Review Article

Post–Quantum Security: Origin, Fundamentals, and Adoption

Johanna Barzen and Frank Leymann*

University of Stuttgart, IAAS, Universitätsstr. 38, 70569 Stuttgart, Germany

https://www.engineegroup.us

Check for updates

Abstract

Based on Shor's quantum algorithm for computing discrete logarithms, powerful enough quantum computers will break current cryptographic protocols. While today's quantum computers are not yet capable enough of running this algorithm successfully, it is expected by many experts that they will be capable enough in the foreseeable future. Thus, actions must be taken to create a new infrastructure that protects society and companies against such attacks. In this contribution, we describe the backgrounds necessary to comprehend these actions.

We first describe the relation between discrete logarithms and two well-known asymmetric security schemes, RSA and Elliptic Curve Cryptography. Next, we present the foundations of lattice-based cryptography which is the basis of schemes that are considered to be safe against attacks by quantum algorithms (as well as by classical algorithms). Then we describe two such quantum-safe algorithms (Kyber and Dilithium) in more detail. Finally, we give a very brief and selective overview of a few actions currently taken by governments and industry as well as standardization in this area. The article has a pedagogical character, not presenting any new research results. Especially it strives towards being self-contained, e.g. the required mathematical foundations to understand post-quantum cryptography are provided and examples are given. Thus, a reader interested in getting a first comprehensive overview of the subject doesn't have to consult text books or several research papers.

1. Introduction

Shor's algorithm [1] for prime factorization is a major breakthrough in quantum computing: It allows to break RSA-based [2] as well as elliptic curve-based cryptosystems exponentially faster than the best-known classical algorithms. Thus, it is a severe threat to the established security infrastructure. However, the quantum resources required to run Shor's algorithm (and thus to break today's security) have been considered at that time to be available in some decades only (if at all). This resulted in awareness of the problem itself but positioning it as a problem that must be dealt with in the far future. These days it seems likely that the required quantum resources will be available in the next few years. Thus, today's security infrastructure is in jeopardy forcing policy makers and industry to urgently focus on this problem.

Prime factorization can be reduced to the period finding of the module exponential function (section 3.1). Quantum computers are able to fast determine global properties of functions like their periods [3], which is exploited in Shor's algorithm by using the quantum Fourier transformation (section 3.2). Effectively, this way discrete logarithms are computed (section 2.5). Computing discrete logarithms also allows the computation of private keys used in elliptic

106

curve cryptography (section 2.6). Thus, a security scheme is needed that cannot be broken with discrete logarithms and that has not been cracked (yet).

Lattice-based cryptography is such a scheme. A lattice is the set of all integer combinations of linear independent vectors (section 4.1) or, equivalently, the set of all integer solutions of linear equation systems. While linear equations can be solved efficiently with classical methods, adding a small random error term to the equation system results in a hard problem (section 4.8) referred to as Learning with Errors. It is this problem that is the underpinning of algebraic lattice cryptography (chapter 5). Furthermore, instead of solving the Learning with Errors problem in vector spaces, modules over rings of residue classes of polynomials are used (section 5.3).

Based on these structures, an asymmetric encryption/decryption mechanism can be defined (section 5.4) as well as a means for signatures (section 5.5). These two mechanisms are recognized to be quantum-safe, i.e. neither classical algorithms nor quantum algorithms are known that can break these security schemes. Because of this, they are standardized as the basis for a future quantum-safe infrastructure (section 6.4).

The importance of becoming quantum-safe has been recognized by several government organizations, who initiated efforts to move today's infrastructure to a post-quantum secure one (section 6.3). This way, vital data like health data, military secrets, etc should be kept confidential. To protect confidential data, companies must become active too, and the time needed for corresponding efforts can be estimated by Mosca's inequation (section 6.2). The corresponding transition is backed-up by open-source software implementing the standardized post-quantum security algorithms (chapter 7).

1.1. Structure of the article

The article is structured as follows: in the first part, we give an overview of fundamental concepts and algorithms underlying today's security infrastructure. The second part sketches roughly a few actions of policy makers. While the first part (sections 2 to 5) contains timeless information, the much shorter second part (sections 6 and 7) is somehow volatile presenting information that might be outdated in a few years and that has to be digested by the reader with corresponding care.

2. Classical asymmetric cryptography

In this section, we first remind a few definitions and facts from number theory related to modular arithmetic [4]. Next, we provide the basics of RSA and elliptic curves which are the underpinnings of today's asymmetric cryptography [5]. Especially, their relation to discrete logarithms is presented.

2.1. Modular arithmetic

A couple of prominent algorithms in cryptography like RSA (section 2.4) or DiffieHellman (section 2.6) are based on modular arithmetic. In this section, we sketch its basics.

For $x \in \mathbb{R}$ let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x (also called the *floor* of x). Then, for $a \in \mathbb{Z}$ and $n \in \mathbb{N}$ (i.e. especially $n \neq 0$), the *remainder* of a divided by n is given by

$$a - \left\lfloor \frac{a}{n} \right\rfloor n$$

and this remainder is denoted by "a mod n" (pronounced "a modulo n"). This is captured by the following

Definition 1: The function

mod: $\mathbb{Z} \times \mathbb{N}^+ \rightarrow \mathbb{N}_0$

$$(a, n) \mapsto a - \left\lfloor \frac{a}{n} \right\rfloor n =: a \bmod n \tag{1}$$

is called *modulo function*.

The definition directly implies the following note:

Note 1: Let $a \in \mathbb{Z}$ and $n \in \mathbb{N}^+$. Then:

$$r = a \bmod n \Leftrightarrow \exists\, k \in \mathbb{Z} : a = kn + r \wedge 0 \leq r < n \tag{2}$$

In case $a \bmod n = b \bmod n$, a and b give the same remainder when being divided by n. This is an important relation between the two numbers a and b that deserves a separate definition:

Definition 2: Let $a, b \in \mathbb{Z}$ and $n \in \mathbb{N}^+$. Then:

$$a \equiv b \pmod{n} :\Leftrightarrow a \bmod n = b \bmod n \tag{3}$$

In this case, a and b are called *congruent* modulo n.

Congruence "$\equiv$" is an equivalence relation, i.e. it is reflexive, symmetric, and transitive [4]. Note that while "$a \equiv b \pmod{n}$" is a Boolean statement (i.e. it is true or false for given a, b, n), "$a \bmod n$" is a natural number (i.e. the remainder of a divided by n).

Definition 3: Let $m \in \mathbb{Z}$ and $n \in \mathbb{N}^+$. The notation $n \mid m$ is defined as follows:

$$n \mid m :\Leftrightarrow \exists\, k \in \mathbb{Z} : m = kn \tag{4}$$

n is called a *divisor* of m (aka n *divides* m aka n is a *factor* of m). The notation $n \nmid m$ means that n does not divide m. In case $n \neq 1$ and $n \neq m$, n is called a *proper divisor* of m.

The next note follows immediately from the definitions before:

Note 2:

$$a \equiv b \pmod{n} \Leftrightarrow n \mid (a - b) \qquad (5)$$

Proving the congruence of two numbers (especially of large numbers) can be very tedious. The following often simplifies related computations:

Principle (Reduce First, Then Compute): Assume that $(a_1a_2 + a_3a_4) \bmod n$ has to be computed, and let $a_i \equiv b_i \bmod n$ for $1 \leq i \leq 4$. Then:

$$(a_1a_2 + a_3a_4) \bmod n = (b_1b_2 + b_3b_4) \bmod n$$

The proof follows directly from the definitions.

This principle allows us to reduce an arithmetic expression that might be cumbersome to compute (in our case $a_1a_2 + a_3a_4$) first and then use an arithmetic expression that might be easier to compute (in our case $b_1b_2 + b_3b_4$) to determine the remainder divided by n.

Example [4]: $(39 \cdot 99 + 95 \cdot 64) \bmod 19$ should be computed. Now, it is $39 \equiv 1 \bmod 19$, $99 \equiv 4 \bmod 19$, $95 \equiv 0 \bmod 19$, and $64 \equiv 7 \bmod 19$. Thus, we can substitute 39 by 1, 99 by 4, 95 by 0, and 67 by 7 and result in

$$(39 \cdot 99 + 95 \cdot 64) \equiv (1 \cdot 4 + 0 \cdot 7) \equiv 4 \bmod 19 = 4$$

which is much simpler than computing $(39 \cdot 99 + 95 \cdot 64)$ first and then finding the remainder divided by 19.

Another example is computing $127^4 \bmod 5$. It is $127 \equiv 2 \bmod 5$. Thus, it is

$$127^4 \equiv 127 \cdot 127 \cdot 127 \cdot 127 \equiv 2 \cdot 2 \cdot 2 \cdot 2 \equiv 16 \bmod 5 = 1$$

In our final example, we want to compute 129537 mod 9. This can be done by using a decimal polynomial representation of 129537 and observing $10 \equiv 1 \bmod 9$:

$$129537 = 1 \cdot 10^5 + 2 \cdot 10^4 + 9 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0$$

$$\equiv 1 \cdot 1^5 + 2 \cdot 1^4 + 9 \cdot 1^3 + 5 \cdot 1^2 + 3 \cdot 1^1 + 7 \cdot 1^0$$

$$\equiv 1 + 2 + 9 + 5 + 3 + 7$$

$$\equiv 27$$

$$\equiv 0 \bmod 9$$

$$= 0$$

Thus, the principle "reduce first then compute" is a fundamental principle in *modular arithmetic* to make computations easier.

2.2. Diophantine equations

Several algorithms in cryptography require solving *linear Diophantine equations*, i.e. equations of the form

$$ax + by = c \qquad (6)$$

with given $a, b, c \in \mathbb{N}$ and where $x, y \in \mathbb{Z}$ (i.e. the solution of the equation) have to be determined. Deciding whether or not a linear Diophantine equation has a solution requires determining the greatest common divisors.

Definition 4: $n \in \mathbb{N}^+$ is called a *common divisor* of $a, b \in \mathbb{Z}$ iff both, $n \mid a$ and $n \mid b$. The *greatest common divisor* of a and b is denoted by $\gcd(a, b)$. a and b are called *coprime* iff $\gcd(a, b) = 1$.

The next Theorem provides a criterion for the solvability of a linear Diophantine equation.

Theorem 1: Let $a, b, c \in \mathbb{N}$. Then: $(\exists\, x, z \in \mathbb{Z} : ax + by = c) \Leftrightarrow \gcd(a, b) \mid c$

Proof: [4].

The famous *Euclidian Algorithm* that is the basis for computing solutions of linear Diophantine equations is given by

Theorem 2: Let $a, b \in \mathbb{N}$. Define $r_0 := a$, $r_1 := b$ and $r_{i+1} := r_{i-1} \bmod r_i$ for $i \geq 1$. Then there exists an $N \in \mathbb{N}$ such that $r_N = 0$ and $r_{N-1} = \gcd(a, b)$.

Proof: [4].

The (extended) Euclidian Algorithm is often applied by using a certain schema when writing down the computation steps (see [4] section 3.2 for the details).

As an example, we apply this schema to compute the greatest common divisor $\gcd(63,17)$ of the numbers 63 and 27 (the numbering of the equations in needed in the next example):

$$63 = 3 \cdot 17 + 12 \quad \text{(d)}$$

$$17 = 1 \cdot 12 + 5 \quad \text{(c)}$$

$$12 = 2 \cdot 5 + 2 \quad \text{(b)}$$

$$5 = 2 \cdot 2 + 1 \quad \text{(a)}$$

$$2 = 2 \cdot 1 + 0$$

Thus, $\gcd(63,17) = 1$, i.e. 63 and 17 are coprime.

Within Theorem 1, the direction "⇐" is called *Bézout's Identity*. Because of its importance, it is stated as a separate lemma.

Lemma 1: Let $a, b \in \mathbb{N}$ and let $g = \gcd(a, b)$. There exist $x, y \in \mathbb{Z}$ such that

$$g = ax + by \qquad (7)$$

The factors x and y are called *Bézout coefficients* for $(a,$

b) and can be determined based on the Euclidian algorithm. For this purpose, the equations of the computations performed according to the Euclidian algorithm are applied "from bottom to top" beginning with the last equation, i.e. the one containing the gcd (see [4] section 3.2 for the details).

As an example, we use the computation of gcd (63,17) before: Using equation (a) results in

$$1 \overset{(a)}{=} 5 - 2.2$$

Based on equation (b), it is $2 \overset{(b)}{=} 12 - 2.5$, which is substituted in the equation before (and so on):

$$1 \overset{(a)}{=} 5 - 2 \cdot 2$$
$$\overset{(b)}{=} 5 - 2 \cdot (12 - 2 \cdot 5) = 5 \cdot 5 - 2 \cdot 12$$
$$\overset{(c)}{=} 5 \cdot (17 - 1 \cdot 12) - 2 \cdot 12 = 5 \cdot 17 - 7 \cdot 12$$
$$\overset{(d)}{=} 5 \cdot 17 - 7 \cdot (63 - 3 \cdot 17 = 5 \cdot 17 - 7 \cdot 63 + 21 \cdot 17$$
$$= 26 \cdot 17 - 7 \cdot 63$$

Thus, the Bézout coefficients for (63, 17) are (−7, 26).

This proceeding can be used to solve an equation important for RSA (section 2.4 (5)) following the proof of

Note 3: Let $a, m \in \mathbb{N}$ be coprime, i.e. gcd$(a, m) = 1$. Then, there exists an $x \in \mathbb{Z}$ such that $ax \equiv 1 \bmod m$.

Proof: Because of lemma 1, there exist $x, y \in \mathbb{Z}$ such that $1 = ax + my$. This implies that $ax - 1 = -my$. According to definition 3, this means $m \mid (ax - 1)$. With note 2 we get $ax \equiv 1 \bmod m$. ■

2.3. Factorization

One of the key algorithms in cryptography is based on the factorization of numbers (see section 2.4). Because of this, we are reminded of the basics of factorization. Factorization means to determine the composition of an integer of prime numbers.

Definition 5: A number $p \in \mathbb{N}$ with no proper divisor is called a *prime number*. The set of all prime numbers is denoted by $\mathbb{P}$.

Every natural number can be represented as a product of prime numbers, and this representation is unique up to the order of the prime numbers in the product: this is the famous *Fundamental Theorem of Arithmetic*. Determining the prime numbers of this representation of a number is referred to as its *factorization*.

Theorem 3: Let $n \in \mathbb{N}$ with $n > 1$. There exist $p_1, ..., p_k \in \mathbb{P}$ ($p_i \neq p_j$ for $i \neq j$) and $n_1, ..., n_k \in \mathbb{N}$ such that

$$n = p_1^{n_1} \cdot p_2^{n_2} \cdot ... \cdot p_k^{n_k} \tag{8}$$

There are many known classical algorithms to compute the factorization of a given number. None of these known algorithms is efficient in the sense that it can compute the factorization in polynomial time. But is has not been proven that a polynomial algorithm cannot exist, nor has it been proven that such an algorithm does exist. Thus, it is open whether or not factorization can be done efficiently. In what follows, we will make use of the *Lemma of Euclid*.

Lemma 2: Let $p \in \mathbb{P}$ and $a, b \in \mathbb{N}$. If $p \mid ab$ then $p \mid a$ or $p \mid b$. ■

Assume $n \in \mathbb{N}$ with $n \mid ab$ and let $n = p_1^{n_1} \cdot p_2^{n_2} \cdot ... \cdot p_k^{n_k}$ the factorization of n. Because

$$ab = mn = mp_1^{n_1} \cdot p_2^{n_2} \cdot ... \cdot p_k^{n_k} = \left(mp_1^{n_1} \cdot ... \cdot p_i^{n_i-1} \cdot ... \cdot p_k^{n_k} \right) \cdot p_i \text{ for}$$

a suitable $m \in \mathbb{N}$, it is $p_i \mid ab$ for each $1 \leq i \leq k$. According to Euclid's lemma before, it is $p_i \mid a$ or $p_i \mid b$ for $1 \leq i \leq k$. Thus, each such p_i is a common divisor of n as well as of a or b. This proves the following corollary:

Corollary 1: Let $n \in \mathbb{N}$ with $n \mid ab$. Then: $\exists\, m \in \mathbb{N} : m \mid n \wedge (m \mid a \vee m \mid b)$.

Especially, if n is not prime, m is a proper divisor of n. ■

Note, that in general $n \mid ab$ does not imply that $n \mid a$ or $n \mid b$. For example: $6 \mid 12 = 3 \cdot 4$ but neither $6 \mid 3$ nor $6 \mid 4$; but of course, 3 is a common divisor of 6 and 3, and 2 is a common divisor of 6 and 4.

2.4. Rivest−Shamir−Adleman cryptosystem

Rivest–Shamir–Adleman (RSA) described a public-key cryptosystem that is based on the assumed hardness of factorization of natural numbers [2]. Public and private keys are determined as follows:

1. Choose two large prime numbers $p, q \in \mathbb{P}$

2. Compute $n = pq$

3. Compute $\varphi(n) = (p - 1)(q - 1)$

4. Choose a small $g \in \mathbb{N}$, with $\varphi(n)$ and g being coprime (i.e. gcd$(\varphi(n), g) = 1$)

5. Determine the solution d of the equation $dg \equiv 1 \bmod \varphi(n)$ (see Note 3)

6. Destroy $p, q, \varphi(n)$

(d, n) is the *public key*, and g is the *private key*.

The following examples are from [2]. Choose the prime number $p = 47$ and $q = 59$ (which are obviously not "large" but fine for the example). Then $n = p \cdot q = 2773$ and $\varphi(n) = 46 \cdot 58 = 2668$. The private key is chosen as $g = 157$ which is valid because $\gcd(2668,157) = 1$, i.e. g and $\varphi(n)$ are coprime. Next, the equation $d \cdot 157 \equiv 1 \bmod 2668$ is solved according to Note 3 by computing the Bézout coefficients for $(2668,157)$ as exemplified in the example after Lemma 1: the solution is $d = 17$. Thus, $(17,2773)$ is the public key.

A message μ is encrypted as follows:

1. μ is mapped to a natural number m

   a. E.g. each letter is substituted by a two-digit number like Blank=00, A=01, B=02, etc., and the numbers corresponding to the letters of the message are concatenated.

   b. The resulting number m must be less than n. If $m \geq n$, the digits of m are split into blocks $m_1,…, m_k$ such that $m_i < n$.

2. Each block is encrypted as $\widehat{m} = m_i^d \bmod n$

3. The encrypted message is the concatenation $\widehat{m} = \widehat{m_1} \cdots \widehat{m_k}$

Example: Let μ = ITS ALL GREEK TO ME. By mapping this message to a natural number as sketched in (1a) and considering (1b) we get m = 0920 1900 0112 1200 0718 0505 1100 2015 0013 0500, i.e. m has been split into blocks of digits the corresponding numbers are less than $n = 2773$. The first block m_1 corresponds to the number 920, thus, according to (2) the encryption of this block results in $\widehat{m}_1 = 920^{17} \bmod 2773 = 948$; the second block corresponds to 1900, i.e. $\hat{m}_2 = 1900^{17} \bmod 2773 = 2342$, etc. Finally, the encrypted message is according to (3) the concatenation of the encrypted blocks: $\hat{m} = 0948\ 2342…$

Decryption is achieved by

4. decrypting each block $\widehat{m}_i$ of the encrypted message m separately

5. by computing $m_i = (\widehat{m}_i)^g \bmod n$

6. and concatenating the decrypted block resulting in the decrypted message $m = m_1 \cdots m_k$

Example: In the encrypted message $\hat{m}$ from above, it is $\hat{m}_1 = 948$, i.e. according to (5) it is $m_1 = (\hat{m}_1)^{157} \bmod 2773 = 984^{157} \bmod 2773 = 920$. With (1a) 920 corresponds to IT (i.e. 09 = I, 20 = T) and so on. Concatenating the decrypted block results in the original message.

Cracking a private key means to compute g. While g is secret, i.e. unknown, d as part of the public key is known. To compute g, the equation $dg \equiv 1 \bmod \varphi(n)$ must be solved which requires in addition to d to know $\varphi(n)$. This, in turn, means that $\varphi(n) = (p - 1) \cdot (q - 1)$ must be known. Because the RSA procedure is well-known, it is further known that n is the product of two prime numbers. Consequently, n must "just" be factorized to get $\varphi(n)$ and then solve $dg \equiv 1 \bmod \varphi(n)$. Since factorization is assumed to be hard, RSA is considered hard to crack, i.e. RSA is considered to be secure (ignoring Shor's algorithm for now).

2.5. Modular exponential function and discrete logarithm

Factorization is closely related to the following function and its inverse, respectively:

Definition 6: For a given $n \in \mathbb{N}$ choose an arbitrary number $a \in \mathbb{N}$ with $0 < a < n$.

The function

$$\exp_a : \mathbb{N}_0 \to \mathbb{N}_0 \text{ with } \exp_a(x) = a^x \bmod n \qquad (9)$$

is called *modular exponential function* with basis a (and modulo n).

As an example, consider the first part of the value table of the modular exponential function with $n = 5$ and basis 2, i.e. :

x	0	1	2	3	4	5	6	7	…
$\exp_2(x)$	1	2	4	3	1	2	4	3	…

As can be seen, it is $\exp_2(x + 4) = \exp_2(x)$. I.e. the function $\exp_2(\cdot)$ with $n = 5$ seems to be periodic with period 4. These terms are defined next.

Definition 7: A function $f : \mathbb{N}_0 \to \mathbb{N}_0$ is called *periodic* $:\Leftrightarrow$ $\exists\, t \in \mathbb{N}\, \forall\, x \in \mathbb{N}_0 : f(x + t) = f(x)$. The smallest such number t is called the *period* of f.

The following is a well-known fact (a corollary of Fermat's Little Theorem) about the periodicity of the modular exponential function:

Lemma 3: The modular exponential function $\exp_a(x) = a^x \bmod n$ is periodic if and only if $\gcd(n, a) = 1$. ∎

Let $\gcd(n, a) = 1$, thus, $\exp_a(x)$ is periodic with period $p \in \mathbb{N}$. It is then $\exp_a(0) = \exp_a(0 + p) = \exp_a(p)$ which in turn is equivalent to $a^p \bmod n = a^0 \bmod n = 1 \bmod n$. By Definition 2, this is equivalent to $a^p \equiv 1 \pmod{n}$, and by Note 2 we know that $n \mid (a^p - 1)$, i.e. n is a divisor of $a^p - 1$. This is summarized by

Note 3: Let $a, n \in \mathbb{N}$ be coprime. $\exp_a(x) = a^x \bmod n$ has period $p \in \mathbb{N}$ if and only if $a^p \equiv 1 \pmod{n}$, i.e. $n \mid (a^p - 1)$. ∎

Let $n \mid (a^p - 1)$ and assume $p \in \mathbb{N}$ is even. Then, $n \mid (a^{p/2} - 1)(a^{p/2} + 1)$, and according to Corollary 1, there is an $m \in \mathbb{N}$ with $m \mid n$ and $m \mid (a^{p/2} - 1)$ or $m \mid (a^{p/2} + 1)$. Thus, n and $(a^{p/2} - 1)$ have a common divisor, or n and $(a^{p/2} + 1)$ have a common divisor. This implies that $\gcd((a^{p/2} - 1), n)$ or $\gcd((a^{p/2} + 1), n)$ is a divisor of n.

This means that if the period of $\exp_a$, $0 < a < n$, can be determined, and the period is even, a divisor of n can easily be computed. And by iteration, the factorization of n results. Before discussing the frequency of this situation, we give the following

Definition 8: The smallest number $y \in \mathbb{N}$ with $a^y \equiv x \bmod n$ (and $0 < a < n$) defines a map

$$\log_a: \mathbb{N}_0 \to \mathbb{N}_0 \text{ with } x \mapsto y \qquad (10)$$

called *discrete logarithm* (of x) with basis a (and modulo n).

As an example, it is $2^4 = 16 \equiv 5 \bmod 11$, with implies $\log_2 5 = 4$ (modulo 11). The usual equations like $\exp_a(\log_a z) = z$ and $\log_a(\exp_a y) = y$ apply the discrete logarithm and modular exponential function.

Computing the period p of $\exp_a$ means to determine $p \in \mathbb{N}_0$ with $a^p \equiv 1 \pmod{n}$ (see Note 3), i.e. to compute the discrete logarithm $p = \log_a 1$. This means that factorization - and, thus, cracking RSA - can be reduced to compute discrete logarithms! But computing discrete logarithms is assumed to be classically hard, just like factorization. Similarly, cracking cryptography based on elliptic curves can be reduced to computing discrete logarithms (see section 2.6).

Remember that the assumption we made in order to determine the period $p \in \mathbb{N}$ of $\exp_a$, $0 < a < n$, is that p is even. Under this assumption, $\gcd((a^{p/2} - 1), n)$ or $\gcd((a^{p/2} + 1), n)$ is a divisor of n. First of all, the question is how often the period is even. Next, the question is whether $\gcd((a^{p/2} - 1), n)$ or $\gcd((a^{p/2} + 1), n)$ are proper divisors of n. The first observation is:

Note 4: $\gcd((a^{p/2} - 1), n) \neq n$.

Proof: Let $\gcd((a^{p/2} - 1), n) = x$. Then, we find $y, z \in \mathbb{N}$ with $zx = a^{p/2} - 1$ and $yx = n$. Thus, $a^{p/2} - 1 = zx = z \cdot n / y = n \cdot z / y$. Assume $x = n$, thus, $y = 1$. This means that $a^{p/2} - 1 = nz \Rightarrow a^{p/2} = nz + 1 \Rightarrow a^{p/2} \equiv 1 \bmod n$. According to Note 3, this means that the period of $\exp_a$ is $p/2$ - which is a contradiction. This proves the claim. ∎

It may still be the case that $\gcd((a^{p/2} - 1), n) = 1$, i.e. that $\gcd((a^{p/2} - 1), n)$ is not a proper divisor of n. In case

$\gcd((a^{p/2} - 1), n) = 1$, Corollary 1 implies that $\gcd((a^{p/2} + 1), n) \neq 1$. Still, it may be that $\gcd((a^{p/2} + 1), n) = n$, and, thus, $\gcd((a^{p/2} + 1), n)$ may not be a proper divisor of n.

Consequently, the bad situation $\gcd((a^{p/2} - 1), n) = 1$ and $\gcd((a^{p/2} + 1), n) = n$ may occur. However, according to Nielsen and Chuang (see [3], Theorem A4.13) the following is known:

Note 5: For more than half of the numbers $0 < a < n$ with $\gcd(n, a) = 1$, the period p of $\exp_a$ is even, and $\gcd((a^{p/2} - 1), n) \neq 1$ or $\gcd((a^{p/2} + 1), n) \neq n$. ∎

Thus, after a couple of attempts a number a will be found such that $\exp_a$ has an even period and that $\gcd((a^{p/2} - 1), n)$ or $\gcd((a^{p/2} + 1), n)$ is a proper divisor of n.

How the period of $\exp_a$ can be determined efficiently is the subject of section 3.

2.6. Elliptic curve cryptography

Elliptic curve cryptography can also be broken by efficient computations of discrete logarithms on quantum computers. In this section, we briefly introduce the underpinnings of elliptic curve cryptography and sketch its relationship to discrete logarithms. For more details about elliptic curves as well as the proofs of the facts given in this section [6].

Note, that an elliptic curve is not an ellipsis. However, the name results from computing the arc length of an ellipsis by means of integrals. With proper parametrization, the integrands of the corresponding integrals contain polynomial functions as used in the definition of the following set C:

Definition 9: Let $C := \{(x, y) \in \mathbb{R}^2 \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{N}\}$ and $a, b \in \mathbb{R}$ with $4a^3 + 27b^2 \neq 0$. Then C is called an *elliptic curve.*

The reason why $\mathcal{N}$ ("the infinitely far point") is added will be clear soon. The condition $4a^3 + 27b^2 \neq 0$ excludes repeated roots that would result in singularities like intersections or cusps of the curve. Figure 1 depicts three elliptic curves to give an impression of their possible shapes.

An elliptic curve C is obviously symmetrical to the x-axis, i.e. for each point $P \in C$ on the curve, its reflection across the x-axis is again a point on the curve, which is denoted by $P^{-1} \in C$. This allows us to define a multiplication $\circ : C \times C \to C$ on a curve (Figure 2) as follows.

1. Let $P \neq Q \in C$ be two distinct points on the curve C and let the line $\overline{PQ}$ intersect the curve C in the point $Z : \overline{PQ} \cap C = Z$. The reflection Z^{-1} of Z across the x-axis is defined as the product of P and Q: $P \circ Q := Z^{-1}$

Figure 1: Sample Shapes of Elliptic Curves.

Figure 2: Operations on an Elliptic Curve.

2. gent to C at R, and let W be the intersection of the tangent with C: $g_R \cap C = W$. The reflection W^{-1} of W across the x-axis is defined as the product of R with itself: $R^2 := W^{-1}$.

3. Let $S \in C$ be a point whose tangent does not intersect C. For this purpose, the curve C is extended by a point at infinity $\mathcal{N}$, and the product of S with itself is this point: $S^2 := \mathcal{N}$.

4. Let $T \in C$ and $T^{-1} \in C$ its reflection across the x-axis. The line connecting T and T^{-1} is parallel to the y-axis and has no intersection with the curve. The product of T and $T-1$ is defined as the infinitely far point $\mathcal{N}$: $T \circ T^{-1} := \mathcal{N}$.

With respect to this operation $\circ$, the reflection P^{-1} of a point $P \in C$ across the x-axis is the inverse element of P, and the infinitely far point $\mathcal{N}$ is revealed as the neutral element of the operation. The use of elliptic curves in cryptography is based on this operation and the corresponding.

Theorem 3: Let C be an elliptic curve. Then $(C, \circ)$ is a commutative group with the neutral element $\mathcal{N}$. ∎

For use of an elliptic curve in a cryptosystem the parameters $a, b \in \mathbb{R}$ of the curve are published as well as a point $G \in C$ called the *generator*. Each participant has a *private key* $s \in \mathbb{N}$ and $G^s =: P \in C$ as the corresponding *public key*. The following example shows how elliptic curve encryption works:

A message μ is encrypted as follows:

1. μ is mapped to a point $M \in C$, e.g.:

    a. μ is considered as a bit string $m \in \mathbb{N}$

    b. Compute y with $y^2 = m^3 + am + b$

    c. Then $M = (m, y) \in C$

2. Chose a random $k \in \mathbb{N}$

3. Compute $E_1 = G^k$ and $E_2 = M \circ P^k$

    • G is the published generator, s is the private key to compute $P = G^s$

4. The encrypted message is $\chi(m) = (E_1, E_2)$

An encrypted message $\chi(m)$ is decrypted as follows:

5. Compute $M = E_2 \circ (E_1)^{-s}$

    • It is $(E_1)^s = (G^k)^s = (G^s)^k = P^k$

    • Thus, it is $E_2 \circ (E_1)^{-s} = M \circ P^k \cdot (E_1)^{-s} = M \circ P^k \circ P^{-k} = M$

6. The decrypted message is $m = \pi_1(M)$

Note, that in practice Elliptic Curve Cryptography (ECC) is not used for encryption of persistent data but for key exchange and signatures. For example, elliptic curve key exchange (a.k.a. ECC Diffie-Hellman) works as follows:

1. As before, let $G \in C$ be the generator of the curve

2. Alice has $G^{s_A} =: P_A \in C$ as the public key, with $s_A \in \mathbb{N}$ as the private key

3. Bob has $G^{s_B} =: P_B \in C$ as the public key, with $s_B \in \mathbb{N}$ as the private key

4. Alice and Bob exchange P_A, P_B

5. Alice computes $K_A = P_B^{s_A}$

6. Bob computes $K_B = P_A^{s_B}$

    • It is $K_A = P_B^{s_A} = (G^{s_B})^{s_A} = (G^{s_A})^{s_B} = P_A^{s_B} = K_B$

    • Thus, Alice and Bob share the same secrete key $K (= K_A = K_B)$

7. Alice and Bob can now use the shared secret key for symmetric encryption A public key is of the form P

= G^s with the publicly known generator $G \in C$. This means that determining the private key s ("cracking elliptic curve cryptography") means computing $s = \log_G P$ where the exponential $\exp_G$ is computed based on "∘" as multiplication. Consequently, cracking elliptic curve cryptography is also reduced to compute discrete logarithms.

Note, that elliptic curves relevant for cryptography are curves whose coefficients are from a finite field $\mathbb{F}_p$ instead of coefficients from $\mathbb{R}$. The principles sketched before remain valid, i.e. this complication is irrelevant for a first treatment and understanding.

3. Shor's algorithm

The algorithm of Shor [1] for factorization is a hybrid algorithm, i.e. it consists of classical computations as well as quantum computations. The same paper describes a modification of the algorithm for computing discrete logarithms. Overall, both algorithms provide an exponential speedup compared with the best-known classical algorithms for solving these two problems. In this section, we very briefly sketch how numbers are factorized with Shor's algorithm.

3.1. The outer classical part

The classical computations can be split into an "outer" classical part that surrounds both, the quantum part (section 3.2) as well as an "inner" classical part (section 3.3). The outer part computes the greatest common divisors and decides whether to iterate the overall algorithm or to stop with a divisor of the number to be factorized.

Let $n \in \mathbb{N}$ be the number to be factorized (n can be assumed to be odd because otherwise its factor "2" is immediately known). Randomly, a number $a \in \mathbb{N}$, $1 < a < n$ is chosen. In the case of $\gcd(a, n) \neq 1$ a divisor of n has been found and the algorithm stops. The algorithm continues with $\gcd(a, n) = 1$, and according to Lemma 3, the function $\exp_a(x) = a^x \bmod n$ is periodic.

Next, the quantum part (section 3.2) and the inner classical part (section 3.3) are used to determine the period of $\exp_a(x) = a^x \bmod n$. If it turns out that p is odd, the algorithm starts over again with a different $1 < a < n$ because we need p even to apply the representation $a^p + 1 = (a^{p/2} - 1)(a^{p/2} + 1)$. Then, according to Note 4, after a couple of attempts, a number a will be found such that p is even, and that $\gcd(a^{p/2} - 1, n)$ or $\gcd((a^{p/2} + 1), n)$ is a proper divisor of n: the algorithm stops with a divisor.

3.2. The quantum part

The details of Shor's algorithm especially its quantum part can be found in most text books on quantum computing like [3]. Here, we briefly sketch the key facts (ignoring some details to simplify the treatise).

First, the integers $0 \leq x \leq N$ (with $N = 2^m$ and $n^2 \leq 2^m < 2n^2$) are prepared in a two-part quantum register by applying a Hadamard transformation:

$$H^{\otimes m} \otimes I\left(|0\rangle|0\rangle\right) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle|0\rangle \tag{11}$$

Next, an oracle is used to compute the value table of the modular exponential function of these numbers:

$$|\alpha\rangle|\beta\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle|a^x \bmod n\rangle \tag{12}$$

Note the similarity with the value table after Definition 6; as discussed there, based on a value table of a function its periodicity can be determined "by inspection". Such an inspection can be done by applying a quantum Fourier transform on the $|\alpha\rangle$-part of the quantum register, and a succeeding measurement of the $|\beta\rangle$-part. The resulting state of the $|\alpha\rangle$-part is:

$$\left(\frac{1}{\sqrt{NA}} \sum_{j=0}^{A-1} \exp\left(\frac{2\pi i}{N} jpy\right)\right)|y\rangle \tag{13}$$

The quantum part ends with measuring the $|\alpha\rangle$-part of the register resulting in the value y.

3.3. The inner classical part

An extensive explanation of the inner classical part is given in Barzen and Leymann [7].

Note that the period p is already reflected in the amplitude of the state $|y\rangle$ reached after the Fourier transform (equation (13)): how can it be derived from this equation?

According to the Born rule of quantum physics, the probability Prob(y) to measure a particular y is the squared modulus of the amplitude of the associated state $|y\rangle$, i.e. based on equation (13):

$$\text{Prob(y)} = \frac{1}{NA} \left| \sum_{j=0}^{A-1} \exp\left(\frac{2\pi i}{N} jpy\right) \right|^2 \tag{14}$$

The sum in equation (14) is a geometric sum, thus, the well-known formula for computing the value of such a sum can be used, and with $q := \exp\left(\frac{2\pi i}{N} py\right)$ we get

$$\text{Prob(y)} = \frac{1}{NA} \left| \frac{1 - q^A}{1 - q} \right|^2 \tag{15}$$

A detailed analysis of equation (15) reveals (see [7], section 3, for the details) that "with high probability" a number k with $0 < k < p - 1$ can be found that

$$\left| \frac{y}{N} - \frac{k}{p} \right| \le \frac{1}{2p^2} \qquad (16)$$

Equation (16) is the pre-condition to apply Legendre's Theorem on continued fractions [7] which states that in this situation, $\frac{k}{p}$ is a convergent of $\frac{y}{N}$. Note that y is known because it has been measured, and N has been chosen at the beginning of the quantum part, i.e. $\frac{y}{N} \in \mathbb{Q}$ is known. Thus, N has been chosen at the beginning of the continued fraction representation $[a_0; a_1,..., a_l]$ of $\frac{y}{N}$ can be derived, and one of the convergents $\left[a_0; a_1,..., a_u \right] = \frac{g_u}{h_u}$ with $1 \le u \le l$ will be a "very good" approximation of $\frac{k}{p}$, i.e. the corresponding denominator h_u will be a candidate for the period p (note, that the actual value of k is not needed because we are only interested in the denominator p). Whether or not this candidate h_u is really the period or not must be checked explicitly.

Consequently, the inner classical part basically consists of a continued fraction analysis of $\frac{y}{N}$ that produces the period p "with high probability". In case a period is not found, the overall algorithm starts over again with a different $1 < a < n$.

3.4. Quantum ressource requirements

Shor's algorithm can efficiently break cryptosystems that are based on factorization or elliptic curves. While its classical parts can be performed on any classical computer today, its quantum part assumes a quantum computer that has error-corrected qubits and operations without errors.

Different authors discuss implementation variants of the quantum part of the factorization algorithm and state the number of logical qubits (i.e. error corrected qubits) and faithful operations [8]. As a very rough average, for RSA2048 the number of logical qubits needed is about 10^4 and the number of operations is about 10^{11} (and the operations tolerate low gate error rates of about 0.1%). A quantum computer with such resources is referred to as a *crypto-relevant quantum computer* (CRQC). At the time this paper has been written (2024), no such quantum computer is available. Note, that the quantum part computing elliptic curve discrete logarithms requires less quantum resources [9].

Considering that the number of physical qubits needed to implement a single logical qubit is estimated to be about 1.000 [10], and considering the corresponding gate errors, a quantum computer that can successfully perform the quantum part of Shor's algorithm seems to be far out. However, recent advancements in quantum error correction indicate that orders of magnitudes fewer physical qubits are needed to realize a logical qubit [11,12]. Thus, a crypto-relevant quantum computer may be sooner available than assumed a few years ago.

To be able to solve problems on today's error-prone quantum computers [13] so-called variational quantum algorithms are used [14]. These algorithms consist of a parameterized quantum circuit that produces a quantum state that is measured, this measurement result is evaluated by the classical part of the algorithms that optimizes the parameters, and the optimized parameter values are used by the parameterized quantum circuit again; this loop is performed until the overall procedure converges. Hereby, the quantum part runs only for a short period of time to avoid the errors on the quantum computer piling up, and only a few qubits are used - i.e. variational quantum algorithms are suitable for today's Noisy and Intermediate-Scale Quantum (NISQ) computers [15].

Yan, et al. claim that their variational quantum algorithm can break RSA2024 with less than 400 qubits, although the number of operations of the quantum part of their algorithm is still too large [16]. Khattar and Yosri implemented this algorithm but failed to scale it beyond 70-bit numbers where the algorithm always failed in their experiments [17]. Thus, it is still open whether variational quantum algorithms can break security based on near-term quantum computers.

3.5. A note on symmetric encryption

Shor's algorithm can break cryptographic schemes based on factorization or elliptic curves, which are asymmetric schemes. Symmetric schemes like AES 256 can be cracked by means of a quantum computer too via a brute-force attack [18]: This is because Grover's quantum algorithm supports an unstructured search with a quadratic speedup [3].

But Grover's algorithm too requires an error-corrected quantum computer. Wang, et al. presented a variational quantum algorithm to realize a brute-force attack on AES-like cryptographic schemes [19]. They also suggested a (non-variational) quantum circuit that can break AES 128 with about 400 logical qubits [20].

Thus, symmetric cryptographic schemes are volatile to quantum attacks too. In contrast to asymmetric schemes, a symmetric scheme can be hardened against such a brute-

force attack by doubling the key size used: since an attack based on the Grover algorithm has quadratic speedup, doubling the key size requires the same effort on a quantum computer that has to be spent on a classical computer for a key of the original key size. Under certain practical assumptions, less than doubling the key size suffice [21].

4. Lattice-based cryptography

Cryptographic schemes that are based on lattices are less known than schemes based on factorization or elliptic curves. But they obey properties that make them candidates for quantum-resistant security mechanisms. In this chapter, we sketch the basics of lattice-based cryptography. Basics about lattices can be found in the lecture notes of Haase, et al. [22], and in addition, Peikert [23] and Silberman [24] discuss lattice problems and corresponding cryptography. Also, Zheng provides proof for many of the facts given here [25].

4.1. Basics about Lattices

A lattice is a set of points in a vector space that is built by taking all linear combinations with integer coefficients of a given set of linear independent vectors.

Definition 10: Let $v_1, ..., v_n \in \mathbb{R}^n$ be linear independent. The set

$$\Lambda < v_1, ..., v_n > := \left\{ \sum_{i=1}^{n} g_i v_i \mid g_i \in \mathbb{Z} \right\} \quad (17)$$

is called a *lattice* in $\mathbb{R}^n$ with basis $\{v_1, ..., v_n\}$.

In the literature a generalized definition can be found with k independent vectors $v_1, ..., v_k \in \mathbb{R}^n$; k is then called the rank of the lattice, and for k = n, the lattice is called full rank. In what follows this generalization is not needed, thus, Definition 10 suffices. Figure 3 shows a lattice in $\mathbb{R}^2$.

It is obvious that a lattice is an abelian subgroup of $\mathbb{R}^n$. Furthermore, a lattice is discrete: for each lattice Λ, an $\varepsilon > 0$ can be found such that for all $x \neq y \in \Lambda$ it is $\|x - y\| \geq \varepsilon$, i.e. two points can be separated by disjoint neighborhoods. In fact, being a discrete abelian subgroup of $\mathbb{R}^n$ is equivalent to being a lattice in $\mathbb{R}^n$ [22]:

Note 6: $\Lambda \subset \mathbb{R}^n$ is a lattice $\Leftrightarrow$ Λ is a discrete abelian subgroup of $\mathbb{R}^n$. ∎

The following observation is important for lattice-based cryptography:

Note 7: Let $M \in GL(n, \mathbb{R})$ (i.e. M is an invertible $n \times n$ matrix). Then, the set $\{x \in \mathbb{Z}^n \mid Mx = 0\}$ is a lattice in $\mathbb{R}^n$.

Proof: Let $m_1, ..., m_n$ denote the columns of M, i.e. $M = (m_1 ... m_n)$. Because M is invertible, $m_1, ..., m_n \subset \mathbb{R}^n$ are linear independent. Thus,

$$\{M x \mid x \in \mathbb{Z}^n\} = \{\textstyle\sum x_i m_i \mid x_i \in \mathbb{Z}\} = \Lambda < m_1, ..., m_n >$$

is a lattice according to Definition 10.

It is obvious that $\{x \in \mathbb{Z}^n \mid Mx = 0\}$ is an abelian subgroup of $\mathbb{R}^n$. As a lattice, $\{Mx \mid x \in \mathbb{Z}^n\}$ is discrete. Obviously, a subset of a discrete set is discrete too, i.e.

$$\{x \in \mathbb{Z}^n \mid Mx = 0\} \subseteq \{Mx \mid x \in \mathbb{Z}^n\}$$

is discrete. Note 6 proves the claim. ∎

4.2. Nearly orthogonal bases

A lattice has many different bases: let $\mathcal{B} = \{b_1, ..., b_n\}$ be a basis of the lattice Λ and let B be the matrix with columns $b_1, ..., b_n$. For an $n \times n$ matrix $\mathcal{U}$ with integer coefficients (i.e. $\mathcal{U} \in \mathbb{Z}^{n \times n}$) and with det $\mathcal{U} = \pm 1$ (a so-called *unimodular* matrix), the columns $c_1, ..., c_n$ of the matrix $C = B \cdot \mathcal{U}$ build a basis $\mathcal{C} = \{c_1, ..., c_n\}$ of Λ (see the Figure 4 for an example).

The two bases shown in Figure 4 have quite different behavior in terms of computing solutions for lattice problems discussed below: a basis with very long vectors and a very small angle between vectors typically requires higher computational effort than a basis with small vectors that are pairwise nearly orthogonal [24].

The ideal basis of the lattice $\mathbb{Z}^n$ is an orthonormal basis $\mathcal{O} = \{o_1, ..., o_n\}$. It spans the unit cube the volume of which
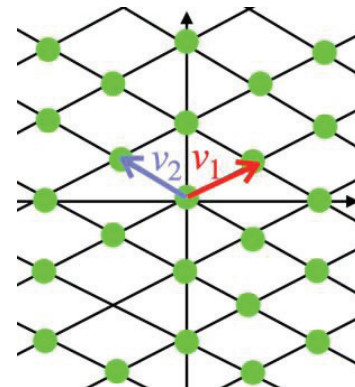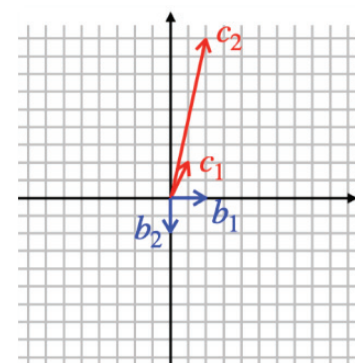

Figure 3: A Sample Lattice.

Figure 4: Two Bases of a Lattice.

is given by the determinant $|\det O| = 1$ where $O = (o_1 \ldots o_n)$ is the matrix with columns o_i. Now, an arbitrary lattice Λ is not generated by an orthonormal basis, but by a basis $\mathcal{B} = \{v_1,\ldots,v_n\}$. If $\{v_1, \ldots, v_n\}$ would be orthogonal (but not necessarily normalized), the volume of the cube spanned by $\{v_1,\ldots,v_n\}$ would simply be $\prod_{i=1}^{n} \|v_i\|$. In case the vectors are not orthogonal they span a parallelepiped with volume $|\det B| = 1$ where $B = (v_1 \ldots v_n)$ is the matrix with columns v_i. Now, the "goodness" of the basis $\mathcal{B}$ can be measured by comparing the hypothetical cube spanned by orthogonal vectors of lengths $\|v_i\|$ and the volume $|\det B|$ of the parallelepiped spanned by $\mathcal{B}$:

Definition 11: Let $v_1,\ldots,v_n \in \mathbb{R}^n$ be linear independent, $\mathcal{B} = \{v_1,\ldots,v_n\}$ and $\boldsymbol{B} = (v_1 \ldots v_n)$ the matrix with columns v_i. The number

$$\delta(\mathcal{B}) := \frac{\prod_{i=1}^{n} \|v_i\|}{|\det B|} \tag{18}$$

is called (*orthogonality*) *defect* of $\mathcal{B}$.

For any matrix $M = (m_1 \ldots m_n)$ it is always $\det M \leq \prod_{i=1}^{n} \|m_i\|_2$ (Hadamard's inequality). Thus,

$$\frac{\prod_{i=1}^{n} \|m_i\|}{|\det M|} \geq 1$$

and consequently $\delta(\mathcal{B}) \geq 1$. If $\mathcal{B}$ is orthogonal then $\det B = \prod_{i=1}^{n} \|v_i\|$, i.e. $\delta(\mathcal{B}) = 1$ and vice versa: this explains the name "orthogonality" defect.

The *Lattice Reduction Problem* asks to determine a basis $\mathcal{B}$ for Λ with minimla $\delta(\mathcal{B})$: This problem is NP-hard. In practice, a *good basis* (not necessarily one with minimal orthogonality defect) of a lattice is chosen and becomes a private key, and a corresponding bad basis - which is computed based on a randomly chosen unimodular matrix - becomes the corresponding public key (see section 4.6).

Next, we present three basic lattice problems that are hard to solve and which are central to cryptography.

4.3. Shortest vector

For a given lattice $\Lambda < v_1,\ldots,v_n >$ and a norm $\|\cdot\|$, the Shortest Vector Problem asks to find a non-zero vector of the lattice that has a minimal length (Figure 5a).

Definition 12: Let $\Lambda < v_1, \ldots, v_n >$ be a lattice and $\|\cdot\|$ be a norm on $\mathbb{R}^n$. With $\lambda_1(\Lambda) := \min_{x \in \Lambda \setminus \{0\}} \|x\|$ the *Shortest Vector Problem* (SVP) asks to determine an

$x^* \in \Lambda$ such that $\|x^*\| = \lambda_1(\Lambda)$.

The importance of this problem is because of [26].

Theorem 4: SVP is NP-hard for $\|\cdot\|_2$ and NP-complete for $\|\cdot\|_\infty$. $\blacksquare$

4.4. Closest vectors

For a given lattice $\Lambda < v_1, \ldots, v >$, a norm $\|\cdot\|$ and a vector $w \in \mathbb{R}^n$, the Closest Vector Problem asks to find a vector of the lattice that is closest to w (Figure 6).

Definition 13: Let $\Lambda < v_1, \ldots, v_n >$ be a lattice, $\|\cdot\|$ be a norm on $\mathbb{R}^n$, and $w \in \mathbb{R}^n$. With $\text{dist}(\Lambda, w) := \min_{x \in \Lambda} \|x - w\|$ the *Closest*
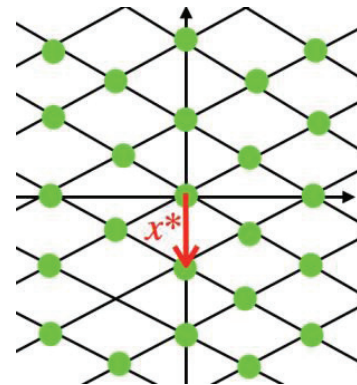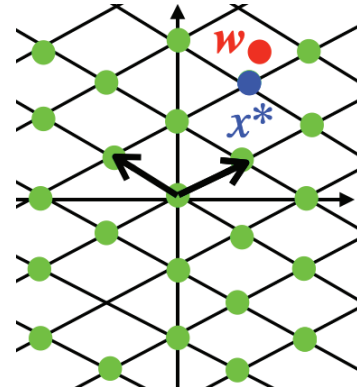

Figure 5a: The Shortest Vector of a Lattice.

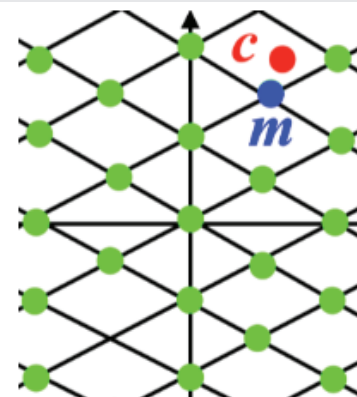**Figure 5b:** The Vector of a Lattice Closest to a Given Vector.

Figure 6: Decrypting by Finding the Closest Vector to a Given Vector Representing a Massage.

Vector Problem (CVP) asks to determine an $x^* \in \Lambda$ such that $\|x^* - w\| = dist(\Lambda, w)$.

As before, the importance of this problem results from work by Micciancio ([26] and Manohar [27]).

Theorem 5: CVP is NP-complete. ∎

4.5. Shortest independent vectors

The Shortest Independent Vectors Problem is a bit more comprehensive: a basis of the lattice has to be found whose longest vector has minimal length across all bases of the lattice.

Definition 14: Let $\Lambda < v_1, ..., v_n >$ be a lattice, and let $\mathfrak{B}$ be the set of all bases of Λ.

With $M(\Lambda) := \min_{\{b_1,...,b_n\} \in \mathfrak{B}} \min_{1 < i < n} \|b_i\|$ the *Shortest Independent Vectors Problem*

(SIVP) asks to determine a basis $\{s_1, ..., s_n\}$ of Λ with $\max_{1 < i < n} \|s_i\| = M(\Lambda)$.

Again, the importance of this problem is because of [26].

Theorem 6: SIVP is NP-complete. ∎

4.6. Example: Goldreich-Goldwasser-Halevi cryptosystem

We sketch a public key cryptosystem that is based on the Closest Vector Problem [28,29]. Let $\mathcal{B} = \{b_1, ..., b_n\}$ be a good basis of the lattice Λ (i.e. $\delta(\mathcal{B}) \approx 1$), B be the matrix with columns $b_1, ..., b_n$, and let $\mathcal{U}$ be a unimodular matrix (see section 4.2). $\mathcal{B}$ is the private key of the cryptosystem. Next, the columns of $\hat{\mathcal{B}} := \mathcal{U}B$ are a basis $\hat{\mathcal{B}}$ of the lattice Λ with $\delta(\hat{\mathcal{B}}) \gg 1$ (i.e. $\hat{\mathcal{B}}$ is a bad basis). $\hat{\mathcal{B}}$ is the public key of the cryptosystem.

The sender encrypts a message m as follows:

- The message m is mapped to a lattice point

  - E.g. m is split into n blocks of characters and each block is mapped to an integer which is used as a coefficient in a linear combination with a good basis) - see $m \in \Lambda$ in Figure 6.

- Randomly select a small vector $e \in \mathbb{R}^n$

- The encrypted message c is $c = m\hat{\mathcal{B}} + e$

- The receiver decrypts c by finding the vector closest to c as follows:

- Compute

$c \cdot B^{-1} = (m\hat{B} + e) \cdot B^{-1} = m\hat{B}B^{-1} + eB^{-1} = m\mathcal{U}BB^{-1} + eB^{-1} = m\mathcal{U} + eB^{-1}$

- Round the result according to [30] to remove eB^{-1}, i.e. $m\mathcal{U}$ is attained

- Finally, $m = m\mathcal{U}\mathcal{U}^{-1}$

Note, that there is a well-known attack (Nguyen's attack) on this cryptosystem [29], thus, it should not be used in practice.

4.7. Average case *vs.* worst case

Cryptography requires that randomly selected instances of problems are hard to solve [23], which is referred to as *average-case hard*. In contrast, the theory of algorithms typically considers hardness in terms of the existence of instances of problems that are hard to solve, which is referred to as *worst-case hard*. For example, computing a prime factor of a number is worst-case hard but not average-case hard: randomly selecting a number will result in an even number with 0.5 probability, and a prime factor "2" is immediately known.

The average-case hardness of certain lattice problems has been proven based on the worst-case hardness of some other lattice problems [31]. As a consequence, cryptographic mechanisms can be designed that are provably average-cased hard unless all (!) instances of some other lattice problem can easily be solved. Learning with errors is such a provable average-case hard problem (see next section).

4.8. Learning with errors

A *ring R* is a set with two binary operations denoted by "+" and "·". In a ring, the pair $(R, +)$ is a group, the multiplication "·" is associative, and there is a multiplicative identity (or *unity*) "1". Both operations combined obey the laws of distributivity. The set of integers $\mathbb{Z}$ is the typical example of a ring. In a ring, elements typically have no multiplicative inverse, i.e. for an element $a \in R$ there is no element $b \in R$ such that $b \cdot a = 1$. If every element of the ring has a multiplicative inverse (and "·" is commutative) a ring is called a *field* (like the real numbers $\mathbb{R}$ or the complex numbers $\mathbb{C}$). See [32] for the details.

Let R be a finite ring, $a_1, ..., a_n \in R^n$ and $t_1, ..., t_n \in R$. Consider an unknown linear function $f: R^n \to R$ with $f(a_i) \approx t_i$, i.e. it is $f(a_i) = t_i$ but with small errors.

The *Learning With Errors* (*LWE*) problem [33] is to determine f based on the set of training pairs $\{(a_1, t_1), ..., (a_n, t_n)\} \subset R^n \times R$.

This problem can be reformulated as follows: choose a fixed $s \in R^n$ ("the secrete"), and pick uniformly random small elements ("the errors") $e_1, ..., e_n \in R$. Then compute $t_i =$

$\langle a_i, s \rangle + e_i$. Given $\{(a_1, t_1), ..., (a_n, t_n)\} \subset R^n \times R$, find s.

If $A = (a_1, ..., a_n) \in R^{n \times n}$ is defined as the matrix with columns $a_1, ..., a_n$, $t = (t_1, ..., t_n)^T \in R^n$ and $e = (e_1, ..., e_n)^T \in R^n$, then the LWE problem is seen as the problem of finding a solution s of the linear equation system

$$A \cdot s + e = t \tag{19}$$

In case no errors occur (i.e. $e = 0$) Gaussian elimination solves the linear equation system $A s = t$ in polynomial time. But when errors are to be considered, the LWE problem turns out to be surprisingly difficult: Regev proves that if the LWE problem can be solved efficiently then solutions of the shortest vector problem SVP can be approximated by an efficient quantum algorithm [33]. Since even the approximate SVP is assumed to be hard, the LWE problem is hard too [34,35].

4.9. Learning with errors as a lattice problem

In cryptographic applications the ring R used in the LWE problem is $\mathbb{Z}_q$ for some prime number $q \in \mathbb{P}$: $\mathbb{Z}_q$ consists of the set of numbers $\{0, 1, ..., q-1\}$. Two elements of $\mathbb{Z}_q$ are added and multiplied modulo q, i.e. for $a, b \in \mathbb{Z}_q$ it is $a + b := a + b \bmod q$ and $a \cdot b := a \cdot b \bmod q$. With these operations, $\mathbb{Z}_q$ is a finite commutative ring [4].

With R substituted by $\mathbb{Z}_q$, $A \in \mathbb{Z}_q^{n \times n}$ and $s, t, e \in \mathbb{Z}_q^n$, the LWE problem in the shape of equation (19) can be written as

$$\left(A | E_n | -t \right) \cdot \begin{pmatrix} s \\ e \\ 1 \end{pmatrix} = 0 \tag{20}$$

where $E_n \in \mathbb{Z}_q^{n \times n}$ is the identity matrix, and "|" denotes concatenation. According to Note 6, the set of integer solutions $\Lambda = \{x \in \mathbb{Z}^{2n+1} | (A|E_n| -t) \cdot x = 0\}$ is a lattice. Obviously, $(s, e, 1)^T \in \mathbb{Z}^{2n+1}$ is such a solution, i.e. a lattice point. Especially, it can be shown that $(s, e, 1)^T \in \Lambda$ solves the shortest vector problem SVP for Λ, i.e. solving the LWE problem means to solve the SVP:

Note 8: The Learning with Errors problem is a lattice problem. ∎

Thus, cryptographic schemes based on the LWE problem belong to the domain of lattice-based cryptography. Micciancio and Regev provide examples of lattice-based schemes for encryption and signature [34]. We sketch a corresponding encryption schema in section 5.4.

5. Algebraic lattice cryptography

Key sizes in LWE get large (see section 4.6 for an example), they grow in the order of n^2 [35]. Large keys are not acceptable in several application domains. Reducing these key sizes to grow linearly in n can be achieved by "adding more algebraic structure" to the lattice. More concrete, the group $\mathbb{Z}_q$ is substituted by the ring of polynomials $\mathbb{Z}_q[X]/(X^n + 1)$ without losing the hardness property of regular LWE [35]. Going even further, the ring of polynomials $\mathbb{Z}_q[X]/(X^n + 1)$ is extended to the module $(\mathbb{Z}_q[X]/(X^n + 1))^k$.

In this chapter, we provide the basics about polynomials and $\mathbb{Z}_q[X]/(X^n + 1)$, and also about modules ([32] for more algebraic depth). We sketch the use of these algebraically extended lattices (also called *algebraic lattices*) in cryptography (Micciancio and Regev in [34] and Regev in [35] provide further details). Based on such lattices Kyber as an encryption mechanism and Dilithium as a signature mechanism are sketched.

5.1. Polynomial rings

When substituting $\mathbb{Z}_q$ by the ring $\mathbb{Z}_q[X]/(X^n + 1)$ the resulting LWE problem is called the *ring-LWE* problem [36]. Beyond that, when the ring $\mathbb{Z}_q[X]/(X^n + 1)$ is even substituted by the module $(\mathbb{Z}_q[X]/(X^n + 1))^k$. The corresponding LWE problem is called *module-LWE*.

The advantage of moving from ring-LWE to module-LWE is as follows [37]: if the security level has to be increased in ring-LWE, the degree n of the polynomials must be increased which in turn increases the computational effort involved. In module LWE the degree of the polynomials can remain the same but the rank k of the module has to be enlarged. The advantage is that any optimization in manipulating polynomials remains the same.

Thus, we must understand the basics of polynomials and how to perform computations with them.

Definition 15: Let R be a commutative ring. A *polynomial* in X over R is an expression $p = a_0 + a_1 X + ... + a_n X^n$ with $a_i \in R$; each a_i is called a *coefficient* of the polynomial. With $a_n \neq 0$ the number n is called the *degree* of the polynomial: $n = \deg p$. $R[X]$ denotes the set of all polynomials in X over R.

Polynomials can be added and multiplied: For $p = \sum_{i=0}^n a_i X^i$ and $q = \sum_{i=0}^m b_i X^i$ in $R[X]$ we define the sum $p + q$ of the two polynomials as

$$p + q = \sum_{i=0}^{\max(m,n)} r_i X^i \text{ with } r_i = a_i + b_i \tag{21}$$

and the product $p \cdot q$ of the two polynomials as

$$p \cdot q = \sum_{i=0}^{m+n} s_i X^i \text{ with } s_i = \sum_{j+k=i} a_j b_k \tag{22}$$

118

With these two operations on $R[X]$, the following can be proven:

Lemma 4: $(R[X], +, \cdot)$ is a commutative ring. ∎

In applications of algebraic lattices like Kyber (see section below), messages are considered as bit strings and they are transformed into polynomials as follows: for a message $m = b_0...b_n$ with $b_i \in \{0, 1\}$ and $b_n \neq 0$, the polynomial from $\mathbb{Z}[X]$ representing m is $P(m) = b_0 + b_1 X + \cdots + b_n x^n$. When adding and multiplying polynomials their coefficients become large, especially when their degree (which corresponds to the number of bits of a message) is large - see formula (22). To avoid inefficiency in the corresponding computations with large numbers, all computations are taken mod q for a prime number $q \in \mathbb{P}$. Thus, the set of polynomials $\mathbb{Z}_q[X]$ is used which ensures that all coefficients of polynomials are less than q.

5.2. Residue classes of polynomials

Also, when multiplying polynomials the degree of the resulting polynomial gets large (i.e. the degrees are summed up - see equation (22)): this results in inefficiencies too. Consequently, the degree of polynomials is limited by using residue classes of polynomials. If f is an *irreducible* polynomial (i.e. f is not the product of two nonconstant polynomials) then the set of all residue classes $\mathbb{Z}_q[X] / f$ is again a commutative ring. The polynomial $1 \cdot X^0$ is the identity element in $\mathbb{Z}_q[X] / f$. Furthermore, if $\deg f = n$ then $\deg g \leq n - 1$ for each $g \in \mathbb{Z}_q[X] / f$. Thus, $\mathbb{Z}_q[X] / f$ contains $n \cdot q$ elements, i.e. $\mathbb{Z}_q[X] / f$ is a finite ring. All hardness properties of the regular LWE problem are inherited by the ring-LWE problem over the ring $\mathbb{Z}_q[X] / f$. In summary:

Lemma 5: Let $f \in \mathbb{Z}_q[X]$ be irreducible. Then, $\mathbb{Z}_q[X] / f$ is a finite commutative ring with unity and card $\mathbb{Z}_q[X] / f = q \cdot \deg f$. ∎

Computing the residue class of a polynomial $g \in \mathbb{Z}_q[X]$ requires dividing g by f and taking the remainder of the division as the result. We briefly remind polynomial division by means of the following example:

- Let $f = x^2 + 1$ be an irreducible polynomial in $\mathbb{Z}[X]$

- Let $p = 4x^5 - x^4 + 2x^3 + x^2 - 1 \in \mathbb{Z}[X]$

- By computing $(4x^5 - x^4 + 2x^3 + x^2 - 1) \div (x^2 + 1)$ and determining the remainder of this division, $p \bmod f \in \mathbb{Z}[X] / (X^2 + 1)$, is computed

- The computation is in analogy to the division of numbers (Figure 7).

  - Determine the factor of x^2 of the denominator to result in the highest term $4x^5$ of the numerator of the division: this is $4x^3$, the first summand of the result of the division.

Figure 7: Example of a Polynomial Division.

- Multiply $4x^3$ by the denominator $x^2 + 1$ which results in $4x^5 + 4x^3$

- Subtract $4x^5 + 4x^3$ this from the numerator

- The highest term of the remaining polynomial is $-x^4$

- Determine the factor of x^2 of the denominator to result in this highest term $-x^4$ of the remaining numerator: this is $-x^2$, the second summand of the result of the division

- Multiply x^2 by $x^2 + 1$ which results in $x^4 + x^2$ ...

- ...

- The remainder of this polynomial division is $2x - 3$

- Thus, $(4x^5 - x^4 + 2x^3 + x^2 - 1) \equiv (2x - 3) \in \mathbb{Z}[X] / (X^2 + 1)$

Note, that the coefficients of the polynomials in the example are from $\mathbb{Z}$. Computations in $\mathbb{Z}_q[X] / (X^2 + 1)$ must perform all computations involving coefficients of the polynomials mod q.

5.3. Modules of polynomials

As mentioned above, increasing the security level of a cryptographic mechanism based on ring-LWE over $\mathbb{Z}_q[X] / f$ requires to increase the degree n of the polynomials, which in turn increases the involved computational effort. To avoid this, the ring $\mathbb{Z}_q[X] / f$ is substituted by $(\mathbb{Z}_q[X] / f)^k$; the latter algebraic structure is called a module.

Definition 16: A *module M* over a ring R is a commutative group $(M, +)$ with an operation $\cdot : R \times M \to M$ of R on M such that the laws of distributivity are satisfied, i.e. $(r_1 + r_2) \cdot m = r_1 m + r_2 m$ and $r \cdot (m_1 + m_2) = r \cdot m_1 + r \cdot m_2$.

If instead of a ring R a field $\mathbb{K}$ is used in the definition, the module over $\mathbb{K}$ becomes a vector space over $\mathbb{K}$. Thus, modules are generalizations of vector spaces. But in general, modules are quite different from vector spaces, e.g.

119

not every module has a basis, and if a module has a basis then different of its bases may have different cardinalities.

The cartesian product R^k of a ring R is a module over the ring R: elements of R^k are added component-wise, and multiplication of an element of R^k with an element of R is done component-wise too - just like vectors in the coordinate space $\mathbb{K}^k$ are added and multiplied with scalars from $\mathbb{K}$. Thus, based on Lemma 5 we get

Lemma 6: Let $f \in \mathbb{Z}_q[X]$ be irreducible, $k \in \mathbb{N}$. Then, $(\mathbb{Z}_q X]/f)^k$ is a finite module with card $(\mathbb{Z}_q[X]/f)^k = (q \cdot \deg f)^k$. ∎

$(\mathbb{Z}_q[X]/f)k$ is a finite module because card $\mathbb{Z}_q[X]/f = q \cdot \deg f$ according to Lemma 5, thus card $(\mathbb{Z}_q[X]/f)^k = (q \cdot \deg f)^k < \infty$.

The notion of a lattice can be generalized to the module $(\mathbb{Z}_q[X]/f)^k$. For this purpose, $\mathbb{Z}_q[X]/f$ is considered as a set of tuples by mapping a polynomial $a_0 + a_1 X + \ldots + a_m X^m$ to the tuple $(a_0, a_1, \ldots, a_m, 0, \ldots, 0) \in \mathbb{Z}_q^{\deg f}$, i.e. the tuple consisting of the coefficients of the polynomial. This mapping is performed for each component polynomial p_i of $(p_1, \ldots, p_k) \in (\mathbb{Z}_q[X]/f)k$, and the tuples are combined. This way, each element of $(\mathbb{Z}_q[X]/f)^k$ becomes an element of $\mathbb{Z}_q^{k.\deg f} \subset \mathbb{Z}^{k.\deg f}$ where lattices (of rank $k \cdot \deg f$) can be defined.

In order to be able to formulate the lattice problems introduced in sections 4.3, 4.4, and 4.5 a norm is needed on $(\mathbb{Z}_q[X]/f)^k$. But the canonical scalar product $\langle (a_i),(b_i) \rangle = \sum a_i b_i$ induces the canonical norm $\|(a_i)\| = \sqrt{\langle (a_i),(a_i) \rangle}$. Thus, lattice problems in $(\mathbb{Z}_q[X]/f)^k$ result.

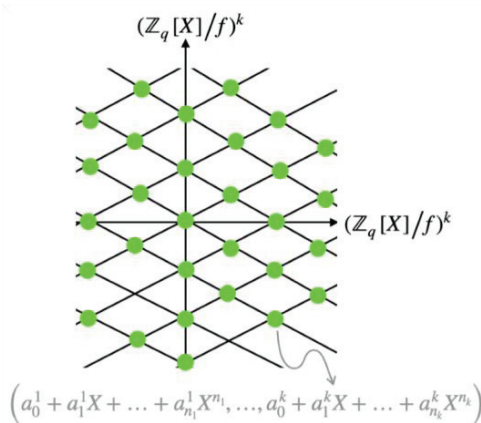Figure 8 depicts this situation: The module which is the foundation of an algebraic lattice is a cartesian product of a finite polynomial ring. Accordingly, the elements of the module are tuples of polynomials. Computations in this module are performed mod q w.r.t. the coefficients of the polynomials, and mod f w.r.t. the polynomials themselves. Different algebraic lattice-based cryptographic mechanisms (like Kyber or Dilithium) use different prime numbers q, different irreducible polynomials f, and different ranks k.

5.4. Kyber

Kyber is an algebraic-lattice-based asymmetric cryptographic mechanism based on a module-LWE problem described above. The specification of Kyber can be found in [38]: it specifies all algorithms used for performing the calculations with polynomials, for rounding, for sampling error vectors, etc. Kyber is using $R_q := \mathbb{Z}_q[X]/(X^n+1)$ with $n = 256$ and $q = 3329$.

5.4.1. Computing Keys: The private and public keys are determined as follows:

1. A randomly chosen $s \in R_q^k = (\mathbb{Z}_q[X]/(X^n+1))^k$ becomes the *private key*. Here, Kyber is using $k \in \{2,3,4\}$ as the *key length*. This key length varies the lattice rank $k \cdot n$ and, thus, the security level implied. For example, with $k = 3$ the lattice rank becomes 768 and the corresponding variant KYBER768 has the security level of AES192.

2. Next, a matrix $A \in R_q^{k \times k}$ is randomly chosen. The module-LWE requires a small randomly generated error vector $e \in R_q^k$, and $t = As + e \in R_q^k$ is computed. The pair (A, t) becomes the *public key.*

5.4.2. Encryption: For each message m to be encrypted the following is performed:

1. First, the following elements are randomly generated: a so-called randomizer vector $r \in R_q^k$ as well as an error vector $e_1 \in R_q^k$ and a single error polynomial $e_2 \in R_q$, both with small coefficients.

2. The message m is considered as a bit string $m = b_0 \ldots b_{n-1}$ and is represented as a polynomial $P(m) = b_0 + b_1 X + \cdots + b_{n-1}X^{n-1}$ (i.e. $b_i \in \{0,1\}$); in this sense, m and $P(m)$ are used interchangeably.

3. $P(m)$ is "scaled" by multiplying it by $\lfloor q/2 \rceil$, i.e. $\lfloor q/2 \rceil \cdot P(m)$ is computed. As a reminder, $\lfloor x \rceil$ is the integer closest to x, e.g. $\lfloor 2.4 \rceil = 2$ or $\lfloor 2.5 \rceil = 3$. Note, that the coefficients of the latter polynomial are 0 or $\lfloor q/2 \rceil$ because the coefficients of $P(m)$ are 0 or 1.

Figure 8: Geometric Situation of Algebraic Lattices.

120

4. Then, $u = A^T r + e_1$ and $v = t^T r + e_2 + m$ are computed.

5. The encrypted message is (u, v).

5.4.3. Decryption: The encrypted message (u, v) is decrypted as follows:

1. First, $\hat{m} = v - s^T u$ is computed by using the secret key s.

   • Note, that $\hat{m}$ is not yet m, but it contains some noise:

$$
\begin{aligned}
\hat{m} = v - s^T u &= \left(t^T r + e_2 + m\right) - s^T\left(A^T r + e_1\right) \\
&= \left((As + e)^T r + e_2 + m\right) - s^T\left(A^T r + e_1\right) \\
&= s^T A^T r + e^T r + e_2 + m - s^T A^T r - s^T e_1 \\
&= e^T r + e_2 + m - s^T e_1 \\
&= m + \underbrace{e^T r + e_2 - s^T e_1}_{\text{noise}}
\end{aligned}
\tag{23}
$$

2. Since e, e_1, e_2 are "small" by definition, the noise $e^T r + e_2 - s^T e_1$ has "small" coefficients too. Thus, the coefficients of $\hat{m}$ are close to the coefficients of $\lfloor q / 2 \rfloor \cdot P(m)$ which are 0 or $\lfloor q / 2 \rfloor$ (see 5.4.2.).

3. Next, the following "rounding" of the coefficients of $\hat{m}$ is performed resulting in $\rho(\hat{m})$:

   • All coefficients of $\hat{m}$ close to $\lfloor q / 2 \rfloor$ are set to "1".

   • All coefficients of $\hat{m}$ close to 0 (or close to q, which is congruent 0) are set to "0".

4. Finally, $\rho(\hat{m}) = m$ is (with very high probability) the original message.

There is a small probability that $\rho(\hat{m})$ is not the original message m, but this failure probability is extremely small (see [39], Theorem 1): for example, for $k \cdot n = 512$ the failure probability is 2^{-139}.

[39] also explains the reasons for the hyper-parameters chosen for Kyber in more detail. The rounding mechanism is in fact more sophisticated than presented here and is described in [30].

5.4.4. Kyber by example: In what follows, we describe Kyber stepwise by example (remark: no guarantee that the computations [involving lots of polynomial multiplications, frequently taking coefficients mod 7 and polynomials mod $(X^4 + 1)$] are error-free!). We use small hyperparameters to highlight the essential processing: As hyperparameters, we set $n = 4$, $q = 7$, and $k = 2$. Thus, our finite polynomial ring is $R_7 = \mathbb{Z}_7[X] / (X^4 + 1)$.

First, we need to choose a private key and compute a corresponding public key:

As private key we choose $S = \begin{pmatrix} x^3 + x + 1 \\ x + 2 \end{pmatrix}$

$A \in R_7^{2 \times 2}$ is chosen randomly as $t = \begin{pmatrix} 4x^3 + 5x + 4 & 3x^3 + 5x^2 \\ 5x^3 + 3x & 6x^2 + 6 \end{pmatrix}$

In order to computed $t = A s + e$ we need a small error vector $e \in R_7^2$ and choose $e = (x^2, x)$

The computation results in (considering the coefficients mod 7 and the polynomials mod $(X^4 + 1)$)

$$
t = \begin{pmatrix} 4x^3 + 5x + 4 & 3x^3 + 5x^2 \\ 5x^3 + 3x & 6x^2 + 6 \end{pmatrix} \cdot \begin{pmatrix} x^3 + x + 1 \\ x + 2 \end{pmatrix} + \begin{pmatrix} x^2 \\ x \end{pmatrix} = \begin{pmatrix} 5x^3 - 2x^2 + 2x - 1 \\ 4x^3 - 4x^2 + 3x + 4 \end{pmatrix}
$$

   • Thus, the public key (A, t) is

$$
\left(\begin{pmatrix} 4x^3 + 5x + 4 & 3x^3 + 5x^2 \\ 5x^3 + 3x & 6x^2 + 6 \end{pmatrix}, \begin{pmatrix} 5x^3 - 2x^2 + 2x - 1 \\ 4x^3 - 4x^2 + 3x + 4 \end{pmatrix} \right)
$$

The message to encrypt is $\mu = $ ITS ALL GREEK TO ME (see section 2.4). For simplicity, we only consider the first letter, i.e. $m = 9$ and we get $m = 1001$ in binary representation. Next, m must be mapped to a polynomial by means of the map P resulting in

$$P(m) = x^3 + 0x^2 + 0x + 1 = x^3 + 1;$$

scaling this polynomial by $\lfloor q / 2 \rfloor = \lfloor 7 / 2 \rfloor = 4$ gives $m = 4x^3 + 4$ (as before, we identify m and $P(m)$).

   • To encrypt m we choose $r = (x^2, 1)^T \in R_7^2$ as randomizer, an error vector $e_1 = (x + 1, 1)^T \in R_7^2$ and as error polynomial $e_2 = x^3 + x \in R_7$.

   • Then we compute $u = A^T r + e_1$ considering mod 7 w.r.t. the coefficients and mod $(X^4 + 1)$ w.r.t. the polynomials:

$$
u = \begin{pmatrix} 4x^3 + 5x + 4 & 5x^3 + 3x \\ 3x^3 + 5x^2 & 6x^2 + 6 \end{pmatrix} \begin{pmatrix} x^2 \\ 1 \end{pmatrix} + \begin{pmatrix} x + 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3x^3 + 4x^2 + 1 \\ 6x2 - 3x - 5 \end{pmatrix}
$$

   • Next, we compute $v = t^T r + e_2 + m$ (again based on the residue classes for the coefficients and the polynomials):

$$
\begin{aligned}
v &= \left(5x^3 - 2x^2 + 2x - 1, \ 4x^3 - 4x^2 + 3x + 4\right)\begin{pmatrix} x^2 \\ 1 \end{pmatrix} + \left(x^3 + x\right) + \left(4x^3 + 4\right) \\
&= 3x^3 + 2x^2 - x - 2
\end{aligned}
$$

   • Thus, the encrypted message is:

$$
(u, v) = \left(\begin{pmatrix} 3x^3 + 4x^2 + 1 \\ 6x^2 - 3x - 5 \end{pmatrix}, 3x^3 + 2x^2 - x - 2 \right)
$$

In order to decrypt the message, the recipient must first compute $\hat{m} = v - s^T u$ based on the private key s. Thus:

$$\hat{m} = \left(3x^3 + 2x^2 - x - 2\right) - \left(x^3 + x + 1, x + 2\right)\begin{pmatrix} 3x^3 + 4x^2 + 1 \\ 6x^2 - 3x - 5 \end{pmatrix} = 3x^3 - x^2 - x - 4$$

Computing the coefficients mod 7 results in

$$\hat{m} = 3x^3 + 6x^2 + 6x + 3$$

and finally rounding the coefficients by ρ, i.e. coefficients close to $\lfloor q / 2 \rfloor = 4$ are set to "1", and coefficients close to 0 (or close to $q = 7$) are set to "0" delivers

$$\rho(\hat{m}) = x^3 + 1 = 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x + 1$$

Thus, $\rho\,(\hat{m}) = P(m)$ and the original message $m = 1001$ results.

5.4.5. Kyber as key encapsulation mechanism: As ECC Diffie-Hellman discussed in section 2.6, Kyber is used as a key encapsulation mechanism (KEM): The asymmetric encryption/decryption protocol described before is mainly used to agree on a symmetric key. Once the symmetric key is agreed on, the information exchanged is encrypted and decrypted based on the shared secret.

5.5. Dilithium

We sketch the essence of Dilithium in this section. Note, that the details are more subtle than sketched (e.g. "rounding" means to use high-order bits of the coefficients of the affected polynomials); the details can be found in [40]. Richter, et al. summarize Dilithium as well as other post-quantum cryptography algorithms [41].

Like Kyber, Dilithium is an algebraic-lattice-based cryptographic mechanism. But instead of encryption/decryption, it is about signatures. Dilithium is also using $R_q = \mathbb{Z}_q[X] / (X^n + 1)$ with $n = 256$ but q is much bigger in Dilithium than in Kyber: $q = 2^{23} - 2^{13} + 1 = 8380417$.

5.5.1. Key generation: Key generation works like in Kyber: a uniformly random $s \in R_q^k$ with small coefficients becomes the *private key*. Next, a uniformly random $A \in R_q^{m \times k}$ as well as a uniformly random $e \in R_q^m$ with small coefficients is chosen; based on the latter two $t = As + e \in R_q^m$ is computed. The pair (A, t) is the *public key*. m and k are chosen in Dilithium in order to result in different security levels [42], i.e. $(m, k) = (4,4)$ for security level 2, $(m, k) = (6,5)$ for security level 3, and $(m, k) = (8,7)$ for security level 5.

5.5.2. Signature generation: Signatures are computed as follows:

1. For each message μ to sign, freshly choose uniformly random $r_1 \in R_q^k$ with small coefficients.

2. Compute $w = \rho(Ar_1) \in R_q^m$

   - ρ is the algorithm rounding coefficients to 0 or $\lfloor q/2 \rfloor$

3. Compute $c = \Psi(\mu \| w) \in R_q$

   - Ψ is a hash function based on the function SHAKE 256 (which is a variant of the SHA-functions [43]) as well as SHA-256 itself [44].

   - The hash function maps its argument to $B_h \subset \mathbb{Z}_q[X] / (X^n + 1)$ which is a set of polynomials with coefficients in $\{-1, 0, + 1\}$ and exactly h coefficients not being "0". Dilithium is using $h = 60$ resulting in more than 2^{256} potential polynomials.

4. Compute $r_2 = r_1 + cs \in R_q^k$

5. The *signature* is (r_2, c). It is sent together with the message μ to the recipient. The recipient can verify the signature as discussed in section 5.5.3. [45] proves that Dilithium is a so-called self-target Module Short Integer Solution Problem (MSISP), i.e. a lattice problem: Given a message μ, find $y \in R_q^{2k+1}$ with $0 < \|y\| \le \beta$ where $y = (r_1, c, r_2)^T$ with $r_1 \in R_q^k$, $c \in B_{60}$, $r_2 \in R_q^k$ such that $\Psi(\mu \| (A \mid s \mid E_k) \cdot y) = c \in B_{60}$. Here, $(A \mid s \mid E_k)$ is the matrix consisting of the matrix A (part of the public key) followed by a single column s (the private key) followed by the identity matrix E_k; the similarity to equation (20) already roughly indicates that this is a lattice problem.

5.5.3. Signature verification: When the signed message $(\mu(r_2, c))$ is received, the following computations are performed to verify the signature (the correctness of this verification procedure is shown in the next section):

1. Compute $\hat{w} = \rho(Ar_2 - tc)$.

   - (A, t) is the public key.

   - (r_2, c) is the signature to verify.

2. Compute $\hat{c} = \Psi(\mu \| \hat{w})$.

   - μ is the message claimed to be signed.

3. In case $\hat{c} = c$ then the signature is valid, otherwise the signature is invalid.

5.5.4. Correctness of signature verification: The following shows that the signature verification procedure is correct:

$$\rho\left(Ar_2 - tc\right) \overset{(a)}{=} \rho\left(A\left(r_1 + cs\right) - \left(As + e\right)c\right)$$
$$\overset{(b)}{=} \rho\left(Ar_1 + Acs - Asc - ec\right)$$
$$\overset{(c)}{=} \rho\left(Ar_1 - ec\right)$$
$$\overset{(d)}{=} \rho\left(Ar_1\right)$$
$$\overset{(e)}{=} w$$

The first equality (a) results from substituting $r_2 = r_1 + cs$ and $t = As + e$. (b) is a simple multiplication, followed by canceling the two terms in the middle in (c). (d) is valid because e has been chosen with small coefficients while $c \in B_{60}$ has coefficients from $\{-1, 0, +1\}$, i.e. they are small too, thus, ec has small coefficients which imply that rounding ρ is not affected by ec. (e) uses the definition of w. The verification succeeds.

Assume that r_2 has not been calculated with the correct private key s (i.e. the substitution in (a)), i.e. the signature is bogus. But t as part of the public key has been computed with the correct private key s. Consequently, the two terms in the middle of (b) would not cancel out and the equation in (c) would not hold. Thus, $\rho(Ar_2 - tc)$ would result in a value $\hat{w} \neq w$. This implies that the step (2) in the verification process results in $\hat{c} = \Psi(\mu \| \hat{w}) \neq c$: the verification fails.

6. Why should I care?

There is a whole spectrum of opinions or estimations, respectively, about when powerful quantum computers will be available (and some even argue that such computers can not be built at all). The first question to ask is: "Powerful for which purpose". It is important to note that different application areas require different characteristics of a quantum computer (in terms of the number of qubits, decoherence time, gate fidelity, etc) to qualify as "powerful". In our context, a quantum computer that can threaten today's cryptographic schemes by being able to execute Shor's algorithm, for example, is referred to as *Cryptographic Relevant Quantum Computer* (CRQC) (see section 3.4).

Scholten, et al. estimate that quantum computers powerful enough for several application areas will be available "in the late 2020" and that cryptographic-relevant quantum computers are (a bit) further out [46]. However,

CRQC is already threatening security now (see section 6.1). The time a particular company has the counter this threat can be roughly estimated (see section 6.2). Awareness of this threat is growing in society (section 6.3), and standards for a quantum-safe infrastructure are being created (sections 6.4 and 6.5).

6.1. "Harvest now, decrypt later"

Cryptographic-relevant quantum computers not only threaten security once they are available but they do already now. An often perceived attack is referred to as "Harvest Now, Decrypt Later" (HNDL): here, an attacker can steal data that is encrypted based on an asymmetric scheme, store it, and decrypt it once a CRQC is available. If data that must be kept confidential and that is persistently stored is encrypted symmetrically, such data at rest is considered quantum-safe (in the sense of section 3.5). However, data at rest that is asymmetrically encrypted is in danger.

Data in motion (e.g. messages or documents exchanged) is typically encrypted based on a symmetric session key which has been agreed upon at the start of the session based on asymmetric encryption. If this message exchange is captured and stored, the data in motion can be decrypted at a later point in time once a CRQC is available: the asymmetric keys will be decrypted allowing to derive of the symmetric key to decrypt the proper data.

6.2. Mosca's inequality

Thus, companies should take proper actions to protect themselves from attacks by a CRQC. These actions include migrating to a cryptographic infrastructure that is quantum-safe. In Figure 9, this time is called $T_{Migrate}$. During this time period, and especially at the end of this period, data is very likely generated by the company that must be kept confidential for a company-specific time, called T_{Confi} in Figure 9. This means that a company must ensure that encrypted data can not be decrypted for a period of $T_{Migrate} + T_{Confi}$.

The time needed to develop a CRQC is denoted by T_{CRQC} in the figure. Once T_{CRQC} is passed, data is at risk and can be decrypted: a "crypto collapse" may occur. While T_{Confi} is
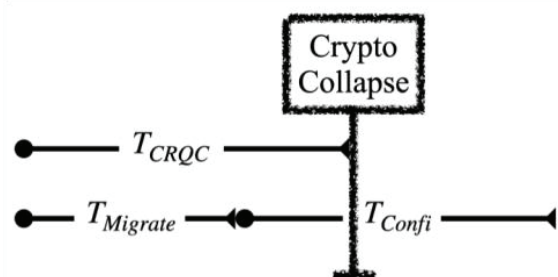

Figure 9: Graphics of Mosca's Inequality.

typically well-known by each company itself, and $T_{Migrate}$ can be estimated by the project management techniques of a company, T_{CRQC} is out of the control of a company. Roadmaps of vendors of quantum computers have been observed and tracked to derive T_{CRQC}. Also, assessments of consulting firms can be considered.

Once the numbers are available, the following inequality known as *Mosca's Inequality* [47] can be evaluated:

$$T_{Migrate} + T_{Confi} > T_{CRQC} \qquad (24)$$

In case the inequality (24) holds, a company may get into trouble: the time data must be kept secret and the time needed to move towards a post-quantum cryptographic infrastructure is bigger than the time needed for (some) vendors to provide a cryptographic relevant quantum computer. Thus, the data of the corresponding company is at risk. If possible, the migration time $T_{Migrate}$ may be shortened in order to make $T_{Migrate} + T_{Confi} \leq T_{CRQC}$ hold. But in case $T_{Confi} > T_{CRQC}$ already holds, the company is in trouble.

6.3. Awareness

Considerations like the ones outlined in the previous section also apply to governments and policymakers in general. As a result the US government, for example, enacted a law [48] "…to encourage the migration of Federal Government information technology systems to quantum-resistant cryptography…". In reaction to this law, a strategy to migrate to post-quantum cryptography has been worked out [49], and the cost to migrate the IT systems of the US government has been estimated to be higher than 7 billion US$ by 2035.

Similarly, the European Commission recommends a coordinated implementation roadmap for the transition to post-quantum cryptography for its member states [50] "…necessary for Europe to look for stronger safeguards, ensuring the protection of sensitive communications and the long-term integrity of confidential information, i.e., by switching to Post-Quantum Cryptography as swiftly as possible". Furthermore, the intend for this roadmap is "…to foster the transition to Post-Quantum Cryptography for the protection of digital infrastructures and services…".

The DigiCert & Ponemon Institute published a study in 2023 about how industrial organizations worldwide are addressing post-quantum-safeness [51]. About 1500 practitioners from the US, EU, and AP have been surveyed who claimed to be acquainted with their organization's work in the area of post-quantum cryptography. The result is that 61% of the practitioners are very much concerned about their organizations not being prepared to address the security threats posed by quantum computing. Even 74% are concerned about "harvest now, decrypt later"

attacks. 23% of the respondents have a strategy towards post-quantum-safeness, and the same percentage (23%) have even no plan to build such a strategy.

In summary, it seems to be that government organizations take post-quantum security more seriously than industrial organizations.

6.4. Standardization

Standards are important for post-quantum security. For example, communication partners must agree on a joint protocol to secure their data exchange. The National Institute of Standards (NIST) is in the process of standardizing quantum-safe algorithms [52,53]. The process consists of several phases: first, submissions of quantum-safe algorithms have been solicited. The submitted algorithms are evaluated, and positively evaluated algorithms become candidates for standards. These candidates are then turned into specifications. The specifications are then reviewed by a global community. Those specifications that did not fail the review became Federal Information Processing Standards (FIPS).

Especially, Kyber and Dilithium have been standardized [54,55]; note that NIST officially uses the names Module-Lattice-Based Key-Encapsulation Mechanism or ML-KEM, and Module-Lattice-Based Digital Signature or ML-DS in its specifications, which reflects the mathematical underpinnings of the algorithms. Other algorithms are under standardization by NIST also or remain candidates for future standardization. Richter, et al. give an overview of the mathematics of the algorithms finally considered by NIST for standardization [41].

To support crypto-agility (next section), NIST identified 14 candidates for standardization in a second round [56]. These candidates are signature algorithms that will be analyzed according to the phases sketched before.

6.5. Crypto-agility

The algorithms that are currently standardized are believed to be quantum-safe. I.e. there are no known classical algorithms or quantum algorithms that can crack them. But there is no guarantee (in the sense of a mathematical proof) that they cannot be broken in the future. In fact, there is some evidence that a couple of them will be finally broken [57]; and one of the NIST candidates has already been broken [58]. However, based on the best knowledge of the majority of the cryptographic community the standards currently being rolled out are post-quantum-safe.

Because of the volatility of being post-quantum-safe, any cryptographic infrastructure that is considered to be post-quantum-safe in total must cope with situations in

which one or more of the algorithms that are considered to be quantum-resistant today will be cracked. It must be possible to easily extract the corresponding algorithms from the infrastructure such that they will not be used anymore. Furthermore, it should be possible to substitute these algorithms with others that are still considered to be quantum-resistant at this point in time. Thus, the infrastructure must be *crypto-agile* [59].

Figure 10 depicts the high-level architecture of such an agile cryptographic infrastructure. It is plugin-based, i.e. the implementations of the algorithms are plugins that can be easily exchanged. Also, more plugins can be added whenever implementations of new post-quantum-safe algorithms appear.

Furthermore, algorithms can be combined. I.e. a classical algorithm C is not necessarily replaced by its quantum counterpart Q, but both algorithms are used together (a.k.a. *hybrid cryptography*). For example, a document may be encrypted based on the classical AES algorithm as well as based on the quantum-safe Kyber algorithm. This is an example of the $C \rightarrow Q$ approach where first the classical algorithm C is applied and next the corresponding quantum algorithm Q is applied on the result of the preceding step. Obviously, $Q \rightarrow C$ is also a valid approach. Classical and quantum algorithms might also be applied in parallel, i.e. a $C\|Q$ approach is followed. For example, a signature may be computed based on a classical algorithm as well as on a quantum algorithm, and finally, both signatures are

Figure 10: Plugin-Based Cryptographic Infrastructure.

Figure 11: Signing a Message Using the $C \rightarrow Q$ Approach.

associated with the signed object. Figure 11 depicts the $C \rightarrow Q$ approach for signing a message: first, the message is signed classically by using the Elliptic Curve Digital Signature Algorithm (ECDSA), and the resulting object is signed by means of Dilithium. The advantage of such combined approaches is that they are at least as secure as the current classical algorithm C, but it allows to gain trust in the quantum algorithm Q and its implementations.

6.6. Potential impact in practice

A question often heard is about the impact of the threat posed by quantum computers on one's own practice. Not surprisingly there is no generally applicable answer to this complex question. Several firms offer corresponding services. Also, publications like the ones by DigiCert & Ponemon Institute [51], Macaulay and Henderson [59], or the National Cybersecurity Center of Excellence [60] (to name but a few) may provide initial guidance.

However, the first step is to understand the problems and associated technologies (the article at hand may help). Next, vulnerable areas (e.g. data in transit, data stored) have to be identified; Mosca's inequality (see section 6.2) may give indications about the urgency to act. Based on this, influences on the current IT infrastructure should be analyzed. Available standards (see section 6.4) are helpful in this. Likely, the emerging architecture should be agile (see section 6.5). Prototypes based on open-source implementations (see Chapter 7) will be helpful.

7. In brief: Sample open source activities

Several of the algorithms that are standardized by NIST (and others) are available as open-source implementations. Also, other implementations that use this code in communication protocols, for example, are available in open source too. In this chapter, we sketch a particular stream of open-source activities having significant industry support while other activities are not mentioned at all.

7.1. Open quantum safe

A consortium named Post-Quantum Cryptography Alliance (PQCA) members of which include AWS, CISCO, Google, IBM, etc. "aims to support the transition to quantum-resistant cryptography" [61]. Amongst other activities, PQCA runs a project called Open Quantum Safe (OQS) that builds implementations of post-quantum algorithms and usages thereof [62].

OQS provides a library called liboqs (see next section) for algorithm implementations. These implementations are also used to build quantum-safe implementations of TLS, SSH, etc. [65]. The latter implementations are built by providing branches of OpenSSL, OpenSSH, for example, that make use of liboqs. The quantum-safe branch of TLS,
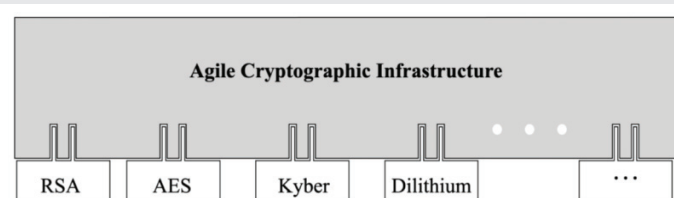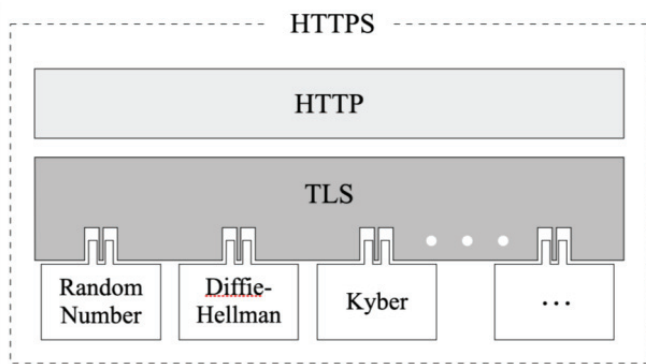
Figure 12: Making HTTPS Quantum-Safe.

for example, has been used to demonstrate how to enable Apache httpd or Nginx, for example, to become quantum-safe.

Figure 12 shows how HTTPS can become quantum-safe: HTTPS is HTTP over TLS. Thus, HTTPS can be turned into a quantum-safe protocol by making TLS quantum-safe. Classically, TLS is using by default an encrypted secret random number-based mechanism or it can be set up to exploit Diffie-Hellman (refer to section 2.6). By using Kyber as a key encapsulation mechanism TLS becomes quantum-safe and this implies the quantum-safeness of HTTPS too.

7.2. Open source activities

A collection of implementations of quantum-safe algorithms is provided by OQS under an MIT license as an open-source C-library on GitHub [64]. Wrappers for other programming languages are provided, for example, Java, Python, and C++. Besides implementations of Kyber and Dilithium (or ML-KEM and ML-DS, respectively) the library contains also implementations of other algorithms for key encapsulation mechanisms and signature schemes.

It is explicitly mentioned on the websites of liboqs that the library is of prototypical nature only, i.e. it should not be used in production environments and it should not be used to protect sensitive data. In contrast, the goal of the Post-Quantum Cryptography Coalition PQCC [65] (whose founding members include members of the PQCA) lists explicitly the creation of corresponding production-quality code, and ensuring cryptographic agility (refer to section 6.5).

7.3. Sandwich

Sandwich is an open source implementation that provides cryptographic agility enabling post-quantum-safeness by using liboqs [66]. It defines a unified API for choosing cryptographic algorithms and protocols at runtime. Thus, no code must be modified to exchange algorithm implementations (but proper configuration files must be passed at runtime).

8. Conclusion

In this paper, we described the origins of the need for efforts subsumed by the term post-quantum security. The relevant number theoretic background has been provided as the underpinning of security schemes based on factorization. The algebra of elliptic curves has been sketched and security schemes on top of it were described. We revealed that both schemes are in fact relying on the hardness of computing discrete logarithms. Shor's algorithm was explained in brief, showing that security schemes relying on discrete logarithms are broken as soon as powerful enough quantum computers become available (which may very well be within the next couple of years).

Lattice-based cryptography is seen as a rescue. We introduced lattices and fundamentally hard problems that provide the base for corresponding security schemes. To avoid huge keys that are implied by high security levels required in practice, algebraic lattices have been introduced and two corresponding cryptographic schemes called Kyber and Dilithium have been discussed.

Standardization efforts in this domain have been reported, and related open-source activities were mentioned. The "harvest now, decrypt later" attack was described and its impact on organizations was revealed. Crypto-agility as an important principle has been emphasized.

Author contributions

Writing – original draft, F.L.; Writing – review & editing, J.B. All authors have read and agreed to the published version of the manuscript.

References

1. Shor PW. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Sci. Statist. Comput. 1997;26. Available from: https://doi.org/10.1137/S0097539795293172

2. Rivest R, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Comm. ACM. 1978;21(2):120-126. Available from: https://doi.org/10.1145/359340.359342

3. Nielsen MA, Chuang IL. Quantum Computation and Quantum Information. Cambridge University Press; 2016.

4. McLeman C, McNicholas E, Starr C. Explorations in Number Theory. Springer Nature; 2022. Available from: https://books.google.co.in/books/about/Explorations_in_Number_Theory.html?id=G7OiEAAAQBAJ&redir_esc=y

5. Delfs H, Knebl H. Introduction to cryptography. 2nd ed. Springer; 2015.

6. Rubinstein-Salzedo S, editor. The Vigenère Cipher. In: Cryptography. Springer Undergraduate Mathematics Series. Springer; 2018;41-54. Available from: https://doi.org/10.1007/978-3-319-94818-8_5

7. Barzen J, Leymann F. Continued fractions and probability estimations in Shor's algorithm: A detailed and self-contained treatise. Appl Math. 2022;2(3):393−432. Available from: https://doi.org/10.3390/appliedmath2030023

8. Gidney C, Ekara M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. Quantum. 2021;5:433. Available from: https://doi.org/10.22331/q-2021-04-15-433

9. Roetteler M, Naehrig M, Svore KM, Lauter K. Quantum resource estimates for computing elliptic curve discrete logarithms. 2017. Available from: https://arxiv.org/abs/1706.06752

10. Cleland AN. An introduction to the surface code. SciPost Phys Lect Notes. 2022;49:1–34. Available from: https://scipost.org/SciPostPhysLectNotes.49

11. Bravyi S, Cross AW, Gambetta JM, Maslov D, Rall P, Yoder TJ. High-threshold and low-overhead fault-tolerant quantum memory. arXiv. 2024 Aug 30. Available from: https://arxiv.org/abs/2308.07915

12. Zu G, Sikander S, Portnoy E, Cross AW, Brown BJ. Non-Clifford and parallelizable fault-tolerant logical gates on constant and almost-constant rate homological quantum LDPC codes via higher symmetries. Available from: https://doi.org/10.48550/arXiv.2310.16982

13. Leymann F, Barzen F. The bitter truth about gate-based quantum algorithms in the NISQ era. Quantum Sci Technol. 2020;5(1):015010. Available from: https://iopscience.iop.org/article/10.1088/2058-9565/abae7d

14. Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, et al. Variational quantum algorithms. Nat Rev Phys. 2021;3(9):625–644. Available from: https://doi.org/10.1038/s42254-021-00348-9

15. Preskill J. Quantum Computing in the NISQ era and beyond. Quantum. 2018;2:79. Available from: https://doi.org/10.22331/q-2018-08-06-79

16. Yan B, Tan Z, Wei S, Jiang H, Wang W, Wang H, et al. Factoring integers with sublinear resources on a superconducting quantum processor. Available from: https://doi.org/10.48550/arXiv.2212.12372

17. Khattar T, Yosri N. A comment on "Factoring integers with sublinear resources on a superconducting quantum processor". arXiv. 2023 Jul 22. Available from: https://arxiv.org/abs/2307.09651

18. Bernstein DJ. Understanding brute force. Semantic Scholar. 2005. Available from: https://www.semanticscholar.org/paper/Understanding-brute-force-Bernstein/3bf374700ec98ea5d1b7658ec85c472f2fe4d952

19. Wang Z, Wei S, Long GL, Hano L. Variational quantum attacks threaten advanced encryption standard based symmetric cryptography. Sci China Inf Sci. 2022;65:200503. Available from: http://dx.doi.org/10.1007/s11432-022-3511-5

20. Wang Z, Wei S, Long GL. A quantum circuit design of AES requiring fewer quantum qubits and gate operations. Front Phys. 2022;17(4):41501. Available from: https://ui.adsabs.harvard.edu/abs/2022FrPhy..1741501W/abstract

21. Fluhrer S. Reassessing Grover's algorithm. Cryptology ePrint Archive, Paper 2017/811. 2017. Available from: https://ia.cr/2017/811

22. Haase Ch, Nill B, Pfaffenholz A. Lecture notes on lattice polytopes. TU Berlin; 2021. Available from: https://www2.mathematik.tu-darmstadt.de/~paffenholz/daten/preprints/20210628_Lattice_Polytopes.pdf

23. Peikert Ch. A Decade of Lattice Cryptography. Foundations and Trends in Theoretical Computer Science. 2016;10(4). Available from: https://eprint.iacr.org/2015/939.pdf

24. Silberman JH. An Introduction to Lattices, Lattice Reduction, and Lattice-Based Cryptography. Brown University; 2020. Available from: https://www.ias.edu/sites/default/files/Silverman_PCMI_Note_DistributionVersion_220705.pdf

25. Zheng Z. Modern Cryptography - 1. Springer; 2022. Available from: https://link.springer.com/book/10.1007/978-981-19-0920-7

26. Manohar N. Hardness of lattice problems for use in cryptography. Harvard University, Cambridge, Massachusetts; 2016. Available from: https://www.semanticscholar.org/paper/Hardness-of-Lattice-Problems-for-Use-in-Manohar/ca5adbdefc035619799a2ba05c49be452df60d38

27. Micciancio D. The hardness of the closest vector problem with preprocessing. IEEE Trans Inf Theory. 2001;47(3):1214–1222. Available from: https://cseweb.ucsd.edu/~daniele/papers/CVPP.pdf

28. Goldreich O, Goldwasser S, Halevi S. Public-key cryptosystems from lattice reduction problems. In: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO); 1997 Aug 17-21; Santa Barbara, California. 112–131.

29. Galbraith SD. Mathematics of public key cryptography. Version 2.0. University of Auckland; 2018. Available from: https://www.math.auckland.ac.nz/~sgal018/crypto-book/main.pdf

30. Babai L. On Lovasz lattice reduction and the nearest lattice point problem. Combinatorica. 1986;6(1):1–13. Available from: https://link.springer.com/article/10.1007/BF02579403

31. Ajtai M. Generating hard instances of lattice problems. Proc 28th ACM Symposium on Theory of Computing. 1996:99–108. Available from: https://dl.acm.org/doi/pdf/10.1145/237814.237838

32. Lang S. Algebra. Springer; 2002. Available from: https://link.springer.com/book/10.1007/978-1-4613-0041-0

33. Regev O. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. Journal of the ACM. 2009;56(6). Available from: https://doi.org/10.1145/1568318.1568324

34. Micciancio D, Regev O. Lattice-based Cryptography. In: Bernstein DJ, Buchmann J, Dahmen E, editors. Post-Quantum Cryptography. Springer; 2009;147–191. Available from: https://link.springer.com/chapter/10.1007/978-3-540-88702-7_5

35. Regev O. The Learning with Errors Problem. IEEE 25th Annual Conference on Computational Complexity. 2010. Available from: https://cims.nyu.edu/~regev/papers/lwesurvey.pdf

36. Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. In: EUROCRYPT 2010: Advances in Cryptology; 2010 May 30-Jun 3; French Riviera, France. 1–23.

37. Mukherjee A, Aikata A, Mert AC, Lee Y, Kwon S, Deryabin M, Roy SS. ModHE: Modular Homomorphic Encryption Using Module Lattices: Potentials and Limitations. Cryptology ePrint Archive, Paper 2023/895. 2023. Available from: https://eprint.iacr.org/2023/895

38. Avanzi R, Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, et al. CRYSTALS-Kyber - Algorithm specifications and supporting documentation (version 3.02). 2021. Available from: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf

39. Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, et al. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In: IEEE European Symposium on Security and Privacy (EuroS&P); 2018 Apr 2-6; London, UK. p. 1–18. Available from: http://dx.doi.org/10.1109/EuroSP.2018.00032

40. Bai S, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, et al. CRYSTALS-Dilithium - Algorithm specifications and supporting documentation (version 3.1). 2021. Available from: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

41. Richter M, Bertram M, Seidensticker J, Tschache A. A Mathematical Perspective on Post-Quantum Cryptography. Mathematics. 2022;10(15):2579. Available from: https://ideas.repec.org/a/gam/jmathe/v10y2022i15p2579-d871032.html

42. Jackson KA, Miller CA, Wang D. Evaluating the security of CRYSTALS-Dilithium in the quantum random oracle model. arXiv. 2024 Dec 13. Available from: https://arxiv.org/abs/2312.16619

43. National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. FIPS PUB 202. 2015. Available from: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

44. Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, et al. CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR Trans Cryptogr Hardw Embedded Syst. 2018;2018(1):238−268. Available from: https://doi.org/10.13154/tches.v2018.i1.238-268

45. Bakos-Lang EC. Worst-case to average-case reductions for the SIS problem: Tightness and security. University of Waterloo; 2019. Available from: https://dspacemainprd01.lib.uwaterloo.ca/server/api/core/bitstreams/70e489d8-fb0a-4f9c-a0c2-e6dd2d122bef/content

46. Scholten TL, William CJ, Moody D, Mosca M, Hurley W, Zeng WJ, et al. Assessing the Benefits and Risks of Quantum Computers. 2024. Available from: https://www.nist.gov/publications/assessing-benefits-and-risks-quantum-computers

47. Mosca M, Piani M. Quantum Threat Timeline Report. Global Risk Institute. 2023. Available from: https://policycommons.net/artifacts/11327929/quantum-threat-timeline-report-2023-dr/12216736/

48. US Government. Public Law 117 - 260 - Quantum Computing Cybersecurity Preparedness Act. 2022. Available from: https://www.govinfo.gov/content/pkg/PLAW-117publ260/pdf/PLAW-117publ260.pdf

49. US Government. Report on Post-Quantum Cryptography. 2024. Available from: https://www.whitehouse.gov/wp-content/uploads/2024/07/REF_PQC-Report_FINAL_Send.pdf

50. European Commission. Commission Recommendation on a Coordinated Implementation Roadmap for the transition to Post-Quantum Cryptography. 2024. Available from: https://www.europeansources.info/record/commission-recommendation-eu-2024-1101-on-a-coordinated-implementation-roadmap-for-the-transition-to-post-quantum-cryptography/

51. DigiCert, Ponemon Institute. Preparing for a safe post-quantum computing future: A global study. 2023. Available from: https://www.digicert.com/content/dam/digicert/pdfs/report/ponemon-preparing-safe-post-quantum-future-report-en-v1.pdf

52. Chen L, Jordan S, Liu Y, Moody D, Peralta R, Perlner R, et al. Report on Post-Quantum Cryptography. NISTIR 8105. 2016. Available from: http://dx.doi.org/10.6028/NIST.IR.8105

53. National Institute of Standards and Technology. Post-Quantum Cryptography. 2023. Available from: https://csrc.nist.gov/projects/post-quantum-cryptography

54. National Institute of Standards and Technology. FIPS 203. Module-Lattice-Based Key-Encapsulation Mechanism Standard. 2024. Available from: https://doi.org/10.6028/NIST.FIPS.203

55. National Institute of Standards and Technology. FIPS 204. Module-Lattice-Based Digital Signature Standard. 2024. Available from: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf

56. National Institute of Standards and Technology. Internal Report IR 8528. Status Report on the First Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process. 2024. Available from: https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8528.pdf

57. NTRU Prime Risk-Management Team. Risks of lattice KEMs. NTRU Prime. 2021. Available from: https://ntruprime.cr.yp.to/latticerisks-20211031.pdf

58. Goodin D. Post-quantum encryption contender is taken out by single-core PC and 1 hour. Ars Technica. 2022. Available from: https://arstechnica.com/information-technology/2022/08/sike-once-a-post-quantum-encryption-contender-is-koed-in-nist-smackdown/

59. Macaulay T, Henderson R. Cryptographic Agility in Practice. InfoSec Global. 2019.

60. National Cybersecurity Center of Excellence. Migration to Post-Quantum Cryptography. NIST Special Publication 1800-38A. 2023. Available from: https://www.nccoe.nist.gov/sites/default/files/2023-04/pqc-migration-nist-sp-1800-38a-preliminary-draft.pdf

61. Post-Quantum Cryptography Alliance. Available from: https://pqca.org/

62. Open Quantum Safe. Available from: https://openquantumsafe.org

63. Open Quantum Safe. Applications and protocols. Available from: https://openquantumsafe.org/applications/

64. liboqs. Available from: https://github.com/open-quantum-safe/liboqs

65. Post-Quantum Cryptography Coalition. Available from: https://www.mitre.org/news-insights/news-release/post-quantum-cryptography-coalition-launches

66. Sandwich. Available from: https://sandbox-quantum.github.io/sandwich/