards poses an issue, as does the challenge of integrating the new primitives into current protocols and infrastructures [49].

If one were to combine a post-quantum primitive with a currently used one, one could overcome most of these challenges: those who do not confide in the security of new cryptography will still be able to rely on the scheme, because of the established primitive. Additionally this will make it easier to adhere to the applicable standards that require the established primitives to be used.

Definition 2.13 (Hybrid encryption)**.** A *hybrid encryption scheme* is a scheme that combines a post-quantum cryptographic primitive and a pre-quantum cryptographic primitive into one.[1]

[1] Remark that this is the definition we use in this thesis. The term *hybrid cryptography* has historically also been used to refer to other concepts.

According to [49], a lot of work remains necessary to upgrade systems like TLS and Tor to fully hybrid systems, while the upgrade needs to happen as soon as possible. Instead a *transitionally secure* protocol can be used where pre-quantum authentication and post-quantum confidentiality is guaranteed.

Definition 2.14 (Transitional security)**.** A protocol is *transitionally secure* if it is secure against active attackers using current computers and against passive adversaries using quantum computers.

This does not secure the user data against active attacks with a quantum computer, but does provide security against passive attacks. In practice this means we can secure the system against adversaries that currently store encrypted internet traffic in order to decrypt it later, when quantum computers do exist. Because these systems are also based upon primitives that are well-known to be secure against attacks with classical computers, they inspire confidence as long as active quantum attacks are not involved.

In [55], Unger and Goldberg explain: "An interesting class of key exchanges are those providing quantum transitional security—classical authentication and quantum-resistant confidentiality [88]. Assuming that the hardness assumptions hold, these schemes preserve forward secrecy against future quantum adversaries. Additionally, these hybrid protocols maintain classical security in the event that the quantum resistance assumptions fail."

Remark that [88] in [55] is [49] in this thesis.

2.5.1 Theoretical work

In 2017 Bindel, Herath, McKague and Stebila [16] investigated the security implications of building post-quantum secure public-key signature schemes, the security implications of the different possible ways signature schemes can be combined, and evaluate the compatibility of current standards with hybrid schemes.

Although we do not focus on signature schemes in this thesis and the combination of key exchange mechanisms works slightly different, we do have a look at the combination methods in their paper.

Combining signature schemes

Different ways of combining schemes are identified and evaluated:

- *Concatenation ($C_{||}$)*: the trivial way of combining schemes is to place two signatures side-by-side and requiring that both verify. The security of this combined scheme

is as strong as the security of the strongest scheme.

- *Weak nesting ($C_{\text{weak}-\text{nest}}$)*: the first signature scheme is used to create a signature for the message: that signature is then signed using the second scheme. The security of this combined scheme relies fully on that of the first scheme, therefore using this combination method is not recommended. The added benefit of the combination is that the integrity of the first scheme can be verified, which is useful if attackers later gain the ability to forge its signatures.
- *Strong nesting ($C_{\text{str}-\text{nest}}$)*: similar to weak nesting, but the second scheme signs both the message and the first signature. This offers the same security as Concatenation, with the added benefit that the two signatures cannot be separated.

Standardizing hybrid signatures

The authors of [16] evaluate several existing standards that make use of signature schemes and describe whether it would be possible to implement one of the hybrid methods mentioned above while adhering to the standard.

- *X.509v3 certificates*: the X.509v3-standard does not support adding multiple sets of certificate-data to one certificate. Therefore there are two ways to use a hybrid certificate while adhering to the standard: either using two separate certificates and letting the application handle the hybrid situation, or using the standard's extension mechanism to add a second certificate. Experimental evaluations show that this does indeed work in practice.
- *TLS*: the TLS standard allows using a form of hybrid encryption, while the mainstream implementations might be a bit more limited due to the allowed signature sizes. In practice, most browsers seem to support reasonable expansions of the message size.
- *CMS & S/MIME*: both concatenated and nested approaches are possible in this standard, however the concatenated approach leads to problems with backwards-compatibility in some clients. The nested approach works well in most cases, the only exception being Mozilla Thunderbird's inability to handle very large signatures.

Chapter 3

A non-interactive key exchange

The lattice-based key exchanges described in Section 2.3.2 differ from the classic Diffie-Hellman exchange (Section 2.3) in one important way: both the reconciliation-based exchange (Protocol 2.3) and the simpler encode-and-extraction-based exchange (Protocol 2.4) need some kind of interaction between the parties, while in a Diffie-Hellman exchange no interaction is necessary. Parties just need to obtain each other's public key. An important advantage of this non-interactiveness is that the key exchange can be done via an online repository, even if one of the parties is offline, or can be done asynchronously if necessary. As described above, this is a feature not yet provided by the current family of RLWE-based key exchanges.

3.1 The Protocol

We now show how the reconciliation-based ring learning with errors key exchange described in the previous chapter can be transformed so Alice and Bob only exchange one public polynomial each, called $\mathbf{b_A}$ and $\mathbf{b_B}$, respectively. This is based on a 'folklore idea' explained by amongst others Vadim Lyubashevsky [40].

Let $R_q = \mathbb{Z}_q[x]/f(x)$ be a lattice where $f(x) = x^n + 1$. We use q and n, respectively the modulus and the dimension, to refer to these values. Protocol 3.1 shows the new exchange. To set up the system, we uniformly random sample the polynomial $\mathbf{a} \leftarrow_R R_q$, Alice and Bob sample secret key polynomials $\mathbf{s_A}$ and $\mathbf{s_B}$, respectively, from a discretisized Gaussian distribution χ over R_q with width parameter σ. They also sample error polynomials $\mathbf{e_A}$ and $\mathbf{e_B}$, respectively, from the same distribution and compute their public keys $\mathbf{b_A} = \mathbf{as_A} + \mathbf{e_A}$ and $\mathbf{as_B} + \mathbf{e_B}$, which are then published.

Using the information received from their counterparts either directly or indirectly, Alice

computes $\mathbf{s_A b_B}$ and Bob computes $\mathbf{s_B b_A}$.

Non-Interactive Key Exchange Protocol

$\mathbf{a} \leftarrow_R R_q$

Alice		**Bob**
$\mathbf{s_A}, \mathbf{e_A} \leftarrow_R \chi$		$\mathbf{s_B}, \mathbf{e_B} \leftarrow_R \chi$
$\mathbf{b_A} \leftarrow \mathbf{a s_A} + \mathbf{e_A}$		$\mathbf{b_B} \leftarrow \mathbf{a s_B} + \mathbf{e_B}$
	$\mathbf{b_A} \longrightarrow$	
	$\mathbf{b_B} \longleftarrow$	
$\mu \leftarrow \text{HighOrderBits}(\mathbf{s_A b_B})$		$\mu \leftarrow \text{HighOrderBits}(\mathbf{s_B b_A})$

Protocol 3.1: A non-interactive Key Exchange Protocol

Remember that the goal of the key exchange is to provide both parties with a shared secret that can then be used to generate a shared key. Observe that

$$\mathbf{s_A b_B} = \mathbf{a s_A s_B} + \mathbf{s_A e_B},$$
$$\mathbf{s_B b_A} = \mathbf{a s_B s_A} + \mathbf{s_B e_A}.$$

Although $\mathbf{s_A b_B} \neq \mathbf{s_B b_A}$[1], a lot of bits are the same in both products. In particular, there is a high probability of them overlapping if we minimize the influence of $\mathbf{e_A}$ and $\mathbf{e_B}$ on the computed product. If we can choose the parameters in such a way that these error polynomials only influence the small-order bits, we can obtain a shared secret μ by looking only at the highest bits of each coefficient. We describe this by stating that Alice computes $\mu_A = \text{HighOrderBits}(\mathbf{s_A b_B})$, and similarly Bob considers μ_B.

To illustrate how this works, we provide an example of the computations in practice, focusing on one coefficient of each polynomial. Numbers are written in binary, big-endian notation. In Example 3.1 we show how executing the protocol provides the two participants with values that lead to computation of a shared secret. In Example 3.2 we show a scenario where the protocol does not lead to a shared secret.

The scenario that occurs in Example 3.2 is unlikely: if we pick our parameters correctly, the probability of such a mismatch should be small. Later we will analyse these failure rates and explain the influence we have on it.

[1] Excluding the negligible chance that they are identical.

First, the shared polynomial is established.	
We focus on one coefficient, given in binary.	
$\mathbf{a} =$	11000101001011000000100110
Alice and Bob randomly generate their parameters.	
$\mathbf{s_A} =$	11010011
$\mathbf{e_A} =$	01111101
$\mathbf{s_B} =$	00100011
$\mathbf{e_B} =$	11001100
They then generate their public keys...	
$\mathbf{b_A} = \mathbf{as_A} + \mathbf{e_A} =$	10100010100000110101001111100111 1
$\mathbf{b_B} = \mathbf{as_B} + \mathbf{e_B} =$	00011010111101010000011011111111 0
...and exchange them.	
Both parties execute the final computation. Their high-order bits match.	
$\mathbf{s_A b_B} =$	10110001101111111010011000100001011010
$\mathbf{s_B b_A} =$	10110001101111111010001111000101001101

Example 3.1: Protocol 3.1 is executed using small values, we look at the value of one coefficient. Observe **es** can not be > 12 bits long. Therefore the bits higher than 12th-order are identical with high probability.

The bits higher than 12-order can differ with small probability.	
Let the numbers be of the form	
$\mathbf{as_A s_B} =$	11111111111111111111111111111111xxxxxxxxxxx
$\mathbf{e_A s_B} =$	0xxxxxxxxxxx
$\mathbf{e_B s_A} =$	1xxxxxxxxxxx
with every $\mathtt{x} \in \{0, 1\}$.	
Now Alice and Bob have values that do not have matching high-order bits.	
$\mathbf{s_A b_B} =$	11111111111111111111111111111111xxxxxxxxxxx
$\mathbf{s_B b_A} =$	00000000000000000000000000000000xxxxxxxxxxx

Example 3.2: Similar to Example 3.1, but now a mismatch occurs and Alice and Bob have different versions of the shared secret.

3.2 Correctness of the Protocol 3.1

The correctness of the protocol depends on achieving matching high-order bits in all coefficients. As q is not a power of 2, we want the second-highest bit to match for each coefficient. We can estimate whether these bits match as follows:

Theorem 3.1 (Correctness). *For $q \leq 2^{k+1}\beta$, $k \geq 0$ and $q = q_l 2^l + q_{l-1}2^{l-1} + \cdots + q_1 2 + q_0$ where $\beta \geq \|\mathbf{s_A e_B}\|_\infty$, $\beta \geq \|\mathbf{s_B e_A}\|_\infty$ and β a power of 2, the probability that Protocol 3.1 is correct, that is, that bit $l-1$ of $\mathbf{s_A b_B}$ equals bit $l-1$ of $\mathbf{s_B b_A}$, is at least $1 - 2^{-k}$ for one coefficient.*

Observe we use the one-but-highest order bit because in the situation where the highest-order bit is 1, reduction modulo q will be applied before all possibilities are used. The consequence is that k will be one lower and thus the failure rate will be increased slightly.

Proof. The polynomials $\mathbf{s_A}, \mathbf{e_A}, \mathbf{s_B}, \mathbf{e_B}$ have small coefficients upper bounded by $\beta = \|\mathbf{se}\|_\infty$.

Since $q \leq 2^1 \cdot 2^k\beta$, all coefficients have at most $\log(q) \leq 1 + k + \log(\beta)$ bits. Only the last $\log(\beta)$ bits of each coefficient can directly be influenced by $\mathbf{e}$. The one-but-highest-order bit $l-1$ can only be changed if bits $l-2, l-3 \ldots 1$ are all 1 so they carry the result of the multiplication. The probability that all these $\log(q) - \beta - 1 = k$ bits are equal to 1 is 2^{-k} and thus the probability that the two coefficients share the highest-order bit is at least $1 - 2^{-k}$. □

Remark that as this is a key exchange, we are exchanging messages with the goal of establishing a symmetric key and since these keys conventionally consist of 256 bits, we only need to encode one bit per coefficient as long as $n \geq 256$, which will be the case in all our parameter sets. Observe that increasing the number of coefficients will increase the probability that a mismatch occurs in one of them: the failure rate of the entire protocol will thus be

$$\Pr(\text{failure}) \leq n \cdot 2^{-k}$$

if the $(l-1)$-th bits of all n coefficients must match.

We observe that the correctness of the protocol depends on the value of k and thus largely on the value of q, and that increasing q leads to a higher probability of correctness. In existing exchanges, this parameter is around 13 or 14 bits long: e.g. in NewHope $n = 1024$, $q = 12289$ and the coefficients are sampled from a centered binomial distribution ψ_{16}, so they have bound $\beta \geq 2^{10}2^4 2^4 = 2^{18}$ [10].

We can attempt to compute the probability that the protocol fails to establish a shared secret if we were to use these parameters: recall $q \leq 2^1 \cdot 2^k\beta$, thus $12289 < 2^{14} =$

$2^1 \cdot 2^{-5} \cdot 2^{18}$ and $k = -5$ which violates the assumption of Theorem 3.1, clearly leading to a system that is not usable.

We thus need to increase q to increase this value k. However, as q becomes larger, the underlying RLWE problem becomes less hard to compute and the security of the key exchange decreases. [56] Therefore we need to increase the dimension of the ring as well. In the following sections we will explain what sizes we need for these variables.

3.3 Security of the protocol

In Section 2.4 the notion of IND-CPA is introduced. We prove that our protocol satisfies this notion, and as such is secure against a passive adversary. As basis, we use the proof Peikert gave in 2014 [45]:

Lemma 3.1 (Lemma 4.1 of [45])**.** *KEM1 is IND-CPA secure, assuming the hardness of R-DLWE$_{q,\chi}$ given two samples.*

Here, *KEM1* is a RLWE-based key exchange based on reconciliation and *R-DLWE$_{q,\chi}$* is the Decision-variant of the RLWE problem (Definition 2.12) with $\mathbf{s} \leftarrow_R \chi$, χ a Discrete Gaussian distribution, and $\mathbf{a} \leftarrow_R R_q$. KEM1 is comparable to Protocol 2.3, but the transmitted values are constructed in a slightly different way. In particular, $\mathbf{u} = \mathbf{e_0 a} + \mathbf{e_1}$ and $\mathbf{v} = \mathbf{g e_0 b} + \mathbf{e_2}$, with $\mathbf{s_0}, \mathbf{s_1}, \mathbf{e_0}, \mathbf{e_1}, \mathbf{e_2} \leftarrow_R \chi$.

Theorem 3.2 (IND-CPA Security)**.** *The protocol introduced in this chapter, described in Protocol 3.1, is IND-CPA secure assuming the hardness of the RLWE decision problem.*

Proof. We take the proof for the IND-CPA security of KEM1 from [45] as the starting point of the proof, and will adapt it to suit our own needs. By Lemma 3.1, KEM1 is secure, assuming the Decision-variant of the RLWE-problem is hard.

Recall that in our protocol, we transmit values $\mathbf{b_A}, \mathbf{b_B}$ over a public channel, where $\mathbf{b_A} = \mathbf{a s_A} + \mathbf{e_A}$ and similarly for $\mathbf{b_B}$. $\mathbf{a} \leftarrow_R R_q$ and $\mathbf{s_A}, \mathbf{s_B}, \mathbf{e_A}, \mathbf{e_B} \leftarrow_R \chi$.

Compare this with KEM1, where $\mathbf{s_0}, \mathbf{s_1}, \mathbf{e_0}, \mathbf{e_1}, \mathbf{e_2} \leftarrow_R \chi$ and then $\mathbf{u} = \mathbf{e_0 a} + \mathbf{e_1}$ and $\mathbf{v} = \mathbf{g e_0 b} + \mathbf{e_2}$ are encapsulated and transmitted.

Observe that the lemma describes two samples: [45] confirms that these are the pairs $(\mathbf{a}, \mathbf{u})$ and $(\mathbf{b}, \mathbf{g})$ in KEM1, which depend on the randomness of $\mathbf{e_1}$ and $\mathbf{e_2}$. Similarly the attacker in our protocol obtains the values of $\mathbf{b_A}$ and $\mathbf{b_B}$, which similarly depend on the randomness of $\mathbf{e_A}$ and $\mathbf{e_B}$.

Therefore, with identical χ and q, $(\mathbf{b_A}, \mathbf{b_B})$ is indistinguishable from $(\mathbf{u}, \mathbf{v})$ and since $(\mathbf{u}, \mathbf{v})$ is indistinguishable from a pair that was sampled uniformly random we conclude

$(\mathbf{b_A}, \mathbf{b_B})$ is indistinguishable from a random sample and thus the protocol is IND-CPA secure. □

Remark that in our protocol, the values transmitted are slightly different than in KEM1: the attacker will even gain less information in our situation, which makes attacks harder.

Also observe that the security of our protocol depends on the fact that the attacker is not able to distinguish the RLWE-samples from random ones. This leaves the hardness of the RLWE-decision problem as the important topic to study in the next chapter.

Chapter 4

Hardness of RLWE

In this chapter we analyze the hardness of the RLWE problem, and thus the security of Protocol 3.1, for different parameter choices.

First, we explain the existing requirements stated in the literature and how large the parameters need to be to achieve small failure probability. We then analyze the hardness of the RLWE problem for different parameter sizes, to ensure that the parameter choices we made are indeed sufficiently large.

4.1 Estimating bits of security

To estimate the hardness of the RLWE problem, we are using the a program by Albrecht [8] written in Sage [54], which computes the security level of LWE for different parameter choices. In particular, it estimates the effectiveness of different types of attacks. For more conventional parameter sizes, there is existing research using this suite of tools, for instance in Albrecht's original paper [8] and in the 2016 thesis by Van Heesch [56]. It makes sense to compare the security of these conventional RLWE-exchanges with the security estimates of our large parameter sets, but it should be noted that the Sage program used has evolved through the years, making use of progress in cryptographic research and improving the implementations. This means the tool now provides better estimates for most types of attacks, and sometimes features new ones altogether. To show this progress, we also compare some of Van Heesch' results with the current estimations using Albrecht's program.

This comparison is shown in Table 4.1. We see for instance that decoding is harder than initially presumed: running the algorithm now yields more bits of security. Additionally, more attacks have been introduced since the original results were obtained in 2015. More

			November 2015		January 2018		
n	$\log(q)$	σ	Distinguishing	Decoding	Decoding	Dual	uSVP
512	19	3.192	147.8	132.4	165.6	169.3	154.8
512	21	3.192	129.1	116.0	135.1	139.3	126.3
512	8	3.192	492.4	479.4	719.8	963.6	790.1
512	12	3.192	281.8	218.5	387.7	416.6	385.4
1024	33	1.596	139.6	129.9	163.9	164.9	154.2
1024	35	1.596	130.1	121.9	146.4	146.8	138.7
1024	18	1.596	294.8	272.0	463.3	467.4	451.7
1024	20	1.596	258.6	238.2	393.1	393.0	379.7
1024	35	3.192	139.1	129.8	162.7	162.7	153.5
1024	37	3.192	129.6	122.0	146.1	145.4	137.3
1024	26	3.192	291.7	270.2	288.8	291.1	276.1
1024	22	3.192	256.1	236.8	390.5	394.9	376.9
1024	37	6.384	138.2	129.3	162.0	161.9	153.5
1024	39	6.384	129.0	120.8	145.0	144.8	136.6
1024	22	6.384	287.2	265.4	457.4	453.8	444.0
1024	23	6.384	269.2	248.7	419.8	422.7	406.4
2048	67	3.192	133.8	129.0	160.1	159.8	153.0
2048	69	3.192	129.2	124.6	151.2	149.3	144.1
2048	37	3.192	285.3	270.0	483.5	484.3	468.4
2048	39	3.192	261.3	247.5	440.5	442.0	426.5

Table 4.1: The parameters proposed by [56] in 2016, analyzed with a modern version of Albrecht's tool.

details about these evolutions will be in in Section 4.2.3.

All estimates in this chapter are computed using commit

`c5763b217ce308433208aaf471728eb9884219a7` of 30 Jan 2018[1]

unless where it is explicitly stated that the version used by Van Heesch was used. In those cases, the estimates were generated using the version at commit

`586cc519cf21acd36b4ce2ea03b44842f2cbca35` of 29 Nov 2015[2]

of the code.

[1] `https://bitbucket.org/malb/lwe-estimator/commits/c5763b2`
[2] `https://bitbucket.org/malb/lwe-estimator/commits/586cc51`

4.2 Attacks on RLWE-schemes

As described in previous chapters, attackers who try to break the RLWE-based schemes are essentially breaking the RLWE-problem, which means that they are trying to find the secret polynomials based on a number of samples. It should be remarked that this comes down to the to the RLWE-search problem, while our security proof reduces the security to the RLWE-decision problem: these are equivalent. There are currently several attacks possible on these schemes.

4.2.1 Risks posed by these attacks

Does the fact that attacks exist put the security of our scheme at risk? The fact that attack strategies have been devised is not a risk per se, as long as they are not easily executed. Let's take a look at what the numbers above mean. We typically measure the security of protocols in *bits of security*, also referred to as the *security level*:

Definition 4.1 (Security level)**.** A protocol provides n bits of security if an attacker would need to perform 2^n elementary operations on average to break it.

For symmetric cryptography, the number of bits of security is typically equivalent to the key length. An attacker who wants to break AES with key size 256 bits, would have to brute force all 2^{256} options and thus has to perform $2^{256} \approx 10^{77}$ operations.

For our protocol, this security level is approximated by making an estimation of how hard the lattice attacks are. Let's take one of the rows from Table 4.1 as an example:

input			⋯	output		
n	$\log(q)$	σ	⋯	Decoding	Dual	uSVP
2048	69	3.192	⋯	151.2	149.3	144.1

To the left, we see the three parameters n, q and σ that were chosen as basis for the RLWE problem. This is used as input for the estimation algorithm.

To the right, the three results are visible:

- *decoding* would take an attacker approximately $2^{151.2}$ operations;
- executing the *Dual-lattice Attack* would take an attacker approximately $2^{149.3}$ operations;
- executing the *Primal uSVP Attack* would take an attacker approximately $2^{144.1}$ operations.

For an attacker armed with these attacks, the Primal uSVP Attack would be the easiest to perform. We thus conclude that an RLWE-based protocol with these parameters has a

security level of 144.1 bits.

4.2.2 Algorithms estimated by Albrecht

The Albrecht estimator described above, implements the following algorithms:

1. Meet-in-the-middle exhaustive search;
2. Coded-BKW [29];
3. Dual-lattice attack and small/sparse secret variant [3];
4. Lattice-reduction and Lattice-enumeration attack [39];
5. Primal attack via uSVP [7, 12];
6. Arora-Ge algorithm [11] using Gröbner bases [6].

The Meet-in-the-middle, Arora-Ge and Coded-BKW are considered less effective then the other three, and are thus not used by default.

4.2.3 Changes over time

Note that the estimations change over the years, since both attack methods and the way they are estimated evolve. If we compare the algorithms in the list above to the ones used at the time of the analysis in 2016 by [56], we for instance observe the following evolutions:

- Meet-in-the-middle, Coded-BKW, Lattice-reduction/enumeration and Arora-Ge were used already;
- the Dual-lattice attack was not considered yet;
- Kannan-embedding was used, which has now been replaced by the Primal attack.

We also observe that the estimations themselves have changed, even if the attacks stayed the same. This has to do with improvements of both the implementation of the estimator, as well as the understanding of the attacks and the way they work. Notably we see in Table 4.1 that Decoding is now estimated to be harder than it was in 2015: this does not mean that the attack itself is harder now than it was then, but that the estimates were coarser at that point.

4.3 Choosing Parameters for Non-Interactive Key Exchange

In this section we analyze the parameters that can be chosen for RLWE-based exchanges, and describe what different parameter sizes mean for the bit security using the tool mentioned in the previous section. We recommend parameter sets for RLWE that are suitable for Protocol 3.1.

4.3.1 Constraints on the parameters

The parameters we need to choose are the modulus q and dimension n of the lattice, as well as the width σ of the error distribution the protocol uses. We need to select parameters that lead to a correct key exchange, while taking into account the hardness of the RLWE problem, which can be analyzed using the algorithm above.

When selecting parameters for the protocol, we have to take into account several aspects. As we know from Theorem 3.1, the failure rate of the protocol depends on the size of q.

In the next paragraph we show that increasing this q immediately makes the lattice problem easier to solve, if we keep n at a conventional size. Increasing n, however, means more coefficients and thus a higher probability that a mismatch occurs with one of them, and means that the coefficients become larger as they are the result of adding up n products.

The goal is thus to find a parameter set that has an acceptable number of bits of security, while providing an acceptable success probability.

Very recent work by Albrecht, Curtis, Davidson, Player, Postlethwaite, Virdia and Wunderer [5] provides an investigation of the number of bits of security provided by other RLWE-based schemes, using the April 2018 version[3] of the tool mentioned above. These security levels range from 101 bits (for NewHope [10] with $n = 512$, $\log q = 14$ and $\sigma = 2.00$)[4] up until 444 bits (for LIMA-2p [53] with $n = 2048$, $\log q = 18$ and $\sigma = 3.16$).

Based on these values, we assume 256 bits to be an acceptable security level for normal purposes. We know from past experiences with other protocols that it might be necessary to scale up to 512 bits of security as future developments occur. This is not because we consider that number of operations to be computable shortly, but it might be possible

[3] Commit `0b16750cf1f5444d3cc4bc943f65fe1eded6cca7`

[4] Remark that for this estimate, the JARJAR variant of NewHope is used, which is actually not recommended by the authors of NewHope because of this small security level. The normal version of NewHope uses $n = 1024$ and yields 233 bits of security.

that more efficient attacks are discovered. It is therefore good to look ahead with a typical dose of cryptographic paranoia.

4.3.2 Acceptable failure rate

Let's look at the failure probabilities of existing protocols: BCNS achieves $2^{-2^{15}}$ with large parameters [17], Peikert [45] does not give an exact number, and NewHope [10] and NewHope-Simple [9] achieve failure probability 2^{-60}. We aim for a success probability not worse than the latter two protocols.

In Theorem 3.1 we stated that the failure probability of our protocol is 2^{-k} per coefficient, for $k = \log(q) - \log(\beta) - 1$ and $\beta \geq \|\mathbf{be}\|_\infty$.

Intuitively, this means the total failure rate is at most $n \cdot 2^{-k}$, and thus $\Pr(\text{success}) \geq 1 - n \cdot 2^{-k}$. We can, however, improve this failure rate: recall that we need to use the protocol to compute a symmetric key, typically with a length of 256 bits. The failure rate we defined here is the probability that the protocol fails to achieve correctness for *any of the coefficients*. This introduces redundancy: if we look at e.g. only the first 256 bits, the failure probability will be decreased by a factor $(256/n)$.

As stated above, we want to achieve a failure probability of less than 2^{-60}. Since this probability in our case largely depends on the bitsize of q, it seems fair to take $\log(q) = 100$ as a starting point. Then

$$\begin{aligned} k &= 100 - \log(\beta) - 1 \\ \Pr(\text{succes}) &= 1 - 256 \cdot 2^{-100+\log(\beta)+1} \end{aligned}$$

We assume for now that this failure rate is indeed less than 2^{-60}, but the size of β depends indirectly on the other variables and thus needs to be verified after selecting those. Remark that most processors use 64-bit words and 64 bits is definitely too small for our q, but we would prefer keeping $\log(q) \leq 128$.

4.3.3 Choosing q and n

As existing literature using the RLWE estimators does not perform tests with larger parameters, we first decided to run a few tests with the algorithm to see how it performs on larger values. Again, a protocol like NewHope uses values $n = 2^{10} = 1024$, $\log(q) = 14$. In our first experiment, we pick a much larger value for n to see what happens; here we decide to choose $n = 2^{14} = 16384$, we take $\sigma = 3.192$ and run for $40 \leq \log(q) \leq 150$. The results of the tests are in Table 4.2 and corresponding Figure 4.1. Indeed we see that the number of bits of security decreases as we make q larger. We also remark that

$\log(q)$	uSVP	Decoding	Dual
40	9451.1	12110.8	9931.3
45	8066.8	9837.0	8494.7
50	6983.0	8263.2	7361.3
55	6116.5	7069.1	6435.0
60	5407.9	6129.7	5701.1
65	4822.8	5383.2	5081.2
70	4334.3	4769.1	4571.0
75	3920.3	4270.7	4124.4
80	3564.0	3848.5	3749.4
85	3256.5	3491.3	3446.4
90	2989.4	3184.5	3149.0
95	2757.3	2934.9	2890.2
100	2550.0	2695.5	2667.6
105	2365.3	2489.0	2480.1
110	2202.6	2312.3	2305.6
115	2057.4	2147.7	2160.6
120	1925.6	2000.6	2016.0
125	1804.1	1870.7	1885.9
130	1697.9	1752.5	1772.4
135	1597.8	1645.5	1670.3
140	1509.8	1549.7	1572.9
145	1426.4	1461.0	1488.9
150	1349.9	1380.5	1411.8

Table 4.2: Analysis for $n = 16384$, $\sigma = 3.192$

computing these estimates takes considerably larger amounts of computing power: on our test setup, computing the first row of results takes over 4.5 minutes, where computing the results for conventional parameter sizes usually takes under 10 seconds.

It also becomes apparent that these numbers are a lot larger than necessary: although a protocol that provides thousands of bits of security is of course interesting, we want to find smaller values to keep the messages small.

So, with $\log(q) = 100$, we will then try to find a suitable n: in Table 4.3 and corresponding Figure 4.2 we see the result of this test. We take $\sigma = 6.384$, as it is the recommended value from [56]. We decrease n first in steps of around 10000, then in steps of around 1000.

Recall that we are looking at cases providing 256 and 512 bits of security. We thus define two parameter sets: *Proposal-256*, with $n = 4001$ and *Proposal-512* with $n = 6007$.

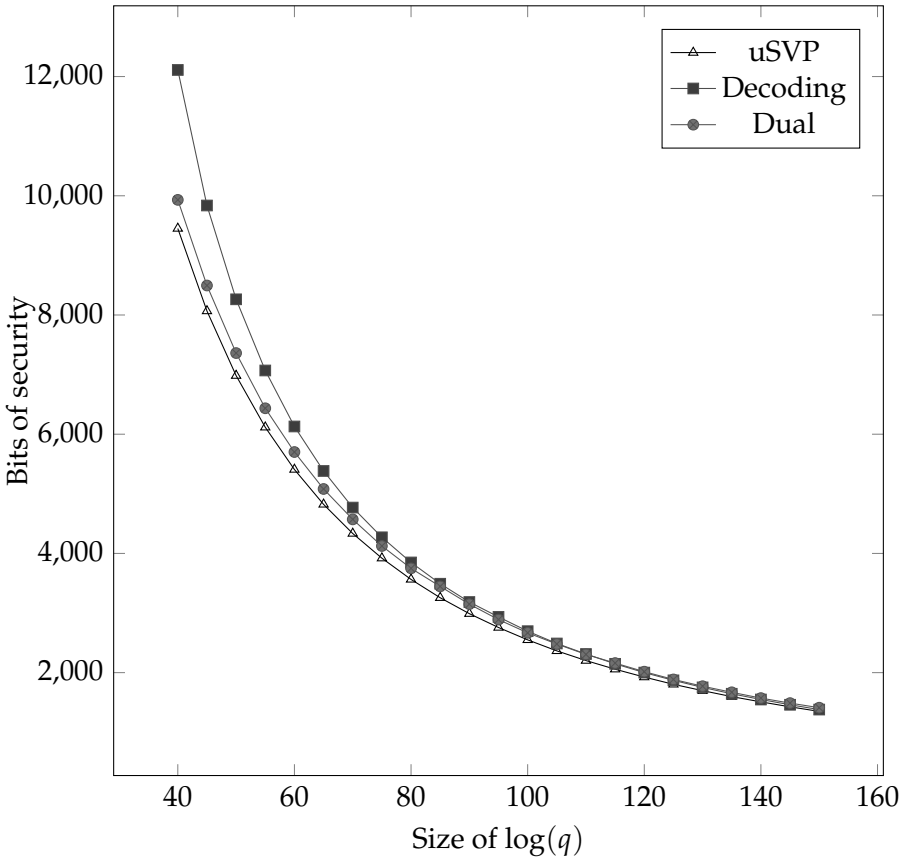

Figure 4.1: Plot of Table 4.2, the result of increasing $\log(q)$ when $n = 16384, \sigma = 3.192$.

n	uSVP	Decoding	Dual
30011	6414.2	6963.9	6697.4
20011	3550.1	3779.0	3731.1
10007	1219.8	1247.8	1224.5
8009	849.1	862.6	866.9
7001	678.3	690.9	693.2
6007	522.7	533.9	534.9
5003	380.1	389.9	385.8
4001	254.7	262.2	259.0
3001	149.8	154.9	155.1
2003	69.5	71.9	72.9
1009	31.8	46.4	47.6

Table 4.3: Decreasing n, keeping $\log(q) = 100, \sigma = 6.384$

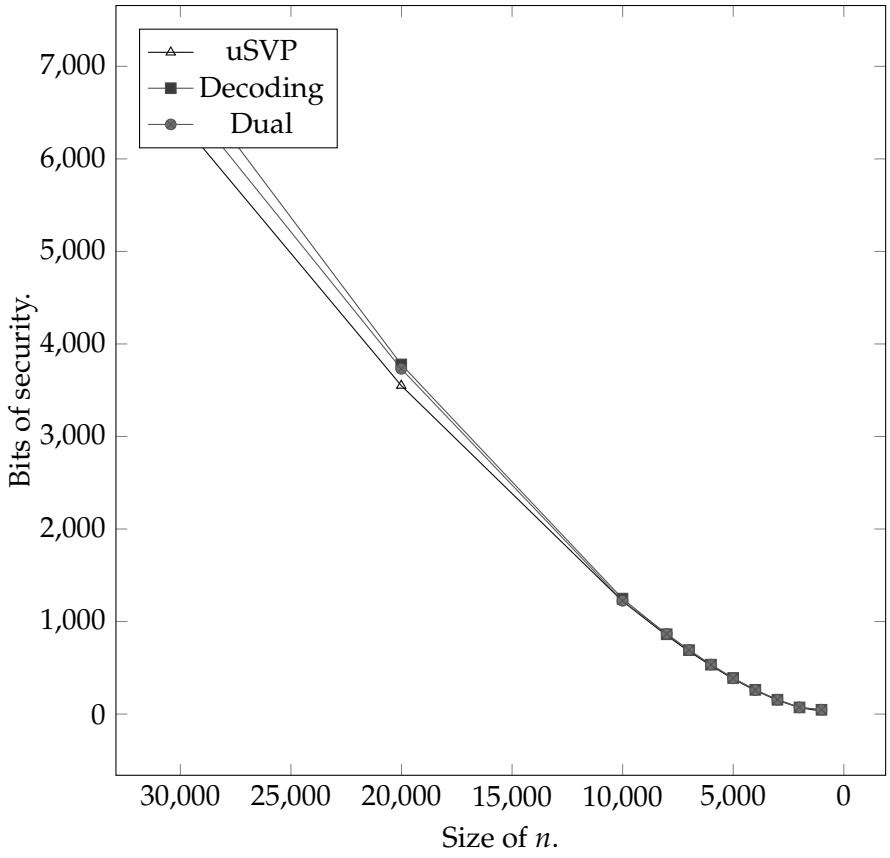

Figure 4.2: Plot of Table 4.3, the result of decreasing n when $\log(q) = 100, \sigma = 6.384$.

A next step is to find suitable sizes of q for these values. Recall that we know $q = 100$ is a suitable size to achieve correctness: however, we want to evaluate whether other options might be better suited and thus we run some tests with $90 \leq \log(q) \leq 110$. The results are in Table 4.5 for 256 bits of security and in Table 4.6 for 512 bits of security.

Indeed, looking at theses tests, taking $\log(q) = 100$ did not seem very far off. For $n = 4001$, we are 2 bits short of the goal of reaching 256 bits of security. This can be fixed simply by decreasing to $\log(q) = 99$. For $n = 6007$, we can increase to $\log(q) = 101$ to end up almost exactly at 512 bits. As these security estimates are of the level we want to achieve, we leave $\sigma = 6.384$ as we chose it.

We then compute the final failure probability for these values, assuming we require correctness of the first 256 bits and not of the whole value of n.

To compute $\beta \geq \|\mathbf{se}\|_\infty$, we need to estimate how large the sizes of $\mathbf{b}$ and $\mathbf{e}$ will be, as they are Gaussian samples.

We use the *tailcutting* method from [24] to achieve this: let $\tau \approx \sqrt{\lambda \cdot 2\ln(2)}$, then with probability $1 - 2^{-\lambda}$ no values larger than $\tau\sigma$ are sampled. We thus let λ be equal to the security level.

This means that for Proposal-256, we have

$$\begin{aligned}
\lambda = 256, n = 4001, \sigma = 6.384 \\
\tau = \sqrt{\lambda \cdot 2\ln(2)} = \sqrt{256 \cdot 2\ln(2)} \approx 18.839 \\
\beta \geq \|\mathbf{se}\|_\infty = n \cdot (\tau\sigma)^2 = 4001(18.839 \cdot 6.384)^2 \\
k = \log(q) - \log(\beta) - 1 \leq 99 - 26 - 1 = 72 \\
\Pr(\text{failure}) \leq 256 \cdot 2^{-72} = 2^8 \cdot 2^{-72} = 2^{-64}
\end{aligned}$$

which is indeed better than the 2^{-60} we were aiming for.

Similar for Proposal-512:

$$\begin{aligned}
\lambda = 512, n = 6007, \sigma = 6.384 \\
\tau = \sqrt{\lambda \cdot 2\ln(2)} = \sqrt{512 \cdot 2\ln(2)} \approx 26.642 \\
\beta \leq \|\mathbf{se}\|_\infty = n \cdot (\tau\sigma)^2 = 6007(26.642 \cdot 6.384)^2 \\
k = \log(q) - \log(\beta) - 1 \leq 101 - 28 - 1 = 72 \\
\Pr(\text{failure}) \leq 256 \cdot 2^{-72} = 2^8 \cdot 2^{-72} = 2^{-64}
\end{aligned}$$

which also meets our goal.

Of course, when using the protocol, we need an actual value of q instead of just its required bitlength. The tool by Albrecht makes these estimations for the largest value

n	$\log(q)$	σ	Pr(fail)	uSVP	Decoding	Dual
4001	99	6.384	2^{-64}	259.8	267.4	264.2
6007	101	6.384	2^{-64}	513.7	524.8	525.8

Table 4.4: Recommended parameter sets

possible with the given bitlength. In terms of this security level, it is thus not a problem to use $q = 2^{99} - 1$ and $q = 2^{101} - 1$ respectively. In practice, we want q to be prime and $q \equiv 1 \mod 2n$. This is for instance necessary when using the Number-Theoretic Transform (NTT) in implementations and when performing hardness reductions.

An example of a suitable value would under those assumptions be

$$q = 633825300114114700748351602441 = 2^{99} - 247.$$

We thus conclude with these two parameter sets for our RLWE-NIKE, given in Table 4.4.

$\log(q)$	uSVP	Decoding	Dual
90	311.7	320.0	318.1
91	305.4	313.2	311.8
92	299.2	306.9	304.6
93	292.1	300.7	298.4
94	287.7	294.5	294.8
95	280.7	288.8	286.8
96	276.3	283.0	280.7
97	271.0	277.4	277.2
98	265.8	272.2	269.4
99	259.8	267.4	264.2
100	254.7	262.2	259.0
101	250.4	257.1	256.4
102	246.1	252.4	251.3
103	241.0	247.3	246.2
104	236.0	243.1	239.4
105	232.6	238.8	236.9
106	228.5	234.3	232.7
107	224.3	230.5	228.5
108	220.2	226.3	224.4
109	216.9	222.2	220.2
110	212.0	218.5	216.1

Table 4.5: Changing $\log(q)$ at $n = 4001, \sigma = 6.384$ (Proposal-256).

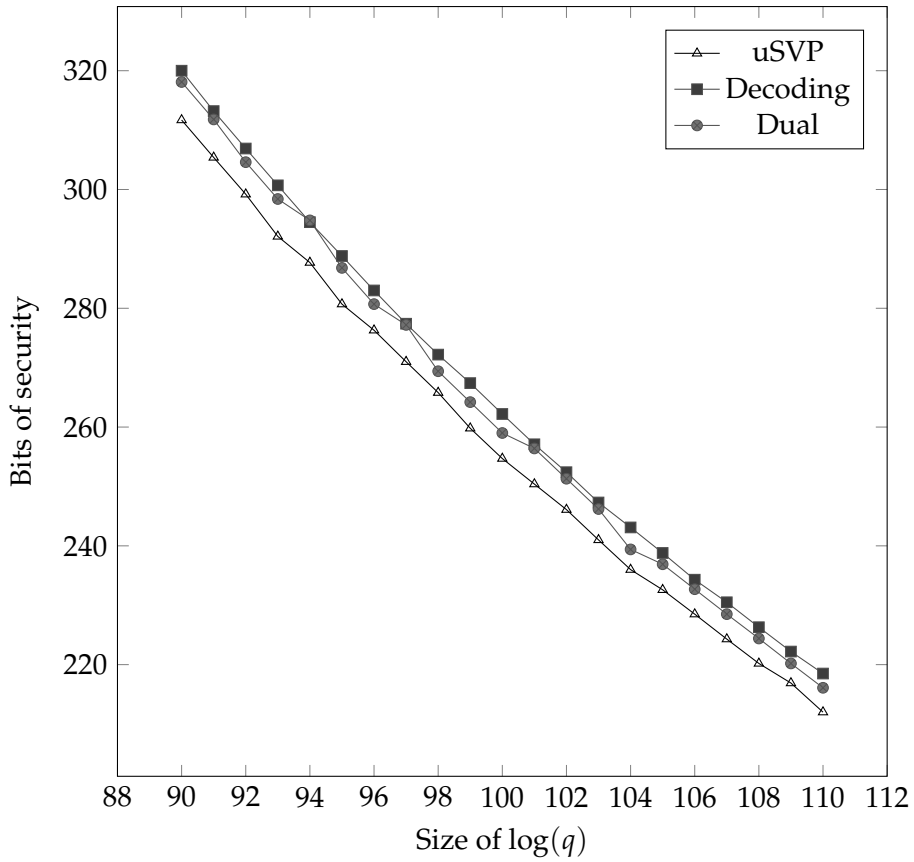

Figure 4.3: Plot of Table 4.5, the result of increasing $\log(q)$ when $n = 4001, \sigma = 6.384$ (Proposal-256).

$\log(q)$	uSVP	Decoding	Dual
90	630.6	643.1	648.5
91	618.1	630.5	635.9
92	606.6	619.1	623.3
93	595.2	606.6	610.8
94	583.9	596.2	599.4
95	572.6	584.9	588.0
96	563.3	574.6	576.7
97	553.1	563.4	565.4
98	542.0	553.2	555.2
99	533.8	544.0	545.0
100	522.7	533.9	534.9
101	513.7	524.8	525.8
102	504.7	515.8	519.8
103	496.7	506.8	506.7
104	488.7	497.8	501.7
105	479.8	488.9	488.8
106	471.9	481.0	473.9
107	463.1	473.1	466.0
108	456.2	465.2	469.0
109	448.4	457.4	450.4
110	441.6	450.3	442.6

Table 4.6: Changing $\log(q)$ at $n = 6007, \sigma = 6.384$ (Proposal-512).

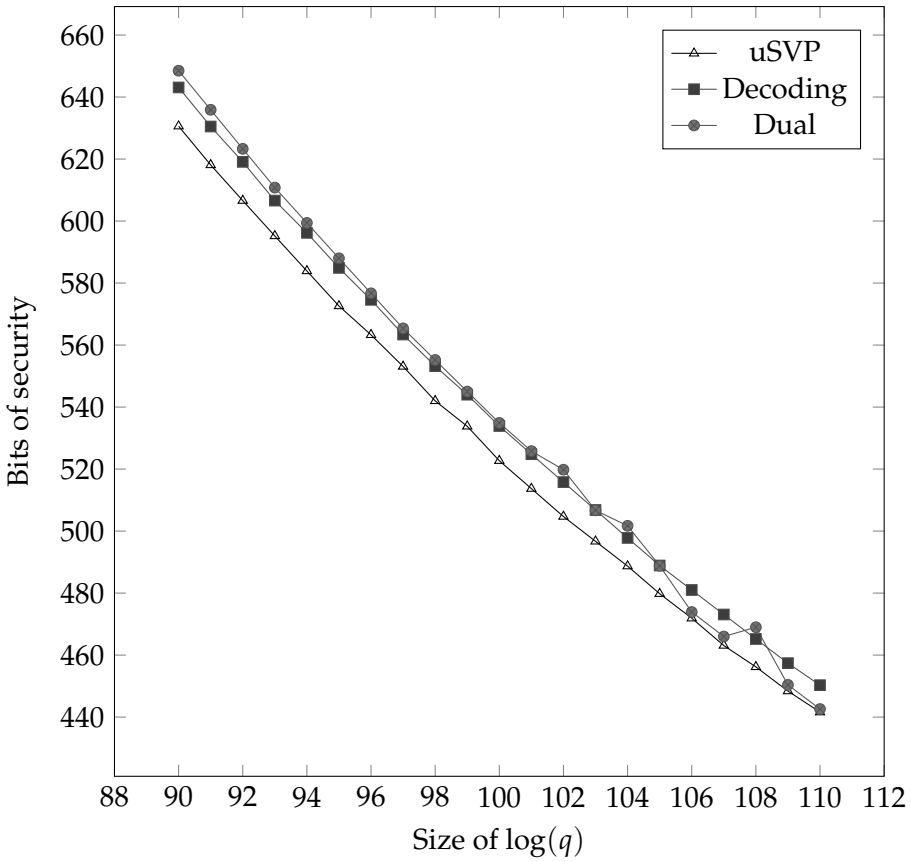

Figure 4.4: Plot of Table 4.6, the result of increasing $\log(q)$ when $n = 6007, \sigma = 6.384$ (Proposal-512).

Chapter 5

Real-world implications

In this chapter we look into the potential of using this protocol in real world scenarios: we first compute the message sizes that will be the results of the larger parameters and look into an existing experiment that tests the impact of large parameters on performance of current network architecture. We then look into some other topics for instance how to combine the system into a hybrid protocol, whether static parameters and keys are possible and whether we can indeed use the protocol for zero-roundtrip scenarios.

5.1 Consequences of using large parameters

A natural consequence of using larger parameters is that the message sizes also increase. Let's look at the sizes for some of the parameters in the previous chapter: for Proposal-256 with $\log(q) = 99$ and $n = 4001$, we get a message size of 396099 bits, which is approximately 48 kB. For Proposal-512 with $\log(q) = 101$ and $n = 6007$, we get a message size of 606707 bits, which is approximately 74 kB.

Again, let's compare this result to the other literature. In NewHope and NewHope-simple, for instance, the message containing the public key has a size of 1824 bytes. Of course, these protocols require other messages between the participants as well, making the total bandwidth requirement 2048 bytes for NewHope and 2176 bytes for NewHope-simple. For Supersingular Isogeny Diffie-Hellman, which is a much more recent and less studied system than lattices, the required key length is only 768 bytes, to achieve 192 bits of security [19]. An overview of this data is given in Table 5.1.

Scheme	Key Length	Bits of security
NewHope [10]	1824 bytes	233
NewHope-simple [9]	1824 bytes	233
SIDH [19]	768 bytes	192
Proposal-256:	49513 bytes	256
Proposal-512:	75839 bytes	512

Table 5.1: Comparison of key lengths for different systems.

5.1.1 Using larger messages in practice

The larger message sizes are inconvenient for some purposes, but will not lead to practical problems: we simply do not anticipate the system to be used for small-scale electronics that are not able to store large keys or compute them within a reasonable time frame. We also assume that the system will always be used with significant processing power, to avoid extremely long delays. The size of the messages is another matter, as these need transmitting over existing infrastructure. As the main advantage of this exchange over other RLWE-exchanges is omitting round-trips, we anticipate this exchange to be used in situations where internet access is very limited, and where the existing internet infrastructure might also be of limited availability.

Google Experiments on Large TLS Messages

As we did not produce a real-world implementation of the protocol as part of this work, we did not manage to test the problems it might lead to in practice. We can, however, take a look at two experiments performed by Google: one in 2016, the other very recently (results published April 2018 [38]).

In 2016, an experiment was conducted by the Google Chrome Team with larger-than-normal TLS messages, encrypted used their CECPQ1-algorithm [18, 37], which employs a form of hybrid cryptography by establishing a 32-byte shared secret through a Curve25519 key exchange, another 32-byte shared secret through NewHope [10] and then feeding these 64 bytes into the TLS Pseudo-Random Function.

Since Google operates both the Chrome browser (client-side) and a lot of popular web services (server side), they were able to implement the suite on both sides and run field tests, where a small percentage of Chrome users participated in the experiment.

Naturally, the TLS key agreement messages are larger than conventional messages. Google did not publish an official paper about the results of the experiment, but developer Adam Langley wrote a blog post [36] in which the results of the test are summarized. Although there were no actual issues with the networking hardware used, the latency

for a small group of users increased significantly. The median increase was only 1 ms, but for the the slowest 5% of users a 20 ms-increase occurred. For the slowest percentile, this increase was even 150 ms.

In the beginning of 2018, another practical experiment was done [38] by the same team, now specifically aimed at testing the effects of larger TLS handshakes: this was done by creating a dummy protocol that sends a TLS request, including a 'key' of a given length, and seeing how long it took servers to respond. For attempts with 3300 bytes, the latency of the handshakes was measured and compared to a control group: for the median desktop user the latency increased by 71.3%, but for the 95th percentile, the latency increase by 117%. A test with 10000 bytes was unfortunately not carried out: 21 out of 2500 servers tested turned out to drop the message altogether and thus that possibility was not pursued.

An interesting remark is that the popularity of NewHope as an RLWE primitive is partially due to the exposure generated by this experiment, which in turn led to an attack paper [26] which later had to be withdrawn due to an error in the proposed attack. Additionally, more interest was spiked in reaction attacks [27]. To avoid these, fresh public keys need to be made for each connection. NewHope and the Google implementation both do this.

5.2 Hybridizing the protocol

As stated in Section 2.5, the introduction of new cryptographic protocols comes with the risk of them being broken, as they have not been studied as extensively as existing ones. Additionally, there will be less trust in their workings, meaning adoption will be slow. We therefore recommend to implement the proposed protocol as part of a hybrid scheme. This can be done by executing both a key exchange using this protocol and an existing NIKE, for instance ECDH, and feeding the resulting shared secrets into a key derivation function that will give us the actual shared key to use.

Of course, this will increase the message sizes: this will, however, not be a significant increase compared to the already large messages, and should thus definitely be done to increase the security.

5.3 Number of roundtrips

A big benefit of non-interactive key exchanges is that they require less interaction between the two parties. In an ideal scenario, Alice and Bob upload their public keys to

some key server and the only thing needed for them to communicate is retrieving the keys from the server: the first message from Alice to Bob can already contain the data she wants to send him. This is referred to as *zero round trip communication*. That would, however mean that the keys are fully static: as the security of this new protocol and of RLWE in general have not been looked at in-depth, it is safer to use ephemeral keys.

Assuming we standardize the system polynomial $\mathbf{a}$ [1], this means Alice and Bob need to create a new key for every communication. That means the key exchange only takes *half a round trip*, that is: one message in each direction. This reduction in the number of round trips is an advantage in scenarios where internet access is limited, not only because it requires less data to arrive but also because these messages do not need to be sent simultaneously. The only requirement is that Alice manages to get one message with her public key out to Bob *at some point in time* and Bob manages to do the same thing at any point in time before or after that. Both parties are able to send information along after they received the key data from the other party. Compare that to the scenario with an interactive exchange: we then need to wait for a moment in time when Alice and Bob are both online and the connection quality is good enough to exchange data between the two. That is not a feasible option in all cases: examples are situations where Alice and/or Bob are in a remote area with only limited internet access, are traveling and thus offline often, or that they want to limit their time online to avoid being tracked and conserve battery life. In all of these cases, it is complicated to exchange a key interactively, especially if the parties are not able to coordinate the time that they are online to perform those roundtrips. A NIKE is then definitely preferable.

[1] Which is also a potential security risk, addressed in Section 6.2.3.

Chapter 6

In conclusion

6.1 Summary and conclusion

We researched the possibility of creating a post-quantum Non-Interactive Key Exchange (NIKE) based on Ring-Learning With Errors (RLWE). We first discussed the basics of (symmetric and asymmetric) cryptography, and how we reason about their security and correctness. We then explained the mathematical knowledge required to understand the primitives relevant to our work. We then explained Ring-Learning With Errors, and how it is possible to base key exchanges upon that primitive. Because these exchanges introduce an error into the shared secret by design, they always need some way to get rid of the discrepancy between the two variants of the shared secret. Usually this is solved using a reconciliation function or an encode-and-extract procedure. This always means interaction is required, and it always means the two parties have a different role in the exchange.

We show in this thesis that it is possible to remove this discrepancy in a different way: by increasing the lattice modulus q significantly, the probability that the highest-order bit of a coefficient is influenced by the error polynomial becomes very small. This means the lattice problem becomes easier to solve: that can be remedied by increasing the lattice dimension n as well. Using the LWE-hardness estimator developed by Martin Albrecht, we estimate how hard known lattice attacks would be when executing them on an RLWE exchange with these parameter sizes. Using those estimations, we see that we would be able to get an exchange with a security level of 256 bits by letting $n = 4001$, $\log(q) = 99$, $\sigma = 6.384$, and of 512 bits by letting $n = 6007$, $\log(q) = 101$ and $\sigma = 6.384$. We validate that using these parameters, the failure rate of the protocol is around 2^{-64} for both sets, which is better than the failure rate of regular RLWE schemes such as NewHope. Note that a disadvantage of this scheme is the large size of the public keys:

for 256 bits of security, they are around 48 kB and for 512 bits of security, they are around 74 kB. Although these large keys using the protocol described in this thesis infeasible for some real-life applications, there are situations where the advantage of a NIKE over an interactive exchange outweighs the larger transmission and computation times that will be required.

6.2 Future work

This thesis is a first step towards an RLWE-NIKE, but it also leads to more research questions about this protocol, or RLWE schemes in general.

6.2.1 Real-world implementation

The work in this thesis shows a non-interactive exchange based on RLWE is indeed possible, but we did not write an implementation for real-world cryptographic purposes. A proof-of-concept implementation in Sage was created, see Appendix A, but is obviously not usable for any practical purposes. A future project would be making such an implementation: not only is this a next step towards possible usage in real life, but it is also a good way to see whether any issues using the protocol would arise in practice, for instance related to the effect of the message size on infrastructures used.

6.2.2 Public Key Validation

As of now, no research has been done into the possibility of public key validation. A common attack on asymmetric cryptographic systems is based on malformed keys: malicious user Mallory announces to Alice that she wants to establish a shared key and sends a p_M along, pretending that it is a valid secret key. In reality, however, p_M is not of the right form and is designed so that Mallory can learn information about Alice's secret by looking at her response. Because Mallory is able to make a lot of connection attempts and Alice keeps responding with values that relate to her long-term public key, and because Mallory can verify whether her guesses are correct by looking at reaction Alice gives, the attack is potentially successful [27].

We currently avoid this scenario by avoiding static keys: the attack requires that Mallory can 'probe' the key of Alice several times, and that is thus avoided. Would we want to avoid these attacks and also implement fully static keys, we would need a method to validate whether a given 'public key' is indeed a public key. In ECDH this is, for instance, achieved by computing whether the given point lies on the curve and has the correct group order.

RLWE based schemes do not have such a method: this intuitively follows from the fact that RLWE-keys are supposed to be indistinguishable from random samples.

Very recently (January 2018) such a method was proposed for RLWE schemes. [22] More research is needed to see whether this method can be applied to the RLWE-NIKE proposed in this thesis.

6.2.3 Risks of fixed polynomial

In the system we propose, the public keys are partially based upon the fixed polynomial **a** which is set as a system-wide constant.

Although the authors of [17] use such a constant, the later work [10] points out the risks concerning this approach. Observe that if this **a** is compromised somehow, *all* keys generated using it might be unsuitable and *all* communication using them might be compromised.

There are two main scenarios that pose a significant risk. First, a standardized parameter **a** could be backdoored by an adversary. In the past, multiple instances of these kinds of backdoors have been inserted into standards by governments and their standardization institutions, most prominently by NIST. While there exist certain "Nothing-up-my-sleeve"-methods that make creating such a backdoor harder, these methods might still leave room for abuse [15].

The second scenario is that a new algorithm is discovered that enables an "All-for-the-price-of-one"-attack method, which means that an attack on one instance of **a** breaks *all* implementations that use that instance. The *Logjam*-attack from the 2015 paper *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice* [1] demonstrates such a scenario: they showed that the computations necessary for breaking Diffie-Hellman largely depend on the group instance, and not on the primes used. Combine this with the fact that a lot of TLS implementations use the default parameters, and it is suddenly very attractive to perform the precomputations for that default group.

The solution proposed in [10] is to avoid the creation of a long-term constant. This is not an acceptable solution in our case, as that would introduce a need for interaction between the parties. We therefore do recommend standardizing this parameter now, but remark that this risk does exist for our system as it does for [17]. More research is needed to be sure whether this is truly a problem. For now, the problem should be mitigated by at the very least using a nothing-up-my-sleeve method for generating **a**. An interesting method cited in [15] is the one proposed by Michael Scott, where he proposes to base the curve parameters for ECC on the first digits of π: this way, anyone knows that these digits are not selected based on some special property or secret backdoor. Similarly,

we could use those digits as the source for the coefficients of **a** instead of generating it randomly.

Bibliography

[1] ADRIAN, D., BHARGAVAN, K., DURUMERIC, Z., GAUDRY, P., GREEN, M., HALDERMAN, J. A., HENINGER, N., SPRINGALL, D., THOMÉ, E., VALENTA, L., VANDERSLOOT, B., WUSTROW, E., ZANELLA-BÉGUELIN, S., AND ZIMMERMANN, P. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *22nd ACM Conference on Computer and Communications Security* (Oct. 2015). 6.2.3

[2] AJTAI, M. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1996), STOC '96, ACM, pp. 99–108. 1.1

[3] ALBRECHT, M. R. On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II* (2017), J. Coron and J. B. Nielsen, Eds., vol. 10211 of *Lecture Notes in Computer Science*, pp. 103–129. 3

[4] ALBRECHT, M. R., CID, C., FAUGÈRE, J.-C., FITZPATRICK, R., AND PERRET, L. On the complexity of the BKW algorithm on LWE. *Des. Codes Cryptography 74*, 2 (2015), 325–354. 1.1

[5] ALBRECHT, M. R., CURTIS, B., DEO, A., DAVIDSON, A., PLAYER, R., POSTLETHWAITE, E., VIRDIA, F., AND WUNDERER, T. Estimate all the {LWE, NTRU} schemes! *Preprint on Github: https://estimate-all-the-lwe-ntru-schemes.github.io/paper.pdf?v=apr18* (Apr. 2018). 4.3.1

[6] ALBRECHT, M. R., FAUGÈRE, J.-C., FITZPATRICK, R., AND PERRET, L. Lazy Modulus Switching for the BKW Algorithm on LWE. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings* (2014), H. Krawczyk, Ed., vol. 8383 of *Lecture Notes in Computer Science*, Springer, pp. 429–445. 6

[7] ALBRECHT, M. R., FITZPATRICK, R., AND GÖPFERT, F. On the Efficacy of Solving LWE by Reduction to Unique-SVP. In *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers* (2013), H. Lee and D. Han, Eds., vol. 8565 of *Lecture Notes in Computer Science*, Springer, pp. 293–310. 5

[8] ALBRECHT, M. R., PLAYER, R., AND SCOTT, S. On the concrete hardness of Learning with Errors. *J. Mathematical Cryptology 9*, 3 (2015), 169–203. 4.1

[9] ALKIM, E., DUCAS, L., PÖPPELMANN, T., AND SCHWABE, P. NEWHOPE without reconciliation. *IACR ePrint: https://cryptojedi.org/papers/newhopesimple-20161217.pdf* (2016), 1–9. 1.1, 2.3.2, 2.4, 4.3.2, 5.1

[10] ALKIM, E., DUCAS, L., PÖPPELMANN, T., AND SCHWABE, P. Post-quantum Key Exchange-A New Hope. In *USENIX Security Symposium* (2016), pp. 327–343. 1.1, 2.3, 2.3.2, 2.3.3, 3.2, 4.3.1, 4.3.2, 5.1, 5.1.1, 6.2.3

[11] ARORA, S., AND GE, R. New Algorithms for Learning in Presence of Errors. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I* (2011), L. Aceto, M. Henzinger, and J. Sgall, Eds., vol. 6755 of *Lecture Notes in Computer Science*, Springer, pp. 403–415. 6

[12] BAI, S., AND GALBRAITH, S. D. Lattice Decoding Attacks on Binary LWE. In *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings* (2014), W. Susilo and Y. Mu, Eds., vol. 8544 of *Lecture Notes in Computer Science*, Springer, pp. 322–337. 5

[13] BELLARE, M., DESAI, A., POINTCHEVAL, D., AND ROGAWAY, P. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings* (1998), pp. 26–45. 2.4

[14] BERNSTEIN, D. J., BUCHMANN, J., AND DAHMEN, E. *Post-Quantum Cryptography*. Springer, 2009. 1.1

[15] BERNSTEIN, D. J., CHOU, T., CHUENGSATIANSUP, C., HÜLSING, A., LAMBOOIJ, E., LANGE, T., NIEDERHAGEN, R., AND VAN VREDENDAAL, C. How to Manipulate Curve Standards: A White Paper for the Black Hat http: //bada55.cr.yp.to. In *Security Standardisation Research - Second International Conference, SSR 2015, Tokyo, Japan, December 15-16, 2015, Proceedings* (2015), L. Chen and S. Matsuo, Eds., vol. 9497 of *Lecture Notes in Computer Science*, Springer, pp. 109–139. 6.2.3

[16] BINDEL, N., HERATH, U., MCKAGUE, M., AND STEBILA, D. Transitioning to a Quantum-Resistant Public Key Infrastructure. In *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings* (2017), T. Lange and T. Takagi, Eds., vol. 10346 of *Lecture Notes in Computer Science*, Springer, pp. 384–405. 2.5.1, 2.5.1

[17] BOS, J. W., COSTELLO, C., NAEHRIG, M., AND STEBILA, D. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *IEEE Symposium on Security and Privacy* (2015), IEEE, pp. 553–570. 2.3.2, 2.3.3, 4.3.2, 6.2.3

[18] BRAITHWAITE, M. Experimenting with Post-Quantum Cryptography. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html, July 2016.
5.1.1

[19] COSTELLO, C., JAO, D., LONGA, P., NAEHRIG, M., RENES, J., AND URBANIK, D. Efficient Compression of SIDH Public Keys. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I* (2017), J. Coron and J. B. Nielsen, Eds., vol. 10210 of *Lecture Notes in Computer Science*, pp. 679–706. 5.1

[20] DAEMEN, J., AND RIJMEN, V. Rijndael for AES. In *AES Candidate Conference* (2000), pp. 343–348. 2.1

[21] DIFFIE, W., AND HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory 22*, 6 (Nov. 1976), 644–654. 1.1, 2.2, 2.3, 2.1

[22] DING, J., RV, S., ALSAYIGH, S., AND CLOUGH, C. How to validate the secret of a Ring Learning with Errors (RLWE) key. *IACR Cryptology ePrint Archive: http://eprint.iacr.org/2018/081 2018* (2018), 81. 6.2.2

[23] DING, J., XIE, X., AND LIN, X. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. *IACR ePrint: http://eprint.iacr.org/2012/688.pdf* (2012). 2.3.2

[24] DUCAS, L., DURMUS, A., LEPOINT, T., AND LYUBASHEVSKY, V. Lattice Signatures and Bimodal Gaussians. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I* (2013), R. Canetti and J. A. Garay, Eds., vol. 8042 of *Lecture Notes in Computer Science*, Springer, pp. 40–56. 4.3.3

[25] DWARAKANATH, N. C., AND GALBRAITH, S. D. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Appl. Algebra Eng. Commun. Comput. 25*, 3 (2014), 159–180. 2.3.2

[26] ELDAR, L., AND SHOR, P. W. An Efficient Quantum Algorithm for a Variant of the Closest Lattice-Vector Problem. *arXiv preprint http://arxiv.org/abs/1611.06999* (2016). 5.1.1

[27] FLUHRER, S. Cryptanalysis of ring-LWE based key exchange with key share reuse. https://eprint.iacr.org/2016/085. 5.1.1, 6.2.2

[28] GROVER, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (1996), ACM, pp. 212–219. 1.1

[29] GUO, Q., JOHANSSON, T., AND STANKOVSKI, P. Coded-BKW: Solving LWE Using Lattice Codes. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I* (2015), R. Gennaro and M. Robshaw, Eds., vol. 9215 of *Lecture Notes in Computer Science*, Springer, pp. 23–42. 2

[30] HELLMAN, M. E. An overview of public key cryptography. *IEEE Communications Magazine 40*, 5 (2002), 42–49. 2.2, 2.3

[31] HOFFSTEIN, J., PIPHER, J., AND SILVERMAN, J. H. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory* (June 1998), vol. 1423 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 267–288. 1.1

[32] KAHN, D. *The Codebreakers*. Weidenfeld and Nicolson, 1974. 1.1

[33] KIMBALL, K. Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms. https://www.federalregister.gov/documents/2016/12/20/2016-30615/announcing-request-for-nominations-for-public-key-post-quantum-cryptographic-algorithms, Dec. 2016. 1.1

[34] KOBLITZ, N. Elliptic curve cryptosystems. *Mathematics of computation 48*, 177 (1987), 203–209. 2.3, 2.2

[35] LANGE, T., AND TAKAGI, T. *Post-Quantum Cryptography–PQCrypto 2017: 8th International Workshop, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, vol. 10346 of *Lecture Notes in Computer Science*. Springer, 2017. 1.1

[36] LANGLEY, A. CECPQ1 results. https://www.imperialviolet.org/2016/11/28/cecpq1.html, Nov. 2016. 5.1.1

[37] LANGLEY, A. Intent to Implement and Ship: CECPQ1 for TLS. https://groups.google.com/a/chromium.org/forum/#!topic/security-

dev/DS9pp2U0SAc, July 2016. 1.1,
5.1.1

[38] LANGLEY, A. Post-quantum confidentiality for TLS.
https://www.imperialviolet.org/2018/04/11/pqconftls.html, 2018. 5.1.1

[39] LINDNER, R., AND PEIKERT, C. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings* (2011), A. Kiayias, Ed., vol. 6558 of *Lecture Notes in Computer Science*, Springer, pp. 319–339. 4

[40] LYUBASHEVSKY, V. Converting NewHope/LWE key exchange to a Diffe-Hellman-like algorithm - Cryptography Stack Exchange.
https://crypto.stackexchange.com/questions/48146/converting-newhope-lwe-key-exchange-to-a-diffe-hellman-like-algorithm/48149, June 2017.
3.1

[41] LYUBASHEVSKY, V., PEIKERT, C., AND REGEV, O. On Ideal Lattices and Learning with Errors over Rings. *J. ACM 60*, 6 (2013), 43:1–43:35. 2.3.2

[42] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of Applied Cryptography*. CRC press, 1996. 1.1

[43] MILLER, V. S. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings* (1985), pp. 417–426. 2.3, 2.2

[44] PEIKERT, C. Public-key cryptosystems from the worst-case shortest vector problem. *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09* (2009), 333. 2.3.2

[45] PEIKERT, C. Lattice cryptography for the internet. In *International Workshop on Post-Quantum Cryptography* (2014), vol. 8772 of *Lecture Notes in Computer Science*, Springer, pp. 197–219. (document), 2.3.2, 3.3, 3.1, 3.3, 4.3.2

[46] REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM 56*, 6 (2009), 1–40. 1.1, 2.3.2

[47] REYNOLDS, M. Google on track for quantum computer breakthrough by end of 2017. https://www.newscientist.com/article/2138373-google-on-track-for-quantum-computer-breakthrough-by-end-of-2017/, June 2017.
1.1

[48] RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126. 1.1

[49] SCHANCK, J. M., WHYTE, W., AND ZHANG, Z. Circuit-extension handshakes for Tor achieving forward secrecy in a quantum world. *Proceedings on Privacy Enhancing Technologies 2016*, 4 (2016), 219–236. 2.5, 2.5, 2.5

[50] SHOR, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing 26*, 5 (Oct. 1997), 1484–1509. 1.1

[51] SHOUP, V. A Proposal for an ISO Standard for Public Key Encryption. *IACR Cryptology ePrint Archive 2001* (2001), 112. 2.2

[52] SINGH, V. A Practical Key Exchange for the Internet using Lattice Cryptography. https://eprint.iacr.org/2015/138, 2015. 1.1

[53] SMART, N. P., ALBRECHT, M. R., LINDELL, Y., ORSINI, E., OSHETER, V., PATERSON, K. G., AND PEER, G. LIMA. *Submission to National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/ round-1-submissions* (2017). 4.3.1

[54] THE SAGE DEVELOPERS. *SageMath, the Sage Mathematics Software System (Version 8.2)*. 2018. http://www.sagemath.org. 4.1

[55] UNGER, N., AND GOLDBERG, I. Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging. *PoPETs 2018*, 1 (2018), 21–66. 2.5

[56] VAN HEESCH, M. *Post-Quantum Key Exchange Based on Ring-LWE*. Master's Thesis, Technische Universiteit Eindhoven: https://research.tue.nl/en/studentTheses/post-quantum-key-exchange-based-on-ring-lwe, 2017. 3.2, 4.1, 4.1, 4.2.3, 4.3.3

Appendix A

Sage implementation of NIKE

On the following pages we provide a simple Sage implementation of the protocol. This is not optimized, nor is it protected against implementation errors, side-channel attacks or other issues. It is thus merely a proof of concept and not to be used for real-world scenarios.

```
import numpy
from sage.stats.distributions.discrete_gaussian_polynomial
    import DiscreteGaussianDistributionPolynomialSampler

from numpy import right_shift as rs

dimension = 4001      # degree of polynomials
modulus = 6338253001141147007483516024 41      # modulus
sigma = 6.192 # sigma

# Quotient polynomial ring
R.<X> = PolynomialRing(GF(modulus))      # Gaussian field of
    integers
Y.<x> = R.quotient(X^(dimension) + 1)    # Cyclotomic field

def generate_error():
    f = DiscreteGaussianDistributionPolynomialSampler(ZZ['x'],
        dimension, sigma)()
    return Y(f)

def generate_polynomial():
    # uniformly sampled from Quotient Polynomial Ring in x over
        Finite Field
    return Y.random_element()

def truncate(poly):
    coefficients = poly.list()
    key = []
    i = 0
    while i < 265:
        coeffbit = rs((poly.list()[i]), log(modulus-1, 2))
        key.append(1 if coeffbit == 1 else 0)
        i = i + 1
    return "".join(map(str, key))

# Shared matrix (A)
shared = generate_polynomial()
```

```
# Alice values
alice_secret = generate_error() # secret generated from error
    distribution
alice_error = generate_error()
alice_value = shared * alice_secret + alice_error

# Bob values
bob_secret = generate_error()    # secret generated from error
    distribution
bob_error = generate_error()
bob_value = shared * bob_secret + bob_error

# Bob key
bob_key = alice_value * bob_secret
bob_key_binary = truncate(bob_key)

# Alice key
alice_key = bob_value * alice_secret
alice_key_binary = truncate(alice_key)

if (alice_key_binary == bob_key_binary):
    print "Keys match!"
    print ((alice_key_binary, 2))
else:
    print "Keys do not match!"
```