Post-Quantum Cryptography: Lattice Based Cryptography

Antonia Fally

2020/21

Contents

1 Introduction

There has been a substantial amount of research on the development of Quantum Computers in recent years. Those machines prove to be way more powerful than traditional computers, in fact they pose a significant threat to today's cryptography systems.
The three core cryptographic functionalities (public key encryption, key exchange and digital signatures) rely heavily on either the integer factorization problem (RSA encryption, for example), the discrete logarithm problem or the elliptic-curve discrete logarithm problem.
The security depends on the difficulty of the theoretic problems and derives from the fact that currently there exists no efficient algorithm to solve them and thus making the full decrpytion infeasible. [4]

Motivation

In 1994 Peter Shor developed a quantum algorithm (called *Shor's algorithm* [13]) that could efficiently factor a number n in polynomial time and therefore breaking the RSA encryption scheme.
Interestingly using a classical algorithm on a quantum computer, and therefore using the quantum computers higher speed to break an encryption in the brute-force manner, wouldn't work. It is possible to try all divisors, but due to the nature of quantum computers, when measuring the outcome of the computation, we would end up with a random possible divisor and with high probability not the right one.
Instead Shor's algorithm works in two parts:
The first part consists of turning the factoring problem into the order finding problem.
The second part uses a quantum computer to solve the order-finding problem. [3] [13]

Therefore, if large scale quantum computers are ever build, they would be able to break essentially all cryptographic systems currently in use. When this will happen is a question not easily answered. While in the past it wasn't clear if large quantum computers are even possible to build, experts nowadays believe that it is merely an engineering challenge and some even predict that it is a matter of only twenty-or-so years. [4]

The good news is that there has also been an increase of research and testing of quantum-resistant cryptography methods or *post quantum cryptography*.
The main families for which post quantum cryptosystems have been proposed are: [4]

- Lattice-based cryptography
- Multivariate-based cryptography
- Hash-based cryptography
- Code-based cryptography

Furthermore there exists also a variety of other systems which do not belong in foresaid categories and that could withstand a quantum computer attack, however there has not been done a lot of research (yet) on their security. For example: The isogeny problem on supersingular curves. [4]

In this paper we will focus on the first approach: Lattice-based cryptography.

2 Basic Definitions

2.1 Vector Space

Let V be a non-empty set with two operations and K a field of scalars:

i) **Vector Addition**: This assigns to any $u, v \in V$ a *sum* $u + v$ in V

ii) **Scalar Multiplication**: This assigns to any $u \in V, k \in K$ a *productku* $\in V$

Then V is called a *vector space* (over the field K) if the following axioms hold for any vectors $u, v, v \in V$:

1. $(u + v) + w = u + (v + w)$
2. There is a vector in V, denoted by 0 and called the *zero vector*, such that, for any $u \in V$,

$$u + 0 = 0 + u = u$$

3. For each $u \in V$, there is a vector in V, denoted by $-u$, and called the *negative* of u, such that

$$u + (-u) = (-u) + u = 0$$

4. $u + v = v + u$
5. $k(u + v) = ku + kv$, for any scalar $k \in K$
6. $(a + b)u = au + bu$, for any scalars $a, b \in K$
7. $(ab)u = a(bu)$, for any scalars $a, b \in K$
8. $1u = u$, for the unit scalar $1 \in K$

[6]

2.2 Lattice

A lattice L is like the vector subspace V, but given a set of basis vectors $\{b_1, \dots b_m\}$ instead of taking the real linear combinations of the b_i we are only allowed to take the *integer* linear combinations of the b_i,

$$L = \left\{ \sum_{i=1}^{m} a_i \cdot b_i : a \in \mathbb{Z} \right\} = \{B \cdot a : a \in \mathbb{Z}^m\}$$

The vectors b_i are called a *basis* of the lattice L. [10]

Any given lattice can have many bases. It's important to note that bases that are formed by vectors almost orthogonal to each other (called *short basis*) are much more useful for solving hard lattice problems.

3 Lattice Problems

3.1 Shortest Vector Problem (SVP)

Find the shortest non-zero vector in the lattice
Approximate Shortext Vector Problem SVP_γ:
find a vector within a factor γ of the shortest [8]

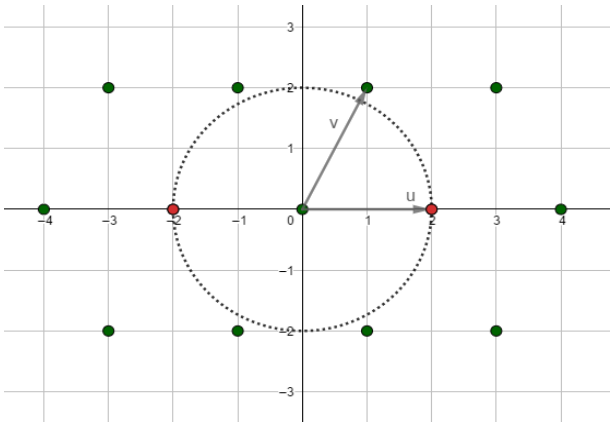


Figure 1: SVP Problem

3.2 Shortest Independent Vector Problem SIVP

Given a basis of L and an approximation factor $\gamma \geq 1$, find a linear independent set $\{y_1, ..., y_d\}$ such that $max_i \mid\mid y_i \mid\mid \leq \gamma\lambda_d(L)$ where $\lambda_i(L) = \min\{\max\{\mid\mid x_1 \mid\mid, ..., \mid\mid x_i \mid\mid\} \mid x_1, ..., x_i \in L$ are linearly independent [5]

3.3 Closest Vector Problem (CVP)

Definition: Given a point in $\mathbb{R}^n$, find the lattice point closest to it [8]

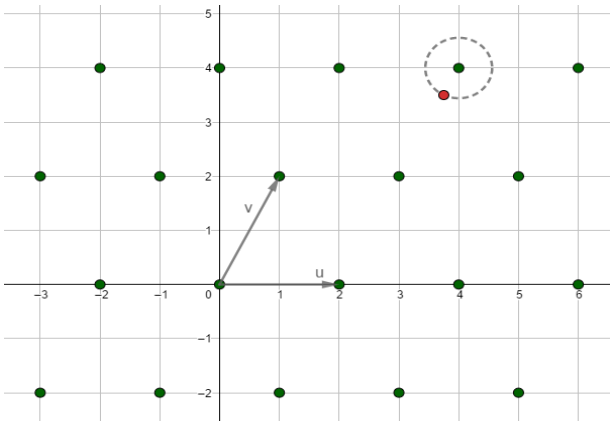


Figure 2: CVP Problem

3.4 Small Integer Solution (SIS)

Given a modulus q, a matrix A (mod q) and a $\mathcal{V} < q$, find $y \in \mathbb{Z}^m$ such that $Ay \equiv 0$ (mod q) and $\mid\mid y \mid\mid \leq \mathcal{V}$ [5]

3.5 Learning With Errors (LWE)

The Learning With Errors problem was introduced by Oded Regev in 2005. [12]

Definition

[12] The LWE problem asks to recover a secret s $\in \mathbb{Z}_q^n$ given a sequence of "approximate" random linear equations on s.
For instance, the input might be

$$13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \text{ (mod 17)}$$
$$6s_1 + 10s_2 + 13s_3 + 1s_4 \approx 3 \text{ (mod 17)}$$
$$10s_1 + 4s_2 + 12s_3 + 16s_4 \approx 12 \text{ (mod 17)}$$

$$9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$$
$$3s_1 + 6s_2 + 4s_3 + 5s_4 \approx 16 \pmod{17}$$
$$\vdots$$
$$6s_1 + 7s_2 + 16s_3 + 2s_4 \approx 3 \pmod{17}$$

3.6 Ring Learning With Errors (RLWE)

The LWE cryptographic method has some issues. One of them is the size of the secret key, another one - as we saw above - is that it takes a whole vector to encrypt one single bit.
Due to the difficulty of solving the RLWE problem, even for quantum computers, it is a promising candidate to be the base of future public key cryptographic methods.

Definition

In Ring-LWE we take "special" matrices A, which arise from polynomial rings. Consider the ring of polynomials $R = \mathbb{Z}[X]/F(X)$ where $F(X)$ is an integer polynomial of degree n. The Ring R is the set of polynomials of degree less than n with integer coefficients, addition being standard polynomial addition and multiplication being polynomial multiplication followed by reduction modulo $F(X)$. [10]

[The RLWE method is a slightly modified version of the Learning With Error method and the RLWE problem is the larger LWE problem.]

4 Hardness of Lattice Problems

4.1 Hardness of SVP and CVP

In 1981 it was shown by van Emde Boas that CVP is NP-hard. Since then the hardness assumption for CVP was strengthened by proving that even finding approximate solutions to CVP is hard. However, proving whether SVP is hard was progressing way slower. It remained an open question almost two decades. Ajtai proved that the SVP is hard for NP under reverse *unfaithful random reductions* (RUR-reductions).
RUR-reductions are probabilistic reductions that map NO instances to NO instances with probability 1 and YES instances to YES instances with non-negligile probability.
Furthermore it was shown that approximating SVP is NP-hard. [9]

4.2 Hardness of the SIS problem

A long sequence of works has established progressively stronger results about the hardness of the SIS problem relative to worst-case lattice problems.

Theorem

For any $m = poly(n)$, any $\beta > 0$, and any sufficiently large $q \geq \beta \cdot poly(n)$, solving $\text{SIS}_{n,q,\beta,m}$ with non-negligible probability is at least as hard as solving the decisional approximate shortest vector problem GAP-SVP_γ and the approximate shortest independent vector problems $\text{SIV}P_\gamma$ (among others) on arbitrary n-dimensoinal lattices (i.e., in the worst case) with overwhelming probability, for some $\gamma = \beta \cdot poly(n)$ [11]

4.3 Hardness of the LWE problem

It is known that LWE is hard based on certain assumptions regarding the worst-case hardness of standard lattice problems such as GAP-SVP (the decision problem of the $SVP\gamma$) and SVIP.
In 2005 Oded Regev showed a quantum reduction from those hard lattice problems to the LWE problem. [12]

Theorem

For any $m = poly(n)$, any modulus $q \leq 2^{poly(n)}$, and any (discretized) Gaussian error distribution χ of parameter $\alpha q \geq 2\sqrt{n}$ where $0 < \alpha < 1$, solving the decision-$\text{LWE}_{n,q,\chi,m}$ probkem is at least as hard as quantumly solving GAP-SVPγ and SIVPγ on arbitrary n-dimensional lattices, for some $\gamma = \tilde{O}(n/\alpha)$. [11]

4.4 Hardness of the R-LWE problem

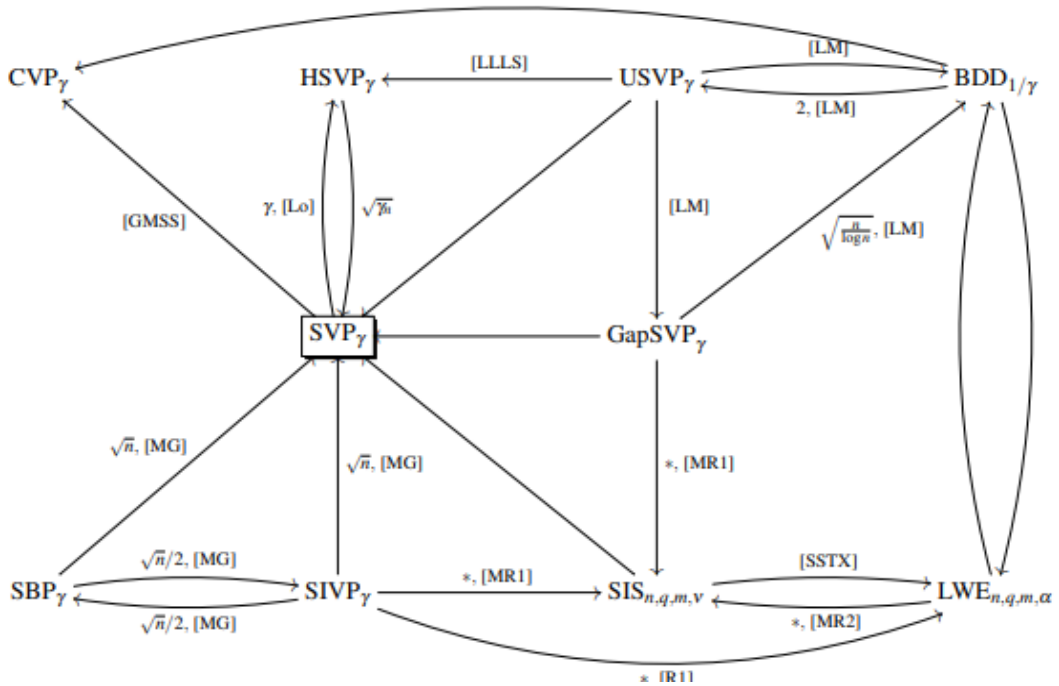
In 2006 Stehlé, Steinfeld, Tanaka and Xayawa showed a quantum reduction from the SIS problem to the LWE problem. [14]
They concluded that hardness decreases with the number of equation and approaches zero for more than q^2 equations.
In 2013, however, Regev, Peikert and Lyubashevsky describe the distribution of the error probability necessary for the independence of the crytosystem hardness from the number of equations. With that they showed the persistence of hardness. [7]

Theorem

For any $m = poly(n)$, cyclotomic ring R of degree n (over $\mathbb{Z}$), and appropriate choices of modulus q and error distribution χ of error rate $\alpha < 1$, solving the R-LWE$_{q,\chi,m}$ problem is at least as hard as quantumly solving the SVP$_\gamma$ problem on arbitrary ideal lattices in R, for some $\gamma = poly(n)/\alpha$. [11]

4.5 Hardness of Lattice Problems Overview

The graphic below shows the relation between the main hard lattice problems. [5]



An arrow from A to B indicates that there exists a polynomial reduction from problem A to problem B.

5 Cryptosystems based on Lattice Problems

There are multiple and various cryptosystems that are based on lattice problems or on their hardness assumption.
Here we are going to take a look at two examples:

- Multi-bit public key encryption with Learning With Errors
- Ring-Learning With Errors Public Key Exchange

5.1 Multi-bit public key encryption with LWE

The goal is to encrypt (and then decrypt) a 4-bit binary value or a number in the range from 0 to 7.

Let's start by rewriting the LWE problem:

$$A \cdot s + e = B \pmod{q}$$

where q is a prime number, s is a one-dimensional matrix representing the secret (private) key, A will be a two-dimensional and B a one-dimensional matrix. The values of A and B are known and they work as public keys. Also e is a one-dimensional matrix that represents the errors or *noise* added.

On a side note: Why is the noise e so important?
Consider the equation without e

$$A \cdot s = B$$

The task is - like before - recovering the secret value of s.
Sadly this system of linear equations is easily solved by the *Gaussian Elimination* algorithm and we can do this in polynomial time.
Therefore the noise e is crucial.

First we create the value for the private key s, then we select the values for the public key A and the error e randomly and lastly we will calculate the second public key B as seen above.
Next we choose a message bit M, which we want to encrypt and transmit.

Encryption

For the encryption itself we take samples from A and B and calculate the two following values

$$u = \sum A_{samples} \ \ (mod \ \ q)$$
$$v = \sum (B_{samples}) + \frac{q}{2} \ \ (mod \ \ q)$$

The encrypted value of M is (u, v) and this is the value that will be transmitted.

Decryption

After receiving the message, we can start decrypting the message in the following way:

$$Dec = v - s \cdot u \ \ (mod \ \ q)$$

$$M = \begin{cases} 0 & if \ \ Dec < \frac{q}{2} \\ 1 & if \ \ Dec > \frac{q}{2} \end{cases}$$

The binary number that is received that way is the binary representative of the original integer message.

Python

5.2 Ring-LWE Public Key Exchange (RLWE-KEX)

Let's look at how a public key exchange based on the RLWE problem between two person Alice and Bob would look like. First we need to establish a value n (a complexity value) which will be the highest co-efficient power to be used.
From that calculate:

$$q = 2^n - 1$$

The values of the coefficients will be limited by $q - 1$, meaning they all will be smaller or equal to $q - 1$. Furthermore all polynomial operations will be conducted with a modulus of q. Then create first public key A of Alice.

$$A = a_{n-1}x^{n-1} + ... + a_2x^2 + a_1x + a_0$$

In order to **not** obtain any powers bigger than n during the calculations and to always stay within certain boundaries, we need to divide A by $(x^n + 1)$. This procedure also gives us the Ring structure.

$$A = \frac{a_{n-1}x^{n-1} + ... + a_2x^2 + a_1x + a_0}{x^n - 1}$$

Additionally create the polynomials for e and s. They will be corresponding to Alice's public key.

$$e_A = e_{n-1}x^{n-1} + ... + e_2x^2 + e_1x + e_0$$

$$s_A = s_{n-1}x^{n-1} + ... + s_2x^2 + s_1x + s_0$$

And as seen before, calculate b_A.

$$b_A = A \cdot s_A + e_A$$

Alice now shares A with Bob, so that Bob can create his own public key. For that he selects his own e and s polynomials:

$$e_B = \bar{e}_{n-1}x^{n-1} + ... + \bar{e}_2x^2 + \bar{e}_1x + \bar{e}_0$$

$$s_B = \bar{s}_{n-1}x^{n-1} + ... + \bar{s}_2x^2 + \bar{s}_1x + \bar{s}_0$$

and calculates b_B with help of Alice's A:

$$b_B = A \cdot s_B + e_B$$

Now they exchange the public keys b_A and b_B.
With b_B from Bob, Alice can now calculate her shared key

$$shared_A = \frac{b_B \cdot s_A}{x^n + 1}$$

Bob now does the same with b_A to obtain his shared key

$$shared_B = \frac{b_A \cdot s_B}{x^n + 1}$$

The next task is to extract the noise from the received values and create an array k that will store the values of the shared key.

$$\begin{cases} if \;\; shared_{Ai} < \frac{q}{4}, & k_i = 0 \\ else \;\; if \;\; shared_{Ai} < \frac{q}{2} & k_i = 1 \\ else \;\; if \;\; shared_{Ai} < \frac{3q}{4} & k_i = 0 \\ else \;\; if \;\; shared_{Ai} < q & k_i = 1 \end{cases}$$

Bob does the same with $shared_B$ and if everything was calculated correctly then Alice and Bob end up with two equal k arrays and thus have created their shared secret key.

6 New Hope

New Hope is quantum resistant key exchange method defined by Erdem Alkim, Leo Ducas, Thomas Poeppelmann and Peter Schwabe in 2017. [1]
It is based on the RLWE problem and was submitted to the NIST-post-quantum crypto project where it made it into Round 2 of the standardization process, but was eventually not selected into Round 3.

The method is defined as: [1]

Parameters: $q = 12289 < 2^{14}$, $n = 1024$ Error distribution: ψ_{16}		
Alice (server)		**Bob (client)**
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \mathsf{Parse}(\mathsf{SHAKE\text{-}128}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \psi_{16}^n$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \xleftarrow{\$} \psi_{16}^n$
$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \mathsf{Parse}(\mathsf{SHAKE\text{-}128}(seed))$
		$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{bs}' + \mathbf{e}''$
$\mathbf{v}' \leftarrow \mathbf{us}$	$\xleftarrow{(\mathbf{u}, \mathbf{r})}$	$\mathbf{r} \xleftarrow{\$} \mathsf{HelpRec}(\mathbf{v})$
$\nu \leftarrow \mathsf{Rec}(\mathbf{v}', \mathbf{r})$		$\nu \leftarrow \mathsf{Rec}(\mathbf{v}, \mathbf{r})$
$\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$		$\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$

Initially Alice generates a 256-bit value called *seed*, then she uses the Shake-128 hashing method to create the polynomial coefficients a.
Next we generate the secret values s, s' and error values e, e' and e''.
Now we calculate b:

$$b = a \cdot s + e$$

seed and b_A are then passed to Bob and with help of the *seed* value and the hash function Shake-128, Bob can now create a.
Bob now computes the values u, v and r in the following manner:

$$u = a \cdot s' + e'$$

$$v = b \cdot s' + e''$$

$$r = HelpRec(v)$$

where the $HelpRec$ function is defined as:

$$\mathsf{HelpRec}(\mathbf{x};b) = \mathsf{CVP}_{\tilde{D}_4}\left(\frac{2^r}{q}(\mathbf{x}+b\mathbf{g})\right) \bmod 2^r,$$

where $b \in \{0,1\}$ is a uniformly chosen random bit.

And CVP defined as:

Algorithm 1 $\mathsf{CVP}_{\tilde{D}_4}(\mathbf{x} \in \mathbb{R}^4)$

Ensure: An integer vector $\mathbf{z}$ such that $\mathbf{Bz}$ is a closest vector to $\mathbf{x}$: $\mathbf{x} - \mathbf{Bz} \in \mathscr{V}$

1: $\mathbf{v}_0 \leftarrow \lfloor \mathbf{x} \rceil$
2: $\mathbf{v}_1 \leftarrow \lfloor \mathbf{x} - \mathbf{g} \rceil$
3: $k \leftarrow (\|\mathbf{x} - \mathbf{v}_0\|_1 < 1)\ ?\ 0 : 1$
4: $(v_0, v_1, v_2, v_3)^t \leftarrow \mathbf{v}_k$
5: **return** $(v_0, v_1, v_2, k)^t + v_3 \cdot (-1,-1,-1,2)^t$

The values u and r are then passed to Alice. From those Alice computes the value v' by

$$v' = u \cdot s$$

From that Alice calculates the shared value $\mathcal{V}$:

$$\mathcal{V} = Rec(v', r)$$

Similarly, Bob computes $\mathcal{V}$:

$$\mathcal{V} = Rec(v, r)$$

where Rec is defined as:

$$\mathsf{Rec}(\mathbf{x},\mathbf{r}) = \mathsf{Decode}\left(\frac{1}{q}\mathbf{x} - \frac{1}{2^r}\mathbf{Br}\right)$$

and *decode* as:

Algorithm 2 Decode($\mathbf{x} \in \mathbb{R}^4/\mathbb{Z}^4$)

Ensure: A bit k such that $k\mathbf{g}$ is a closest vector to $\mathbf{x}+\mathbb{Z}^4$:
$\mathbf{x} - k\mathbf{g} \in \mathcal{V} + \mathbb{Z}^4$

1: $\mathbf{v} = \mathbf{x} - \lfloor \mathbf{x} \rceil$
2: **return** 0 if $\|\mathbf{v}\|_1 \leq 1$ and 1 otherwise

The function Rec() computes one key bit from a vector x with 4 coefficients in $\mathbb{Z}_q$ and a reconciliation vector $r \in \{0,1,2,3\}^4$.

Lastly the value of the shared key μ is obtained by the SHA-256 hash value of $\mathcal{V}$.

References

[1] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25Th {USENIX} security symposium ({USENIX} security 16)*, pages 327–343, 2016.

[2] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[3] Stephanie Blanda. `https://blogs.ams.org/mathgradblog/2014/04/30/shors-algorithm-breaking-rsa-encryption/`, 2014.

[4] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[5] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. *IACR Cryptol. ePrint Arch.*, 2012:533, 2012.

[6] Seymour Lipschutz and Marc Lipson. *Schaum's outline of theory and problems of linear algebra*. Erlangga, 2001.

[7] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013.

[8] Prabhakaran Manoj. `https://www.cse.iitb.ac.in/mp/teach/advcrypto/f17/slides/19.pdf`, 2017.

[9] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM journal on Computing*, 30(6):2008–2035, 2001.

[10] Smart Nigel P. *Cryptography Made Simple*. Springer, 2016.

[11] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.

[12] Oded Regev. The learning with errors problem. *Invited survey in CCC*, 7(30):11, 2010.

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

[14] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 617–635. Springer, 2009.