An Analysis of Lattice-Based Key Exchange Protocols

Master-Thesis von Susanne Rieß
Tag der Einreichung:

1. Gutachten: Prof. Dr. Johannes Buchmann
2. Gutachten: Nina Bindel

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Mathematik
Fachbereich Informatik
Kryptographie und Computeralgebra

An Analysis of Lattice-Based Key Exchange Protocols

Vorgelegte Master-Thesis von Susanne Rieß

1. Gutachten: Prof. Dr. Johannes Buchmann
2. Gutachten: Nina Bindel

Tag der Einreichung:

Affidavit

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, March 2, 2016

Susanne Rieß

Abstract

In view of the expected cryptanalysis it is important to find alternatives for currently used cryptographic primitives, which are often based on number theoretic problems. In the past years, several key exchange protocols were developed which base there security on presumably quantum-resistant problems, such as lattice problems. We analyze and compare those lattice-based protocols. Three unauthenticated and five authenticated protocols are part of our study. We choose parameters for bit security levels of approximately 100 and 192 bit and use those parameters in our C++ implementations. These implementations are run on a server and the running times and parameter choices are the base for our comparison. We conclude that the protocol by Bos et al. (IEEE Security & Privacy 2015) and the protocol by Lyubashevsky et al. (Eurocrypt 2010) are the best performing unauthenticated protocols, while the protocol by Fujioka et al. (ASIACCS 2013) and the protocol by Peikert (PQCrypto 2014) are the best performing authenticated key exchange protocols.

Acknowledgement

This thesis consumed a lot of dedication, work, and research. Still, the great support and effort of some people contributed to the result. First, I would like to thank Professor Johannes Buchmann and Nina Bindel for the possibility of writing my thesis in the research field of lattice-based cryptography. I express my gratitude to Nina Bindel for her constant support, helpful discussions and guidance throughout the process of this thesis. Moreover, I would like to thank my family and friends for their moral support and encouragement during difficult phases of this research.

Contents

List of Figures

List of Tables

Notation

$\mathbb{N}$	Set of natural numbers
$\mathbb{Z}$	Set of integers
$\mathbb{Q}$	Set of rational numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
$\mathbb{Z}_n$	Set of integers modulo n
$\mathbb{Z}_n^\times$	Multiplicative group of invertible elements modulo n
$\mathcal{R}$	Commutative ring with 1
$\mathcal{R}^\times$	Multiplicative group of invertible elements of $\mathcal{R}$
$\langle x_1, \ldots, x_n \rangle$	The ideal generated by the elements $x_1, \ldots, x_n$
$\|\|x\|\|$	The ℓ_2-norm for $x \in \mathbb{C}^n$, also denoted by $\|\| \cdot \|\|_2$
$f(x) = O(g(x))$	$\lim_{x\to\infty} \|f(x)/g(x)\| < \infty$
$f(x) = \omega(g(x))$	$\lim_{x\to\infty} \|f(x)/g(x)\| = \infty$
$\rho_{\sigma,v}(x)$	The probability function of the spherical continuous Gaussian distribution over $\mathbb{R}^n$ with center $v \in \mathbb{R}^n$ and standard deviation $\sigma \in \mathbb{R}$
$D_{\mathbb{Z}^n,\sigma,v}$	Probability function of the discrete spherical Gaussian distribution over $\mathbb{Z}^n$
$\triangle(X, Y)$	The statistical distance of the two discrete random variables X and Y on the countable set V
$\mathcal{R}[X]$	The ring of polynomials in one variable with coefficients in $\mathcal{R}$
K	Algebraic number field
O_K	Ring of integers of the algebraic number field K
$[K : \mathbb{Q}]$	Degree of the field extension $K/\mathbb{Q}$
$\varrho_1, \ldots, \varrho_r$	The $\mathbb{Q}$-homomorphisms from K to $\mathbb{R}$, called real embeddings
$\tau_1, \overline{\tau_1} \ldots, \tau_s, \overline{\tau_s}$	The $\mathbb{Q}$-homomorphisms from K to $\mathbb{C}$ which are not real embeddings, called complex embeddings
$\Phi_m(X)$	m-th cyclotomic polynomial
K_m	m-th cyclotomic number field
Γ	Denotes a lattice which is a discrete additive subgroup of $\mathbb{R}^n$
$\lambda_1(\Gamma)$	The length of the shortest non-zero vector in the lattice Γ
Γ^*	The dual lattice of a lattice Γ
$\Lambda_q(A)$	$= \{y \in \mathbb{Z}^m : y = A^\top s \mod q \text{ for some } s \in \mathbb{Z}^n\}$, a q-ary lattice
$\Lambda_q^\perp(A)$	$= \{y \in \mathbb{Z}^m : Ay = 0 \mod q\}$, a q-ary lattice
κ	Denotes the security parameter
$\mathcal{H}_{r,s}$	$= \{x \in \mathbb{R}^r \times \mathbb{C}^{2s} : x_{r+j} = \bar{x}_{r+s+j} \text{ for all } j = 1, \ldots, s\} \subset \mathbb{C}^{r+2s=n}$
$\bar{x}$	$= a - ib$ the complex conjugate of $x = a + ib$ with $a, b \in \mathbb{R}$
$j : K \to \mathcal{H}_{r,s}$	The canonical embedding
R	$= \mathbb{Z}[X]/\langle f \rangle$ where $f \in \mathbb{Z}[X]$ of degree n
R_q	$= R/qR$
$c : R \to \mathbb{R}^n$	The coefficient embedding
$Tr_{K/\mathbb{Q}}$	The field trace of K
I^V	The dual ideal of the fractional ideal $I \subset K$
$K_{\mathbb{R}}$	$= K \otimes \mathbb{R}$
$r \xleftarrow{\$} \mathcal{M}$	Element r is drawn uniformly at random from a finite set $\mathcal{M}$
$r \leftarrow \chi$	Element r is drawn according to the probability distribution χ

Abbreviations

SVP	Shortest vector problem
γ-SVP	γ-approximate SVP
$GapSVP_\gamma$	Decisional γ-approximate SVP
BDD_α	α-bounded distance decoding problem
SIS	Short integer solution problem
LWE	Search learning with errors problem
$DLWE$	Decisional learning with errors problem
R-LWE	Search ring learning with errors problem
R-$DLWE$	Decisional ring learning with errors problem
R-SIS	Ring short integer solution problem
KEM	Key encapsulation mechanism
PKE	Public key encryption scheme
IND-CPA	Indistinguishability under chosen-plaintext attacks
IND-CCA	Indistinguishability under chosen-ciphertext attacks
MAC	Message authentication code
PRF	Pseudorandom function
KDF	Key derivation function
Sig	Signature scheme
KE	Key exchange protocol
AKE	Authenticated key exchange protocol
passive PPT	Secure against passive probabilistic polynomial-time adversaries
$ACCE$	Secure in the authenticated and confidential channel establishment
wPFS	Weak perfect forward secrecy
KCI	Key compromise impersonation
MEX	Maximal exposure attacks
CK^+	Security model that combines the Canetti-Krawczyk security model with wPFS, and security against KCI and MEX
BR	Bellare-Rogaway security model
auth. CCA	Secure authenticated CCA encryption scheme
SK	Security model by Canetti and Krawczyk [CK02] that implies wPFS
StdM	Standard model
ROM	Random oracle model
$FO(\cdot)$	Fujisaki-Okamoto transformation of an IND-CPA KEM into an IND-CCA KEM
JD	KE by Ding, Xie, and Lin [JD12]
ring-JD	KE by Ding, Xie, and Lin [JD12], based on the R-LWE problem
BCNS	KE by Bos, Costello, Naehrig, and Stebila [BCNS14]
FSXY12	AKE by Fujioka, Suzuki, Xagawa, and Yoneyama [FSXY12]
FSXY13	AKE by Fujioka, Suzuki, Xagawa, and Yoneyama [FSXY13]
two-pass ZZD	two-pass AKE by Zhang, Zhang, Ding, Snook, and Dagdelen [ZZD+14]
one-pass ZZD	one-pass AKE by Zhang, Zhang, Ding, Snook, and Dagdelen [ZZD+14]
Peikert	AKE by Peikert [Pei14]
LPR	PKE by Lyubashevsky, Peikert, and Regev [LPR12]
SS13	PKE by Stehlé and Steinfeld [SS13]
BLISS	Signature scheme by Ducas, Durmus, Lepoint, and Lyubashevsky [DDLL13]
ms	milliseconds

Introduction

Cryptography is involved in many parts of our daily life, e.g. credit cards, internet banking, electronic voting etc. An important aspect of cryptography is methods of exchanging a secret key through a possibly eavesdropped conversation without sharing any secret before. Those methods are called key exchange protocols. Many of the currently used key exchange protocols are based on the assumption that the discrete logarithm or factoring certain numbers are hard problems. These assumptions do not hold true for quantum algorithms. Therefore, we compare key exchange protocols that are based on lattice problems as those are assumed to be quantum resistant. That makes them possible candidates for replacing today's methods and achieving quantum resistant security.

The aim of this thesis is to compare lattice-based key exchange protocols. The comparison is based on a theoretical and a practical analysis. In the theoretical comparison we analyse the key exchange protocols in terms of the involved cryptographic building blocks. For the practical analysis we determine parameters for approximate bit security levels of 100 and 192 bit by using the LWE-Estimator by Albrecht et al. [APS15]. With those parameters we run our C++ implementation which is the base for our running time analysis. Furthermore, we use those parameters to calculate the amount of bits that need to be communicated in the run of each protocol.

We start with some basic definitions and mathematical background information in Section 1. This also contains a brief introduction to algebraic number fields and their special case, namely cyclotomic number fields. This is intended to motivate lattices and lattice-based cryptography from a mathematical perspective. Section 2 is devoted to the study of lattices and lattice-based problems. Most relevant for this thesis are the short integer solution problem and the learning with errors problem since they exist in a ring-variant. This adds additional structure and thus makes mathematical operations faster and reduces the needed memory space. Most of the key exchange protocols that we analyze in this thesis are based on the decisional ring learning with errors problem. This problem is based on ideal lattices and is defined in Section 3. In Section 4 we define cryptographic primitives and security notations. We explain and analyze different authenticated and unauthenticated key exchange protocols in Section 5 and 6 respectively. The unauthenticated key exchange protocols are the ring and non-ring version of the JD protocol by Ding et al. [JD12] and the BCNS protocol by Bos et al. [BCNS14]. The authenticated key exchange protocols are two protocols by Fujioka et al. [FSXY12], [FSXY13], the one- and two-pass version of the ZZD protocol by Zhang et al. [ZZD+14], and Peikert's protocol [Pei14]. For each protocol the individual steps are explained, the underlying security assumption is stated, and some restrictions for parameter choices are given to assure the correctness and security of the key exchange protocol. The used security model is stated but not explained in further detail since this is out of scope of this thesis. In Section 7 we use Albrecht et al.'s LWE-Estimator [APS15] to find safe parameters for bit security levels of approximately 100 and 192 bit for each previously discussed protocol. A first theoretical analysis showing the connection of the different protocols is given in Section 8, while Section 9 contains the running time analysis. We run a C++ implementation and analyze the running times of small building blocks of the protocols using the parameters from Section 7. This is the basis for our conclusion about the efficiency and practicality of post-quantum lattice-based key exchange protocols.

1 Preliminaries

In this section we define some mathematical objects from algebraic number theory that are used later. We follow the definitions stated in [Neu92], [JS14], [LPR12], and [ZZD+14].

1.1 Mathematical Basics

Let x be a vector in $\mathbb{C}^n$ or $\mathbb{R}^n$. When speaking of the length of vector x we mean the ℓ_2-*norm*

$$\|x\|_2 = \left(\sum_{j=1}^{n} |x_j|^2 \right)^{1/2},$$

also denoted by $\|\cdot\|$. Some results hold for any norm, but in most cases only the ℓ_2-norm is of interest to us. The *maximum norm* of vector x in $\mathbb{C}^n$ or $\mathbb{R}^n$ is defined by $\|x\|_\infty = max_{i\in\{1,\dots,n\}}|x_i|$.

To describe the asymptotic behaviour of functions, we use the Landau notation. Let f and g be real functions. With $f(x) = O(g(x))$ we denote that f has an asymptotic upper bound g, i.e. $\lim_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$. With $f(x) = \omega(g(x))$ we denote that f dominates g asymptotically, i.e. $\lim_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| = \infty$. The expression $\tilde{O}(f(x))$ denotes that there exists k such that $f(x) = O(g(x)log^k(g(x)))$.

In this thesis a ring is defined as a commutative ring with 1 denoted by $\mathcal{R}$. We denote the set of multiplicative inverses by $\mathcal{R}^\times \subset \mathcal{R}$. The elements are called *units*.

An *integral domain* is a ring that has no *zero divisors*, i.e. for $x \in \mathcal{R} \setminus \{0\}$ there exists no element $y \in \mathcal{R} \setminus \{0\}$ such that $xy = 0$. In this thesis a ring is assumed to be an integral domain if not announced differently.

An element $y \notin \mathcal{R}^\times \cup \{0\}$ is called *irreducible* if $y = ab$ for $a, b \in \mathcal{R}$ implies that either a or b is a unit. An element $y \notin \mathcal{R}^\times \cup \{0\}$ is called *prime* if whenever y divides a product of two ring elements $a, b \in \mathcal{R}$, it follows that y divides a or y divides b.

An *ideal* I of $\mathcal{R}$ is an additive subgroup in $\mathcal{R}$ that is closed under multiplication by ring elements. In other words, I is a subset of $\mathcal{R}$ such that for all $a, b \in I$ and $r \in \mathcal{R}$ it holds that $-a \in I$, $a + b \in I$, and $ar \in I$. We write $I = \langle x_1, \dots, x_n \rangle$ to denote that the ideal I is generated by the elements $x_1, \dots, x_n \in \mathcal{R}$.

1.2 Gaussian Distribution and Rejection Sampling

We give a definition of the discrete Gaussian distribution according to [LPR12] and [ZZD+14].

The probability function of the one-dimensional continuous Gaussian distribution with standard deviation $\sigma \in \mathbb{R}$ and center at $v \in \mathbb{R}$ is given by

$$\rho_{\sigma,v}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-v)^2}{2\sigma^2} \right).$$

While in general an n-dimensional Gaussian distribution requires an $n \times n$ covariance matrix, we restrict ourselves to a continuous spherical Gaussian distribution with center at $v \in \mathbb{R}^n$. The reason is that the distribution is a product distribution where the coordinates are independent and identically distributed. As we see in Section 3, this is possible because of the structure of the underlying ring. A spherical Gaussian distribution has a diagonal covariance matrix with $\sigma \in \mathbb{R}_{>0}$ on its diagonal. The probability function is defined as

$$\rho_{\sigma,v}(x) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(\frac{-\| x - v \|^2}{2\sigma^2}\right).$$

This can also be viewed as sampling each coordinate of a vector x according to the one-dimensional Gaussian distribution. The probability function of the discrete Gaussian distribution over $\mathbb{Z}^n$ is given by

$$D_{\mathbb{Z}^n,\sigma,v}(x) = \frac{\rho_{\sigma,v}(x)}{\rho_{\sigma,v}(\mathbb{Z}^n)}.$$

If $v = 0$, we write $D_{\mathbb{Z}^n,\sigma}$ instead of $D_{\mathbb{Z}^n,\sigma,0}$.

It should be noted that the probability function of the Gaussian distribution is by some also defined as $\rho_{r,v}(x) = \exp\left(\frac{-\pi\|x-v\|^2}{r^2}\right)$. This corresponds to $\sigma = \frac{r}{\sqrt{2\pi}}$ in $D_{\mathbb{Z}^n,\sigma,v}(x)$ as the factor $\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n$ cancels in the fraction.

With $r \xleftarrow{\$} \mathcal{M}$ we denote that an element r is drawn uniformly at random from a finite set $\mathcal{M}$ and with $r \leftarrow \chi$ we denote that an element r is drawn according to the probability distribution χ.

The following lemma, used by Bos et al. [BCNS14] and Zhang et al. [ZZD+14], states that the product of two Gaussian distributed polynomials is approximately Gaussian distributed .

Lemma 1.1. *Let $D_{\mathbb{Z},\sigma}$ and $D_{\mathbb{Z},\theta}$ be two Gaussian distributions and $x, y \in \mathbb{Z}[X]$ of degree $n-1$. Furthermore, let the coefficients of x and y be distributed according to $D_{\mathbb{Z},\sigma}$ and $D_{\mathbb{Z},\theta}$ respectively. The coefficients of the product $xy \mod X^n + 1$ are approximately distributed like $z \leftarrow D_{\mathbb{Z},\sigma\theta\sqrt{n}}$.*

Let V be a countable set and let X and Y be two given discrete random variables on V. The *statistical distance* is defined as $\triangle(X,Y) = \frac{1}{2}\sum_{x\in V} |Pr[X = x] - Pr[Y = x]|$.

The statistical distance is used in the rejection sampling theorem, which is used by two of the analyzed protocols.

Theorem 1.2. *[Lyu12][Rejection Sampling] Let $V \subset \mathbb{Z}^n$ such that all elements have norm less than T, $\beta = \omega(T\sqrt{\log n}) \in \mathbb{R}$ and $\psi : V \to \mathbb{R}$ be a probability distribution. Then there exists a constant $M = O(1)$, called rejection constant, such that the distribution of the following algorithm $\mathcal{A}$:*
1. Sample $z_1 \leftarrow \psi$.
2. Sample $z \leftarrow D_{\mathbb{Z}^n,\beta,z_1}$.
3. Output (z, z_1) with probability $\min(\frac{D_{\mathbb{Z}^n,\beta}(z)}{M D_{\mathbb{Z}^n,\beta,z_1}(z)}, 1)$.
is within statistical distance $\frac{2^{-\omega(\log n)}}{M}$ of the distribution of the following algorithm $\mathcal{F}$:

1. Sample $z_1 \leftarrow \psi$.
2. Sample $z \leftarrow D_{\mathbb{Z}^n,\beta}$.
3. Output (z, z_1) *with probability* $\frac{1}{M}$.
More concretely, if $\beta = \tau T$ *for any positive* τ, *then* $M = exp(\frac{12}{\tau} + \frac{1}{2\tau^2})$ *and the statistical difference between the output of the two algorithms is* $\frac{2^{-100}}{M}$ *and the probability that* $\mathscr{A}$ *outputs something is at least* $\frac{1-2^{-100}}{M}$.

A function f is said to be *negligible in the security parameter* κ (also denoted $negl(\kappa)$) if for every $c > 0$ there exists an $N > 0$ such that $f(\kappa) < 1/\kappa^c$ for all $\kappa > N$.

An event A holds true with *overwhelming probability* if there exists a negligible function $negl(\kappa)$, such that $Pr(A) \geq 1 - negl(\kappa)$.

1.3 Algebraic Number Theory

To give a mathematical classification of lattices and lattice-based cryptography, we give a brief introduction to the topic of ring and field extensions, algebraic number fields, and the ring of integers, following the definitions of [Neu92] and [JS14].

A *ring extension* of the ring A is a set B such that $A \subset B$ and B is a ring. A *field extension* L of K, often denoted L/K, is defined similarly.

By $A[X]$ we denote the smallest ring extension of A to a ring that contains A and X. Hence, $A[X] = \{a_n X^n + \cdots + a_1 X + a_0 : n \in \mathbb{N}, a_i \in A, i = 0, \ldots n\}$. Let K be a field. Define $K(X)$ to be the smallest field that contains K and X.

Let L/K be a field extension of K and let $\alpha \in L$. Then the *minimal polynomial* $m_{\alpha,K}$ of α over K is the polynomial in $K[X]$ with smallest degree that has leading coefficient 1 and root α.

Definition 1.3. *An algebraic number field* K *is a finite field extension of the field of rational numbers* $\mathbb{Q}$.

That means the degree of the field extension $[K : \mathbb{Q}]$ is finite. In other words, K can be seen as a $\mathbb{Q}$-vector space of dimension $[K : \mathbb{Q}]$.

Let K be an algebraic number field. An element $x \in K$ is called *algebraic number*. This means that x is a root of a non-zero polynomial f with coefficients in $\mathbb{Q}$ or $\mathbb{Z}$. Hence, for every algebraic number the minimal polynomial over $\mathbb{Q}$ exists. If all coefficients of the polynomial are integers and the leading coefficient equals 1, the element x is called *integral element* of the algebraic number field K.

Definition 1.4. *Let* K *be an algebraic number field. The ring of integers* O_K *is the ring of all integral elements of* K.

To see that ideals in the ring of integers O_K of an algebraic number field K have a $\mathbb{Z}$-basis, we define finitely generated modules. Let M be an $\mathscr{R}$-module of the ring $\mathscr{R}$. M is called *finitely generated* if there exist $x_1, \ldots, x_n \in M$ such that every element of M can be written as a finite linear combination of $x_1, \ldots, x_n$ with coefficients in $\mathscr{R}$.

Theorem 1.5. *Let O_K be the ring of integers of an algebraic number field K with $[K:\mathbb{Q}]=n$. Then any finitely generated O_K-submodule $I \neq 0$ has a $\mathbb{Z}$-basis $x_1,\dots,x_n \in I$ with n elements.*

It follows that every ideal of O_K (as they are always finitely generated O_K-submodules) and O_K itself have a $\mathbb{Z}$-basis. Furthermore, it can be shown that this basis is automatically a $\mathbb{Q}$-basis of K. That means that any $\mathbb{Z}$-basis of O_K has $n=[K:\mathbb{Q}]$ elements. For proofs refer to [Neu92].

Let K be an algebraic number field with $[K:\mathbb{Q}]=n$. There exist exactly n different $\mathbb{Q}$-homomorphisms from K to $\mathbb{C}$, denoted by $\{\pi_1,\dots,\pi_n\}=Hom_{\mathbb{Q}}(K,\mathbb{C})$. They are also called $\mathbb{Q}$-embeddings. A $\mathbb{Q}$-embedding π is called real, if $\pi(K)\subset\mathbb{R}$, else it is called complex. We denote the real embeddings by $\{\varrho_1,\dots,\varrho_r\}$ and the complex embeddings as complex conjugate pairs by $\{\tau_1,\overline{\tau_1},\dots,\tau_s,\overline{\tau_s}\}$, with $n=r+2s$. The complex conjugate embedding $\overline{\tau}$ is defined via the complex conjugation $\overline{\tau}(a)=\overline{\tau(a)}$ [Neu92].

1.4 Cyclotomic Number Fields

One example of algebraic number fields are cyclotomic number fields. They are the foundation of ideal-lattice-based cryptography. First, we define and study cyclotomic polynomials.

Definition 1.6. *The m-th cyclotomic polynomial $\Phi_m(X)\in\mathbb{Z}[X]$ is the polynomial*

$$\Phi_m(X)=\prod_{k\in\mathbb{Z}_m^\times}(X-\zeta_m^k),$$

where $\zeta_m=e^{2\pi i/m}$.

An m-th root of unity is called *primitive* if it is not a k-th root of unity for some $k<m$. All m-th primitive roots are given by ζ_m^k for $k\in\mathbb{Z}_m^\times$, where $\mathbb{Z}_m$ denotes $\mathbb{Z}/m\mathbb{Z}$. Hence, the m-th cyclotomic polynomial is the polynomial whose roots are the primitive m-th roots of unity in $\mathbb{C}$.

$\Phi_m(X)$ is an irreducible polynomial with leading coefficient 1. Hence, it is the minimal polynomial of ζ_m. Via the map $\psi:\mathbb{Z}[X]\to\mathbb{Z}[\zeta_m]$, $X\mapsto\zeta_m$ we can conclude the following isomorphism

$$\mathbb{Z}[X]/\langle\Phi_m(X)\rangle\tilde{=}\mathbb{Z}[\zeta_m].$$

Euler's phi-function is denoted by $\varphi(x)$ and counts the elements in $\mathbb{Z}_x$ that are relatively prime to x. The degree of $\Phi_m(X)$ is given by $\varphi(m)=n$, since elements in $\mathbb{Z}_m$ have a multiplicative inverse if and only if they are relatively prime to m.

Let $p\in\mathbb{Z}$ be a prime number. The cyclotomic polynomial $\Phi_p(X)$ can be written as $\Phi_p(X)=1+X+\cdots+X^{p-1}$, see [JS14].

Lemma 1.7. *Let $m=p^k$. Then $\Phi_m(X)=\Phi_p(X^{m/p})$. In particular, $\Phi_{2^k}(X)=X^n+1$, where $n=\varphi(2^k)=2^{k-1}$.*

Proof. It suffices to show that $\Phi_m(X)$ and $\Phi_p(X^{m/p})$ have the same degree and the same roots. With $\Phi_p(X^{m/p})=\Phi_p(X^{p^{k-1}})$ the degree of $\Phi_p(X^{m/p})$ equals $\varphi(p)p^{k-1}=p^k(1-1/p)=\varphi(m)$,

where $\varphi(m)$ is the degree of $\Phi_m(X)$.
An element $\zeta \in \mathbb{C}$ is a root of $\Phi_p(X)$ if $\zeta^p = 1$ holds. Hence, $\zeta \in \mathbb{C}$ is a root of $\Phi_p(X^{m/p})$ if $(\zeta^{m/p})^p = 1$ holds. It follows that $\zeta \in \mathbb{C}$ with $(\zeta^{m/p})^p = 1$ is an m-th root of unity and thus a root of the polynomial $\Phi_m(X)$. □

Now we can define the algebraic number fields of cyclotomic polynomials.

Definition 1.8. *The m-th cyclotomic number field is defined as* $K_m = \mathbb{Q}(\zeta_m)$.

The degree of the field extension is $[K_m : \mathbb{Q}] = \varphi(m) = n$, since the degree of the minimal polynomial of ζ_m is n. Hence, a $\mathbb{Q}$-basis B of K_m is given by $B = \{1, \zeta_m, \ldots, \zeta_m^{n-1}\}$. This is also a $\mathbb{Z}$-basis of O_{K_m} and $O_{K_m} = \mathbb{Z}[\zeta_m]$.

2 Lattices and Lattice Problems

2.1 Lattices

As we compare key exchange protocols that are based on lattice problems, we define lattices first and look at some of their properties combining definitions from [LPR12], [MR08], and [BAT]. Those definitions are used to define and analyze several lattice and lattice-related problems in Section 2.2 and Section 2.3.

Definition 2.1. *A lattice is a discrete additive subgroup of* $\mathbb{R}^n$.

An equivalent definition of a lattice is given in the following lemma.

Lemma 2.2. $\Gamma \subset \mathbb{R}^n$ *is a lattice if and only if there exist linearly independent vectors* $b_1, \dots, b_m \in \mathbb{R}^n$ *such that* $\Gamma = \{z_1 b_1 + \cdots + z_m b_m : z_i \in \mathbb{Z}\}$.

Proof. "⇒": Let Γ be a discrete additive subgroup of $\mathbb{R}^n$. Discrete additive subgroups of $\mathbb{R}^n$ are isomorphic to $\mathbb{Z}^m$ for some $m \in \mathbb{N}$ [SKR05]. Hence, Γ has a $\mathbb{Z}$-basis, i.e. $\Gamma = \{z_1 b_1 + \cdots + z_m b_m : z_i \in \mathbb{Z}\}$, where $b_1, \dots, b_m$ are linearly independent vectors in $\mathbb{R}^n$.

"⇒": Let $\Gamma = \{z_1 b_1 + \cdots + z_m b_m : z_i \in \mathbb{Z}\}$, where $b_1, \dots, b_m$ are linearly independent vectors in $\mathbb{R}^n$. Associativity, commutativity, and discreteness follow from the structure of $\mathbb{Z}$ and $\mathbb{R}$. The neutral element $0 \in \mathbb{R}^n$ is an element of Γ since $0 = \sum_{i=1}^{m} 0 \cdot b_i$. It remains to show that every element $v = z_1 b_1 + \cdots + z_m b_m \in \Gamma$ with $z_i \in \mathbb{Z}$ has an additive inverse, which follows from $z_i \in \mathbb{Z}$. □

The vectors $b_1, \dots, b_m$ in Lemma 2.2 are called basis of the lattice Γ. The basis matrix B is the matrix with columns $b_1, \dots, b_m$. We write $\Gamma(B)$ for a lattice generated by basis B. By definition we see that $\Gamma(B) = \{Bz : z \in \mathbb{Z}^m\}$. If $n = m$ we say that the lattice has full rank. From now on we only consider full-rank lattices.

Definition 2.3. *Let* $\Gamma \subset \mathbb{R}^n$ *be a lattice. The length of the shortest non-zero vector in* Γ *is denoted by*

$$\lambda_1(\Gamma) = \min_{0 \neq x \in \Gamma} \|x\|.$$

More generally, the smallest radius of a ball that contains k linearly independent vectors of the lattice Γ *is denoted by* $\lambda_k(\Gamma)$.

Some computational lattice problems use special latices such as the dual lattice or q-ary lattices.

Definition 2.4. *Let* $\Gamma \subset \mathbb{R}^n$ *be a lattice. The dual lattice* Γ^* *of* Γ *is defined as*

$$\Gamma^* = \{x \in \mathbb{R}^n : \langle x, z \rangle \in \mathbb{Z} \quad \forall z \in \Gamma\},$$

where $\langle \cdot, \cdot \rangle$ *is the inner product on* $\mathbb{R}^n$ *defined as* $\langle x, z \rangle = \sum_{i=1}^{n} x_i z_i$.

Definition 2.5. *A q-ary lattice is a lattice* $\Gamma \in \mathbb{R}^n$ *that satisfies* $q\mathbb{Z}^n \subset \Gamma \subset \mathbb{Z}^n$ *for some* $q \in \mathbb{Z}$.

Let A be an $n \times m$ matrix with entries in $\mathbb{Z}_q$. Two important examples for q-ary lattices that are used in lattice-based cryptography are

$$\Lambda_q(A) = \{y \in \mathbb{Z}^m : y = A^\top s \bmod q \text{ for some } s \in \mathbb{Z}^n\} \text{ and}$$
$$\Lambda_q^\perp(A) = \{y \in \mathbb{Z}^m : Ay = 0 \bmod q\}.$$

A basis of a lattice Γ in $\mathbb{R}^n$ can have at most n elements and is not unique. Different bases for the same lattice always have the same amount n of basis vectors and those are linearly independent. Not every set of n linearly independent vectors in Γ is necessarily a basis for lattice Γ, but can also span a subset of Γ.

Example 2.6. *Let $B = \{e_1, e_2, \ldots, e_n\}$ and $B' = \{2e_1, 2e_2, \ldots, 2e_n\}$ be two bases, where e_i denotes the i-th standard basis vector. Then the lattice with basis B' is contained in the lattice with basis B, but it is not equal to it.*

2.2 Computational Lattice Problems

In the following we define several computational lattice problems together with facts and assumptions about our today's ability to solve them. Definitions and results follow [MR08], [Reg09], and [LM09].

We call a problem $\mathscr{P}$ *hard* if there exists no probabilistic polynomial-time algorithm that solves $\mathscr{P}$ with non-negligible probability. A problem $\mathscr{P}$ is called *NP-hard* if for every problem $\mathscr{L}$ in NP there exists a polynomial-time reduction from $\mathscr{L}$ to $\mathscr{P}$. A problem $\mathscr{L}$ is in *NP* if a given solution of $\mathscr{L}$ can be verified as a solution in polynomial time.

One important lattice problem is the Shortest Vector Problem (SVP) and its approximate version.

Definition 2.7 (Shortest Vector Problem (SVP))**.** *Given a lattice $\Gamma \subset \mathbb{R}^n$, find a non-zero vector v in the lattice Γ with length equal to $\lambda_1(\Gamma)$.*

There is no known quantum algorithm that can solve this problem in polynomial time [MR08]. An often used variant of the SVP is its approximate version defined below.

Definition 2.8 (γ-approximate SVP (γ-SVP))**.** *Given a lattice $\Gamma \subset \mathbb{R}^n$ and an approximation factor $\gamma \in \mathbb{R}_{\geq 1}$, find a non-zero vector v in the lattice Γ such that*

$$\|v\| \leq \gamma \lambda_1(\Gamma).$$

For $\gamma = 1$ one obtains γ-$SVP = SVP$. The γ-SVP is also called *search* γ-SVP. Its decisional version is denoted by $GapSVP_\gamma$.

Definition 2.9 (Decisional γ-approximate SVP ($GapSVP_\gamma$))**.** *Given a lattice $\Gamma \subset \mathbb{R}^n$, an approximation factor $\gamma \in \mathbb{R}_{\geq 1}$, and $d \in \mathbb{R}$, decide whether $\lambda_1(\Gamma) \leq d$ or whether $\lambda_1(\Gamma) > \gamma d$.*

For a full rank lattice $\Gamma \subset \mathbb{R}^n$ there exists no known quantum algorithm that can solve γ-SVP in polynomial time for $\gamma = poly(n)$. Furthermore, it is shown that it is NP-hard for approximation factors less than $\sqrt{2}$ [Mic98].

The $GapSVP_\gamma$ is related to another lattice problem, namely the α-bounded distance decoding problem (BDD_α). Lyubashevsky and Micciancio [LM09] showed that for all $\gamma > 2\sqrt{n/\log n}$ there exists a polynomial-time Turing reduction from $GapSVP_\gamma$ to $BDD_{\frac{1}{\gamma}\sqrt{n/\log n}}$ in the ℓ_2-norm, with n being the number of basis vectors of the respective lattices. That means that there exists a polynomial-time algorithm that could solve $GapSVP_\gamma$ if it has access to an oracle which can solve $BDD_{\frac{1}{\gamma}\sqrt{n/\log n}}$. Hence, $GapSVP_\gamma$ is not harder than $BDD_{\frac{1}{\gamma}\sqrt{n/\log n}}$. In fact, the $BDD_{1/\gamma}$ problem and the $GapSVP_\gamma$ are equivalent up to polynomial approximation factors [LM09]. Now we define the BDD_α problem.

Definition 2.10 (α-Bounded Distance Decoding Problem (BDD_α))**.** *Given a lattice $\Gamma \in \mathbb{R}^n$, an approximation factor $\alpha \in \mathbb{R}_{\geq 1}$, and a vector $x \in \mathbb{R}^n$ with distance less than $\alpha\lambda_1(\Gamma)$ from the lattice Γ, find the lattice vector $v \in \Gamma$ such that*

$$\|v - x\| = min_{u \in \Gamma}\|u - x\|.$$

There exists also another version of BDD_α, where one searches for the lattice vector $v \in \Gamma$ such that $\|v - x\| = \alpha\lambda_1(\Gamma)$. Both problems are equivalent under a polynomial-time reduction [LM09].

The BDD_α problem is known to be NP-hard for any constant factor $\alpha > 1/\sqrt{2}$ [LLM06]. Typically α is chosen to be $\frac{1}{2}$.

2.3 Computational Lattice-Based Problems

In this section we define two problems that are most relevant for the rest of this thesis, namely the short integer solution problem (SIS) and the learning with errors problem (LWE). Both problems can be seen as lattice problems, namely as the previously defined lattice problems γ-SVP and BDD_α.

Definition 2.11 (Short integer solution problem (SIS))**.** *[Ajt96] Given $n, m > 0$, $q \geq 2$, $\beta \in \mathbb{R}$, and $A \in \mathbb{Z}_q^{n \times m}$, find a non-zero vector $z \in \mathbb{Z}^m$ with $\|z\| \leq \beta$ such that $Az \equiv 0 \mod q$.*

The learning with errors problem (LWE) exists in two version, the decisional variant and the search variant. It was first defined by Regev [Reg09].

Definition 2.12 (The LWE Distribution)**.** *Let n and $q \geq 2$ be non-negative integers, $s \in \mathbb{Z}_q^n$, and χ be a distribution over $\mathbb{Z}$. The LWE distribution is denoted by $A_{s,\chi}$ and outputs pairs $(a, b = \langle a, s\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $a \xleftarrow{\$} \mathbb{Z}_q^n$ is uniformly distributed and $e \leftarrow \chi$ is drawn from the distribution χ.*

The distribution χ is also called error distribution as it is used to sample error terms from it. This distribution is often chosen to be the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ as defined in Section 1.2.

Definition 2.13 (Search Learning with errors problem (LWE))**.** *Let n and $q \geq 2$ be non-negative integers, $s \xleftarrow{\$} \mathbb{Z}_q^n$, and χ be a distribution over $\mathbb{Z}$. Given polynomially many samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ from the LWE distribution $A_{s,\chi}$, find $s \in \mathbb{Z}_q$.*

Regev [Reg09] showed that for $2 \leq q \leq poly(n)$ prime the search LWE problem is equivalent to the decisional LWE problem defined as follows.

Definition 2.14 (Decisional Learning with errors problem ($DLWE$))**.** *Let n and $q \geq 2$ be non-negative integers, $s \xleftarrow{\$} \mathbb{Z}_q^n$, and χ be a distribution over $\mathbb{Z}$. Given polynomially many samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, decide whether they are drawn from the LWE- distribution $A_{s,\chi}$ or whether they are drawn uniformly random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

It can be shown that choosing s according to the error distribution does not weaken the hardness of the problem [ACPS09]. This form is called normal form of the LWE. We denote its decisional variant for variables n, q, and χ by $DLWE_{n,q,\chi}$.

We can also write the pairs $(a_i, b_i = \langle a_i, s \rangle + e_i)$ for $i \in \{1, \ldots, m\}$ with $m = poly(n)$ in a matrix and vector notation [MR08]: $(A, b = As + e)$, where $A \in \mathbb{Z}_q^{m \times n}$, $s \in \mathbb{Z}_q^n$ and $e \in \mathbb{Z}^m$.

The SIS and the LWE problem can be seen as lattice problems as well. The SIS problem is in fact a γ-SVP where one searches for a short vector z in the q-ary lattice defined by $\Lambda_q^\top(A) = \{z \in \mathbb{Z}^m : Az = 0 \mod q\}$. The search LWE problem can be seen as a BDD_α problem with lattice $\Lambda^\top(A) = \{y \in \mathbb{Z}^m : y = A^\top s \mod q \text{ for some } s \in \mathbb{Z}^n\}$.

We call a problem $\mathscr{P}$ *worst-case hard* if at least one instance of the problem is hard. In contrast, problem $\mathscr{P}$ is called *average-case hard* if the average instance of the problem is hard. That means that worst-case hardness is weaker than average-case hardness. A worst-to-average-case reduction is a reduction from a worst-case problem to an average-case problem.

The approximate SVP has a worst-to-average-case reduction to the LWE problem, i.e. if there exists an algorithm to solve the LWE problem with $q, m \leq \text{poly(n)}, q$ prime, and $\sigma > \sqrt{n/2\pi}$, then there exists a polynomial-time algorithm that solves the approximate SVP in any n-dimensional lattice in the worst-case. This result was first proven by Regev [Reg09] for a polynomial-time quantum algorithm. Peikert [Pei09] proved that the same holds for a polynomial-time non-quantum classical algorithm.

3 Ideal Lattices and Ideal Lattice Problems

While the LWE and SIS problem offer high security, cryptographic primitives based on them are often very inefficient. That means that key sizes and computation times are at least quadratic in the main security parameter [LPR12]. To improve on that, versions of the LWE and SIS problem are used that work on so called ideal lattices. Ideal lattices are lattices with some additional properties. Through this additional mathematical structure, the needed memory space is reduced and computational operations are faster. The resulting *R-LWE* and *R-SIS* problems are the base for the cryptographic primitives analyzed in this thesis.

3.1 Ideal Lattices

By Definition 2.1, any lattice is a group. Ideal lattices have additional structure, namely the structure of an ideal.

We give a general definition of ideal lattices. Afterwards, we look at more specific examples, following [BAT] and [LPR12].

Definition 3.1. *Let K be an algebraic number field with $[K:\mathbb{Q}] = n$ and $I \subset O_K$ be an ideal in its ring of integers O_K. Let $\Theta : K \to \mathbb{R}^n$ be an additive field homomorphism. Then $\Theta(I)$ is a lattice in $\mathbb{R}^n$. It is called ideal lattice.*

We write I instead of $\Theta(I)$ when it is clear from the context which homomorphism Θ is used.

Two homomorphisms, the canonical and the coefficient embedding, are often used in lattice-based cryptography. We give a definition of both with a focus on the coefficient embedding on cyclotomic fields since all key exchange protocols that are analyzed in this thesis are based on the coefficient embedding for cyclotomic fields.

3.1.1 The Canonical Embedding

Let $\mathcal{H}_{r,s} = \{x \in \mathbb{R}^r \times \mathbb{C}^{2s} : x_{r+j} = \bar{x}_{r+s+j} \text{ for all } j = 1,\dots,s\} \subset \mathbb{C}^n$, where $n = r + 2s$ and $\bar{x}_j$ denotes the complex conjugate of x_j. That means for $x_j = a + ib \in \mathbb{C}$ with $a, b \in \mathbb{R}$ the complex conjugate is given by $\bar{x}_j = a - ib$.

Using the definition of the real and complex embeddings $\{\varrho_1,\dots,\varrho_r\}$ and $\{\tau_1,\overline{\tau_1},\dots,\tau_s,\overline{\tau_s}\}$ (see Section 1.3), we give a definition of the canonical embedding according to [Neu92].

Definition 3.2. *Let K be an algebraic number field with $[K:\mathbb{Q}] = n$ and O_K its ring of integers. Then the canonical embedding is defined by*

$$j : K \to \mathcal{H}_{r,s} \subset \mathbb{R}^r \times \mathbb{C}^{2s}; a \mapsto (\varrho_1(a),\dots,\varrho_r(a),\tau_1(a),\dots,\tau_s(a),\overline{\tau_1(a)},\dots,\overline{\tau_s(a)}).$$

Theorem 3.3. *[Neu92] Let $0 \neq I \subset O_K$ be an ideal. Then $j(I)$ is a full rank lattice in $\mathcal{H}_{r,s} \cong \mathbb{R}^n$.*

With this theorem we obtain that $j(I)$ is an ideal lattice.

The isomorphism between $\mathcal{H}_{r,s}$, with an inner product inherited from $\mathbb{C}^n$, and R^n as inner product spaces holds via the map $f : \mathcal{H}_{r,s} \to \mathbb{R}^n : x \mapsto Dx$ with block matrix

$$D = \begin{pmatrix} Id_r & 0 \\ 0 & D_{2s} \end{pmatrix},$$

where Id_r is the r-dimensional identity matrix and

$$D_{2s} = \frac{1}{\sqrt{2}} \begin{pmatrix} Id_s & Id_s \\ iId_s & -iId_s \end{pmatrix}.$$

3.1.2 The Coefficient Embedding

Let f be a polynomial of degree n in $\mathbb{Z}[X]$ and define the ring $R = \mathbb{Z}[X]/\langle f \rangle$. The elements of R can be represented by polynomials of degree less than n with coefficients in $\mathbb{Z}$. Hence, $(1, X, \ldots, X^{n-1})$ is a $\mathbb{Z}$-basis of R and any element $a \in R$ can be represented uniquely by $a = \sum_{j=0}^{n-1} a_j X^j$ with $a_j \in \mathbb{Z}$.

Definition 3.4. *Let $R = \mathbb{Z}[X]/\langle f \rangle$ and $f \in \mathbb{Z}[X]$ be a polynomial of degree n. The coefficient embedding is defined via the map*

$$c : R \to \mathbb{R}^n; a = \sum_{j=0}^{n-1} a_j X^j \mapsto (a_0, \ldots, a_{n-1}).$$

The map c is an additive ring homomorphism.

As an example for ideal lattices in combination with the coefficient embedding, we have a closer look at cyclotomic fields since the coefficient embedding can also be defined for cyclotomic fields, see [BAT] and [LPR12].

In Section 1.4 we see that the m-th cyclotomic field $K_m = \mathbb{Q}(\zeta_m)$ has $\mathbb{Q}$-basis $(1, \zeta_m \ldots, \zeta_m^{n-1})$, which is also a $\mathbb{Z}$-basis of O_{K_m}. Hence, $O_{K_m} \cong \mathbb{Z}[\zeta_m]$. We know that the m-th cyclotomic polynomial Φ_m is the minimal polynomial of $\zeta_m = e^{2\pi i/m}$. Hence, the following holds

$$\mathbb{Z}[X]/\langle \Phi_m \rangle \cong \mathbb{Z}[\zeta_m] \cong O_{K_m}.$$

Using these isomorphisms we see that for an ideal $I \subset O_{K_m}$ we obtain an ideal lattice $c(I) \subset \mathbb{Z}^n$ under the coefficient embedding since c can be extended to an additive field homomorphism on $K_m = \mathbb{Q}(\zeta_m)$ in this case.

In the ring of integers of the m-th cyclotomic field for $m = 2^k$, ideal lattices have some interesting structure. As stated in Section 1.4, the cyclotomic polynomial is $\Phi_m(X) = X^n + 1$ in this case. This means that $X^n = -1 \mod \Phi_m(X)$. Thus, a multiplication of a representative $a = \sum_{j=0}^{n-1} a_j X^j$ of an element in $R = \mathbb{Z}[X]/\langle X^n + 1 \rangle$ by X can be seen as an anti-cyclic shift. By that we mean

$$a \mapsto \sum_{j=1}^{n-1} a_{j-1} X^j - a_{n-1}.$$

Example 3.5. *Let $a(X) = 1 + 2X + 3X^3$ be the representative of an element in $R = \mathbb{Z}[X]/\langle f(X)\rangle$ with $f(X) = X^4+1$. Multiplying $a(X)$ by X results in a polynomial with representative $b(X) = -3+X+2X^2$ in R. Polynomials in R can be associated with vectors in $\mathbb{Z}^4$ via the coefficient embedding. That means that multiplying $c(a(X)) = (1,2,0,3)$ with $c(X)$ results in $c(b(X)) = (-3,1,2,0)$. Hence, it is a rotation of the coefficients with a negation of the new first coefficient.*

Rings of the form $R = \mathbb{Z}[X]/\langle f\rangle$, where f is an irreducible polynomial in $\mathbb{Z}[X]$ with leading coefficient 1, have infinitely many equivalence classes. However, in lattice-based cryptography one deals with finite rings of the form $R_q = \mathbb{Z}_q[X]/\langle f\rangle$. This ring has q^n different equivalence classes.

Example 3.6. *The ring $\mathbb{Z}_3/\langle X^2+1\rangle$ has nine equivalence classes, which are represented by $\overline{0}, \overline{1}, \overline{2}, \overline{X}, \overline{2X}, \overline{1+X}, \overline{2+X}, \overline{1+2X}, \overline{2+2X}$.*

In that case we denote $R_q = \mathbb{Z}_q[X]/\langle f\rangle$ and we can identify R_q with $\mathbb{Z}_q^n$ via the coefficient embedding defined above. In that sense we also call representatives of equivalence classes of R elements or vectors. Through this association the ℓ_2-norm of an equivalence class of R is defined as the ℓ_2-norm of the associated vector in $\mathbb{Z}_q^n$.

Let $R_q = \mathbb{Z}_q[X]/\langle X^n+1\rangle$, i.e. R_q is isomorphic to the ring of integers of the $m = 2n = 2^k$-th cyclotomic field modulo q. Via the coefficient embedding, sampling an element from R_q from a discrete Gaussian distribution can be done by sampling each coefficient independently according to a one-dimensional discrete Gaussian distribution.

3.2 LWE and SIS on ideal lattices

A variant of the *LWE* problem on ideal lattices was first given by Lyubashevsky, Peikert, and Regev [LPR12] and is called *R-LWE*. We follow their general definition of the *R-LWE* problem and state it additionally for the special case of $R = \mathbb{Z}[X]/\langle X^n+1\rangle$. We also give a definition of the *SIS* problem on ideal lattices and analyse the needed memory space when using the *R-LWE* and the *R-SIS* problem compared to the *LWE* and *SIS* problem.

Definition 3.7. *Let K be an algebraic number field with $[K:\mathbb{Q}] = n$. Its field trace is defined by*

$$Tr_{K/\mathbb{Q}} : K \to \mathbb{Q}; x \mapsto \sum_{i=1}^{r} \varrho_i(x) + \sum_{j=1}^{s} (\tau_j(x) + \overline{\tau_j(x)}),$$

where $\{\varrho_1, \ldots, \varrho_r\}$ and $\{\tau_1, \overline{\tau_1}, \ldots, \tau_s, \overline{\tau_s}\}$ are the real and complex embeddings defined in Section 1.3.

Definition 3.8. *Let K be an algebraic number field with ring of integers O_K. A fractional ideal $I \subset K$ is a set I such that $dI \subset O_K$ is an ideal for some $d \in O_K$.*

Every fractional ideal I has a $\mathbb{Z}$-basis with $[K:\mathbb{Q}] = n$ elements. Similarly to the proof of Theorem 3.3, it can be shown that $j(I)$ is an ideal lattice of full rank in $\mathcal{H}_{r,s}$ [Neu92].

Definition 3.9. *Let $I \subset K$ be a fractional ideal. Its dual ideal I^V is defined as*

$$I^V = \{x \in K : Tr_{K/\mathbb{Q}}(xI) \subset \mathbb{Z}\}.$$

It holds that $j(I^V) = \overline{j(I)^*}$ and $I^V = I^{-1}O_K^V$, see [Con09].

Let K be an algebraic number field, $R = O_K$ its ring of integers, $q \geq 2$ be an integer, and $K_{\mathbb{R}}$ be the field tensor product $K_{\mathbb{R}} = K \otimes \mathbb{R} \cong \mathscr{H}_{r,s}$ [LPR12].

Following Lyubashevsky et al. [LPR12], we define the ring learning with errors distribution.

Definition 3.10 (The R-LWE Distribution)**.** *Let $s \in R_q^V$ and ϕ be an error distribution over $K_{\mathbb{R}}$. The R-LWE distribution is denoted by $A_{R,s,\phi}$ and outputs pairs $(a, b = as + e \mod qR^V) \in R_q \times K_{\mathbb{R}}/qR^V$, where $a \xleftarrow{\$} R_q$ is uniformly random and $e \leftarrow \phi$.*

The ring learning with errors problem is defined as follows.

Definition 3.11 (Search Ring-Learning With Errors Problem (*R-LWE*))**.** *Let Ψ be a family of distributions over $K_{\mathbb{R}}$. Given arbitrarily many independent samples (a, b) from $A_{R,s,\phi}$ for arbitrary $\phi \in \Psi$ and $s \in R_q^V$, find s.*

As in the original version of the *LWE* problem, there exists a decisional version of the search *R-LWE* problem, namely the *R-DLWE* problem.

Definition 3.12 (Decisional Ring-Learning With Errors Problem (*R-DLWE*))**.** *Let $s \xleftarrow{\$} R_q^V$ and $\phi \xleftarrow{\$} \Psi$ be uniformly distributed. Given arbitrarily many independent samples $(a, b) \in R_q \times K_{\mathbb{R}}/qR^V$, decide whether they are drawn from the R-LWE distribution $A_{R,s,\psi}$ or whether they are drawn uniformly random from $R_q \times K_{\mathbb{R}}/qR^V$.*

The hardness of the *R-LWE* problem can be reduced to the hardness of the *SVP* over ideal lattices. For more details and a proof refer to [LPR12].

Theorem 3.13. *Let K_m be the m-th cyclotomic field of degree $n = \varphi(m)$ with ring of integers $R = O_K$. Let $\alpha < \sqrt{\log n/n}$ and $q = poly(n) \geq 2$ be prime with $q \equiv 1 \mod m$ such that $\alpha q \geq \omega(\sqrt{n})$. Then there exists a polynomial-time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$ approximate SVP on ideal lattices to R-DLWE given l R-LWE samples, with fixed spherical Gaussian error distribution $D_{\mathbb{Z}^n,\beta}$, where $\beta = \alpha\left(\frac{nl}{\log(nl)}\right)^{1/4}$.*

We look at the *R-LWE* problem for the special case of the m-th cyclotomic ring where $m = 2^k$. That means $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$ holds, where $n = 2^{k-1}$.

In this special case it holds that $nR^V = R$. Hence, pairs $(a, b = (a \cdot s)/q + e \mod R^V)$ can be transformed to pairs $(a, \tilde{b})$, where $\tilde{b} = b \cdot n = (a \cdot \tilde{s})/q + \tilde{e}$ with $\tilde{s} \in R_q$ and $\tilde{e} \in R$ [LPR12].

In cryptography a discretized version of the *R-LWE* is often used by computing b in a finite set instead of $K_{\mathbb{R}}/R^V$.

In the authenticated and non-authenticated key exchange protocols analyzed in Section 5 and 6, we only consider the *R-DLWE* problem for the $m = 2^{l+1}$-th cyclotomic field with ring of integers $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$, where $n = 2^l$ for $l > 0$. Furthermore, the element $s \in \mathbb{R}_q$ is drawn from the error distribution ϕ instead of uniformly random. We state this version of the *R-DLWE* problem

explicitly in the following definition and denote it by $R\text{-}DLWE_{m,n,q,\chi}$.

Definition 3.14 ($R\text{-}DLWE$ problem for $m = 2^{l+1}$ ($R\text{-}DLWE_{m,n,q,\chi}$)). *Let $m = 2n$, $n = 2^l$, $l > 0$, q be an integer, $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$, χ be a distribution over R, and $s \leftarrow \chi$. Given arbitrarily many pairs $(a, b) \in R_q \times R_q$, decide whether they are drawn from the R-LWE distribution $A_{R,s,\chi}$ or whether they are drawn uniformly random from $R_q \times R_q$.*

The hardness of the $R\text{-}DLWE_{m,n,q,\chi}$ problem is equivalent to the hardness of the original $R\text{-}DLWE$ problem of Definition 3.12. For a proof refer to [LPR13]. Brakerski et al. [BV11] proved that taking an additional factor $t \in \mathbb{Z}_q^*$ in front of the error term e, i.e. looking at pairs $(a, b = as + te)$ instead of $(a, b = as + e)$, does not weaken the hardness of the $R\text{-}DLWE_{m,n,q,\chi}$ problem either. This form of the $R\text{-}DLWE_{m,n,q,\chi}$ problem is also used in some protocols.

Now we give a definition of the $R\text{-}SIS$ in the ring $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ following [LS12].

Definition 3.15 ($R\text{-}SIS$). *Given $a_1, \ldots, a_t \in R_q$, find $y_1, \ldots, y_t \in R$ such that $\sum_{i=1}^{t} a_i y_i = 0 \mod q$ and $0 <\| y \|\leq \beta$ for some fixed $\beta \in \mathbb{R}$.*

Since in this specific setting the multiplication of an element $a_i \in R_q$ with X is just an anti-cyclic shift, the $R\text{-}SIS$ can be seen as an SIS with the matrix A consisting of the rotation matrices of the elements a_i, i.e. $A = (rot(a_1), \ldots, rot(a_t))$ with

$$rot(a_i) = \begin{pmatrix} a_{i,0} & -a_{i,n-1} & \cdots & -a_{i,1} \\ a_{i,1} & a_{i,0} & \cdots & -a_{i,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i,n-1} & a_{i,n-2} & \cdots & a_{i,0} \end{pmatrix}, a_i = \sum_{j=0}^{n-1} a_{i,j} X^j.$$

For the $R\text{-}LWE$ problem a similar process is possible, just that we take the transpose of A. That means that the $R\text{-}LWE$ problem can be seen as an LWE problem with matrix

$$A = \begin{pmatrix} rot(a_1) \\ rot(a_2) \\ \vdots \\ rot(a_t) \end{pmatrix}.$$

The advantage of the $R\text{-}LWE$ and $R\text{-}SIS$ problem compared to the LWE and SIS problem is that the needed memory space is much less. This is because one does not need to save the whole matrix A but just one column of each rotation matrix $rot(a_i)$. Saving the matrix A from above requires saving nt elements of $\mathbb{Z}_q$ while the matrix A has $n^2 t$ entries. Hence, the additional structure of ideal lattices reduces the needed memory space by a factor of n.

4 Cryptographic Primitives and Definitions

Key exchange protocols, especially authenticated protocols, often use cryptographic primitives such as key encapsulation mechanism, public key encryption schemes, and signature schemes. Definitions are given below together with often used security properties.

A *key encapsulation mechanism* (KEM) is a scheme in which a key k is produced as the output of the senders encapsulation algorithm using only the receivers public encapsulation key. More formally, a KEM $\mathscr{E}$ is defined via the three probabilistic polynomial-time algorithms $\mathscr{E} = (KeyGen, EnCap, DeCap)$ defined as follows [FSXY13]:

KeyGen(1^κ)	input: security parameter κ output: a pair consisting of a decapsulation and an encapsulation key (dk, ek)
EnCap(ek)	input: encapsulation key ek output: ciphertext $c \in C$ and a key $k \in K$, where C is the ciphertext space and K the key space
DeCap(c, dk)	input: decapsulation key dk and a ciphertext $c \in C$, where C is the ciphertext space output: key $k \in K$, where K is the key space, or a decapsulation failure symbol $\perp$.

A KEM is said to be *correct* if there exists a negligible function $negl(\kappa)$ in the security parameter κ such that for any $(dk, ek) \leftarrow KeyGen(1^\kappa)$ and $(c, k) \leftarrow EnCap(ek)$, it holds that $Pr[DeCap(c, dk) = k] \geq 1 - negl(\kappa)$.

A *public key encryption scheme* (PKE) $\mathscr{E}$ is defined via the following three probabilistic polynomial-time algorithms $\mathscr{E} = (Gen, Enc, Dec)$ [FSXY13]:

Gen(1^κ)	input: security parameter κ output: a pair consisting of a secret and a public key (sk, pk)
Enc(M, pk)	input: public key pk and a plaintext $M \in \mathscr{M}$, where $\mathscr{M}$ is the message space output: ciphertext $c \in C$, where C is the ciphertext space
Dec(c, sk)	input: secret key sk and a ciphertext $c \in C$, where C is the ciphertext space output: plaintext $M \in \mathscr{M}$, where $\mathscr{M}$ is the message space or a decryption failure symbol $\perp$.

A PKE is said to be *correct* if for any $M \in \mathscr{M}$ and $(sk, pk) \leftarrow Gen(1^\kappa)$, there exists a negligible function $negl(\kappa)$ in the security parameter κ such that $Pr[Dec(Enc(M, r_e, pk), sk) = M] \geq 1 - negl(\kappa)$ holds. The PKE is called *perfectly correct* if $Pr[Dec(Enc(M, r_e, pk), sk) = M] = 1$.

When a PKE (or KEM) $\mathscr{E} = (Gen, Enc, Dec)$ is used in a key exchange protocol, $\mathscr{E}$ is often required to fulfill a security property such as IND-CPA or IND-CCA security. Those security properties are defined via the IND-CPA and the IND-CCA game defined below, following [KL07].

The IND-CPA game:
The key generation algorithm of a PKE $\mathscr{E} = (Gen, Enc, Dec)$ (KEM) is run and outputs $(sk, pk) \leftarrow Gen(1^\kappa)$. The adversary $\mathscr{A}$ receives pk and has access to an encryption oracle $Enc(\cdot, pk)$. $\mathscr{A}$ chooses two messages $m_0, m_1 \in \mathscr{M}$. Afterwards, a random bit $b \in \{0, 1\}$ is chosen and the ciphertext $c \leftarrow Enc(m_b, pk)$ is computed by the oracle and given to $\mathscr{A}$. The

adversary can continue to query the encryption oracle. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{A}$ wins the game, else $\mathcal{A}$ loses.

Definition 4.1 (IND-CPA). *A $PKE\ \mathcal{E} = (Gen, Enc, Dec)$ (KEM) is said to be IND-CPA secure if for all probabilistic, polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $negl(\kappa)$ in the security parameter κ such that*

$$Pr[\mathcal{A} \text{ wins the IND-CPA game}] \leq \frac{1}{2} + negl(\kappa).$$

A stronger security property for PKE schemes is given by indistinguishability under chosen-ciphertext attacks (IND-CCA).

The IND-CCA game:
The key generation algorithm of a $PKE\ \mathcal{E} = (Gen, Enc, Dec)$ $(KEM)$is run and outputs $(sk, pk) \leftarrow Gen(1^\kappa)$. The adversary $\mathcal{A}$ receives pk and has access to a decryption oracle $Dec(\cdot, sk)$. $\mathcal{A}$ chooses two messages $m_0, m_1 \in \mathcal{M}$. Afterwards, a random bit $b \in \{0,1\}$ is chosen and the ciphertext $c = Enc(m_b, pk)$ is computed and given to $\mathcal{A}$. The adversary can continue to query the decryption oracle but may not ask for the decryption of c. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{A}$ wins the game, else $\mathcal{A}$ loses.

Definition 4.2 (IND-CCA). *A $PKE\ \mathcal{E} = (Gen, Enc, Dec)$ (KEM) is said to be IND-CCA secure if for all probabilistic, polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $negl(\kappa)$ in the security parameter κ such that*

$$Pr[\mathcal{A} \text{ wins the IND-CCA game}] \leq \frac{1}{2} + negl(\kappa).$$

An IND-CPA secure KEM can be obtained from an IND-CPA secure PKE by taking $K = \mathcal{M}$ and vice versa. Therefore, we also write (sk, pk) for the key pair (dk, ek) and note that a KEM can be instantiated by a PKE scheme. Fujikoa et al. [FSXY13] make use of this in their authenticated key exchange protocol described in Section 6.2. In that case the public key encryption algorithms Gen, Enc, Dec replace the KEM algorithms $KeyGen, EnCap, DeCap$. The algorithm $KeyGen(1^\kappa)$ outputs a decapsulation-encapsulation key pair (dk, ek) which corresponds to the secret-public key pair $(sk, pk) \leftarrow Gen(1^\kappa)$. The ciphertext c and key k as outputs $(c, k) \leftarrow EnCap(ek)$ of the encapsulation algorithm correspond to the randomly chosen input message M and the ciphertext output $c \leftarrow Enc(M, pk)$ of the encryption algorithm. The output $k \leftarrow DeCap(c, dk)$ of the decapsulation algorithm corresponds to the output $M \leftarrow Dec(c, sk)$ of the decryption algorithm. That means that the key of the KEM is derived as a randomly chosen message of the PKE.

A *signature scheme* Π is defined via the following three probabilistic polynomial-time algorithms $\Pi = (Sig.Gen, Sign, Ver)$ [PW07]:

Sig.Gen(1^κ)	input: security parameter κ output: a pair of verification and signing key (vk, sk)
Sign(sk, M)	input: signing key sk and a message $M \in \mathcal{M}$, where $\mathcal{M}$ is the message space output: signature σ
Ver(vk, M, σ)	input: verification key vk, a message $M \in \mathcal{M}$, and a signature σ output: 1 if σ is a valid signature for message M, else 0.

A signature scheme is called *complete* if for any $(vk, sk) \leftarrow Sig.Gen(1^\kappa)$ and any $M \in \mathcal{M}$, there exists a negligible function $negl(\kappa)$ in the security parameter κ such that $Pr[Ver(vk, M, Sign(sk, M)) = 1] \geq 1 - negl(\kappa)$ holds. The signature scheme is called *perfectly complete* if $Pr[Ver(vk, M, Sign(sk, M)) = 1] = 1$.

The security property for signature schemes that we use in this thesis is defined via the signature game [KL07].

The signature game:
The key generation algorithm $Sig.Gen$ of a signature scheme $\Pi = (Sig.Gen, Sign, Ver)$ generates a valid key pair $(vk, sk) \leftarrow Sig.Gen(1^\kappa)$ and sends the verification key to the adversary $\mathcal{A}$. $\mathcal{A}$ has access to a signing oracle $Sign(sk, \cdot)$ which returns valid signatures for any message $M \in \mathcal{M}$ of $\mathcal{A}$'s choice. $\mathcal{A}$ outputs a pair (M', σ'). Let $\mathcal{Q}$ be the set of all messages whose signatures were previously requested by $\mathcal{A}$. The adversary $\mathcal{A}$ wins the game if $Ver(vk, M', \sigma') = 1$ and $M' \neq M$ for any message $M \in \mathcal{Q}$.

Definition 4.3. *A signature scheme is called existentially unforgeable under adaptive chosen message attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $negl(\kappa)$ in the security parameter κ such that*

$$Pr[\mathcal{A} \text{ wins the signature game}] \leq negl(\kappa).$$

All authenticated key exchange protocols that are analyzed in this thesis use hash functions.

Definition 4.4 (Hash Function)**.** *A hash function H is a function that takes a string of arbitrary length as input and outputs a string of fixed length l:*

$$H : \{0,1\}^* \rightarrow \{0,1\}^l.$$

A hash function is called *collision resistant* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ the probability to find x, x' with $x \neq x'$ and $H(x) = H(x')$ is negligible.

Such a hash function is often instantiated by SHA-256. SHA-256 outputs elements of 256 bits and achieves a security of 128 bits against collision attacks. In this thesis we instantiate all hash functions by SHA-256.

Hash functions can also be used to obtain message authentication codes. A *message authentication code* (MAC) Π consists of the following three probabilistic polynomial-time algorithms $\Pi = (Gen, \mathcal{O}, Vrfy)$:

Gen(1^κ)	input: security parameter κ output: a uniformly distributed key $k \in \{0,1\}^\kappa$
$\mathcal{O}(k, M)$	input: key k and a message $M \in \{0,1\}^*$ output: $t \in \{0,1\}^*$ for a message $M \in \{0,1\}^*$, where t is called MAC tag
Vrfy(k, M, t)	input: key k, a message $M \in \{0,1\}^*$, and a MAC tag $t \in \{0,1\}^*$ output: 1 if t is a valid MAC tag for message M, else 0

Furthermore, for all $k, M \in \{0,1\}^*$, $Pr[Vrfy(k, M, \mathcal{O}(k, M)) = 1] > 1 - negl(\kappa)$ holds true.

The MAC game:
The key generation algorithm $Gen(1^\kappa)$ of a MAC $\Pi = (Gen, \mathcal{O}, Vrfy)$ generates a random key $k \in \{0,1\}^\kappa$. The adversary $\mathcal{A}$ is given access to the oracle $\mathcal{O}(k,\cdot)$ which returns $(M,t) \leftarrow \mathcal{O}(k,M)$ on input M. The adversary $\mathcal{A}$ wins the game if it finds (M',t') such that $Vrfy(k,M',t') = 1$ and $M \neq M'$ for all previously queried M.

Definition 4.5. *A MAC is called existentially unforgeable under adaptive chosen message attacks (or simply secure), if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function $negl(\kappa)$ in the security parameter κ such that*

$$Pr[\mathcal{A} \text{ wins the MAC game}] \leq negl(\kappa).$$

A MAC can be obtained through a collision resistant hash function H by taking $\mathcal{O}(k,M) = HMAC(k,M) = H((k \oplus opad \| H(k \oplus ipad) \| M))$, where $opad, ipad$ are constants. This means that a MAC can be seen as a keyed hash function. Hence, this only works if both parties share a key.

Since this is the case in this thesis whenever message authentication codes are needed, we again use SHA-256 as an instantiation for MACs.

Another cryptographic primitive that is used in one of the analyzed authenticated key exchange protocols is a pseudorandom function.

Definition 4.6 (PRF)**.** *[KL07] A keyed function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is called pseudorandom function (PRF) if it is efficient, length preserving and no probabilistic polynomial-time distinguisher D can distinguish with more than negligible probability the output of $F(k,\cdot)$ from the output of a truly random function $f_\kappa(\cdot)$, where the length of the input equals κ for both functions, i.e.*

$$|Pr[D^{F_k(\cdot)}(1^\kappa) = 1] - Pr[D^{f_\kappa(\cdot)}(1^\kappa) = 1]| \leq negl(\kappa),$$

where $k \xleftarrow{\$} \{0,1\}^\kappa$ and f_κ chosen uniformly random from the set of functions mapping κ-bit strings to κ-bit strings.

The final shared session key of the sender and the receiver is in some cases the output of a certain function. This function can be either a pseudorandom function or a so called key derivation function.

Definition 4.7 (KDF)**.** *[FSXY12] A key derivation function (KDF) is a function $D : Salt \times Dom \to Rng$, with finite domain Dom, finite range Rng and non-secret random salt $Salt$. Furthermore, for the security parameter κ no probabilistic polynomial-time distinguisher D can distinguish with more than negligible probability the output of D from the output of a truly random function f on Rng.*

We conclude this section with a definition of k-bit security of a cryptographic scheme.

Definition 4.8. *Let Π be a cryptographic scheme and $\mathcal{A}$ be an algorithm that breaks the security of Π in time $t_\mathcal{A}$ with probability $\varepsilon_\mathcal{A}$. The scheme Π is called k-bit secure if $\frac{t_\mathcal{A}}{\varepsilon_\mathcal{A}} \geq 2^k$.*

5 Key Exchange Protocols

A key exchange protocol (KE) is a method that describes how different parties can generate a common secret key via a public network communication [KL07]. Those parties do not share any secret information before. That means that several parties want to agree on a common key that no other party is able to obtain or compute while their only way of communication is possibly eavesdropped. More formally, in a KE protocol each party computes a public-secret-key pair and sends his public key to the other party. Using their own public-secret key pair and the other party's public key, they compute a shared session key.

In this section we analyse three lattice-based KE protocols. We call the first two protocols JD and ring-JD and they were developed by Ding et al. [JD12]. The third protocol was developed by Bos et al. [BCNS14] and we call it BCNS protocol.

For each protocol we give a short introduction including a statement of the lattice problem on which the security of the protocol is based. The used security model is stated but not defined since this is out of the scope of this thesis. A definition of needed additional functions together with a description of the protocol follows. Furthermore, we quote lemmas that state parameter conditions for the correctness of the protocol, i.e. under which the computed session keys of the two parties are the same.

5.1 The JD and ring-JD Key Exchange by Ding, Xie, and Lin

Ding et al. [JD12] propose one key-exchange protocol based on the LWE problem and one based on its ring variant. We call them JD and ring-JD from now on. The basic idea can be seen as a Diffie-Hellman key exchange protocol based on the LWE and $R\text{-}LWE$ problem.

Both key exchange protocols are secure against passive probabilistic polynomial-time (PPT) adversaries if the $DLWE_{n,q,D_{\mathbb{Z}^n,\sigma}}$ and $R\text{-}DLWE_{2n,2^l,q,\chi}$ problem with factor $t = 2$ are hard. For a proof and an exact definition of this security model refer to [JD12].

The JD and the ring-JD protocol use some specific functions. Similar methods are also used in Section 6.3 by the ZZD protocols. Ding et al. [JD12] define the functions δ_0, δ_1, S and E via

$$\delta_0 : \mathbb{Z}_q \to \{0,1\}, x \mapsto \begin{cases} 0 & \text{if } x \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor] \\ 1 & \text{otherwise,} \end{cases}$$

$$\delta_1 : \mathbb{Z}_q \to \{0,1\}, x \mapsto \begin{cases} 0 & \text{if } x \in [-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1] \\ 1 & \text{otherwise,} \end{cases}$$

$$S : \mathbb{Z}_q \to \{0,1\}, x \mapsto \delta_b(x) \text{ where } b \in \{0,1\} \text{ uniformly random, and}$$

$$E : \mathbb{Z}_q \times \{0,1\} \to \{0,1\}, (x,\delta) \mapsto \left(x + \delta \frac{q-1}{2} \mod q \right) \mod 2.$$

For any odd $q > 2$ if $x \in \mathbb{Z}_q$ is chosen uniformly random, then $E(x,\delta) \in \{0,1\}$ is uniformly distributed conditioned on $\delta \leftarrow S(x)$ [JD12], i.e. for any $\delta, b' \in \{0,1\}$ it holds that

$$Pr_{x \xleftarrow{\$} \mathbb{Z}_q, b \xleftarrow{\$} \{0,1\}}[E(x,\delta) = b' | \delta_b(x) = \delta] = \frac{1}{2}.$$

Let $q > 8$ be odd. Ding et al. [JD12] show that for all $x, y \in \mathbb{Z}_q$ such that $|x - y| \leq \frac{q}{4} - 2$ and $x - y$ is even, the equality $E(x, S(y)) = E(y, S(y))$ holds. The proof is very similar to the proof of Lemma 6.5. This is used to make sure that both parties compute the same shared session key $SK_S = SK_R$.

The two versions of the key exchange protocol are depicted in Figure 1 and Figure 2. A description of the JD protocol follows now.

Public parameters: The protocol depends on the parameters n, q, and σ. Let $n, q \in \mathbb{N}$, q be an odd prime, and σ be the standard deviation of the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$. Furthermore, let $A \in \mathbb{Z}_q^{n \times n}$ be a uniformly sampled matrix.

Initiation: The sender computes a public-secret key pair by sampling the secret key $s_S \leftarrow D_{\mathbb{Z}^n,\sigma}$ and computing the public key $p_S = As_S + 2e_S \mod q$, where $e_S \leftarrow D_{\mathbb{Z}^n,\sigma}$. The public key is sent to the receiver.

Response: The receiver computes a public-secret key pair by sampling the secret key $s_R \leftarrow D_{\mathbb{Z}^n,\sigma}$ and computing the public key $p_R = A^T s_R + 2e_R \mod q$, where $e_R \leftarrow D_{\mathbb{Z}^n,\sigma}$. The receiver samples an additional error term $e'_R \leftarrow D_{\mathbb{Z},\sigma}$, computes $k_R = p_S^T s_R + 2e'_R \mod q$ and $\delta \leftarrow S(k_R)$, and obtains the session key $SK_R = E(k_R, \delta)$. The public key p_R and the bit δ are finally sent to the sender.

Finish: The sender samples $e'_S \leftarrow D_{\mathbb{Z},\sigma}$, computes $k_S = s_S^T p_R + 2e'_S \mod q$, and obtains the session key $SK_S = E(k_S, \delta)$.

The sender and the receiver share only one secret bit after running the protocol described in Figure 1. To extent it to multiple shared secret bits, each party chooses secret matrices $S_S, S_R \in \mathbb{Z}_q^{n \times n}$ instead of secret vectors $s_S, s_R \in \mathbb{Z}^n$.

Sender		**Receiver**
$s_S, e_S \leftarrow D_{\mathbb{Z}^n,\sigma}$		$s_R, e_R \leftarrow D_{\mathbb{Z}^n,\sigma}$
$p_S = As_S + 2e_S \mod q$		$p_R = A^T s_R + 2e_R \mod q$
	$\xrightarrow{p_S}$	$e'_R \leftarrow D_{\mathbb{Z},\sigma}$
		$k_R = p_S^T s_R + 2e'_R \mod q$
	$\xleftarrow{p_R,\delta}$	$\delta \leftarrow S(k_R)$
$e'_S \leftarrow D_{\mathbb{Z},\sigma}$		$SK_R = E(k_R, \delta)$
$k_S = s_S^T p_R + 2e'_S \mod q$		
$SK_S = E(k_S, \delta)$		

Figure 1: The JD key exchange protocol

Lemma 5.1 (Correctness)**.** *[JD12] If* $16(\sigma\pi)^2 n \leq \frac{q}{4} - 2$*, then* $SK_S = SK_R$ *with overwhelming probability.*

Ding et al. [JD12] suggested the parameters $n = \lambda$, $q \approx \lambda^4$, and $\sigma = \frac{\lambda}{\sqrt{2\pi}}$. These parameters not only guarantee correctness but also satisfy $\sigma\sqrt{2\pi} > \sqrt{n}$, which is a condition for the worst-to-average-case reduction from the SVP to the LWE problem (see Subsection 2.3).

From this key exchange protocol, Ding et al. [JD12] obtained a key exchange protocol based on the *R-DLWE* problem. The functions δ_0, δ_1, S, and E are extended coefficient-wise to elements in R_q. A short overview of the protocol, which we call ring-JD, is given in Figure 2.

Public parameters: The protocol depends on the parameters n, q, and σ. Let $n = 2^l$ be the degree of the field extension of the m-th cyclotomic field K_m for $m = 2n$ and q be a prime such that $q \equiv 1 \mod 2n$. Let σ be the parameter of the error distribution χ such that $Pr[\|x\| > \sqrt{2\pi n}\sigma : x \leftarrow \chi] \leq negl(n)$ holds. Furthermore, let $a \in R_q$ be a uniformly sampled polynomial.

Initiation: The sender computes a public-secret key pair by sampling the secret key $s_S \leftarrow \chi$ and computing the public key $p_S = as_S + 2e_S \mod q$, where $e_S \leftarrow \chi$. The public key is sent to the receiver.

Response: The receiver computes a public-secret key pair by sampling the secret key $s_R \leftarrow \chi$ and computing the public key $p_R = as_R + 2e_R \mod q$, where $e_R \leftarrow \chi$. The receiver samples an additional error term $e'_R \leftarrow \chi$, computes $k_R = p_S s_R + 2e'_R \mod q$ and $\delta \leftarrow S(k_R)$, and obtains the session key $SK_R = E(k_R, \delta)$. The public key p_R and the n-bit string δ is finally sent to the sender.

Finish: The sender samples $e'_S \leftarrow \chi$, computes $k_S = s_S p_R + 2e'_S \mod q$, and obtains the session key $SK_S = E(k_S, \delta)$.

A distribution χ that depends on the parameter σ such that $Pr[\|x\| > \sqrt{2\pi n}\sigma : x \leftarrow \chi] \leq negl(n)$ holds can be obtained by taking the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ with parameter σ. This follows from Lemma 5.2.

Lemma 5.2. *[MR07] For any $\sigma \geq \omega(\sqrt{\log(n)})$ we find $Pr[\|x\| > \sqrt{2\pi n}\sigma : x \leftarrow D_{\mathbb{Z}^n,\sigma}] \leq 2^{-n}$.*

Sender		**Receiver**
$s_S, e_S \leftarrow \chi$		$s_R, e_R \leftarrow \chi$
$p_S = as_S + 2e_S \mod q$		$p_R = as_R + 2e_R \mod q$
	$\xrightarrow{p_S}$	$e'_R \leftarrow \chi$
		$k_R = p_S s_R + 2e'_R \mod q$
	$\xleftarrow{p_R, \delta}$	$\delta \leftarrow S(k_R)$
$e'_S \leftarrow \chi$		$SK_R = E(k_R, \delta)$
$k_S = s_S p_R + 2e'_S \mod q$		
$SK_S = E(k_S, \delta)$		

Figure 2: The ring-JD key exchange protocol

Equivalently to Lemma 5.1 the correctness of the scheme is guaranteed by the following lemma.

Lemma 5.3 (Correctness)**.** *[JD12] If $16\pi(n\sigma)^2 \leq \frac{q}{4} - 2$, then $SK_S = SK_R$ with overwhelming probability.*

Ding et al. [JD12] propose the parameters $n = \lambda$, $q \approx \lambda^4$ and $\sigma = \frac{\lambda}{\sqrt{2\pi\lambda}}$, which fulfill the conditions for the correctness given above.

Both key exchange schemes by [JD12] are only proven to be secure in the two-user setting. A multi-user variant is proposed in the same paper, but its security is not yet proven.

5.2 The BCNS Key Exchange by Bos, Costello, Naehrig, and Stebila

Bos, Costello, Naehrig, and Stebila [BCNS14] introduced a key exchange protocol that replaces the traditional number theoretic key exchange in the Transport Layer Security protocol (TLS) [DR06] by one based on the *R-LWE* problem. Together with *RSA* or elliptic curve digital signatures it forms an authenticated key exchange protocol. We only analyze the lattice-based unauthenticated key exchange protocol, which we call BCNS protocol. Furthermore, we note that Alkim et al. [ADPS15] recently proposed a new version of the BCNS protocol, which can be interesting for future research.

This new TLS protocol is secure in the authenticated and confidential channel establishment ($ACCE$) security model, which is based on the Bellare-Rogaway model. A definition of the security model exceeds the scope of this thesis. For more details refer to Bos et al. [BCNS14].

The key exchange protocol uses some specific rounding functions and concepts that were first defined by Peikert [Pei14]. Therefore, the resulting key exchange protocol is a special case of Peikert's key encapsulation mechanism which is described in Subsection 6.4.

Peikert proved that the BCNS protocol is IND-CPA secure for parameters $n = 2^l$, q, and χ, if the $R\text{-}DLWE_{2n,2^l,q,\chi}$ problem is hard to solve [Pei14].

First, we define the needed functions. Afterwards, we give a description of the scheme, which is depicted in Figure 3, and state parameters for its correctness.

The general rounding function is denoted by

$$\lfloor \cdot \rceil : \mathbb{R} \to \mathbb{Z}, x \mapsto z, \text{ where } x \in [z - 1/2, z + 1/2).$$

Definition 5.4. *For a positive modulus q the modulus rounding function is defined by*

$$\lfloor \cdot \rceil_{q,2} : \mathbb{Z}_q \to \mathbb{Z}_2, x \mapsto \left\lfloor \frac{2}{q} x \right\rceil \mod 2.$$

The cross rounding function is defined via

$$\langle \cdot \rangle_{q,2} : \mathbb{Z}_q \to \mathbb{Z}_2, x \mapsto \left\lfloor \frac{4}{q} x \right\rceil \mod 2.$$

Let q be an even modulus. Then we obtain

$$\lfloor v \rceil_{q,2} = \begin{cases} 0 & v \in \{0, 1, \ldots, \lfloor \frac{q}{4} \rceil - 1\} \cup \{\lfloor \frac{3q}{4} \rceil, \ldots, q-1\} \\ 1 & v \in \{\lfloor \frac{q}{4} \rceil, \ldots, \lfloor \frac{3q}{4} \rceil - 1\} \end{cases}.$$

Let $I_0 = \{0, 1, \ldots, \lfloor \frac{q}{4} \rceil - 1\}$ and $I_1 = \{-\lfloor \frac{q}{4} \rfloor, \ldots, -1\}$ be two sets. In $\mathbb{Z}_q$ the elements of I_1 can be represented by the elements in $\{\lfloor \frac{3q}{4} \rceil, \ldots, q-1\}$. This is denoted by $I_1 = \{\lfloor \frac{3q}{4} \rceil, \ldots, q-1\}$

mod q. In the same way we obtain $\{\lfloor \frac{q}{4} \rceil, \ldots, \lfloor \frac{3q}{4} \rceil - 1\} = \left(\left(I_0 + \frac{q}{2}\right) \cup \left(I_1 + \frac{q}{2}\right)\right) \mod q$. Analysing the cross rounding function $\langle \cdot \rangle_{q,2}$, we obtain

$$\langle v \rangle_{q,2} = \begin{cases} 0 & v \in I_0 \cup \left(I_0 + \frac{q}{2}\right) \\ 1 & v \in I_1 \cup \left(I_1 + \frac{q}{2}\right). \end{cases}$$

If q is an odd integer, the modular rounding function on $\mathbb{Z}_q$ would be biased. Therefore, Peikert [Pei14] introduced a randomized doubling function

$$dbl : \mathbb{Z}_q \to \mathbb{Z}_{2q}, x \mapsto 2x - e, \text{ where } e = \begin{cases} -1 & \text{with probability } 1/4, \\ 0 & \text{with probability } 1/2, \\ 1 & \text{with probability } 1/4. \end{cases}$$

Lemma 5.5. *[Pei14] Let q be an odd integer and $v \in \mathbb{Z}_q$ be chosen uniformly random. Then $\lfloor dbl(v) \rceil_{2q,2}$ is uniformly distributed given $\langle dbl(v) \rangle_{2q,2}$.*

Before looking at the key exchange protocol, one more function needs to be defined that is used to calculate the shared session key. Let I'_b be defined by replacing q by $2q$ in the set I_b, i.e. we obtain $I'_1 = \{-\lfloor \frac{q}{2} \rfloor, \ldots, -1\}$ and $I'_0 = \{0, 1, \ldots, \lfloor \frac{q}{2} \rceil - 1\}$. The reconciliation function rec is defined as

$$rec : \mathbb{Z}_{2q} \times \mathbb{Z}_2 \to \mathbb{Z}_2, (v, b) \mapsto \begin{cases} 0 & \text{if } v \in \left(\left(I'_b + \left[-\frac{q}{4}, \frac{q}{4}\right)\right) \bmod 2q\right), \\ 1 & \text{otherwise.} \end{cases}$$

Lemma 5.6. *Let q be odd and $w = v + e \in \mathbb{Z}_q$ with $w, e \in \mathbb{Z}_q$ and $2e \pm 1 \in \left[-\frac{q}{4}, \frac{q}{4}\right) \subset \mathbb{Z}_q$. Then it holds that $rec(2w, \langle dbl(v) \rangle_{2q,2}) = \lfloor dbl(v) \rceil_{2q,2}$.*

Proof. On the left-hand side of the equation we obtain $rec(2w, \langle dbl(v) \rangle_{2q,2}) = 0$ if and only if

$$2w = 2v + 2e \in \left(I'_b + \left[-\frac{q}{4}, \frac{q}{4}\right)\right) \bmod 2q \text{ and } \langle dbl(v) \rangle_{2q,2} = b.$$

Let $\langle dbl(v) \rangle_{2q,2} = b$. This holds if and only if $dbl(v) \in I'_b \cup \left(I'_b + q\right)$.
In this case we obtain on the right-hand side of the equation $\lfloor dbl(v) \rceil_{2q,2} = 0$ if and only if $dbl(v) \in I'_b$. With $dbl(v) = 2v - \tilde{e}$ and $\tilde{e} \in \{-1, 0, 1\}$, this is equivalent to

$$2w = 2v + 2e = dbl(v) + \tilde{e} + 2e \in \left(I'_b + \left[-\frac{q}{4}, \frac{q}{4}\right)\right).$$

Hence, the claim follows. □

The functions $\lfloor \cdot \rceil_{2q,2}, \langle \cdot \rangle_{2q,2}$, and dbl can be extended to functions over R_q by applying them to each coefficient.

A description of the key exchange protocol BCNS as depicted in Figure 3 follows.

Public parameters: The protocol depends on the parameters n, q, and σ. Let $n = 2^l$ be the degree of the field extension of the m-th cyclotomic field K_m for $m = 2n$ and q be an

Sender		Receiver
$s_S, e_S \leftarrow \chi$		$s_R, e_R \leftarrow \chi$.
$p_S = as_S + e_S$		$p_R = as_R + e_R$
	$\xrightarrow{p_S}$	$e'_R \leftarrow \chi$
		$v = p_S s_R + e'_R$
		$\bar{v} \leftarrow dbl(v)$
	$\xleftarrow{p_R, c}$	Set $c = \langle \bar{v} \rangle_{2q,2}$
$SK_S = rec(2p_R s_S, c)$		$SK_R = \lfloor \bar{v} \rceil_{2q,2}$

Figure 3: The BCNS key exchange protocol

odd positive integer. Let σ be the parameter of the discrete Gaussian distribution χ over $R = \mathbb{Z}[X]/\langle X^n + 1 \rangle$ and $a \in R_q$ be a uniformly sampled polynomial.

Initiation: The sender computes a public-secret key pair by sampling the secret key $s_S \leftarrow \chi$ and computing the public key $p_S = as_S + e_S \mod q$, where $e_S \leftarrow \chi$. The public key is sent to the receiver.

Response: The receiver computes a public-secret key pair by sampling the secret key $s_R \leftarrow \chi$ and computing the public key $p_R = as_R + e_R \mod q$, where $e_R \leftarrow \chi$. The receiver samples an additional error term $e'_R \leftarrow \chi$ and computes $v = p_S s_R + e'_R \mod q$ and $\bar{v} = dbl(v)$. The polynomial $\bar{v}$ is passed to the functions $\lfloor \cdot \rceil_{2q,2}$ and $\langle \cdot \rangle_{2q,2}$ to compute $c = \langle \bar{v} \rangle_{2q,2}$ and the session key $SK_R = \lfloor \bar{v} \rceil_{2q,2}$. The public key p_R and the n-bit string c are sent to the sender.

Finish: The sender obtains the session key SK_S via $SK_S = rec(2p_R s_S, c)$.

To guarantee a 128-bit security level, Bos, Costello, Naehrig, and Stebila suggest parameters $n = 1024$, $q = 2^{32} - 1$, and $\chi = D_{\mathbb{Z},\sigma}(x)$ with $\sigma = 8/\sqrt{2\pi}$.

Lemma 5.7 (Correctness). *[BCNS14] Let $n = 1024$, $q = 2^{32} - 1$, and $\chi = D_{\mathbb{Z},\sigma}(x)$ with $\sigma = 8/\sqrt{2\pi}$. If both parties execute the BCNS protocol honestly, the probability that the two keys SK_S and SK_R are the same is greater than $1 - 2^{-2^{17}}$.*

6 Authenticated Key Exchange Protocols

A key exchange protocol that also authenticates the identities of the involved parties is called authenticated key exchange protocol (AKE). More formally, in an AKE protocol each party has a static public-secret key pair. The static public key is certified with the party's identity. When running the protocol, each party generates an ephemeral secret key and, depending on that, a corresponding ephemeral public key. The public key is sent to the other party. Each party computes a shared session state by using the ephemeral and the static keys. This shared session state is passed to a key derivation function which outputs the session key [ZZD+14]. Each invocation of the protocol is called session and identified by a session identity denoted by sid. The session identity sid usually consists of public information such as the identities of the involved parties and public keys. By I_S and I_R we denote the identities of the sender and the receiver respectively.

In the following subsections we describe and analyze five different lattice-based authenticated key exchange protocols. The first two protocols were developed by Fujioka, Suzuki, Xagawa, and Yoneyama and we call them FSXY12 and FSXY13. Zhang, Zhang, Ding, Snook, and Dagdelen constructed a two- and a one-pass AKE called two-pass ZZD and one-pass ZZD respectively. The last protocol was developed by Peikert and we refer to it as the Peikert protocol.

Most of those AKE protocols make use of other cryptographic primitives like key encapsulation mechanisms, public key encryption schemes, and signature schemes.

For each protocol we give a short introduction including a statement of the lattice problem on which the security of the protocol is based. The used security model is stated but not defined since this is out of the scope of this thesis. A definition of needed additional functions together with a description of the protocol follows. Furthermore, we quote lemmas that state conditions for the correctness and security of the protocol.

6.1 The FSXY12 Key Exchange by Fujioka, Suzuki, Xagawa, and Yoneyama

Fujioka, Suzuki, Xagawa, and Yoneyama developed a generic construction of an authenticated key exchange protocol from key encapsulation mechanisms [FSXY12]. It is a generalization with stronger security based on a previous construction by Boyd et al. [BCNP08, BCNP09] that was based on the Diffie-Hellman assumption.

The constructed AKE satisfies one of the strongest security models for AKE under certain conditions, namely CK^+ security in the standard model. CK^+ is a combination of several security models, namely the Canetti-Krawczyk (CK) security model, security against key compromise impersonation (KCI), weak perfect forward secrecy (wPFS), and security against maximal exposure attacks (MEX). For more details and definitions of those attacks, see [FSXY12].

The AKE consists of three different pseudorandom functions, a key derivation function, and two not necessarily different key encapsulation mechanisms with certain properties. However, the resulting AKE protocol is not necessarily lattice-based unless one decides to use a lattice-based instantiation.

In the following we describe the needed cryptographic primitives before looking at the generic protocol in Figure 4. An overview of the quantities of the used primitives is given in Table 8 in Section 8.

Let κ be the security parameter and

$$kem = (KeyGen, EnCap, DeCap),$$
$$wkem = (wKeyGen, wEnCap, wDeCap)$$

be two key encapsulation mechanisms with random space $\mathcal{RS}_G$ for the key generation algorithms and random space $\mathcal{RS}_E$ for the encapsulation algorithms. That means that the key generation and the encapsulation algorithms have an additional random input from those random spaces which is used as a random seed. Furthermore, let

$$F : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E,$$
$$F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E, \text{ and}$$
$$G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$$

be three pseudo-random functions with key space $\mathcal{FS}$ and $|\mathcal{FS}| = \kappa$.
Let

$$kdf : Salt \times K \to \mathcal{FS}$$

be a key derivation function with a non-secret salt $s \in Salt$, where $Salt$ is the salt space and K is the key space of the key encapsulation mechanisms kem and $wkem$.

Public parameters: The protocol depends on the security parameter κ, the cryptographic primitives of two KEMs, three PRFs, and one KDF. Let F, F', and G, as defined above, be the three PRFs, $kem = (KeyGen, EnCap, DeCap)$ and $wkem = (wKeyGen, wEnCap, wDeCap)$ be the two KEMs, and kdf be the needed KDF.

Key generation: Both parties generate a static public-secret key pair $(sk, pk) \leftarrow KeyGen(1^\kappa, r)$, where $r \xleftarrow{\$} \mathcal{RS}_G$ is a seed for the key generation algorithm $KeyGen$. Furthermore, $\sigma \xleftarrow{\$} \mathcal{FS}$ and $\sigma' \xleftarrow{\$} \{0,1\}^\kappa$ are part of the static secret keys.

Initiation: The sender samples seeds $r_S \xleftarrow{\$} \{0,1\}^\kappa$, $r'_S \xleftarrow{\$} \mathcal{FS}$, and $r_{TS} \xleftarrow{\$} \mathcal{RS}_G$ and generates an ephemeral public-secret key pair $(sk_T, pk_T) \leftarrow wKeyGen(1^\kappa, r_{TS})$. He computes $(c_S, k_S) \leftarrow EnCap(F_{\sigma_S}(r_S) \oplus F'_{r'_S}(\sigma'_S), pk_R)$, where $F_{\sigma_S}(r_S) \oplus F'_{r'_S}(\sigma'_S)$ serves as random input seed, and sends (c_S, pk_T) to the receiver.

Response: The receiver samples seeds $r_R \xleftarrow{\$} \{0,1\}^\kappa$, $r'_R \xleftarrow{\$} \mathcal{FS}$, and $r_{TR} \xleftarrow{\$} \mathcal{RS}_E$. He computes $(c_R, k_R) \leftarrow EnCap(F_{\sigma_R}(r_R) \oplus F'_{r'_R}(\sigma'_R), pk_S)$ and $(c_T, k_T) \leftarrow wEnCap(r_{TR}, pk_T)$, where $F_{\sigma_R}(r_R) \oplus F'_{r'_R}(\sigma'_R)$ and r_{TR} respectively serve as random input seeds, and sends (c_R, c_T) to the sender. The receiver obtains $k_S \leftarrow DeCap(c_S, sk_R)$ and, through the KDF, $k'_1 = kdf(s, k_S)$, $k'_2 = kdf(s, k_R)$, and $k'_3 = kdf(s, k_T)$. With the session identity $sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$ he obtains the session key $SK = G_{k'_1}(sid) \oplus G_{k'_2}(sid) \oplus G_{k'_3}(sid)$.

Sender		Receiver
Long-term keys		Long-term keys
$\sigma_S \xleftarrow{\$} \mathcal{FS}$, $\sigma'_S \xleftarrow{\$} \{0,1\}^\kappa$		$\sigma_R \xleftarrow{\$} \mathcal{FS}$, $\sigma_R \xleftarrow{\$} \{0,1\}^\kappa$
$r \xleftarrow{\$} \mathcal{RS}_G$		$r' \xleftarrow{\$} \mathcal{RS}_G$
$(s_S, p_S) \leftarrow KeyGen(1^\kappa, r)$		$(sk_R, pk_R) \leftarrow KeyGen(1^\kappa, r')$
$r_S \xleftarrow{\$} \{0,1\}^\kappa$		
$r'_S \xleftarrow{\$} \mathcal{FS}$		
$r_{TS} \xleftarrow{\$} \mathcal{RS}_G$		
(c_S, k_S)		
$\leftarrow EnCap(F_{\sigma_S}(r_S) \oplus F'_{r'_S}(\sigma'_S), pk_R)$		
$(sk_T, pk_T) \leftarrow wKeyGen(1^\kappa, r_{TS})$	$\xrightarrow{(c_S, pk_T)}$	
		$r_R \xleftarrow{\$} \{0,1\}^\kappa$
		$r'_R \xleftarrow{\$} \mathcal{FS}$
		$r_{TR} \xleftarrow{\$} \mathcal{RS}_E$
		(c_R, k_R)
		$\leftarrow EnCap(F_{\sigma_R}(r_R) \oplus F'_{r'_R}(\sigma'_R), pk_S)$
	$\xleftarrow{(c_R, c_T)}$	$(c_T, k_T) \leftarrow wEnCap(r_{TR}, pk_T)$
$k_R \leftarrow DeCap(c_R, s_S)$		$k_S \leftarrow DeCap(c_S, sk_R)$
$k_T \leftarrow wDeCap(c_T, sk_T)$		
$k'_1 = kdf(s, k_S)$		$k'_1 = kdf(s, k_S)$
$k'_2 = kdf(s, k_R)$		$k'_2 = kdf(s, k_R)$
$k'_3 = kdf(s, k_T)$		$k'_3 = kdf(s, k_T)$
Session identity, session key:		Session identity, session key:
$sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$		$sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$
$SK = G_{k'_1}(sid) \oplus G_{k'_2}(sid) \oplus G_{k'_3}(sid)$		$SK = G_{k'_1}(sid) \oplus G_{k'_2}(sid) \oplus G_{k'_3}(sid)$

Figure 4: The FSXY12 authenticated key exchange protocol

Finish: The sender computes $k_R \leftarrow DeCap(c_R, s_S)$ and $k_T \leftarrow wDeCap(c_T, sk_T)$. Through the KDF he obtains $k'_1 = kdf(s, k_S)$, $k'_2 = kdf(s, k_R)$, and $k'_3 = kdf(s, k_T)$. With the session identity $sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$ he computes the session key $SK = G_{k'_1}(sid) \oplus G_{k'_2}(sid) \oplus G_{k'_3}(sid)$.

Theorem 6.1. *[FSXY12] If $kem = (KeyGen, EnCap, DeCap)$ is an IND-CCA secure and κ-min-entropy KEM, $wkem = (wKeyGen, wEnCap, wDeCap)$ is an IND-CPA secure and κ-min-entropy KEM, F, F', and G are PRFs, and kdf is a KDF, then the AKE construction depicted in Figure 4 is CK^+ secure in the standard model.*

For a proof refer to [FSXY12, appendix].

We refrain from a definition of κ-min-entropy, as lattice-based PKE schemes are κ-min-entropy KEMs if the message space is larger than the security parameter κ and the message is chosen

uniformly at random [FSXY12]. As seen in Section 4, the uniformly chosen message of the PKE corresponds to the derived key of the encapsulation and decapsulation algorithms of the KEM in this case.

This authenticated key exchange protocol is correct as long as both KEMs are correct.

Key encapsulation mechanisms based on the hardness of the ring LWE problem that fulfill the above conditions exist as well as PRFs constructed from the ring LWE assumption. Hence, from this generic construction of an AKE, an AKE based on lattices can be obtained. However, all known lattice-based KEMs that are IND-CCA secure in the standard model need huge keys and the resulting AKE is therefore neither practical nor efficient. Therefore, this authenticated key exchange protocol is mostly of theoretical interest. A more practical variant of this construction is given in Subsection 6.2.

6.2 The FSXY13 Key Exchange by Fujioka, Suzuki, Xagawa, and Yoneyama

As mentioned above, the authenticated key exchange by Fujioka et al. [FSXY12], which is CK^+ secure in the standard model, is very inefficient when instantiated by lattice-based KEMs. Fujioka et al. [FSXY13] propose a more practical generic post-quantum CK^+ secure AKE by using one-way secure KEMs in the random oracle model. That means that by using hash functions that are modelled as random oracles, the IND-CCA secure KEM can be obtained from an IND-CPA secure KEM. Therefore, this AKE can be seen as a practical variant of the one described in Section 6.1. We call it FSXY13 protocol. Fujioka et al. [FSXY13] prove CK^+ security in the random oracle model but do not consider quantum random oracle models.

Compared to the FSXY12 protocol, the FSXY13 protocol only needs two KEMs with certain properties and two hash functions that are modelled as random oracles.

In the following we describe the needed cryptographic primitives for this generic AKE protocol as well as the needed Fujisaki-Okamoto (FO) transformation that transforms an IND-CPA secure KEM into an IND-CCA secure KEM. An overview of the generic protocol is depicted in Figure 5 and a summary of its building blocks and their quantities can be seen in Table 8 in Section 8.

Let κ be the security parameter and

$$kem = (KeyGen, EnCap, DeCap),$$
$$wkem = (wKeyGen, wEnCap, wDeCap)$$

be two key encapsulation mechanisms with random space $\mathcal{RS}_G$ for the key generation algorithms and random space $\mathcal{RS}_E$ for the encapsulation algorithms. Again that means that the key generation and the encapsulation algorithms have an additional random input from those random spaces which is used as a random seed. Furthermore, let

$$H_1 : \{0,1\}^* \to \mathcal{RS}_E \text{ and}$$
$$H_2 : \{0,1\}^* \to \{0,1\}^\kappa$$

be two hash functions.

Sender		Receiver
Long-term keys		Long-term keys
$r \xleftarrow{\$} \mathcal{RS}_G$		$r' \xleftarrow{\$} \mathcal{RS}_G$
$(sk_S, pk_S) \leftarrow KeyGen(1^\kappa, r)$		$(sk_R, pk_R) \leftarrow KeyGen(1^\kappa, r')$
$r_S \xleftarrow{\$} \{0,1\}^\kappa$		
$r_{TS} \xleftarrow{\$} \mathcal{RS}_G$		
$(c_S, k_S) \leftarrow EnCap(H_1(r_S, sk_S), pk_R)$		
$(sk_T, pk_T) \leftarrow wKeyGen(1^\kappa, r_{TS})$	$\xrightarrow{c_S, pk_T}$	
		$r_R \xleftarrow{\$} \{0,1\}^\kappa$
		$r_{TR} \xleftarrow{\$} \mathcal{RS}_G$
		$(c_R, k_R) \leftarrow EnCap(H_1(r_R, sk_R), pk_S)$
	$\xleftarrow{c_R, c_T}$	$(c_T, k_T) \leftarrow wEnCap(r_{TR}, pk_T)$
$k_R \leftarrow DeCap(c_R, sk_S)$		$k_S \leftarrow DeCap(c_S, sk_R)$
$k_T \leftarrow wDeCap(c_T, sk_T)$		
Set session identity, session key		Set session identity, session key
$sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$		$sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$
$SK = H_2(k_S, k_R, k_T, sid)$		$SK = H_2(k_S, k_R, k_T, sid)$

Figure 5: The FSXY13 authenticated key exchange protocol

Public parameters: The protocol depends on the security parameter κ, the cryptographic primitives of two KEMs and two hash functions. Let $kem = (KeyGen, EnCap, DeCap)$ and $wkem = (wKeyGen, wEnCap, wDeCap)$ be the two KEMs and H_1 and H_2 be the two hash functions.

Key generation: Both parties generate a static public-secret key pair $(sk, pk) \leftarrow KeyGen(1^\kappa, r)$, where $r \xleftarrow{\$} \mathcal{RS}_G$ is a random seed for the key generation algorithm.

Initiation: The sender samples seeds $r_S \xleftarrow{\$} \{0,1\}^\kappa$ and $r_{TS} \xleftarrow{\$} \mathcal{RS}_G$ and generates an ephemeral public-secret key pair $(sk_T, pk_T) \leftarrow wKeyGen(1^\kappa, r_{TS})$. He computes $(c_S, k_S) \leftarrow EnCap(H_1(r_S, s_S), pk_R)$, where $H_1(r_S, s_S)$ serves as random input seed, and sends (c_S, pk_T) to the receiver.

Response: The receiver samples seeds $r_R \xleftarrow{\$} \{0,1\}^\kappa$ and $r_{TR} \xleftarrow{\$} \mathcal{RS}_E$. He computes $(c_R, k_R) \leftarrow EnCap(H_1(r_R, sk_R), pk_S)$ and $(c_T, k_T) \leftarrow wEnCap(r_{TR}, pk_T)$, where $H_1(r_R, sk_R)$ and r_{TR} respectively serve as random input seeds, and sends (c_R, c_T) to the sender. The receiver obtains $k_S \leftarrow DeCap(c_S, sk_R)$ and with the session identity $sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$ the session key $SK = H_2(k_S, k_R, k_T, sid)$.

Finish: The sender computes $k_R \leftarrow DeCap(c_R, s_S)$ and $k_T \leftarrow wDeCap(c_T, sk_T)$. With the session identity $sid = (I_S, I_R, pk_S, pk_R, c_S, pk_T, c_R, c_T)$ he computes the session key $SK = H_2(k_S, k_R, k_T, sid)$.

Theorem 6.2. *[FSXY13] If $kem = (KeyGen, EnCap, DeCap)$ is an OW-CCA secure KEM and $wkem = (wKeyGen, wEnCap, wDeCap)$ an OW-CPA secure KEM, then the AKE depicted in Figure 5 is CK$^+$ secure, where H_1 and H_2 are modelled as random oracles.*

This FSXY13 protocol is correct if both KEMs are correct.

OW in OW-CCA and OW-CPA stands for one-way. We refrain from a formal definition, since Fujioka et al. [FSXY13] instantiate the two needed secure KEMs by IND-CPA secure public key encryption schemes. Using PKE schemes instead of KEMs is possible because passively secure KEMs can be easily converted to passively secure PKEs and vice versa. For more details refer to Section 4.

To obtain the needed efficient OW-CCA secure PKE, Fujioka et al. [FSXY13] apply the Fujisaki-Okamoto conversion to transform an IND-CPA secure PKE into an IND-CCA secure PKE. This conversion is called FO transformation. Since for sufficiently large plaintext spaces IND-CCA security implies OW-CCA security [FO99], one obtains the needed OW-CCA security. We give a short description of the FO transformation and refer to [FO99] for more details.

Let $wpke = (wGen, wEnc, wDec)$ be an OW-CPA secure PKE scheme with message space $w\mathcal{M}$, random spaces $w\mathcal{R}\mathcal{S}_E$ and $w\mathcal{R}\mathcal{S}_G$ and ciphertext space wC. By applying the FO transformation to $wpke$ we obtain an OW-CCA secure PKE scheme $pke = (Gen, Enc, Dec) = FO(wpke)$ with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ such that $w\mathcal{M} = \mathcal{M} \times \mathcal{R}\mathcal{S}_E$ and $wC = C$ holds. Let $H : \{0,1\}^* \to w\mathcal{R}\mathcal{S}_E$ be a hash function. The FO transformation outputs

$\mathrm{Gen}(1^n, r_g) = wGen(1^n, r_g)$,
$\mathrm{Enc}(M, r_e, pk) = wEnc((M, r_e), H(M, r_e), pk)$, and
$$\mathrm{Dec}(c, sk) = \begin{cases} \perp & \text{if } wDec(c,sk) = \perp \text{ or } c \neq wEnc(wDec(c,sk), H(wDec(c,sk), pk)) \\ M & \text{if } c = wEnc(wDec(c,sk), H(wDec(c,sk), pk)) \text{ with } (M, r_e) = wDec(c,sk). \end{cases}$$

Fujioka et al. [FSXY13] give two suggestions for IND-CPA secure public key encryption schemes based on the *R-LWE* problem, namely the PKE by Lyubashevsky, Peikert and Regev and the PKE by Stehlé and Steinfeld. A description of both is given in the following subsections. In both PKEs the random input of the key generation and the encryption algorithm is then used as random seed for the sampling processes.

6.2.1 The LPR Public Key Encryption Scheme by Lyubashevsky, Peikert, and Regev

The LPR scheme is a modified version of the public key encryption scheme described in [LPR12] and [LPR13] that guarantees perfect correctness.

Let n be a power of two and $q \equiv 1 \mod 2n$ be a prime. Furthermore, identify a polynomial in R_q with its image under the coefficient embedding, hence, as a vector in $\mathbb{Z}_q^n$. Elements in $\mathbb{Z}_q$ are represented by elements in $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$.

Public parameters: The LPR scheme depends on the parameters n, q, p, t, and k. Let $n = 2^l$ be the degree of the field extension of the m-th cyclotomic field K_m for $m = 2n$ and q be an odd prime such that $q \equiv 1 \mod 2n$. Let $p = 2$ and $t = \omega(\sqrt{\lg n})$. Furthermore, let $\chi = \lfloor N(0, k^2/2\pi) \rceil$ be a distribution defined through the parameter $k \leq \sqrt{\frac{q-p}{2(2n+1)pt^2}}$ and $a \in R_q$ be a uniformly sampled polynomial.

Key generation: First, the polynomials $s_S, e_S \leftarrow \chi$ are sampled. If $\|s_S\|_\infty, \|e_S\|_\infty \leq kt$, the algorithm continues, else it resamples. The public key $p_S = as_S + e_S \in R_q$ is computed and the public-secret key pair $pk = (a, p_S), sk = s_S$ is obtained.

Encryption: The algorithm samples polynomials $s_R, e_R, e'_R \leftarrow \chi$. If $\|s_R\|_\infty, \|e_R\|_\infty, \|e'_R\|_\infty \leq kt$, the algorithm continues, else it resamples. The public key $p_R = as_R + e_R \in R_q$ and the polynomial $v = p_S s_R + e'_R \in R_q$ are computed. The algorithm samples the key $k' \xleftarrow{\$} w\mathcal{M} = \{0,1\}^n$, encrypts it via computing $c = v + \lfloor \frac{q}{p} k' \rceil$, and obtains the ciphertext $C = (p_R, c)$.

Decryption: The algorithm computes $d = c - p_R s_S \in R_q$ and outputs the decrypted key $k' = \lfloor \frac{p}{q} d \rceil \mod p$.

The distribution $\lfloor N(0, k^2/2\pi) \rceil$ is statistically indistinguishable from $D_{\mathbb{Z}^n, k/\sqrt{2\pi}}$.

For a proof that the LPR scheme is perfectly correct and that the following security theorem holds, refer to [FSXY13] and personal conversation with the authors. When the LPR scheme is used as an instantiation of the KEM for the FSXY13 protocol, the random input seeds are used as seeds for sampling from the distribution χ.

Theorem 6.3. *[FSXY13] The PKE obtained by applying the FO conversion to the LPR scheme with the random oracle $H : \{0,1\}^* \to w\mathcal{RS}_E$ and the sets $\mathcal{M} = \mathcal{RS}_E = \mathbb{Z}_p^{\frac{n}{2}}$, $w\mathcal{M} = \mathbb{Z}_p^n$, $w\mathcal{RS}_E = \{0,1\}^*$, and $wC = R_q^2$, is IND-CCA secure if the R-DLWE$_{2n,2^l,q,\chi}$ problem is hard for $q \equiv 1 \mod 2n$.*

6.2.2 The SS13 Public Key Encryption Scheme by Stehlé and Steinfeld

The SS13 scheme is a modified version of the public key encryption scheme by Stehlé and Steinfeld [SS13]. Fujioka et al. [FSXY13] added a resampling step to make the PKE scheme by Stehlé et al. perfectly correct.

As in the LPR scheme, we associate polynomials in R_q with their image under the coefficient embedding.

Public parameters: The SS13 scheme depends on the parameters n, q, p, t, k, ϵ, and σ. Let $n = 2^l$ be the degree of the field extension of the m-th cyclotomic field K_m for $m = 2n$ and q be an odd prime such that $q \equiv 1 \mod 2n$. Let $p = 2$ and $t = \omega(\sqrt{\lg n})$. Furthermore, let $\chi = \lfloor N(0, k^2/2\pi) \rceil$ be a distribution defined through the parameter $k \leq \sqrt{\frac{q-p}{2(2n+1)pt^2}}$ and $\sigma \geq 2n\sqrt{\ln(8nq)}q^{\frac{1}{2}+2\epsilon}$ with $\epsilon \in (0, \frac{1}{4})$ be the parameter of the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$. Let $a \in R_q$ be a uniformly sampled polynomial.

Key generation: A polynomial $f' \leftarrow D_{\mathbb{Z}^n,\sigma}$ is sampled and $f = pf' + 1 \in R_q$ is computed. If $f \notin R_q^*$ or $\|f\|_2 \geq 4p\sqrt{n}\sigma$, the algorithm resamples f', else it continues. A polynomial $g \leftarrow D_{\mathbb{Z}^n,\sigma}$ is sampled. If $g \notin R_q^*$ or $\|g\|_2 \geq \sqrt{n}\sigma$, the algorithm resamples g, else it continues. The polynomial $h = pg/f \in R_q^*$ is computed and the public-secret key pair $pk = h$, $sk = f$ is obtained.

Encryption: Polynomials $s, e \leftarrow \chi$ are sampled. If $\|s\|_\infty, \|e\|_\infty \leq kt$, the algorithm continues, else it resamples. The algorithm samples the key $k' \xleftarrow{\$} \mathbb{Z}_2[X]$ of degree $n-1$, encrypts it via computing $c = hs + pe + k' \in R_q$, and outputs the ciphertext c.

Decryption: The algorithm computes $d = cf \in R_q$ and outputs the decrypted key $k' = d \mod p$.

For a proof of perfect correctness of the SS13 scheme and the following security theorem refer to [FSXY13] and personal conversations with the authors. When the SS13 scheme is used as an instantiation of the KEM for the FSXY13 protocol, the random input seeds are used as seeds for sampling from the distributions $D_{\mathbb{Z}^n,\sigma}$ and χ.

Theorem 6.4. *[FSXY13] The PKE obtained by applying the FO conversion to the SS13 scheme with the random oracle $H : \{0,1\}^* \to w\mathcal{R}\mathcal{S}_E$ and the sets $\mathcal{M} = \mathcal{R}\mathcal{S}_E = \mathbb{Z}_p^{\frac{n}{2}}$, $w\mathcal{M} = R/pR$, $w\mathcal{R}\mathcal{S}_E = \{0,1\}^*$, and $wC = R_q$, is IND-CCA secure if the R-DLWE$_{2n,2^l,q,\chi}$ problem is hard for $q \equiv 1 \mod 2n$.*

6.3 The ZZD Key Exchange by Zhang, Zhang, Ding, Snook, and Dagdelen

Zhang, Zhang, Ding, Snook, and Dagdelen [ZZD+14] described an implicit AKE, i.e. no signature is needed to authenticate the identities of the involved parties. It is the first post-quantum AKE of its kind. There exists a two-pass and a one-pass version of the protocol which we call two-pass ZZD and one-pass ZZD respectively.

First, we define the needed cryptographic primitives and two additional functions. Afterwards, the protocol is described and some steps of the protocol are explained. A short overview of the two-pass ZZD protocol is depicted in Figure 6. The correctness condition is derived and the hardness assumption is stated. The one-pass ZZD protocol is described in Subsection 6.3.1.

The one- and two-pass ZZD protocols need a hash function and a key derivation function defined as

$$H_1 : \{0,1\}^* \to D_{\mathbb{Z}^n,\gamma} \text{ and}$$
$$H_2 : \{0,1\}^* \to \{0,1\}^n.$$

Furthermore, the two functions Cha and Mod_2 are needed, which work similar to the functions in the protocol of Section 5.1. More formally, let $q > 2$ be an odd prime and define the set $E = \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$ and the functions

$$Cha : \mathbb{Z}_q \to \mathbb{Z}_2, v \mapsto \begin{cases} 0 & \text{if } v \in E, \\ 1 & \text{otherwise,} \end{cases}$$

and

$$Mod_2 : \mathbb{Z}_q \times \{0,1\} \to \{0,1\}, (v,b) \mapsto \left(\left(v + b\frac{q-1}{2} \right) \mod q \right) \mod 2.$$

The following lemma proven by Zhang et al. [ZZD+14] guarantees that the derived session keys SK_S and SK_R of the sender and the receiver are the same.

Lemma 6.5. *Let q be an odd prime and $v, e \in \mathbb{Z}_q$ such that $|e| < \frac{q}{8}$. Then it holds that $Mod_2(v, Cha(v)) = Mod_2(v + 2e, Cha(v))$.*

Proof. Let $v \in \mathbb{Z}_q$. We find that $v + Cha(v)\frac{q-1}{2} \mod q \in E = \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$, i.e. if we add $2e$, we obtain $v + Cha(v)\frac{q-1}{2} \mod q + 2e \in \{-\lfloor \frac{q}{2} \rfloor, \ldots, \lfloor \frac{q}{2} \rceil\}$. Hence, $v + Cha(v)\frac{q-1}{2} \mod q + 2e = v + Cha(v)\frac{q-1}{2} + 2e \mod q = w + Cha(v)\frac{q-1}{2} \mod q$. Hence, we obtain $Mod_2(v, Cha(v)) = Mod_2(v + 2e, Cha(v))$. □

The functions Cha and Mod_2 can be extended to elements of R_q by applying them coefficient-wise to the image of elements of R_q under the coefficient embedding c.

A description of the two-pass ZZD protocol follows and is depicted in Figure 6.

Public parameters: The protocol depends on the parameters n, q, δ, β, M, a hash function, and a key derivation function. Let $n = 2^l$ be the degree of the field extension of the m-th cyclotomic field K_m, for $m = 2n$ and $q > 0$ be an odd prime such that $q \equiv 1 \mod 2n$. Let δ and β be the parameters of the two error distributions $\chi_\delta = D_{\mathbb{Z}^n,\delta}$ and $\chi_\beta = D_{\mathbb{Z}^n,\beta}$. Furthermore, let M be the rejection constant determined by Theorem 1.2 and $a \in R_q$ be a uniformly sampled polynomial. Let the hash function and the KDF be given by H_1 and H_2 as defined above.

Key generation: Both parties compute a static public-secret key pair by sampling the secret key $s, e \leftarrow \chi_\delta$ and computing the public key $p = as + 2e \mod q$.

Initiation: The sender samples $r_S, f_S \leftarrow \chi_\beta$ and computes $x_S = ar_S + 2f_S$. With hash function H_1 the sender derives $d_S = H_1(I_S, I_R, x_S)$ and computes $\hat{r_S} = s_S d_S + r_S$ and $\hat{f_S} = e_S d_S + f_S$. Let $z = (c(\hat{r_S}), c(\hat{f_S})) \in \mathbb{Z}^{2n}$ and $z_1 = (c(s_S d_S), c(e_S d_S)) \in \mathbb{Z}^{2n}$. The sender repeats the initiation protocol with probability $1 - \min(\frac{D_{\mathbb{Z}^{2n},\beta}(z)}{MD_{\mathbb{Z}^{2n},\beta,z_1}(z)}, 1)$. Afterwards, x_S is sent to the receiver.

Response: The receiver samples $r_R, f_R \leftarrow \chi_\beta$ and computes $x_R = ar_R + 2f_R$. With hash function H_1 the receiver derives $d_R = H_1(I_R, I_S, x_R, x_S)$ and computes $\hat{r_R} = s_R d_R + r_R$ and $\hat{f_R} = e_R d_R + f_R$. Let $z = (c(\hat{r_R}), c(\hat{f_R})) \in \mathbb{Z}^{2n}$ and $z_1 = (c(s_R d_R), c(e_R d_R)) \in \mathbb{Z}^{2n}$. The receiver repeats the response protocol with probability $1 - \min(\frac{D_{\mathbb{Z}^{2n},\beta}(z)}{MD_{\mathbb{Z}^{2n},\beta,z_1}(z)}, 1)$. The receiver samples $g_R \leftarrow \chi_\beta$, computes $d_S = H_1(I_S, I_R, x_S)$, and derives the session state $k_R = (p_S d_S + x_S)(\hat{r_R}) + 2d_S g_R$. The n-bit strings $w_R = Cha(k_R)$ and $\sigma_R = Mod_2(k_R, w_R)$ are computed and the session key $SK_R = H_2(I_S, I_R, x_S, x_R, w_R, \sigma_R)$ is derived. Afterwards, x_R and w_R are sent to the sender.

Finish: The sender samples $g_S \leftarrow \chi_\beta$ and computes $d_R = H_1(I_R, I_S, x_R, x_S)$ and the session state $k_S = (p_R d_R + x_R)(\hat{r_S}) + 2d_R g_S$. The sender derives $\sigma_S = Mod_2(k_S, w_R)$ and obtains the session key $SK_S = H_2(I_S, I_R, x_S, x_R, w_R, \sigma_S)$.

If we can apply Theorem 1.2 to z and z_1 of the two-pass ZZD protocol depicted in Step 4 and Step 7 in Figure 6, we achieve that r_i and $\hat{r_i}$, and f_i and $\hat{f_i}$ are statistically indistinguishable, for $i \in \{S, R\}$. In step 2 the message x_i is computed by $x_i = ar_i + 2f_i$, i.e. r_i is used as a signature. To guarantee that the signature contains the secret key s_i and that it is hard to obtain s_i given x_i, the closeness of r_i and $\hat{r_i}$ is necessary.

	Sender		Receiver
	Long-term keys		Long-term keys
1.	$s_S, e_S \leftarrow \chi_\delta$		$s_R, e_R \leftarrow \chi_\delta$
	$p_S = as_S + 2e_S$		$p_R = as_R + 2e_R$
2.	$r_S, f_S \leftarrow \chi_\beta$		$r_R, f_R \leftarrow \chi_\beta$
	$x_S = ar_S + 2f_S$		$x_R = ar_R + 2f_R$
3.	$d_S = H_1(I_S, I_R, x_S)$		
	$\hat{r_S} = s_S d_S + r_S$		
	$\hat{f_S} = e_S d_S + f_S$		
4.	$z = (c(\hat{r_S}), c(\hat{f_S})) \in \mathbb{Z}^{2n}$		
	$z_1 = (c(s_S d_S), c(e_S d_S)) \in \mathbb{Z}^{2n}$		
	Repeat steps 2-4 with probability $1 - \min(\frac{D_{\mathbb{Z}^{2n},\beta}(z)}{MD_{\mathbb{Z}^{2n},\beta,z_1}(z)}, 1)$		
5.		$\xrightarrow{x_S}$	
6.			$d_R = H_1(I_R, I_S, x_R, x_S)$
			$\hat{r_R} = s_R d_R + r_R$
			$\hat{f_R} = e_R d_R + f_R$
7.			$z = (c(\hat{r_R}), c(\hat{f_R})) \in \mathbb{Z}^{2n}$
			$z_1 = (c(s_R d_R), c(e_R d_R)) \in \mathbb{Z}^{2n}$
			Repeat steps 2, 6, 7 with probability $1 - \min(\frac{D_{\mathbb{Z}^{2n},\beta}(z)}{MD_{\mathbb{Z}^{2n},\beta,z_1}(z)}, 1)$
8.		$\xleftarrow{x_R}$	
9.	$g_S \leftarrow \chi_\beta$		$g_R \leftarrow \chi_\beta$
	$d_R = H_1(I_R, I_S, x_R, x_S)$		$d_S = H_1(I_S, I_R, x_S)$
	$k_S = (p_R d_R + x_R)(\hat{r_S}) + 2d_R g_S$		$k_R = (p_S d_S + x_S)(\hat{r_R}) + 2d_S g_R$
10.		$\xleftarrow{w_R}$	$w_R = Cha(k_R)$
11.	$\sigma_S = Mod_2(k_S, w_R)$		$\sigma_R = Mod_2(k_R, w_R)$
	$SK_S = H_2(I_S, I_R, x_S, x_R, w_R, \sigma_S)$		$SK_R = H_2(I_S, I_R, x_S, x_R, w_R, \sigma_R)$

Figure 6: The two-pass ZZD authenticated key exchange protocol

Combining [MR07] and [Ban93], we can conclude that for large enough σ all samples from $D_{\mathbb{Z}^n,\sigma}$ are small.

Lemma 6.6. *Let $\sigma = \omega(\sqrt{\log n}) \in \mathbb{R}$ and $t > \frac{1}{\sqrt{2\pi}}$ constant, then $Pr_{x \leftarrow D_{\mathbb{Z}^n,\sigma}}[\|x\| > t\sigma\sqrt{n}] \leq \frac{1}{2}(t\sqrt{2\pi e}e^{-\pi t^2})^n$.*

This can be used to guarantee that Theorem 1.2 is applicable. With Lemma 1.1, $z_1 = (c(s_i d_i), c(e_i d_i))$ is Gaussian distributed with parameter $\delta\gamma\sqrt{n}$. Applying Lemma 6.6 with $t = \frac{1}{2}$ to $\sigma = \sqrt{n}\delta\gamma$, we obtain $Pr[\|e_i d_i\|, \|e_i d_i\| \leq \frac{1}{2}\delta\gamma n] \leq 2 \cdot 0.943^n$. That means that the norm of z_1 is less than $\frac{1}{2}\delta\gamma n$ with high probability. Hence, $\frac{1}{2}\delta\gamma n$ serves as the bound T in Theorem 1.2, which is in this case applicable if $\beta \geq \frac{1}{2}\tau\delta\gamma n$ holds. Furthermore, Theorem 1.2 implies that

the expected number of repetitions of the Steps 2-4 and 2,6, and 7 in Figure 6 is given by the rejection constant $M = exp(\frac{12}{\tau} + \frac{1}{2\tau^2})$.

Next, we analyze the condition under which the two computed secret keys are the same. The proof is based on ideas by Zhang et al. [ZZD+14].

Lemma 6.7 (Correctness). *The sender and the receiver compute the same shared secret key $SK_S = SK_R$ with overwhelming probability if $16 \cdot 7\beta^2\sqrt{n} < q$.*

Proof. The sender and the receiver compute the same shared secret session key if $\sigma_S = \sigma_R$. This is the case if $k_S = k_R + 2e$ with $|e| < \frac{q}{8}$ as seen in Lemma 6.5. It holds that

$$\begin{aligned} k_S &= (p_R d_R + x_R)(s_S d_S + r_S) + 2 d_R g_S \\ &= (a s_R d_R + 2 e_R d_R + a r_R + 2 f_R)(s_S d_S + r_S) + 2 d_R g_S \\ &= a(s_R d_R + r_R)(s_S d_S + r_S) + 2(e_R d_R + f_R)(s_S d_S + r_S) + 2 d_R g_S \end{aligned}$$

and similarly

$$k_R = a(s_R d_R + r_R)(s_S d_S + r_S) + 2(e_S d_S + f_S)(s_R d_R + r_R) + 2 d_S g_R.$$

Hence, $SK_S = SK_R$ holds if

$$\|((e_R d_R + f_R)(s_S d_S + r_S) + d_R g_S) - ((e_S d_S + f_S)(s_R d_R + r_R) + d_S g_R)\|_\infty < \frac{q}{8}.$$

For that is suffices to guarantee that $\|((e_R d_R + f_R)(s_S d_S + r_S) + d_R g_S)\|_\infty$ and $\|((e_S d_S + f_S)(s_R d_R + r_R) + d_S g_R)\|_\infty$ are small. We only show this for the first one, as the argument for the second is analogous.

By Theorem 1.2 and Lemma 1.1 we can already conclude that $(e_R d_R + f_R)(s_S d_S + r_S)$ is Gaussian distributed with parameter $\beta^2\sqrt{n}$ and $d_R g_S$ is Gaussian distributed with parameter $\gamma\beta\sqrt{n}$. Therefore, $\|(e_R d_R + f_R)(s_S d_S + r_S)\|_\infty \leq 6\beta^2\sqrt{n}$ with overwhelming probability. Furthermore, $\|d_R g_S\|_\infty \leq 6\beta\gamma\sqrt{n}$ holds true with overwhelming probability since erfc(6) $\approx 2^{-55}$ ($erfc(x/\sigma\sqrt{2}) = 1 - erf(x/\sigma\sqrt{2}) = 1 - Pr[\|a\| \leq x | a \leftarrow D_\sigma]$).

This yields

$$\|(e_R d_R + f_R)(s_S d_S + r_S) + d_R g_S\|_\infty \leq 6\beta^2\sqrt{n} + 6\beta\gamma\sqrt{n} \leq 7\beta^2\sqrt{n}.$$

The same holds for $\|(e_S d_S + f_S)(s_R d_R + r_R) + d_S g_R\|_\infty$. Hence, if $16 \cdot 7\beta^2\sqrt{n} < q$, $SK_S = SK_R$ holds with overwhelming probability. □

Zhang et al. [ZZD+14] proved that the security of the two-pass ZZD protocol relies on the *R-DLWE* problem.

Theorem 6.8. *[ZZD+14] Let n be a power of 2 satisfying $0.97n \geq 2\kappa$ for security parameter κ, $q > 203$ be a prime satisfying $q \equiv 1 \mod 2n$, and $\beta = \omega(\delta\gamma n\sqrt{n \log n})$. Then the two-pass ZZD protocol is secure in the Bellare-Rogaway (BR) security model in the random oracle model if R-$DLWE_{2n,2^l,q,\chi_\delta}$ with additional factor $t = 2$ is hard.*

6.3.1 The One-Pass ZZD Protocol

Zhang et al. [ZZD+14] also proposed a one-pass variant of their AKE protocol, called one-pass ZZD. We depict it in Figure 7. The main difference is that the receiver is neither supposed to do rejection sampling, nor to send anything to the sender. Hence, this reduces the needed computations of this scheme. On the other hand, this protocol, unlike the two-pass ZZD protocol, does neither satisfy wPFS nor is it resistant against replay attacks. However, a weaker security statement for one-pass protocols can be proven, namely that the one-pass ZZD protocol is a secure authenticated CCA encryption scheme in the random oracle model [ZZD+14]. For the one-pass ZZD protocol a smaller q suffices for correctness. More concretely, q can be chosen β/δ-times smaller than in the two-pass ZZD protocol. The public parameters and the key generation algorithm are the same as in the two-pass ZZD protocol. A description of the initiation and finish algorithm follows.

	Sender		**Receiver**
	Long-term keys		Long-term keys
1.	$s_S, e_S \leftarrow \chi_\delta$ $p_S = as_S + 2e_S$		$s_R, e_R \leftarrow \chi_\delta$ $p_R = as_R + 2e_R$
2.	$r_S, f_S \leftarrow \chi_\beta$ $x_S = ar_S + 2f_S$		
3.	$d_S = H_1(I_S, I_R, x_S)$ $\hat{r_S} = s_S d_S + r_S$ $\hat{f_S} = e_S d_S + f_S$		
4.	$z = (c(\hat{r_S}), c(\hat{f_S})) \in \mathbb{Z}^{2n}$ $z_1 = (c(s_S d_S), c(e_S d_S)) \in \mathbb{Z}^{2n}$ Repeat steps 2-4 with probability $1 - \min(\frac{D_{\mathbb{Z}^{2n},\beta}(z)}{MD_{\mathbb{Z}^{2n},\beta,z_1}(z)}, 1)$		
5.	$g_S \leftarrow \chi_\beta$ $k_S = p_R \hat{r_S} + 2g_S$		
6.	$w_S = Cha(k_S)$		
7.		$\xrightarrow{x_S, w_S}$	$g_R \leftarrow \chi_\delta$ $d_S = H_1(I_S, I_R, x_S)$ $k_R = (p_S d_S + x_S)s_R + 2d_S g_R$
8.	$\sigma_S = Mod_2(k_S, w_S)$ $SK_S = H_2(I_S, I_R, x_S, w_S, \sigma_S)$		$\sigma_R = Mod_2(k_R, w_S)$ $SK_R = H_2(I_S, I_R, x_S, w_S, \sigma_R)$

Figure 7: The one-pass ZZD authenticated key exchange protocol

Initiation: The sender samples $r_S, f_S \leftarrow \chi_\beta$ and computes $x_S = ar_S + 2f_S$. With hash function H_1 the sender derives $d_S = H_1(I_S, I_R, x_S)$ and computes $\hat{r_S} = s_S d_S + r_S$ and $\hat{f_S} = e_S d_S + f_S$. Let $z = (c(\hat{r_S}), c(\hat{f_S})) \in \mathbb{Z}^{2n}$ and $z_1 = (c(s_S d_S), c(e_S d_S)) \in \mathbb{Z}^{2n}$. The sender repeats the initiation protocol with probability $1 - \min(\frac{D_{\mathbb{Z}^{2n},\beta}(z)}{MD_{\mathbb{Z}^{2n},\beta,z_1}(z)}, 1)$. The sender samples $g_S \leftarrow \chi_\beta$ and derives $k_S = (p_R \hat{r_S}) + 2g_S$. The n-bit strings $w_S = Cha(k_S)$ and $\sigma_S = Mod_2(k_S, w_S)$ are computed

and the session key $SK_R = H_2(I_S, I_R, x_S, w_S, \sigma_S)$ is derived. Afterwards, x_S and w_S are sent to the receiver.

Finish: The receiver samples $g_R \leftarrow \chi_\delta$ and computes $d_S = H_1(I_S, I_R, x_S)$ and the session state $k_R = (p_S d_S + x_S)s_R + 2d_S g_R$. The receiver derives $\sigma_R = Mod_2(k_R, w_S)$ and obtains the session key $SK_S = H_2(I_S, I_R, x_S, w_S, \sigma_R)$.

6.4 The Peikert Key Exchange by Peikert

Chris Peikert [Pei14] designed a lattice-based AKE from a generic AKE developed by Canetti and Krawczyk. He showed that the Diffie-Hellman key exchange mechanism can be replaced by any passively secure key encapsulation mechanism. We call the proposed authenticated key exchange protocol Peikert protocol.

Now we describe the steps of the Peikert protocol and depict it in Figure 8.

Public parameters: Let κ be the security parameter. The Peikert protocol, consists of an IND-CPA secure $KEM\ wkem = (wKeyGen, wEnCap, wDeCap)$ with key space K, a digital signature scheme $SIG = (Sig.Gen, Sign, Ver)$, a message authentication code $\Pi = (Gen, \mathcal{O}, Vrfy)$ with key space K' and message space $\{0,1\}^*$, and a pseudorandom function $F : K \times \{0,1\} \rightarrow K'$.

Key generation: Both parties have a long-term verification-signing key pair (vk, sk).

Initiation: The sender generates a public-secret key pair $(s_S, p_S) \leftarrow wKeyGen(1^\kappa)$ and sends p_S to the receiver.

Response: The receiver computes $(c, k) \leftarrow wEnCap(p_S)$ and the session state $(k_0, k_1) = (F_k(0), F_k(1))$, where k_0 is the session key. He computes a MAC tag $t_R \leftarrow \mathcal{O}(k_1, (1, sid, I_R))$ for message $(1, sid, I_R)$ and key k_1. Furthermore, the receiver computes a signature $\sigma_R = Sign(sk_R, (1, sid, p_S, c))$ with his long-term signing key sk_R for the message $(1, sid, p_S, c)$. The MAC tag t_R, the signature σ_R, and c are sent to the sender.

Finish sender: The sender computes $k \leftarrow wDeCap(c, s_S)$ and the session state $(k_0, k_1) = (F_k(0), F_k(1))$, where k_0 is the session key. The sender verifies the signature σ_R of the message $(1, sid, p_S, c)$ with the long-term verification key vk_R of the receiver by computing $Ver(vk_R, (1, sid, p_S, c), \sigma_R)$. Furthermore, he verifies the MAC tag t_R of the message $(1, sid, I_R)$ with key k_1 by computing $Vrfy(k_1, (1, sid, I_R), t_R)$. If both verifications are positive, he accepts k_0 as session key. He computes a MAC tag $t_S \leftarrow \mathcal{O}(k_1, (0, sid, I_S))$ for the message $(0, sid, I_S)$ with key k_1. Moreover, the sender computes a signature $\sigma_S = Sign(sk_S, (0, sid, p_S, c))$ for message $(0, sid, p_S, c)$ with his long-term signing key sk_S. The MAC tag t_S and the signature σ_S are sent to the receiver.

Finish receiver: The sender verifies the signature σ_S of the message $(0, sid, p_S, c)$ with the long-term verification key of the sender by computing $Ver(vk_S, (0, sid, p_S, c), \sigma_S)$. Furthermore, he verifies the MAC tag t_S of the message $(0, sid, I_S)$ with key k_1 by computing $Vrfy(k_1, (0, sid, I_S), t_S)$ and accepts k_0 as the session key.

Theorem 6.9. *The Peikert protocol is SK secure in the post-specified peer model, if* $SIG = (Sig.Gen, Sign, Ver)$ *and* $MAC = (Gen, \mathcal{O}, Vrfy)$ *are existentially unforgeable under chosen-*

message attacks, $wkem = (wKeyGen, wEnCap, wDeCap)$ is an IND-CPA secure KEM, and F is a secure pseudorandom function.

SK security ensures for example weak perfect forward secrecy. A definition of this and the post-specific peer model is out of the scope of this thesis. For more details and the proof of Theorem 6.9, refer to [Pei14] and [CK02].

Sender		**Receiver**
Long-term signing and verification key (vk_S, sk_S)		Long-term signing and verification key (vk_R, sk_R)
$(s_S, p_S) \leftarrow wKeyGen(1^\kappa)$	$\xrightarrow{p_S}$	
		$(c,k) \leftarrow wEnCap(p_S)$ $(k_0,k_1) = (F_k(0), F_k(1))$ $t_R \leftarrow \mathcal{O}(k_1, (1, sid, I_R))$ $\sigma_R \leftarrow Sign(sk_R, (1, sid, p_S, c))$
	$\xleftarrow{c,\sigma_R,t_R}$	
$k \leftarrow wDeCap(c, s_S)$, $(k_0,k_1) = (F_k(0), F_k(1))$ $Ver(vk_R, (1, sid, p_S, c), \sigma_R)$ $Vrfy(k_1, (1, sid, I_R), t_R)$ $t_S \leftarrow \mathcal{O}(k_1, (0, sid, I_S))$		
$\sigma_S \leftarrow Sign(sk_S, (0, sid, p_S, c))$ Session key $k_0 = F_k(0)$	$\xrightarrow{c,\sigma_S,t_S}$	$Ver(vk_S, (0, sid, p_S, c), \sigma_S)$ $Vrfy(k_1, (0, sid, I_S), t_S)$ Session key k_0

Figure 8: The Peikert authenticated key exchange protocol

Peikert himself suggested an IND-CPA secure lattice-based KEM. Its description and possible choices of parameters are part of the next subsection.

6.4.1 A Key Encapsulation Mechanism by Peikert

Peikert's KEM uses the functions $rec, dbl, \lfloor\cdot\rceil_2$, and $\langle\cdot\rangle_2$ from Section 5.2, which were first defined by Peikert in 2014.

Public parameters: The KEM depends on the parameters m, n, q, and r. Let $m > 0$ be an integer for the m-th cyclotomic field K_m with ring of integers R and degree $n = \varphi(m)$. Let q be an odd integer which is coprime with every integer dividing m. Furthermore, let $\chi = \lfloor \frac{\hat{m}}{g} \tilde{D}_r \rceil$ be a discretized error distribution , where $\tilde{D}_r$ has probability function $\rho_r(x) = \exp(\frac{-\pi x^2}{r^2})/r$. The factor $\frac{\hat{m}}{g}$ is the translation from R^V to R and $\hat{m}$ equals $\frac{m}{2}$ if m is even and $\hat{m}$ equals m if m is odd. Let $a \xleftarrow{\$} R_q$ be a uniformly sampled polynomial.

Furthermore, we define

$$g = \prod_{odd\ prime\ p|m} (1 - \zeta_p) \in R, \text{ where } \zeta_p = e^{2\pi i/p}.$$

Since q is coprime with every odd prime dividing m, g is coprime with q. Therefore, q is often chosen to be prime such that $q \equiv 1 \mod m$ which implies the coprimality condition.

Key generation: The algorithm samples $s_S, e_S \leftarrow \chi$, computes $p_S = as_S + e_S \in R_q$, and outputs the public-secret key pair $pk = p_S, sk = s_S$.

Encapsulation: On input p_S the algorithm samples $s_R, e_R, e'_R \leftarrow \chi$ and computes the polynomials $p_R = as_R + e_R \in R_q$ and $v = gp_S s_R + e'_R \in R_q$. It derives $c = \langle dbl(v) \rangle_2$ and outputs the ciphertext $C = (p_R, c)$. The session key k is computed by $k = \lfloor dbl(v) \rceil_2 \in R_2$ and also output.

Decapsulation: On input C the algorithm computes $w = gp_R s_S \in R_q$ and outputs the session key $k = rec(w, c)$.

The correctness of the protocol depends on certain parameter conditions given by the following lemma.

Lemma 6.10 (Correctness). *[Pei14] If $\|gs_S\|_2 \leq l$, $\|ge_S\|_2 \leq l$ and $\frac{q^2}{64} \geq (r'^2(2l^2 + n) + \frac{\pi}{2})\omega^2$ for some $\omega > 0$ and with $r'^2 = r^2 + \frac{2\pi rad(m)}{m}$, then the key encapsulation mechanism described above is correct with probability at least $1 - 2n\exp(3\delta - \pi\omega^2)$ for some $\delta \leq 2^{-n}$.*

The radical $rad(m)$ of m is the product of all prime numbers dividing m, i.e.

$$rad(m) = \prod_{p|m,\ p\ prime} p.$$

If m is a power of two, we find that $g = 1$ as there is no odd prime dividing m and hence, the product is empty. For $g = 1$ this scheme is exactly the same as the BCNS protocol from Section 5.2. It is possible to take the same parameters to achieve the same bit security level.

Using the facts that $rad(m)/m \leq 1$, $\|ge_S\|_2, \|gs_S\|_2 \leq \hat{m}(r+1)\sqrt{n}$ except with probability 2^{-n}, and $r'^2 \leq r^2 + 2\pi$, one can calculate that those parameters fulfill Lemma 6.10.

Theorem 6.11. *Peikert's key encapsulation mechanism is IND-CPA secure if the R-DLWE$_{m,n,q,\chi}$ problem is hard given two samples.*

For a proof refer to [Pei14].

6.4.2 The Signature Scheme BLISS

Peikert's key exchange needs a signature scheme, that is existentially unforgeable under chosen message attacks. In this case we choose BLISS (Bimodal Lattice Signature Scheme), introduced by Ducas et al. [DDLL13], which is currently the fastest lattice-based signature scheme satisfying the security condition. Its security is based on the assumption that the *R-SIS* problem is hard.

In the following we describe the scheme, explain the individual steps and the reason for its efficient running time.

The high efficiency is mainly due to a new procedure that samples very efficiently from a Gaussian distribution over $\mathbb{Z}^m$.

We start with a much simpler signature scheme. Assume we choose a verification-signing key pair $pk = (A, T) = (A, AS \mod q) \in \mathbb{Z}^{n\times m} \times \mathbb{Z}^{n\times n}$ and $sk = S \in \mathbb{Z}^{m\times n}$ small. To sign a message M, we sample $y \leftarrow D_{\mathbb{Z}^m,\sigma}$, compute $c = H(Ay \mod q, M)$ and $z = Sc + y$, where H is a hash function with random outputs in $\mathbb{Z}^n$ with small norms. The signature of message M is then given by (z, c). Since y is distributed according to $D_{\mathbb{Z}^m,\sigma}$, we find that $z = Sc + y$ is distributed according to $D_{\mathbb{Z}^m,\sigma,Sc}$. To make sure that the signature does not leak any information about the secret signing key S, we use Theorem 1.2 and obtain that z appears to be distributed according to $D_{\mathbb{Z}^m,\sigma}$. The repetition rate of the rejection sampling process is $M = \exp(1 + \frac{1}{2\tau^2}) \approx \exp(1)$ for $\sigma = \tau\|Sc\|$. Ducas et al. [DDLL13] show that a much lower rejection rate is possible if $z = Sc + y$ is changed to $z = bSc + y$ for $b \xleftarrow{\$} \{-1, 1\}$. This yields that z is distributed according to $\frac{1}{2}D_{\mathbb{Z}^m,\sigma,Sc} + \frac{1}{2}D_{\mathbb{Z}^m,\sigma,-Sc}$. To assure that z appears to be distributed according to $D_{\mathbb{Z}^m,\sigma}$, we find for the same repetition rate $M = \exp(1)$, a lower value $\sigma = \|Sc\|/\sqrt{2}$. We examine an upper bound of this norm below.

To verify a given signature, the verification algorithm checks if $\|z\|_2 \leq B_2 = \eta\sqrt{m}\sigma$ for some $\eta \in [1.1, 1.4)$ (z is distributed according to $D_{\mathbb{Z}^m,\sigma}$, which is concentrated around $\|z\|_2 = \sqrt{m}\sigma$), $\|z\|_\infty \leq \frac{q}{4}$ (this is needed in the security proof), and $c = H(Az - Tc \mod q, M)$ hold.

This verification algorithm does not always verify a correct signature as correct when taking $z = bSc + y$ as above, because

$$Az - Tc = A(bSc + y) - Tc = Ay + bTc - Tc.$$

Hence, for $Ay = Az - Tc \mod q$ to be true, one needs $-Tc = Tc \mod q$, which is only true for $T = 0$ if q is prime.

To overcome this problem, a modulus of $2q$ instead of q is needed and $AS = T = qI \mod 2q$ is required in the key generation algorithm. For more details refer to [DDLL13].

Furthermore, $N_\varsigma(S) = \max_{I\subset\{1,\dots,n\},\#I=\varsigma} \sum_{i\in I} \left(\max_{J\subset\{1,\dots,n\},\#J=\varsigma} \sum_{j\in J} T_{i,j}\right)$, where $T = S^tS$, is an upper bound for $\|Sc\|$, i.e. $\|Sc\|^2 \leq N_\varsigma(S)$ for any real matrix S and $c \in \{0, 1\}^n$ such that exactly ς entries of c are equal to 1. $N_\varsigma(S)$ is used during the key generation to make sure that the number of repetitions in the rejection sampling process does not become too big.

We describe the final signature scheme and explain some steps of the signing algorithm below. Let $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, where n is a power of two and $q \equiv 1 \mod 2n$ is a prime number.

Key generation: The algorithm chooses uniform polynomials $f, g \in R_q$ with exactly d_1 entries in $\{\pm 1\}$ and d_2 entries in $\{\pm 2\}$ and obtains $S = (s_1, s_2)^t = (f, 2g + 1)^t$. If $N_\varsigma(S) \geq 5C^2(d_1 + 4d_2)\varsigma$ resampling takes place, else $a_q = \frac{2g+1}{f} \mod q$ (if f is not invertible restart). The algorithm outputs the verification key $pk = A = (a_1 = 2a_q, q - 2) \in R_{2q}^{1\times 2}$ and the corresponding signing key $sk = S = (s_1, s_2)^t \in R_{2q}^{2\times 1}$.

Signing: The algorithm samples $y_1, y_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$ and computes $u = \zeta a_1 y_1 + y_2 \mod 2q$ and $c = H(\lfloor u\rceil_d \mod p, M)$ for message M. A random bit b is chosen and $z_1 = y_1 + (-1)^b s_1 c, z_2 = y_2 + (-1)^b s_2 c$ is computed. Repeat from the start with probability $1 - \frac{1}{M\exp(-\frac{\|Sc\|^2}{2\sigma^2})\cosh(\frac{\langle z,Sc\rangle}{\sigma^2})}$. Set $z_2^\dagger = (\lfloor u\rceil_d - \lfloor u - z_2\rceil_d) \mod p$ and output the signature $(z_1, z_2^\dagger, c)$ for message M.

Verifying: On input $(z_1, z_2^\dagger, c)$, reject if $\|(z_1|2^d z_2^\dagger)\|_2 > B_2$. If $\|(z_1|2^d z_2^\dagger)\|_\infty > B_\infty$, reject. Accept $(z_1, z_2^\dagger, c)$ as a signature for message M if and only if $c = H(\lfloor \zeta a_1, z_1 + \zeta q c \rceil_d + z_2^\dagger \mod p, M)$.

We find $AS = 2a_q f + (q-2)(2g+1) = 2gq + q$. This equals 0 modulo q, 1 modulo 2 and q modulo $2q$. We saw before that this is exactly what we need to ensure that the verification algorithm verifies correct signatures as correct.

The signing algorithm looks slightly different than explained before. This is due to the fact that we compress the signature, i.e. we drop the low-order bits of z_2. We only give the necessary definitions, for a proof of correctness, refer to [DDLL13].

Let d be the number of bits to be dropped. Then every integer $x \in [-q, q)$ can be uniquely written as $x = \lfloor x \rceil_d 2^d + [x \mod 2^d]$, where $[x \mod 2^d] \in [-2^{d-1}, 2^{d-1})$ are the low order bits. Furthermore, ζ is chosen such that $\zeta(q-2) = 1 \mod 2q$ and $p = \lfloor 2q/2^d \rfloor$. The hash function H uniformly outputs elements in $\{0,1\}^n$ with exactly ς entries equal to 1.

Through the described rejection sampling and dropping high order bits, the signature scheme BLISS obtains its high efficiency in running time as well as signature sizes. Therefore, we instantiate the signature scheme in the Peikert protocol by BLISS.

7 Parameter Selection and Bit Security

To estimate the bit security of the key exchange protocols described above, we use the LWE-Estimator by Albrecht, Player, and Scott [APS15].

Except for BLISS and the JD protocol, all protocols are based on the *R-LWE* problem. Since the introduction of *R-LWE*, the question arises whether the additional structure of ideal lattices can be used to solve an *R-LWE* instance more efficiently than an *LWE* instance. Up to now no such efficient algorithm is applicable to our schemes. Therefore, we use the LWE-Estimator to estimate the hardness of *R-LWE* since *R-LWE* can be seen as an instantiation of *LWE*.

The LWE-Estimator estimates the bit-hardness of the *LWE* problem against four kinds of different attacks: embedding approaches, decoding attacks, the BKW algorithm, and the Arora-Ge algorithm. We give a brief and informal description of those attacks and refer the reader to [APS15] for more details. Moreover, we do not consider the Arora-Ge algorithm in our analysis since it requires a very large number of samples. Therefore, it is not applicable in our case.

The embedding attack, called kannan by the LWE-Estimator, reduces the *LWE* problem to a *uSVP* instance, which is then solved via basis reduction. Basis reduction algorithms are for example BKZ, LLL, enumeration or sieving algorithms. The decoding attack, called dec by the LWE-Estimator, reduces the *LWE* instance to a *BDD* instance. The BKW algorithm solves *LWE* via the *SIS* strategy. That means it solves a *DLWE* instance by finding a short vector v such that $\langle v, a \rangle = 0$ holds.

The bit security depends on the three values n, q, and α of the *LWE* problem, where $\alpha = \sqrt{2\pi}\frac{\sigma}{q}$ for the Gaussian parameter σ in $D_{\mathbb{Z},\sigma}$. For $\alpha q \geq 8$ the continuous Gaussian distribution with $\sigma = \frac{\alpha q}{\sqrt{2\pi}} \geq \frac{8}{\sqrt{2\pi}}$ approximates the discrete Gaussian distribution well [ZZD+14]. Therefore, we choose α and q such that they fulfill this condition.

We determine two sets of parameters for each protocol that satisfy the needed conditions for correctness and for which the LWE-Estimator estimates a bit security level of approximately 100 and 192 bit. All ring-based protocols require that $n = 2^l$ holds. Hence, n is sparsely populated. Therefore, it often is not possible to reach exactly 100 or 192 bit.

For increasing values of q the LWE-Estimator returns a decreasing bit security. Furthermore, higher values of q increase the running time of mathematical operations since the involved numbers are greater. Hence, we keep q as small as possible.

Mostly the parameter restrictions that guarantee correctness give a lower bound for q that depends approximately linearly or quadratic on αq. Hence, choosing αq as small as possible yields the smallest q. As mentioned above, a lower bound for αq is given by 8.

In most protocols the parameter condition for correctness is given by

$$(\alpha q)^x y + z \leq q,$$

where $x \in \{1, 2\}$, y is a variable independent of q and α, and z is small. Increasing the value of α yields a greater right-hand side. That implies a greater q, but q appears with a higher power on the left-hand side of the inequality which results in a contradiction. Hence, increasing the

value of αq does neither yield a higher bit security level nor a faster running time. Therefore, we always use $\alpha = 8/q$.

Hence, there are not many possibilities to change parameters to obtain a bit security level of 100 and 192 bit. We derive and state the best possible approximations.

All estimations by the tool are based on version f69b17a published on November 10th, 2015.

7.1 A detailed Parameter and Bit Security Analysis

Parameters for JD

For the JD protocol the parameter restrictions for correctness are given in Lemma 5.1. Additionally, q has to be an odd prime number. Ding et al. suggest to take the parameters $n = \lambda, q \approx \lambda^4$, and $\sigma = \frac{\lambda}{\sqrt{2\pi}}$, which satisfy this condition for $q \geq 33$:

$$16(\sigma\pi)^2 n = 8\lambda^3 \leq \frac{\lambda^4}{4} - 2 = \frac{q}{4} - 2.$$

Those parameters are chosen to also satisfy the worst-case-to-average-case reduction. Since we do not consider this for the parameters of the other schemes, we do not consider it here either. Hence, we take $\sigma = \alpha q/\sqrt{2\pi}$ with $\alpha = 8/q$ and obtain $q > 32 \cdot 64n + 8$. We reach a bit security level of 100 bit for $n = 411$ and $q = 841741$ and a bit security of 192 bit for $n = 790$ and $q = 1617929$ under the decoding attack.

Parameters for ring-JD

The ring-JD protocol requires $n = 2^l$ and an odd prime q such that $q \equiv 1 \mod 2n$ holds. With $\sigma = \alpha q/\sqrt{2\pi}$ and the parameter conditions for correctness given in Lemma 5.3, we obtain

$$16\pi(n\sigma)^2 = 8(nq\alpha)^2 \leq \frac{q}{4} - 2.$$

Ding et al. proposed the parameter relations $n = \lambda, q \approx \lambda^4$, and $\sigma = \frac{\lambda}{\sqrt{2\pi\lambda}}$. Again those are chosen to satisfy the worst-case-to-average-case reduction and hence, we do not use them. Instead we stick to $\alpha = 8/q$ and obtain

$$q > 2048n^2 + 8, \text{ such that } q \equiv 1 \mod 2n.$$

With the smallest q that fulfills the above conditions, we obtain bit security levels of 76, 150, and 306 bit under the decoding attack (see Table 1). These are not very close to the bit security levels of 100 and 192 bit, but the best we achieve.

Parameters for BCNS

Bos et al. [BCNS14] are the only authors who give explicit parameters (see Section 5.2). The LWE-Estimator estimates a bit security of 145 bit for those parameters. Since the BCNS key exchange protocol is the same as Peikert's KEM from Section 6.4 for $n = 2^l$, we choose parameters according to Lemma 6.10 to guarantee correctness. With $\alpha q = 8$ this yields

$$q > \sqrt{64\omega^2\left(\left(64 + \frac{2\pi}{n}\right)(128n^3 + 2n^2 + n + 32n^{2.5}) + \frac{\pi}{2}\right)},$$

Table 1: Parameters with the bit security under the best attack estimated by the LWE-Estimator for the ring-JD protocol

n	512	1024	2048
q	536881153	2147493889	8589987841
α	$8/q$	$8/q$	$8/q$
Bit Security	76	150	306
under	dec	dec	dec

where $\omega > 0$ influences the probability of incorrectness, which is less than $2n\exp(3 \cdot 2^{-128} - \pi\omega^2)$. With $\omega = \sqrt{\frac{\ln(2n/er)}{\pi}}$, we obtain an error probability of less than the value of parameter er. The parameter q only needs to be odd, not prime. We take $er = 2^{-128}$ and the smallest possible value for q and state the parameter sets with the best achieved bit security in Table 2. We can not get closer to the security levels of 100 and 192 bit. To obtain a higher security level, a smaller q is required and hence, the probability for incorrectness increases.

Table 2: Parameters with the bit security under the best attack estimated by the LWE-Estimator for an error probability of less than 2^{-128} for the BCNS protocol. The parameter ω determines the error probability, which is 2^{-128}.

n	512	1024	2048
q	46565383	131964963	374164875
α	$8/q$	$8/q$	$8/q$
ω	5.52	5.54	5.56
Bit Security	91	180	371
Best Attack	dec	dec	dec

Parameters for LPR

The LPR scheme requires $n = 2^l$ and an odd prime q such that $q \equiv 1 \mod 2n$ holds. Furthermore, the condition $k \leq \sqrt{\frac{q-p}{2(2n+1)pt^2}}$ with $p = 2$ for $\chi = \lfloor N(0, k^2/2\pi) \rceil \approx D_{\mathbb{Z}^n, k/\sqrt{2\pi}}$ restricts possible choices and by definition $k = \alpha q$ holds. With $k \approx 8$ we obtain

$$256(2n+1)t^2 + 2 \leq q.$$

The value $t = \omega(\sqrt{\log_2(n)})$ influences the number of resampling in the scheme. More precisely, the probability that no resampling is required equals $erf(\sqrt{\pi}t)$. Hence, we obtain $t \geq 0.66$ for a probability of 90% and $t \geq 1.032$ for 99%. We try different combinations of n and t, calculate the corresponding q and test the bit security with the LWE-Estimator. The parameters that yield the desired bit security are $n = 512, t = 7$, and $q = 12865537$ for 100 bit and $n = 1024, t = 10$, and $q = 52457473$ for 192 bit under the decoding attack.

Parameters for SS13

In Section 6.2.2 we list the parameter restrictions to guarantee correctness for the SS13 scheme. Let $n = 2^l$ and q be an odd prime such that $q \equiv 1 \mod 2n$. Furthermore, $\sigma \geq 2n\sqrt{\ln(8nq)}q^{\frac{1}{2}+2\epsilon}$ and $k \leq \frac{q}{4n\sigma tp(4p+1)}$ hold, where $t = \omega(\sqrt{\log_2(n)})$ and $k = \alpha q$. Combining both we obtain

$$q > 72 \cdot 8n\sigma t = 576n\sigma t > 1152^2 t^2 n^4 \ln(8nq)$$

with $k = 8$ and $p = 2$. For $n = 512$ this yields $q > 2^{56}$, which is rather inefficient.

Cabarcas, Weiden, and Buchmann analyzed this PKE and the LPR scheme in [CWB14]. Their conclusion supports our hypothesis since they conclude that the LPR scheme is more efficient. Therefore, we do not analyze the SS13 scheme any further.

Parameters for two-pass ZZD

The two-pass ZZD protocol requires $n = 2^l$ and an odd prime q such that $q \equiv 1 \mod 2n$ holds. Lemma 6.7 states that for the correctness of the protocol, $q > 16 \cdot 7\beta^2\sqrt{n}$ with $\beta = \frac{1}{2}\tau\delta^2 n$ has to hold. The parameter $\delta = \alpha q/\sqrt{2\pi}$ denotes the Gaussian parameter. The repetition rate for the rejection sampling is given by $M = \exp(\frac{12}{\tau} + \frac{1}{2\tau^2})$.

With $\tau = 12$, we obtain $M = 2.72$, $n = 1024$, $q = 14186338877441$, $\alpha = 8/q$, and $\beta = 62914.56$. For those parameters the LWE-Estimator estimates a bit security of 100 bit under the decoding attack. For a higher bit security we need $n = 2048$. Since the tool fails to return a result, we restate the estimations given by Zhang et al. [ZZD+14] in Table 3.

Table 3: Parameters with their bit security under the LWE-Estimator for the one-pass ZZD protocol and parameters with their bit security determined by Zhang et al. for the two-pass ZZD protocol. The parameter M is the rejection constant, τ determines M, and δ, β, and γ are parameters of Gaussian distributions.

Protocol	two-pass	two-pass	one-pass	one-pass
n	2048	2048	512	1024
q	$1125899906949121 \approx 2^{50}$	$\approx 2^{47}$	255111169	721563649
δ	3.397	3.397	3.2	3.2
β	$425396.146 \approx 2^{18.7}$	$2^{17.1}$	$31457.28 \approx 2^{14.9}$	$62914.56 \approx 2^{15.9}$
γ	3.397	3.397	3.192	3.192
τ	36	12	12	12
M	1.396	2.728	2.728	2.728
Bit Security	210	230	81	160
estimated by	[ZZD+14]	[ZZD+14]	LWE-Estimator	LWE-Estimator

Parameters for one-pass ZZD

In the one-pass ZZD protocol, q can be chosen β/δ-times smaller than in the two-pass ZZD protocol. Hence,

$$q > 16 \cdot 7\beta\delta\sqrt{n} = 16 \cdot 7(\frac{1}{2}\tau\delta^2 n)\delta\sqrt{n} = 112(\frac{1}{2}\tau 3.2^2 n)3.2\sqrt{n}$$

holds. For different values of n, τ, and the resulting smallest q, it is not possible to get close to a bit security level of 100 or 192 bit. For $n = 512$ we obtain a security level of 81 for $\tau = 12$. Smaller values of τ increase the bit security but also the rejection constant M to a value above 3. This is not practical. For $n = 1024$ we obtain a bit security level between 151 and 160 bit for values of τ between 12 and 36. For $n = 2048$ and $\tau = 36$ the bit security level is 313 bit under the decoding attack. Hence, bit security levels of 81 and 160 bit are the best we achieve. Concrete parameters are stated in Table 3.

Parameters for BLISS

Ducas et al. [DDLL13] state different possible parameter sets for their signature scheme BLISS. Since the LWE-estimator does not return any bit security level estimation for those parameters, we restate three parameter sets given by Ducas et al. [DDLL13] with its estimated security level under a BKZ 2.0 attack simulation. For more details refer to [DDLL13].

In Table 4 the parameter τ determines the repetition constant M and the parameter of the LWE-Estimator, which is usually denoted by α, can be obtained via $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$.

Table 4: Parameters with their bit security determined by Ducas et al. for BLISS.

Bit Security	128	160	190
$m = 2n$	512	512	512
q	12289	12289	12289
σ in $D_{\mathbb{Z},\sigma}$	215	250	271
δ_1, δ_2 as secret key densities($d_i = \lceil \delta_i n \rceil$)	0.3, 0	0.42, 0.03	0.45, 0.06
τ in rejection constant $M = exp(\frac{1}{2\tau^2})$	1	0.7	0.55
ς for N_ς	23	30	39
Secret key resampling constant C	1.62	1.75	1.88
d dropped bits in z_2	10	9	8
B_2, B_∞ as verification bounds	12872, 2100	10206, 1760	9901, 1613

7.2 An Overview of the Parameter Selection and Bit Security

In this section we give an overview in Table 5 and Table 6 of all previously proposed parameters and their bit security for bit security levels of approximately 100 and 192 bit. The bit security level is estimated by the LWE-Estimator or if that is not possible cited from other sources .

The parameter α in Table 5 and Table 6 that is used in the LWE-Estimator stands for the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ with $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$.

Table 5: Overview of the parameters of all protocols for a bit security level of approximately 100 bit.

	JD	ring-JD	BCNS	LPR
n	411	512	512	512
q	$841741 \approx 2^{19.7}$	$536881153 \approx 2^{29}$	$46565383 \approx 2^{25.5}$	$12865537 \approx 2^{23.6}$
α	$\frac{8}{q}$	$\frac{8}{q}$	$\frac{8}{q}$	$\frac{8}{q}$
Bit Security under	100 decoding	76 decoding	91 decoding	100 decoding

	two-pass ZZD	one-pass ZZD	BLISS
n	1024	512	512
q	$\approx 2^{43.7}$	$255111169 \approx 2^{27.9}$	$12289 \approx 2^{13.6}$
α	$\frac{8}{q}$	$\frac{8}{q}$	$\frac{\sqrt{2\pi}215}{q}$
Bit Security under	100 decoding	81 decoding	128 [DDLL13]

Table 6: Overview of the parameters of all protocols for a bit security level of approximately 192 bit.

	JD	ring-JD	BCNS	LPR
n	790	1024	1024	1024
q	$1617929 \approx 2^{20.6}$	$2147493889 \approx 2^{31}$	$131964963 \approx 2^{27}$	$52457473 \approx 2^{25.6}$
α	$\frac{8}{q}$	$\frac{8}{q}$	$\frac{8}{q}$	$\frac{8}{q}$
Bit Security best attack	192 decoding	150 decoding	180 decoding	192 decoding

	two-pass ZZD	one-pass ZZD	BLISS
n	2048	1024	512
q	$\approx 2^{50}$	$721563649 \approx 2^{29.4}$	$12289 \approx 2^{13.6}$
α	$\frac{\sqrt{2\pi}3.397}{q}$	$\frac{8}{q}$	$\frac{\sqrt{2\pi}271}{q}$
Bit Security best attack	210 [ZZD+14]	160 decoding	190 [DDLL13]

8 Theoretical Comparison

The aim of this section is to compare key exchange protocols from a theoretical point of view including the underlying hardness assumptions, similarities, building blocks, and needed cryptographic primitives.

All previously described protocols are described in the two-user setting. However, except of the JD and ring-JD protocols, all protocols are proven to be secure in the multi-user setting. Ding et al. [JD12] propose a multi-user variant, but it is not proven to be secure yet.

Table 7: Overview of the protocols, their underlying hardness assumption, a statement whether the protocol is an AKE or not, the security model in which it is proven to be secure, and the message-passes. For an explanation of the security model abbreviations refer to the list of abbreviations on page 9.

	Hardness Assumption	Is the protocol authenticated?	Security Model	Message -passes
JD	$DLWE$	no	passive PPT	2-pass
ring-JD	$R\text{-}DLWE$	no	passive PPT	2-pass
BCNS	$R\text{-}DLWE$	no	IND-CPA	2-pass
FSXY12	depends on instantiation	yes	CK^+ in StdM	2-pass
FSXY13 with LPR/SS13	$R\text{-}DLWE$	yes	CK^+ in ROM	2-pass
two-pass ZZD	$R\text{-}DLWE$	yes	BR in ROM	2-pass
one-pass ZZD	$R\text{-}DLWE$	yes	auth. CCA	1-pass
Peikert	$R\text{-}DLWE, R\text{-}SIS$	yes	SK	3-pass

We compare eight different lattice-based key exchange protocols. Five of them are authenticated. All key exchange protocols are listed with the underlying hardness assumption, the used security model, and their message-passes in Table 7. As Table 7 indicates, the underlying hardness assumption is always the decisional variant of the learning with errors problem. Only the signature scheme BLISS used in Peikert's AKE relies on the hardness assumption of the SIS problem. The generic protocol by Fujioka et al. [FSXY12] depends on the choice of instantiation. Except for one version of Ding et al.'s protocol, all protocols are defined over rings.

Most protocols need either additional functions or other cryptographic primitives like hash functions, pseudorandom functions, or key encapsulation mechanisms with certain security properties. Additional cryptographic primitives are only used in the authenticated protocols. Figure 9 shows that the five different authenticated key exchange protocols often rely on the same cryptographic primitives. The BCNS protocol can be used as one of the needed cryptographic primitives, namely as an IND-CPA secure KEM.

Furthermore, it can be seen how many different cryptographic primitives are needed in which protocol. The generic protocols, which are not necessarily lattice-based but can be instantiated as such, tend to need more cryptographic primitives in general. Since especially the FSXY12, the FSXY13, and the Peikert protocol are generic with a couple of cryptographic primitives, we give a more detailed overview of those protocols. In Table 8 we decompose the protocols in smaller building blocks and state the quantities for the three different protocols. All three protocols need an IND-CPA secure KEM. Where the two protocols by Fujioka et al. require twice an IND-

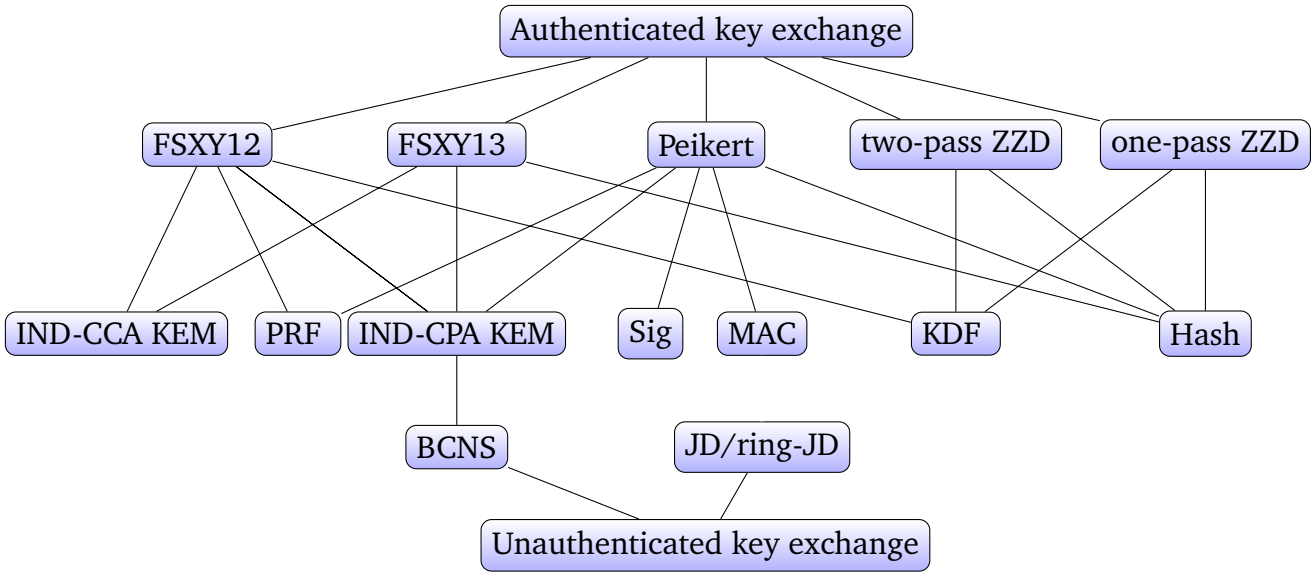

Figure 9: Key exchange protocols and their building blocks, where Sig abbreviates signature scheme and $Hash$ abbreviates hash function. For all other abbreviations refer to the list of abbreviations on page 9.

CCA secure KEM, Peikert's protocol uses a signature scheme twice. This is also visible in the communication bits. The two protocols by Fujioka et al. communicate three ciphertexts while Peikert's protocol replaces two of the ciphertexts by signatures. Additionally, Peikert's protocol uses message authentication codes which contribute to the communication bits. The amount of additional functions such as PRFs, KDFs or hash functions is much higher in the FSXY12 protocol compared to the FSXY13 protocol and Peikert's protocol.

Table 8: Quantities of building blocks and primitives for FSXY12, FSXY13, and Peikert.

Building Block	Quantity in FSXY12	Quantity in FSXY13	Quantity in Peikert
IND-CPA KEM	1	1	1
IND-CCA KEM	2	2	
Sig			2
Hash function		4	
MAC			2
PRF	10		4
KDF	6		
xor	6		
Communication bits	1 public key 3 ciphertexts	1 public key 3 ciphertexts	2 Signatures, 2 MAC tags, 1 public key, 1 ciphertext

Peikert's protocol needs a key encapsulation mechanism that is IND-CPA secure. A possible candidate is stated in the original paper [Pei14]. It is available in an IND-CPA and an IND-CCA secure version and could therefore be a possible candidate for the AKEs by Fujioka et al. from Section 6.1 and 6.2. Peikert's AKE furthermore needs a signature scheme. We propose to

use BLISS. Looking at Table 8, we notice that Peikert's protocol is the only one that explicitly requires a signature scheme.

All three unauthenticated key exchange protocols are very concrete and need no additional cryptographic primitives. The two variants of the authenticated key exchange protocols by Zhang et al. [ZZD+14] are also very concrete. They only need a KDF and an additional hash function.

Furthermore, some protocols share the same needed additional functions. The key exchange by Bos et al. [BCNS14] in Section 5.2 for example uses the same reconciliation function as Peikert in his key encapsulation mechanism from Section 6.4. Having a closer look we actually see that the first one is just a special case of the latter. We notice the same in the authenticated key exchange protocol by Zhang et al. from Section 6.3. The functions Cha and Mod_2 work in the same way as the functions of the key exchange by Ding, Xie, and Lin from Section 5.1.

A decomposition of each protocol in building blocks that are either cryptographic primitives or mathematical operations are given in appendix A together with the quantity of each building block and its running times for the two parameter sets. The running time is the result of a C++ implementation and is discussed in the next section.

9 Running Time Analysis

In this section we compare the running times of the separate operations of the different protocols. The implementation is written in C++ and was run on a server with a 3.60GHz Intel(R) Core(TM) i7-4790 CPU processor with four cores and 8GB RAM. We measure the time of each mathematical operation with the C++ function std::clock() and take the mean of 10000 runs if not stated differently.

9.1 Gaussian Sampling Algorithms

First, we compare the timings of the following four different Gaussian sampling algorithms: rejection sampling [GPV08], inverting the cumulative distribution function (inverse-CDF) [Pei10], the Ziggurat algorithm [BCG+13], and the Knuth-Yao algorithm [DG14]. A description of those methods is out of the scope of this thesis.

We measure the average running time of sampling n elements from the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$. In this case the average is taken over *Nr of runs* runs instead of 10000 runs. In the analyzed protocols the largest value of σ equals 425396 and the smallest value of σ equals 3.19. We test all four Gaussian sampling algorithms for those two values of σ and state the result of the performance test in Table 9. For both values of σ the Knuth-Yao sampling method has the fastest running time. Hence, we choose this sampling method in all protocols.

Table 9: Running times of Gaussian samplers. The times refer to the average running time of sampling n elements from $D_{\mathbb{Z},\sigma}$ over $No.\ of\ runs$ runs. Times are given in ms.

Parameter			Running time			
n	σ	No. of runs	Rej. Sampling	Inverse-CDF	Knuth-Yao	Ziggurat
2048	3.19	10000	5 ms	0.8155 ms	0.2445 ms	2 ms
2048	425396	100	7.5622 ms	4.7646 ms	0.831 ms	3.6264 ms

9.2 Main Mathematical Operations of Key Exchange Protocols

We now analyze the running time of small building blocks or mathematical operations of the different protocols. Therefore, we first give an overview of the measured times for operations that are part of all protocols. We state rudimentary results about those measured running times and compare the running times of different operations as well as the running times of different protocols for the same operation. Afterwards, we turn to a more detailed and separate analysis of some of the protocols.

In Table 10 we state the running times of the four main mathematical operations of each protocol. We list the times for the parameters of a bit security level of 100 and 192 bit. The table contains the three explicit key exchange protocols analyzed in Section 5, both variants of the ZZD protocol and the KEM by Lyubashevsky et al. [LPR12] which is used in the FSXY13 protocol. We obtain the running times of the FSXY13 protocol and Peikert's protocol by adding the running times of their building blocks. This is done in Section 9.4. We refrain from a running time analysis of the FSXY12 protocol since Fujioka et al. [FSXY13] state that the FSXY13 protocol is its more practical version.

The four mathematical operations which are used in each of those six protocols are sampling from the Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ for $\sigma = 3.19$, taking a polynomial or vector times two, multiplying two polynomials, and adding two polynomials. In case of the non-ring version of the JD key exchange we take vector addition in place of polynomial addition and replace the polynomial multiplication by a multiplication of a matrix with a vector. Additionally to those four building blocks, the table contains the times for sampling from the Gaussian distribution $D_{\mathbb{Z}^n,\beta}$. This is part of the two ZZD protocols, which needed the two different Gaussian distributions $D_{\mathbb{Z}^n,\delta}$ and $D_{\mathbb{Z}^n,\beta}$. In this section the parameter δ equals σ.

The times stated in Table 10 are the result of running our C++ implementation and taking the mean of the measured times of 10000 runs. Since almost each operation occurs several times in each protocol, the listed times are again the means of those measured times. Gaussian sampling for variable σ, for example, occurs four times in the key exchange by Bos et al. [BCNS14]. Hence, the stated times 0.101 and 0.192 are the mean of those four measured running times which are already the means of 10000 runs of the protocol. Running times are rounded and given in milliseconds.

Table 10: Running times of mathematical operations. Times are given in ms.

	JD non-ring	JD ring	BCNS	LPR	ZZD two-pass	ZZD one-pass
bit security 100						
n, $\log_2(q)$	411, 19.7	512, 29	512, 25.5	512, 23	1024, 43.7	512, 27.9
Gauss sampling σ	0.099	0.099	0.101	0.140	0.188	0.101
Gauss sampling β	-	-	-	-	0.311	0.159
2 times Poly/Vect.	0.026	0.032	0.043	0.059	0.064	0.030
Poly/Matrix Mult.	3.622	0.862	0.863	0.867	2.373	0.866
Poly/Vector Add.	0.015	0.017	0.021	0.020	0.033	0.017
bit security 192						
n, $\log_2(q)$	790, 20.6	1024, 31	1024, 27	1024, 25.6	2048, 50	1024, 29.4
Gauss sampling σ	0.187	0.191	0.192	0.272	0.377	0.191
Gauss sampling β	-	-	-	-	0.790	0.313
2 times Poly/Vect.	0.049	0.062	0.085	0.118	0.129	0.063
Poly/Matrix Mult.	13.441	1.815	1.819	1.819	5.133	1.811
Poly/Vector Add.	0.030	0.034	0.041	0.039	0.066	0.033

When looking at all running times of Table 10, we immediately see that the slowest operation is the multiplication of a matrix with a vector. Even for relatively small values of n and q the result is already four to five times slower than the times of polynomial multiplication. For bigger values of n and q this difference gets even bigger. Hence, polynomial multiplication is time-wise more efficient than multiplying a matrix with a vector. Nonetheless, polynomial multiplication remains the slowest mathematical operation in each protocol. It is five to ten times slower than Gaussian sampling, which is the second slowest operation. The Gaussian sampling for variable β is slower than the one for variable σ since β is much larger than σ. The fastest operation is the addition of two polynomials or vectors. Sorting those four mathematical operations by their running time, results in the same order for all protocols.

In almost all cases the value of n doubles when increasing the bit security level from 100 to 192 bit. The running times of all mathematical operations also increase by a factor of approximately two.

Gaussian sampling in the LPR scheme is approximately 1.4 times slower than for comparable variable values in other protocols. This is because the times include checking for the necessity of resampling depending on the absolute value of the sampled polynomial (see Section 6.2.1). For our choice of the variable t, we find with our implementation that in no of the 10000 runs of the protocol a resampling took place. Hence, one can think about removing this step to improve the performance of the protocol.

The running time of doubling a vector or polynomial is completely different in different protocols. This is because the doubled polynomials look different in different protocols. In both JD protocols and in both ZZD protocols the polynomials are Gaussian sampled polynomials. In the remaining two protocols the polynomials are uniformly distributed. Therefore, taking a Gaussian sampled polynomial times two seems to be faster than taking a uniformly sampled polynomial times two.

A similar observation can be made for polynomial addition. This operation is slower in the BCNS protocol and the LPR scheme than in the other protocols. In most protocols the polynomial additions are between two polynomials that are statistically indistinguishable from Gaussian sampled polynomials. In the two mentioned protocols, additions between other polynomials are involved as well. In the BCNS protocol, for example, one calculates $2v + e$, where v can be seen as uniformly distributed in R_q^n and e is a polynomial with coefficients in $\{-1, 0, 1\}$. The average running times of polynomial addition between Gaussian sampled polynomials are similar in all analyzed protocols.

For polynomial multiplication the running time increases noticeably for very large values of q. Each protocol additionally computes one or more constants as bounds for other functions. This can be anything from dividing the modulus q by two, subtracting one from a constant involving q, or rounding a fraction. Those operations take between 0.0001 and 0.0002 milliseconds and do not seem to depend significantly on the value of q. Hence, we do not consider them any further.

9.3 Individual Functions and Peculiarities of some Protocols

Aside from the four mathematical operations discussed in the previous section, each protocol contains some special functions, some protocols need uniformly sampled variables, and some use hash functions. The running time of those operations are analyzed in this section. Afterwards, other peculiarities of some of the protocols are stated.

Running times of individual functions of the protocols and sampling uniformly distributed elements

The running times of the additional functions used in each protocol are listed in Table 11 and are rather small. They range between the running time for polynomial addition and approximately half of the running time for Gaussian sampling. As for the previously analyzed operations, the running times increase by a factor of around two when changing from low to high bit security.

The special functions of the LPR scheme have slightly higher running times than the ones of the other protocols. Besides, the two-pass ZZD protocol is fairly fast for the fact that it has much higher values of n and q.

Table 11: Running times of special functions. Times are given in ms.

Protocol	Operation	Time for level 100	Time for level 192
ring JD	Sampling uniform bit string	0.025	0.049
	Function $S(\cdot)$	0.051	0.100
	Function $E(\cdot,\cdot)$	0.037	0.053
BCNS	Sampling uniform polynomial in $\{-1,0,0,1\}$	0.026	0.051
	Crossrounding function $\langle\cdot\rangle_{2q,2}$	0.020	0.039
	Rounding function $\lfloor\cdot\rceil_{2q,2}$	0.016	0.032
	Reconciliation function $rec(\cdot,\cdot)$	0.048	0.094
LPR	Sampling uniform bit string	0.026	0.050
	Polynomial times constant (part of encoding)	0.037	0.074
	Polynomial times two (part of decoding)	0.059	0.118
	Last step of decoding (part of decoding)	0.070	0.140
two-pass ZZD	Function Cha	0.086	0.173
	Function Mod_2	0.034	0.067
one-pass ZZD	Function Cha	0.047	0.093
	Function Mod_2	0.038, 0.066	0.076, 0.132

Hash Function SHA-256

All authenticated key exchange protocols need one or more hash functions. In our case we always instantiate them by SHA-256 and apply the hash function to elements of $\mathbb{Z}_q^{x\cdot n} \times \{0,1\}^{y\cdot n}$, where $x, y \in \mathbb{N}$. The running time depends on the value of x, y, n and q. As expected, the times increase with larger input and hashing elements of $\mathbb{Z}_q$ takes much longer than hashing bits. For example hashing a 1024-bit-string takes 0.01 ms while hashing an element in $\mathbb{Z}_q^{1024}$ takes between 0.035 and 0.05 milliseconds for different values of q.

The non-ring JD protocol

The non-ring based version of the JD key exchange protocol has very small times for the protocol execution without the key generation. This is because it results in just one shared secret bit instead of a shared secret bit string of length n. To make this protocol comparable to the other protocols, one would need to compute n shared secret bits. However, this requires secret key matrices instead of vectors which results in more matrix-vector multiplications. As seen in Table 10, this is by far the most inefficient operation and hence, we conclude that the ring-based version is the faster and more efficient option.

The ZZD protocols

When running the one-pass variant, we notice that the evaluation of the function $Mod_2(\cdot,\cdot)$ takes more time when it is run by the sender than when it is run by the receiver. This is because

the sender needs to convert a polynomial to a vector before evaluating the function. This is not necessary in the run of the receiver since the conversion already took place in the evaluation of the function $Cha(\cdot)$.

Moreover, the running time for multiplying a Gaussian sampled polynomial by two or by a uniformly sampled polynomial, is a bit larger for the larger Gaussian parameter β than it is for the Gaussian parameter σ. For the parameters of a bit security level of 100 for the two-pass variant, the running time of doubling a polynomial is 0.063 ms for parameter σ and 0.066 ms for parameter β. Differences of the same magnitude can be observed for level 192 as well as for both levels of the one-pass variant.

The Peikert protocol

The authenticated key exchange protocol proposed by Peikert [Pei14] consists of several building blocks as stated in Table 8. This means the total running time of this protocol is obtained by adding the running time of the signature scheme BLISS twice to the running time of the KEM. The KEM is similar to the key exchange protocol BCNS. Furthermore, four times the evaluation of a PRF and four times the evaluation of a HashMAC needs to be added. The running time of BLISS is given in Table 12 according to Ducas et al. [DDLL13].

Table 12: Running times for BLISS. Times are given in ms.

Operation	Time for bit security 128	Time for bit security 192
Sign	0.124 ms	0.375 ms
Verify	0.030 ms	0.032 ms

9.4 Overall Running Times

We summarize the most relevant information from the running time analysis before continuing on a bigger scale.

- The order of the running time of the four main mathematical operations from slow to fast is the same for each protocol, namely polynomial multiplication, Gaussian sampling, doubling a polynomial, and adding two polynomials. We note that we do not use any further optimization of polynomial multiplication, such as fast Fourier transforms (FFT) or number theoretic transforms (NTT).
- Each protocol needs some additional functions, which are usually rather fast compared to other mathematical operations.
- Most operations take approximately twice the time when changing to a higher security level, i.e. when doubling n. Usually that factor is a bit smaller than two, except for polynomial multiplication, where it is a bit greater than two.
- The running time of multiplications and additions does depend on the coefficients of the polynomials, but the differences are in most cases very small.

To conclude which protocols are most efficient, we summarize the overall running times and communication bits for each protocol and for a high and low bit security level in Table 13. The overall running time is divided into three running times, which are the key generation and the

protocol of the receiver and the sender. For the unauthenticated key exchange protocols the key exchange is part of each run of the protocol for each party. By contrast, in the authenticated protocols it is only run once and then kept the same for each run of the protocol.

We add the LPR scheme to the three unauthenticated key exchange protocols from Section 5 since it can be seen as a key exchange protocol under certain conditions. As we see in Section 9.3, the non-ring JD protocol is less efficient than its ring-version and hence, not interesting for our comparison. The BCNS protocol is faster and needs to communicate fewer bits than the ring-JD. In the same sense the BCNS protocol is better than the LPR scheme, except from the total time of protocol receiver where the LPR scheme is faster. In fact the LPR scheme is not far off in the rest either and the running time of the key generation process can be improved by dropping the check for the necessity of resampling. Besides, its bit security level is even slightly higher than the one of the BCNS protocol. The downside is relatively many communication bits. Nonetheless, this key exchange can be a good choice as well. The ring-JD protocol has slightly slower running times and its amount of communication bits if neither low nor high. Looking at the bit security level, we find that it is lower than for the other two protocols. With this we conclude that the LPR scheme from Section 6.2.1 and the BCNS protocol from Section 5.2 are the two most efficient unauthenticated protocols.

Our analysis contains four authenticated key exchange protocols from Section 6. The generic FSXY12 protocol is not included since Fujioka et al. state that the FSXY13 protocol is its more practical version [FSXY13]. The one-pass ZZD protocol outperforms the other three clearly in its amount of communication bits. It is faster than the two-pass ZZD protocol but slower than the FSXY13 and the Peikert protocol. However, one important part of Peikert's protocol is the signature scheme BLISS, whose running times are cited from Ducas et al. [DDLL13]. This means that the timings were not obtained on the same server via the same implementation. Since Ducas et al.'s implementation is also a proof-of-concept implementation, we assume that the measured times are still comparable. This assumption is also based on the fact that our running times for the ZZD protocols are comparable to the ones derived by Zhang et al. [ZZD$^+$14]. The two-pass ZZD protocol is much slower than all other protocols. Its running time is, even without counting the time for repetition, three times slower than FSXY13. Also the number of communication bits is much higher for the two-pass ZZD protocol than for the other protocols. As Zhang et al. [ZZD$^+$14] show, the running time of both ZZD protocols can be significantly reduced by choosing parameters for a lower repetition rate. However, this also reduces the bit security. Hence, we conclude that two out of four authenticated key exchange protocols have rather good running times and few communication bits. Since the running time of FSXY13 is approximately two times slower than the one of the Peikert protocol, we conclude that the best results are obtained by Peikert's protocol.

Table 13: Overall running times and communication bits. Times are given in ms. Communication bits are given as the total amount of communication bits.

Protocol			Low Bit Security	High Bit Security
non-ring JD	Bit security		100	192
	Communication bits	$2n\log_2(q)+1$	16180	32590
	KeyGen	total time	3.859	13.895
	Protocol Receiver	total time	0.013	0.023
	Protocol Sender	total time	0.012	0.022
ring JD	Bit security		76	150
	Communication bits	$2n\log_2(q)+n$	30208	64512
	KeyGen	total time	1.108	2.293
	Protocol Receiver	total time	1.123	2.306
	Protocol Sender	total time	1.047	2.158
BCNS	Bit security		91	180
	Communication bits	$2n\log_2(q)+n$	26596	56270
	KeyGen	total time	1.082	2.236
	Protocol Receiver	total time	1.130	2.341
	Protocol Sender	total time	0.943	1.975
LPR	Bit security		100	192
	Communication bits	$3n\log_2(q)$	36276	78780
	KeyGen	total time	1.167	2.402
	Protocol Receiver	total time	1.094	2.266
	Protocol Sender	total time	1.011	2.107
FSXY13	Bit security		100	192
	Communication bits	$4n\log_2(q)$	48368	105040
	KeyGen	total time	1.167	2.402
	Protocol Receiver	total time	5.58	11.53
	Protocol Sender	total time	5.497	11.371
two-pass ZZD	Bit security		100	210
	Communication bits	$2n\log_2(q)+n$	90500	206848
	KeyGen	total time	2.842	6.050
	Protocol Receiver	total time	29.623	41.396
	Protocol Sender	total time	29.364	40.974
one-pass ZZD	Bit security		81	160
	Communication bits	$n\log_2(q)+n$	14810	31157
	KeyGen	total time	1.117	2.288
	Protocol Receiver	total time	2.867	5.963
	Protocol Sender	total time	9.413	19.514
Peikert	Bit security		91	180
	Communication bits	$3n\log_2(q)+$ $2\cdot 256+2\log_2(\sigma)$	50838	96381
	KeyGen	total time	?	?
	Protocol Receiver	total time	2.41	5.056
	Protocol Sender	total time	2.223	4.69

10 Conclusion

Three unauthenticated and five authenticated key exchange protocols are analyzed in this thesis. We include the LPR protocol in our analysis of unauthenticated key exchange protocols since those protocols can be seen as key encapsulation mechanisms under certain conditions. Since Fujioka et al. [FSXY13] already announce themselves that the FSXY13 protocol is the more practical version of the FSXY12 protocol, we only discuss the FSXY13 protocol further. For each protocol we either suggest or quote parameters for bit security levels of approximately 100 and 192 bit, compare them from a theoretical point of view and in terms of their running time. This section gives an overall conclusion and suggests possible future research.

The selected protocols are based on different security models, which makes it hard to compare them in security aspects. A detailed security analysis would be out of the scope of this thesis. Nonetheless, we state a short summary that is based on the original papers of the protocols. Since the unauthenticated JD protocol is not proven secure in the multi-user setting yet, one might prefer the BCNS or the LPR protocol. Both are IND-CPA secure key encapsulation mechanisms. One of the strongest security models for authenticated protocols is the CK^+ security model, which is used in the FSXY13 protocol, but this does not imply that the other AKEs are weaker. Unlike all other AKEs, the one-pass ZZD protocol is weak to replay attacks and does not fulfill wPFS. This is because it has only one message-pass. A more detailed analysis could be part of future research.

We look at the amount of cryptographic primitives involved in each AKE. The protocol by Peikert uses MACs and pseudorandom functions. A disadvantage of the latter is that it lacks an efficient and direct construction from lattice problems [ZZD+14]. The security of the two ZZD protocols and the FSXY13 protocol relies completely on the $R\text{-}LWE$ problem. The advantage of both ZZD protocols is that they are the only protocols that do not use an explicit signature.

The amount of exchanged ring elements is seven for the FSXY13 protocol and thus much higher than for the Peikert, the two-pass ZZD, and the one-pass ZZD protocols, which need three, two, and one, respectively. But the amount of communication bits is still higher in the two-pass version of ZZD than in any other protocol because of the size of q. The one-pass version of ZZD has the fewest communication bits followed by the FSXY13 protocol. In the unauthenticated case the BCNS protocol uses the fewest communication bits.

We analyze the different protocols in terms of their running times on a server. As seen in Table 10 the fastest unauthenticated key exchange is the BCNS protocol, closely followed by the LPR protocol. The latter can be improved by removing the error checking step in the Gaussian sampling procedure. In the authenticated case Peikert's protocol is the fastest protocol followed by the FSXY13 protocol. At the cost of bit security the one-pass ZZD protocol can achieve similar running times to those two protocols by choosing a smaller rejection constant.

To summarize, the LPR and the BCNS protocol are fast and secure unauthenticated key exchange protocols. While the LPR protocol has the potential for faster running times, the BCNS protocol provides fewer communication bits. The conclusion for the authenticated protocols is more complex. The FSXY13 and Peikert's protocol have good running times. The one-pass ZZD protocol has few communication bits and can possibly achieve good running times but is based on a weaker security model since it is a one-pass protocol. The Peikert and the FSXY13 protocol

have more communication bits but are based on a stronger security model. Overall, Peikert's protocol has the fastest running times and is therefore our preferred authenticated key exchange protocol.

For future research it can be interesting to investigate the possibility of using the LPR protocol in Peikert's AKE and Peikert's key encapsulation mechanism BCNS in the FSXY13 protocol. This is because they are both IND-CPA secure key encapsulation mechanisms and perform well in our analysis. Another possible instantiation of IND-CPA secure KEMs is given by the PKE of Linder and Peikert [LP11]. Furthermore, it can be interesting to analyze the new version of the BCNS protocol by Alkim, Ducas, Pöppelmann, and Schwabe [ADPS15].

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai, *Fast cryptographic primitives and circular-secure encryption based on hard learning problems*, Advances in Cryptology - CRYPTO 2009 (Shai Halevi, ed.), Lecture Notes in Computer Science, vol. 5677, Springer Berlin Heidelberg, 2009, pp. 595–618 (English).

[ADPS15] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe, *Post-quantum key exchange - a new hope*, Cryptology ePrint Archive, Report 2015/1092, 2015, `http://eprint.iacr.org/`.

[Ajt96] M. Ajtai, *Generating hard instances of lattice problems (extended abstract)*, Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '96, ACM, 1996, pp. 99–108.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott, *On the concrete hardness of learning with errors*, Cryptology ePrint Archive, Report 2015/046, 2015, `http://eprint.iacr.org/`.

[Ban93] W. Banaszczyk, *New bounds in some transference theorems in the geometry of numbers*, Mathematische Annalen **296** (1993), no. 1, 625–635 (English).

[BAT] SCOTT CHRISTOPHER. BATSON, *On the relationship between two embeddings of ideals into geometric space and the shortest vector problem in principal ideal lattices*, Ph.D. thesis, Raleigh, North Carolina.

[BCG+13] Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden, *Discrete ziggurat: A time-memory trade-off for sampling from a gaussian distribution over the integers*, Cryptology ePrint Archive, Report 2013/510, 2013, `http://eprint.iacr.org/`.

[BCNP08] Colin Boyd, Yvonne Cliff, Juan M. Gonzalez Nieto, and Kenneth G. Paterson, *Efficient one-round key exchange in the standard model*, Cryptology ePrint Archive, Report 2008/007, 2008, `http://eprint.iacr.org/`.

[BCNP09] Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson, *One-round key exchange in the standard model*, IJACT **1** (2009), no. 3, 181–199.

[BCNS14] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila, *Post-quantum key exchange for the tls protocol from the ring learning with errors problem*, Cryptology ePrint Archive, Report 2014/599, 2014, `http://eprint.iacr.org/`.

[Ber09] Daniel J. Bernstein (ed.), *Post-quantum cryptography*, Springer, Berlin [u.a.], 2009.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan, *Fully homomorphic encryption from ring-lwe and security for key dependent messages*, Proceedings of the 31st Annual Conference on Advances in Cryptology (Berlin, Heidelberg), CRYPTO'11, Springer-Verlag, 2011, pp. 505–524.

[CK02] Ran Canetti and Hugo Krawczyk, *Security analysis of ike's signature-based key-exchange protocol*, Cryptology ePrint Archive, Report 2002/120, 2002, `http://eprint.iacr.org/`.

[Con09] Keith Conrad, *The different ideal*, 2009.

[CWB14] Daniel Cabarcas, Patrick Weiden, and Johannes Buchmann, *On the efficiency of provably secure ntru*, Post-Quantum Cryptography (Michele Mosca, ed.), Lecture Notes in Computer Science, vol. 8772, Springer International Publishing, 2014, pp. 22–39 (English).

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky, *Lattice signatures and bimodal gaussians*, Cryptology ePrint Archive, Report 2013/383, 2013, `http://eprint.iacr.org/`.

[DG14] Nagarjun C. Dwarakanath and Steven D. Galbraith, *Sampling from discrete gaussians for lattice-based cryptography on a constrained device*, Appl. Algebra Eng. Commun. Comput. **25** (2014), no. 3, 159–180.

[DR06] T. Dierks and E. Rescorla, *The transport layer security (tls) protocol*, IETF RFC 4346, 2006.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto, *How to enhance the security of public-key encryption at minimum cost*, Public Key Cryptography, Lecture Notes in Computer Science, vol. 1560, Springer Berlin Heidelberg, 1999, pp. 53–68 (English).

[FSXY12] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama, *Strongly secure authenticated key exchange from factoring, codes, and lattices*, Cryptology ePrint Archive, Report 2012/211, 2012, `http://eprint.iacr.org/`.

[FSXY13] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama, *Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism*, Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (New York, NY, USA), ASIA CCS '13, ACM, 2013, pp. 83–94.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '08, ACM, 2008, pp. 197–206.

[JD12] Xiaodong Lin Jintai Ding, Xiang Xie, *A simple provably secure key exchange scheme based on the learning with errors problem*, Cryptology ePrint Archive, Report 2012/688, 2012, `http://eprint.iacr.org/`.

[JS14] Jens Carsten Jantzen and Joachim Schwermer, *Algebra*, 2014.

[KL07] Jonathan Katz and Yehuda Lindell, *Introduction to modern cryptography (chapman & hall/crc cryptography and network security series)*, Chapman & Hall/CRC, 2007.

[LLM06] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio, *On bounded distance decoding for general lattices*, Approximation, Randomization, and Combinatorial Op-

timization. Algorithms and Techniques (Josep Díaz, Klaus Jansen, José D.P. Rolim, and Uri Zwick, eds.), Lecture Notes in Computer Science, vol. 4110, Springer Berlin Heidelberg, 2006, pp. 450–461 (English).

[LM09] Vadim Lyubashevsky and Daniele Micciancio, *On bounded distance decoding, unique shortest vectors, and the minimum distance problem*, Proceedings of CRYPTO 2009 (Santa Barbara, CA, USA), Lecture Notes in Computer Science, vol. 5677, IACR, Springer, August 2009, pp. 577–594.

[LP11] Richard Lindner and Chris Peikert, *Better key sizes (and attacks) for lwe-based encryption*, Proceedings of the 11th International Conference on Topics in Cryptology: CT-RSA 2011 (Berlin, Heidelberg), CT-RSA'11, Springer-Verlag, 2011, pp. 319–339.

[LPR12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, *On ideal lattices and learning with errors over rings*, Cryptology ePrint Archive, Report 2012/230, 2012, `http://eprint.iacr.org/`.

[LPR13] ———, *A toolkit for ring-lwe cryptography*, Cryptology ePrint Archive, Report 2013/293, 2013, `http://eprint.iacr.org/`.

[LS12] Adeline Langlois and Damien Stehle, *Worst-case to average-case reductions for module lattices*, Cryptology ePrint Archive, Report 2012/090, 2012, `http://eprint.iacr.org/`.

[Lyu12] Vadim Lyubashevsky, *Lattice signatures without trapdoors*, Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques (Berlin, Heidelberg), EUROCRYPT'12, Springer-Verlag, 2012, pp. 738–755.

[Mic98] Daniele Micciancio, *The shortest vector in a lattice is hard to approximate to within some constant*, Electronic Colloquium on Computational Complexity (ECCC) **5** (1998), no. 16.

[MR07] Daniele Micciancio and Oded Regev, *Worst-case to average-case reductions based on gaussian measures*, SIAM J. Comput. **37** (2007), no. 1, 267–302.

[MR08] ———, *Lattice-based cryptography*, Post-quantum Cryptography (D. J. Bernstein and J. Buchmann, eds.), Springer, 2008.

[Neu92] Jürgen Neukirch, *Algebraische zahlentheorie*, Springer, Berlin [u.a.], 1992.

[Pei09] Chris Peikert, *Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract*, Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '09, ACM, 2009, pp. 333–342.

[Pei10] ———, *An efficient and parallel gaussian sampler for lattices*, Proceedings of the 30th Annual Conference on Advances in Cryptology (Berlin, Heidelberg), CRYPTO'10, Springer-Verlag, 2010, pp. 80–97.

[Pei14] Chris Peikert, *Lattice cryptography for the internet*, Cryptology ePrint Archive, Report 2014/070, 2014, `http://eprint.iacr.org/`.

[PW07] Chris Peikert and Brent Waters, *Lossy trapdoor functions and their applications*, Cryptology ePrint Archive, Report 2007/279, 2007, `http://eprint.iacr.org/`.

[Reg09] Oded Regev, *On lattices, learning with errors, random linear codes, and cryptography*, J. ACM **56** (2009), no. 6, 34:1–34:40.

[SKR05] I.R. Shafarevich, A.I. Kostrikin, and M. Reid, *Basic notions of algebra*, Encyclopaedia of Mathematical Sciences, Springer Berlin Heidelberg, 2005.

[SS13] Damien Stehlé and Ron Steinfeld, *Making ntruencrypt and ntrusign as secure as standard worst-case problems over ideal lattices*, Cryptology ePrint Archive, Report 2013/004, 2013, `http://eprint.iacr.org/`.

[Win12] *2nd biu winter school*, 2012, `http://crypto.biu.ac.il/2nd-biu-winter-school-lattice-based-cryptography-and-applications`.

[ZZD+14] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen, *Authenticated key exchange from ideal lattices*, Cryptology ePrint Archive, Report 2014/589, 2014, `http://eprint.iacr.org/`.

A Building Blocks and Running Times

Table 14: Building blocks and running times of unauthenticated key exchange protocols. Times are given in ms and refer to the average running time over 10000 runs of one computation of the corresponding building block.

Protocol	Building Block	Quantity protocol/KeyGen	Time for bit security 100	Time for bit security 192
non-ring JD	Gauss Sampling on $\mathbb{Z}^n$	0 / 4	0.099	0.187
	Vector times two	0 / 2	0.026	0.049
	Matrix-Vector Multiplication	0 / 2	3.622	13.441
	Vector Addition	0 / 2	0.015	0.030
	Gauss Sampling on $\mathbb{Z}$	2	0.0019	0.0031
	Number times two	2	0.0007	0.0009
	Scalar Multiplication	2	0.0089	0.0175
	Number addition	2	0.0003	0.0003
	Sampling uniform bit	1	0.0004	0.0005
	Function S	1	0.0003	0.0006
	Function E	2	0.0006	0.0005
ring JD	Gauss Sampling on R_q	2 / 4	0.099	0.191
	Polynomial times two	2 / 2	0.032	0.062
	Polynomial Multiplication	2 / 2	0.862	1.815
	Polynomial Addition	2 / 2	0.017	0.034
	Sampling uniform n-bit string	1	0.025	0.049
	Function S	1	0.050	0.099
	Function E	2	0.037	0.053
BCNS	Gauss Sampling on R_q	1 / 4	0.101	0.192
	Polynomial Multiplication	2 / 2	0.863	1.819
	Polynomial/Vector Addition	1 / 2	0.017	0.034
	dbl	1	0.112	0.221
	$\langle\cdot\rangle_{2q,2}$	1	0.020	0.039
	$\lfloor\cdot\rceil_{2q,2}$	1	0.016	0.032
	Two times p_R	1	0.033	0.065
	rec	1	0.048	0.094
LPR	Gauss Sampling on R_q	5/4	0.14	0.272
	Polynomial Multiplication	4/2	0.867	1.819
	Polynomial/Vector Addition	5/2	0.02	0.039
	Sampling uniform bit string	1	0.026	0.05
	Polynomial times constant	1	0.037	0.074
	Polynomial times two	1	0.059	0.118
	Rounding in decoding	1	0.07	0.14

Table 15: Building blocks and running times of authenticated key exchange protocols. Times are given in ms and refer to the average running time over 10000 runs of one computation of the corresponding building block.

Protocol	Building Block	Quantity protocol/KeyGen	Time for bit security 100	Time for bit security 192
FSXY13	LPR	3	4.439	9.177
	Hash function H_1	2	0.017	0.032
	Hash function H_2	2	0.013	0.023
two-pass ZZD	Gaussian sampling χ_δ	0 / 4	0.188	0.377
	Gaussian sampling χ_β	$2 + 4M$	0.311	0.790
	Polynomial times two	2+2M / 2	0.064	0.129
	Polynomial Multiplication	6+6M / 2	2.373	5.133
	Polynomial Addition	4+6M / 2	0.033	0.066
	Hash function H_1 on R_q	1+M	0.050	0.108
	Hash function H_1 on R_q^2	1+M	0.096	0.213
	Cha	1	0.086	0.173
	Mod_2	2	0.034	0.067
	$KDF\ H_2$ on $R_q^2 \times \{0,1\}^{2n}$	2	0.109	0.239
one-pass ZZD	Gaussian sampling χ_δ	1 / 4	0.101	0.191
	Gaussian sampling χ_β	2M+1	0.159	0.313
	Polynomial times two	2+M / 2	0.030	0.063
	Polynomial Multiplication	4+3M / 2	0.866	1.811
	Polynomial Addition	3+3M / 2	0.017	0.033
	Hash function H_1 on R_q	1+M	0.019	0.035
	Cha	1	0.047	0.093
	Mod_2	2	0.038, 0.066	0.076, 0.132
	$KDF\ H_2$ on $R_q \times \{0,1\}^{2n}$	2	0.025	0.049
Peikert	Pseudorandom function	4	0.006	0.01
	BCNS	1	4.237	8.788
	BLISS	2	0.154	0.407
	MAC	4	0.016	0.026