Lecture Notes 7: Post-Quantum Cryptography and Homomorphic Encryption

Haiyang Xue and Yerkezhan Sartayeva

March 26, 2023

Summary

In this lecture, post-quantum cryptography and homomorphic encryption are discussed. First, a definition of post-quantum cryptography is given, and reasons why it is necessary are described, followed by existing solutions and NIST's status on post-quantum cryptography standardisation. Then an overview of fully homomorphic and partially homomorphic encryption is provided, followed by a discussion on bootstrapping and four generations of fully homomorphic encryption.

1 Post-Quantum Cryptography

Cryptosystems are essential for securing communications between two parties and are widely in use today. Even though they are known to be computationally secure, quantum computers will pose a significant threat to the security of existing cryptosystems once they are available, which, with a funding of 1 billion USD, could happen in the next two decades [MP22]. Therefore, it is necessary to develop new cryptographic algorithms that are resistant to quantum attacks. This is known as post-quantum cryptography.

1.1 A Big Picture of Cryptographic Algorithms

When a client sends a request to a server, they agree on cryptographic algorithms and exchange security parameters over TLS (Transport Layer Security) to ensure secure communication. This is done to ensure the CIA triad, i.e., confidentiality, integrity and authentication. Confidentiality is needed to make sure adversaries cannot eavesdrop on the messages exchanged between the client and the server. Integrity is needed to prevent adversaries from changing the original message. Authentication is needed to verify the identity of the sender. Based on Kerckhoff's principle, all cryptographic algorithms are public, only their keys are private.

Suppose A wants to send a message m to B. First, A and B will use a key exchange mechanism (this will be discussed in more detail later) to share symmetric keys k_{enc} (encryption key) and k_{auth} (authentication key). Then A will use a symmetric encryption algorithm,

e.g., AES-128 (Advanced Encryption Standard) [DR02], to encrypt m with k_{enc} to generate a ciphertext c as $c \leftarrow Enc_{k_{enc}}(m)$. A will also generate a MAC (message authentication code) tag using a MAC tag generation algorithm Mac on c with an authentication key k_{auth} to prove it has access to the key and that c was indeed sent from A as $t \leftarrow Mac_{k_{auth}}(H(c))$, where H is a collision-resistant hash function. Hash functions are needed for domain extension, i.e., to allow generating MACs on messages of arbitrary length. Note that the hash function was applied on the ciphertext c, which is known as the "encrypt-then-authenticate" approach. According to theorem 5.7 in [KL20], MAC schemes based on the "encrypt-then-authenticate" approach are secure. When B receives a ciphertext and its tag c' and t', respectively, B needs to make sure that it originated from A, so B calculates a MAC tag $t' \leftarrow Mac_{k_{auth}}(c')$ for the received ciphertext c' and runs a MAC verification algorithm $Vrfy_{k_{auth}}(c', t')$. If $Vrfy_{k_{auth}}(c', t') = 1$, then $t = t'$ and $c = c'$, so MAC tag verification is successful. After verification, B decrypts c' using k_{enc} to obtain the original message $m' = Dec_{k_{enc}}(c') = m$. Symmetric encryption ensures confidentiality, while MACs ensure message integrity and authentication because forging a modified message $m' \neq m$ such that its tag $t' = t$ is computationally hard [KL20].

A major issue in symmetric encryption is how to securely share encryption and authentication keys. Public-key encryption schemes have computationally secure key exchange mechanisms and also support encryption and message authentication, but because they are more computationally expensive, they are used for authenticating public security parameters and key exchange, while symmetric encryption is used for encrypting the original message after the secret key is agreed on using public-key cryptography [KL20]. In public-key cryptography, each party has two keys: a public key and a private key. Anyone can get hold of public keys and use them to encrypt messages. Anyone can encrypt a message m using $A's$ public key pk_A and an asymmetric encryption scheme like RSA (Rivest-Shamir-Adleman) to obtain the ciphertext as $c \leftarrow Enc_{pk_A}(m)$, but only A can recover the original message using its private key $m \leftarrow Dec_{sk_A}(c)$. Recovering the private key based on the public key is computationally hard under the discrete logarithm assumption, which will be discussed later.

Before A and B can communicate, A and B exchange shared keys for symmetric encryption and message authentication. One way to do this is to use public-key encryption, e.g., padded RSA or ElGamal-based encryption [Elg85]. If A wants to communicate with B, A picks a symmetric random key, encrypts it with B's public key and sends the encrypted key to B. B recovers the shared key after verifying it originated from A. Another way for A and B to obtain shared keys is to participate in authenticated key exchange, e.g., the Diffie-Hellman key exchange [Hel76].

When sharing public security parameters such as public keys in public-key cryptography, it is important to verify their integrity and the identity of the sender. This can be achieved using digital signatures, which are analogous to MAC tags in private-key cryptography. If A wants to associate its identity with a message m, it signs it with its secret key sk_A to generate its signature as $\sigma \leftarrow Sign_{sk_A}(H(m))$, where H is a collision-resistant hash function. Applying a hash function on the message ensures it is of fixed length, which reduces the cost of running a digital signature algorithm. According to theorem 13.4 in [KL20], signature schemes that are secure for messages of length l and use collision-resistant hash functions before signing messages are secure. Then anyone can verify that σ is the signature of m using a digital signature verification algorithm $Vrfy_{vk_A}(H(m))$, where H is a hash function

and vk_A is the verification key for A's signatures, which is public. If $Vrfy_{vk_A}(H(m)) = 1$, then digital signature verification is successful, so the receiver can decrypt the ciphertext and recover the original message using its secret key for decryption, which must be distinct from the secret key used for signature generation [KL20]. Digital signatures are similar to MACs, but there are three important distinctions:

- A digital signature can be verified by anyone, whereas a MAC tag can only be verified by parties sharing k_{auth}.
- A digital signature can only be generated by one party, whereas a MAC tag can be generated by either party sharing k_{auth}.
- Digital signatures ensure non-repudiation, meaning that once A signs a message m, it cannot deny having signed it because its verification key is tied to its secret key. A MAC tag, however, could have been generated by either party, so it is hard to attribute authorship of a message to any party.

An important issue with public-key cryptography is how to associate public keys with their owners and protect their integrity. This can be done using trusted third-party certificate authorities (CAs). If B wants to prove its identity to A, it sends its certificate $Cert_B$ to A signed by a trusted third party T with its private key sk_T. Then A finds the public key pk_T of T in its local storage (browsers come hardwired with the public keys of trusted authorities) and verifies $Cert_B$. If certificate verification is successful, A extracts B's public key from the certificate and can use it for encryption, for example.

As discussed previously, the basic goal of cryptography is to ensure message privacy, message integrity and message authentication. Message privacy is ensured by encryption, and integrity and authentication are ensured by digital signatures or MACs. There are three levels of security for message privacy:

- IND-eva (indistinguishability in the presence of an eavesdropper): given two different messages and the ciphertext of one of the messages, the adversary cannot tell which message the ciphertext belongs to (definition 3.8 in [KL20]
- IND-CPA (indistinguishability in chosen-plaintext attacks): given oracle access to the encryption algorithm of an encryption scheme, meaning even if the adversary is allowed to choose any plaintext it wants, it still cannot match a ciphertext to its original message (definition 3.21 in [KL20]
- IND-CCA (indistinguishability in chosen-ciphertext attacks): given oracle access to the encryption algorithm and the decryption algorithm of an encryption scheme, meaning even if the adversary is allowed to choose any plaintext it wants and can decrypt any ciphertext that it eavesdrops by but did not request itself, it still cannot match a ciphertext to its original message (definition 5.1 in [KL20]).

All three apply to private-key encryption schemes, whereas only IND-CPA and IND-CCA security are applicable to public-key encryption schemes. The security of message authentication schemes, which ensure message integrity and authentication, is defined by their unforgeability. According to definition 4.2 in [KL20], a MAC scheme is secure if the

likelihood of an adversary forging a tag for a message of its choice given oracle access to the Mac algorithm of the scheme is negligible. This is referred to as uf-CMA (existentially unforgeable under an adaptive chosen-message attack) security. A similar definition applies to digital signatures (definition 13.2 in [KL20]). According to definition 5.3 in [KL20], a private-key encryption scheme that is IND-CCA-secure and is unforgeable is called an authenticated encryption scheme. As for the security of hash functions, it is defined by their collision resistance (definition 6.2 in [KL20]), which means that it is hard to find a pair of messages (m, m') such that $H(m) = H(m')$, where H is a hash function. This is important because it means that, given the digest of a message, the adversary should not be able to recover the original message in MAC and digital signature schemes. There are two types of hash functions: keyed and unkeyed. Keyed hash functions accept a random key as a parameter, whereas unkeyed hash functions do not. Keyed hash functions are stronger since they have no hardcoded collisions, but unkeyed functions are used in practice because they are computationally collision-resistant [KL20].

Different types of public-key and private-key cryptographic algorithms are used in practice. Symmetric-key encryption algorithms can either be based on block ciphers or stream ciphers. Block ciphers accept messages of fixed length, whereas stream ciphers can accept inputs of arbitrary length. Block-cipher-based symmetric-key encryption algorithms used in practice include AES-CBC (AES with Cipher-Block Chaining mode) and AES-CTR (AES with Counter block cipher mode). Symmetric-key-based message authentication schemes used in practice include CBC-MAC (cipher block chaining message authentication code) and HMAC (hash-based message authentication code). Examples of authenticated encryption schemes used in practice are AES-GCM (AES with Galois/Counter Mode), AES-CCM (AES with Counter with Cipher Block Chaining-Message Authentication Code) and AES-OCB (AES with offset codebook mode). As for public-key cryptography, hashed RSA-3072 [RSA78a], Schnorr signature scheme and ECDSA (Elliptic Curve Digital Signature Algorithm) [JMV01] are used for digital signatures and passed RSA and ElGamal-based encryption schemes are used for public-key encryption. Key exchange mechanisms based on Diffie-Hellman key exchange can be used used to share symmetric keys for encryption. When it comes to practical constructions of hash functions, examples include SHA2-256 [Bry12], SHA2-512, SHA3-256, etc.

Cryptographic algorithms are now widely used for secure communications over the Internet in multiple protocols. For example, the TLS protocol is used for encrypted communication over the HTTPS protocol. The SSH (Secure Socket Shell) protocol allows users to remotely work on another computer in a secure manner. IPSec (Internet Protocol Security) is similar to TLS but operates over an Internet Protocol network, i.e., it is positioned one layer lower in the OSI hierarchy than TLS. Thanks to these algorithms, people can safely perform financial transactions on the Internet, access sensitive data, send text messages, etc. Software companies also use digital signatures to authenticate their software when it is downloaded by users.

The security of cryptographic algorithms depends on the hardness of the problems they are based on. Algorithms used in practice are not perfectly secure, they are computationally hard to break, meaning their security is predicated on the computational resources of classical computers. For example, RSA digital signature schemes and RSA encryption schemes are based on the assumption that the following problems are hard:

- integer factorisation;
- the RSA problem, which states that, given Z_N^*, which is a group consisting of integers co-prime to N of order $\phi(N)$, where N is the product of two large primes p and q, e is an integer that is co-prime to $\phi(N)$, and ϕ is the Euler totient function, recovering $x \in Z_N^*$ from y, where $y = x^e \bmod N$, is computationally hard.

Diffie-Hellman key exchange is predicated on the hardness of:

- the discrete logarithm (DL) problem, which states that, given g^x, where g is a generator of a finite cyclic prime-order group G, and x is an integer, recovering x from g^x is hard;
- the decisional Diffie-Hellman (DDH) problem, which states that distinguishing between g^{ab} and g^z given g^a, g^b, g^z is hard, where g^a, g^b, g^z are chosen uniformly at random and g is a generator of a finite cyclic prime-order group G.

Elliptic-curve-based schemes are also based on the assumption that DL and DDH problems are hard. As for the security of symmetric-key schemes, it is based on the security of pseudorandom functions and the collision resistance of the hash functions they use.

1.2 Quantum Threats

Computational security assumes that adversarial attacks are carried out on classical computers, i.e., it does not take quantum computers into account. However, quantum computers are expected to pose a significant threat to the security of existing cryptosystems since they can become powerful enough to run exponentially more operations than classical computers. In classical computers, information is represented in terms of bits, where a bit is either 0 or 1. The equivalent of a bit in quantum computing is called a qubit. The computational advantage of quantum computers stems from the fact that, unlike a classical bit, a qubit can represent 0, 1 or any proportion of 0 and 1 in between, meaning that it can work with multiple states simultaneously. The computational supremacy of quantum computing has already been demonstrated by Google in 2019 [AAB+19]. Their research shows that a quantum computer working with 53 qubits can take 200 seconds to solve a task that would have taken a classical computer 10,000 years. However, quantum computers are still in development and have not been applied on real problems since they have a number of issues such as the inability to correct errors and allow qubits to interact [Bal21]. Nevertheless, quantum computers are expected to become much more powerful in a few decades, and this section will discuss in more detail how this computational power can jeopardise the security of existing cryptosystems.

1.2.1 Shor's Algorithm

One of the most devastating algorithms for existing cryptosystems has been proposed by Shor [Sho94]. The author designed Las Vegas algorithms for finding discrete logarithms and factors of large numbers in polynomial time on quantum computers, which has devastating implications for public-key cryptosystems based on these problems such as RSA, ECC (elliptic curve cryptography), FFC (finite field cryptography) and Diffie-Hellman key exchange.

Shor's algorithm factors integers by solving the period-finding problem using Fast Fourier Transform [ST21]. The period-finding problem lies in finding the smallest integer $r \in [0, N]$ such that $a^r \bmod N = 1$, where N and a are positive integers that share no common factors, and $a \in [0, N-1]$. In other words, r is the period of a function $f : e \to a^e \bmod N$. Shor's algorithm can solve this problem in $O(d^3)$ time, where $d = \log_2 N$, i.e., d is the number of digits in N. It computes a quantum Fourier transform to approximate the superposition of periods of f and measures it to obtain a random period [BL17]. If one can find the order r of a cyclic group $< a >$, where $a \in Z_N$ is a generator of Z_N^*, according to proposition 9.55 in [KL20], $r|\phi(N)$, so one can reveal the prime factors of N based on $\phi(N)$ because $\phi(N) = N\Pi_{p|N}(1 - \frac{1}{p})$. In the case of RSA, if $N = pq$, then $\phi(N) = (p-1)(q-1)$. If one finds the order r of a cyclic group with one generator $a \in Z_N$ and finds the order r' of a cyclic group with another generator a', one can recover another factor of $(p-1)(q-1)$ with high probability [BL17]. Knowing the factors of N enables the adversary to find the secret key d of any public key e since $[ed \bmod \phi(N)] = 1$, so the adversary can recover d by finding the multiplicative inverse of e with respect to $\phi(N)$.

According to Bernstein and Lange [BL17], the period-finding routine in Shor's algorithm can also be used to find discrete logarithms. Let g be a generator of a group G of prime order q. Given $h = g^k \in G$, Shor's algorithm can be used to recover k. To demonstrate this, suppose there is a periodic function $f : (x_1, x_2) \to g^{x_1} h^{x_2} \bmod q$, where $h = g^k \bmod q$. Then Shor's algorithm can be used to find the period of f, i.e., (ω_1, ω_2) such that $f(x_1, x_2) = f(x_1 + \omega_1, x_2 + \omega_2)$. Then $g^{\omega_1} h^{\omega_2} = 1 = g^{\omega_1} g^{k\omega_2} = g^{\omega_1 + k\omega_2}$. Therefore, $k\omega_2 \equiv -\omega_1 \bmod q$, meaning that k can be recovered as $k = -\omega_1/\omega_2 \bmod q$ by finding a non-zero pair (ω_1, ω_2). If multiplication modulus q is replaced with point addition on an elliptic curve modulus q, Shor's algorithm jeopardises elliptic curve cryptography as well. In general, Shor's algorithm compromises the security of cryptosystems that rely on the hardness of integer factorisation and discrete logarithm problems, i.e., RSA, ECC, Diffie-Hellman key exchange, FFC, etc., i.e., public-key cryptographic algorithms. Since symmetric-key cryptography is not predicated on these problems, Shor's algorithm does not pose a threat to it. Bernstein and Lange state that it will take billions of operations on quantum computers to break RSA and ECC. Quantum computing might not be able to meet these computational requirements in the future, but there is no guarantee that quantum computers will not become powerful enough to compromise the security of existing systems, so quantum-computing-resistant algorithms are required.

1.2.2 Grover's Algorithm

Another quantum threat has been described by Grover [Gro96], who showed that database search can be performed in $O(\sqrt{N})$ time on quantum computers, where N is the size of the database. This may not seem problematic since database search time complexity can be improved to $O(\log N)$ if the database is sorted in $O(N \log N)$ time, but what if sorting is not possible? According to [BL17], Grover's algorithm can be better described as searching for the roots of a function f, i.e., finding x such that $f(x) = 0$. Grover's algorithm only requires $O(\sqrt{N})$ calculations of f on superpositions of inputs and has been shown by Grover to be within a constant factor of the optimal quantum algorithm for database search. Grover's algorithm poses a threat to symmetric-key cryptography. This can be demonstrated with

AES. Suppose there is a ciphertext $c = AES_k(p_1, p_2)$ for some plaintexts p_1 and p_2. Then Grover's algorithm can reduce the number of operations needed to find a k that maps p_1 and p_2 to c from $O(N)$ to $O(\sqrt{N})$), meaning that, if the key k is 128 bits in length, for example, then only $\sqrt{2^{128}} = 2^{64}$ operations are required to find the key. This problem extends to other symmetric-key encryption algorithms like triple DES (Data Encryption Algorithm) that are parameterised by secret keys. To address this problem, the key size should simply be doubled [Flu17] to bring the complexity of key search from $O(2^{N/2})$ to $O(2^{(N/2)\cdot 2}) = O(2^N)$. Information-theoretic MACs like GMAC (Galois Message Authentication Code) [MV04] and Poly1305 [Ber05] assume the adversary has unlimited computational power, so they are secure against quantum computing attacks [BL17]. However, hash functions must also use longer outputs because, given a hash h of a message m, searching for m will take $O(\sqrt{|h|})$ operations [Pre22] with Grover's algorithm.

In summary, it is evident that public-key cryptography is under significant threat from Shor's and Grover's algorithms, whereas private-key cryptography seems to only be threatened by Grover's algorithm, but this issue can be addressed by simply doubling the key size, as discussed previously. Grover showed that his algorithm is optimal, so doubling the key size should be sufficient to ensure quantum security. However, this makes these schemes more computationally expensive. This change is rarely noticeable in symmetric-key cryptography but is more significant in public-key cryptography [BL17]. To address the quantum threat of Shor's algorithm, on the other hand, requires replacing existing cryptographic algorithms with new ones that are quantum-safe.

1.3 Post-Quantum Cryptography and Quantum Key Distribution

Shor's and Grover's algorithms motivated researchers to design quantum-safe cryptosystems. There are two approaches to do this. One way is to design cryptosystems for quantum computers and build physical quantum key distribution (QKD) channels for symmetric-key cryptography, assuming devices at both ends are quantum computers [MNR+20].

Definition 1.1 (Quantum Key Distribution). *Quantum key distribution is defined as the act of producing and sharing symmetric cryptographic keys between two remote quantum machines based on quantum physics [MNR+20].*

According to [MNR+20], quantum key agreement is implemented at the lowest level of the QKD network architecture and will be integrated into optical communications. However, the main obstacle in QKD development is that the key rate is currently restricted and depends on the length of the channel. QKD channels would enable a continuous exchange of keys between users, which would allow for larger key sizes and, therefore, stronger security. However, mechanisms on how to allow users to establish identical keys from a continuous stream would have to be designed.

Aside from quantum cryptography, another option is to design cryptographic schemes that cannot be broken with quantum computers. This is referred to as post-quantum cryptography and will be the focus of this part of the lecture.

Definition 1.2 (Post-Quantum Cryptography). *Post-quantum cryptography refers to cryptosystems resistant to attacks by quantum computers [MR22].*

The security of existing cryptosystems is predicated on the hardness of computationally hard problems such as integer factorisation and finding discrete logarithms. Similarly, quantum-safe cryptosystems are based on quantum hard problems. Examples of potential quantum hard problems are described in section 1.6.

1.4 Why Post-Quantum Cryptography?

In 2016, The National Institute of Standards and Technology (NIST) initiated a standardisation process of post-quantum cryptography, which is expected to be finalised by 2024. Even though currently quantum computers are not powerful enough to break existing cryptosystems [MR22], there are significant reasons to start developing quantum-secure cryptosystems. First, active developments are being made in making quantum computers more powerful. According to NIST's report on post-quantum cryptography [CCJ+16], quantum computers will be able to break 2000-bit RSA in a few hours by 2030. IBM has been making strides in increasing the computational power of their quantum chips and has recently made a breakthrough by building a 433-qubit quantum computer [Bal21]. Boudot et al. [BGG+20] recently demonstrated the ability to factorise a 795-bit RSA number and solve discrete logarithm problems of the same size. These examples demonstrate that the performance of quantum computers will only improve in the future, rendering existing public-key cryptosystems insecure. Another reason why measures must be taken to address quantum threats now is that there might be adversaries that have been collecting encrypted data for years and are waiting to decrypt it with quantum computers [MR22].

1.5 The Status of NIST's Project on Post-Quantum Cryptography

NIST launched the standardisation process of post-quantum cryptography in 2016 in the form of a public competition, which accepted public-key encryption schemes, digital signature schemes and key exchange mechanisms [MR22]. Three rounds of the competition have been completed so far. The first round received 86 submissions, out of which 69 met the minimum requirements. Most submissions were based on lattices and error-correcting codes. After conducting collaborative security analysis, in 2019 [AAAS+19], NIST announced that 26 out of 69 systems would proceed to the second round of the competition for evaluation. In 2020 [MAA+20], seven finalists were identified and proceeded to the third stage. In 2022 [AAC+22], it was announced that CRYSTALS-Kyber would be standardised as a post-quantum public-key encryption algorithm and a key exchange algorithm. As for digital signatures, CRYSTALS–Dilithium, FALCON and SPHINCS+ were selected for standardisation. Four more key exchange mechanisms were chosen to proceed to the fourth round. NIST continues its call for proposals to diversify its portfolio of post-quantum cryptographic algorithms but expects to publish the final standard by 2024. Kyber is based on module learning with errors, and Dilithium is a lattice-based digital signature scheme.

1.6 Candidates for Post-Quantum Cryptography

As mentioned previously, post-quantum cryptography is based on quantum hard problems. This section will discuss candidate problems for post-quantum cryptography.

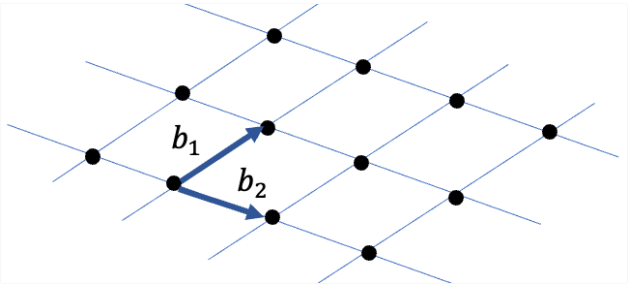

Figure 1: Example of a 2D lattice with basis vectors b_1 and b_2.

1.6.1 Lattice-based Cryptography

One promising direction in post-quantum cryptography is the use of lattices. As opposed to classical cryptography, which is based on average-case problems, lattice-based cryptography is based on worst-case problems [App16]. For example, integer factorisation is hard on average over a certain distribution. That is why lattice-based cryptography is a promising candidate for post-quantum cryptography.

Definition 1.3 (Lattice). *According to [MR22], a lattice $\mathcal{L}$ is a set of points where each point is a linear combination of $n \in \mathbb{N}$ vectors from a set of vectors $B = \{b_1, ..., b_n\}$, which are called basis vectors and where B is called the basis of the lattice.*

$$\mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{v=1}^{n} c_v b_v : c_v \in \mathbb{Z} \right\}$$

Please refer to Figure 1 for an illustration of a lattice in 2D space.

There are two major quantum NP-hard problems associated with lattices: shortest vector problem (SVP) and closest vector problem (CVP). The complexity of the problems lies in the fact that a lattice can be represented with infinitely many bases. The longer and more skewed the basis vectors of a lattice are, the harder it is to solve SVP and CVP. Please refer to [ABSS97, Mic01] for proofs of NP-hardness of CVP and SVP respectively.

Theorem 1.1. *For sufficiently large n, SVP and CVP are NP-hard [Mic01, ABSS97].*

As the name suggests, SVP lies in finding the shortest non-zero vector in a lattice $\mathcal{L}$ given that its basis is public. In this case, the secret key is one of the shortest vectors. CVP requires one to find the closest vector $CV(t) \in \mathcal{L}$ to a vector $t \in \mathbb{R}^n$, i.e., such that $|CV(t) - t|$ is minimised. [ABSS97] showed that even approximating the closest vector is NP-hard. Suppose α defines the quality of an estimate, i.e., $|CV(t) - t| < \alpha \min_{v \in \mathcal{L}} |v - t|$. As α and n increase, the cost of approximating the closest vector becomes prohibitive, as is shown in Figure 2.

CVP lies at the heart of many lattice-based encryption schemes. For example, NTRU [HPS98] uses a p-coefficient polynomial $z = z_0 + z_1 x + z_2 x^2 + ... + z_{p-1} x^{p-1}$ as the public key and a pair of polynomials (d, e) as the secret key, where $z_i \in [0, q-1]$. The ciphertext for a secret key (d, e) is then calculated as $c = ((zd + e) \bmod x_p - 1) \bmod q$, where $\bmod x_p - 1$ means that $x_j = x_{j \bmod p}$. Then polynomial pairs (u, v) that satisfy $((zu - v) \bmod x_p - 1) \bmod q = 0$

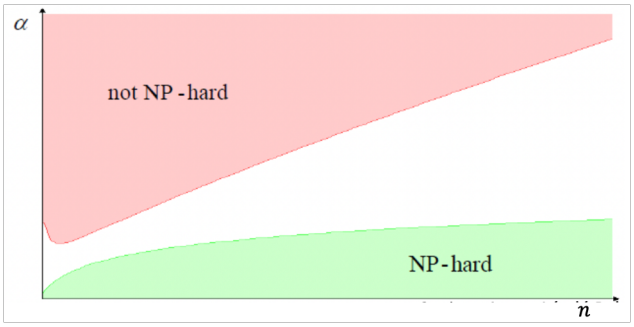

Figure 2: Approximating $CV(t)$ in a lattice becomes an NP-hard problem for large lattices and a low margin of error

constitute a $2p$-dimensional lattice $\mathcal{L}$, and the adversary's goal is to recover the secret key (d, e) based on the ciphertext c by finding the closest vector to $(0, c)$, which is $(d, c-e)$ [BL17].

Lattice-based cryptography supports homomorphic encryption and offers security parameters of reasonable size and runs relatively efficiently [MR22]. There have been attempts at designing lattice-based digital signature schemes but they suffer from large signature sizes [BL17]. One of the most promising lattice-based digital signature schemes is Lyubashevsky's system [Lyu12], which has shorter signature sizes but has no thorough security proof.

1.6.2 Hash-based Signatures

As discussed previously, for a secure hash function H, recovering the pre-image x of a hash $H(x)$ is computationally hard. Lamport [Lam79] designed a one-time digital signature scheme based on this problem. In his scheme, the message space is $[0, 1]$, meaning that signatures are computed bit-by-bit. For a bit j, two random strings x_{j0} and x_{j1} are picked and act as the secret key of that bit. The verification key is $(h(x_{j0}), h(x_{j1}))$, where h is a collision-resistant hash function. If, after decryption, the receiver discovers that a bit $x_j = b$, where b is either 0 or 1, the receiver computes the hash value of x_{jb} and compares it to $h(x_{jb})$ in the public key. Overall, for an m-bit message, the sender must pick $2m$ random strings as the secret key $SK = (x_{10}, x_{11}, x_{20}, x_{21}, ..., x_{m0}, x_{m1})$ and compute their hash values $VK = (h(x_{10}), h(x_{11}), h(x_{20}), h(x_{21}), ..., h(x_{m0}), h(x_{m1}))$. For example, for a message 011101, $SK = (x_{10}, x_{21}, x_{31}, x_{41}, x_{50}, x_{61})$. It is important to pick distinct secret keys for each bit and never reuse any x_{ij}, otherwise, the security of the signature will degrade [BL17]. That is why this digital signature scheme is called a one-time scheme. A major problem with it though is the large size of the verification key. If each hash value in the verification key has 256 bits, then the total size of SK for a 256-bit message is $256 \times 256 \times 2 = 131072$ bits, or around 16kB, and the total size of VK is also 16kB. To address this issue, Merkle [Mer01] proposed a binary tree structure for verifying a signature based on the tree's root, i.e., the verification key is the root of the Merkle tree. Suppose a $VK = (V_1, V_2, ..., V_{2^k})$ has 2^k keys/hashes. Then the hashes are split into consecutive pairs (V_i, V_j), which constitute the leaves of a binary tree. Then, for each pair, a new hash value is computed and set as the parent of the pair. The process is repeated recursively for each layer until the root hash is computed. The root hash is made public as the verification key. The signature consists of nodes needed to pave a path

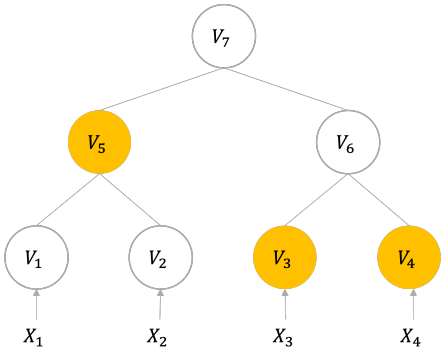

Figure 3: Demonstration of Lamport's one-time signature based on a Merkle tree. Yellow nodes are part of the signature of X_3.

to the Merkle tree root. To illustrate this, consider a sample binary tree as in Figure 3. To verify the hash of X_3, only V_3, V_4 and V_5 need to be shared because V_6 can be computed using V_3 and V_4. Then $h(V_5, V_6)$ gives V_7. If V_7 matches the hash value in the verification key, signature verification is successful.

Because the same keys cannot be reused in this scheme, it is stateful, meaning it must keep track of all previously used keys, which can be problematic. Stateless hash-based signatures, on the other hand, have large signature sizes [BL17].

1.6.3 Isogeny-based Cryptography

Isogeny-based cryptography is based on elliptic curves. It is an attractive option for post-quantum cryptography because it has much smaller key sizes that are comparable to those of existing systems [DK22]. Before discussing isogeny-based cryptography, let us briefly go over the definition of elliptic curves.

Definition 1.4 (Elliptic Curve). *The set of solutions of a polynomial equation in $a, b \in \mathbb{Z}_p$ of the form $y^2 = x^3 + ax + b \mod p$ is defined as an elliptic curve, where $4a^3 + 27b^2 \neq 0 \mod p$ and $p \geq 5$ is a prime [KL20]. The addition of two points P and Q on an elliptic curve E is defined as the projection of the intersection point of a line that goes through P and Q on E (i.e., the projection of a third point that is neither P nor Q but intersects with E). According to Bezout's theorem, any line cuts an elliptic curve E in exactly three points.*

Please refer to Figure 4 for an illustration of point addition. In the figure, $P = P_1$, $Q = P_2$, $P_1 + P_2 = -P_3$, and $-P_3$ is a projection, or image, of P_3.

Definition 1.5 (Scalar Multiplication of a Point on an Elliptic Curve). *Scalar multiplication of a point P by a scalar n on an elliptic curve is defined as $[n]P = \underbrace{P + ... + P}_{n}$, where $+$ denotes point addition.*

In traditional ECC, a point on an elliptic curve is mapped onto its image on the same curve. Similarly, at a high level, in isogeny-based cryptography, a whole elliptic curve is mapped onto its image in the elliptic curve space. A brief description of isogenies will be given in this section but readers are encouraged to refer to original sources for a detailed

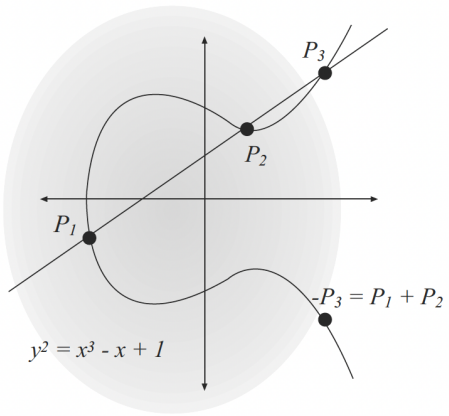

Figure 4: Illustration of point addition on an elliptic curve [KL20].

explanation. To understand isogenies, it is necessary to define a few terms first. The following definitions are taken from [DK22] and [ISA19].

Definition 1.6 (Rational Map). *A rational map ϕ is a one-to-one key-value mapping between two curves, where the key is any point $P = (x, y)$ from the input curve and the value is the ratio of polynomials p_1, p_2, q_1 and q_2, which forms another elliptic curve, as follows: $\phi(x_1, y_1) = (\frac{p_1(x,y)}{q_1(x,y)}, \frac{p_2(x,y)}{q_2(x,y)}) = (x_2, y_2)$.*

Definition 1.7 (Group Homomorphism). *Group homomorphism is defined as preserving addition, i.e., making sure the value of a function on the addition of two points is equivalent to the sum of the function values computed on the two points separately: $\phi(P+Q) = \phi(P)+\phi(Q)$.*

Definition 1.8 (Isogeny). *An isogeny is a rational map $\phi : E_0 \to E_1$ between two curves E_0 and E_1 that is also a group homomorphism.*

In other words, an isogeny defines a mapping between a curve and its image, which is itself an elliptic curve. Please refer to Figure 5 for a visual example.

Definition 1.9 (Isomorphism). *An isogeny is said to be an isomorphism if it is a bijection, i.e., it is a one-to-one mapping. Its curves are said to be isomorphic.*

Scalar multiplication by n is also an isogeny with the definition of addition revised accordingly for elliptic curve addition. One useful property of isomorphic elliptic curves is that, even though they are different, their j-invariants are the same, and the j-invariant can be used as the shared key. The proof of the above theorem is beyond the scope of the lecture.

Definition 1.10 (j-invariant of an Elliptic Curve). *The the j-invariant of an elliptic curve $E : y^2 = x^3 + ax + b$ is*

$$j(E) = \frac{6912a^3}{4a^3 + 27b^2} \tag{1}$$

Theorem 1.2. *Isomorphic elliptic curves have the same j-invariant.*

This formed the basis of an isogeny-based key exchange mechanism called SIDH (Supersingular Isogeny-Based Diffie-Hellman) [JDF11], which was submitted to NIST's competition. However, it was recently shown to be insecure [CD22], but other isogeny-based schemes like CSIDH [CLM+18] have not yet been shown to be vulnerable.

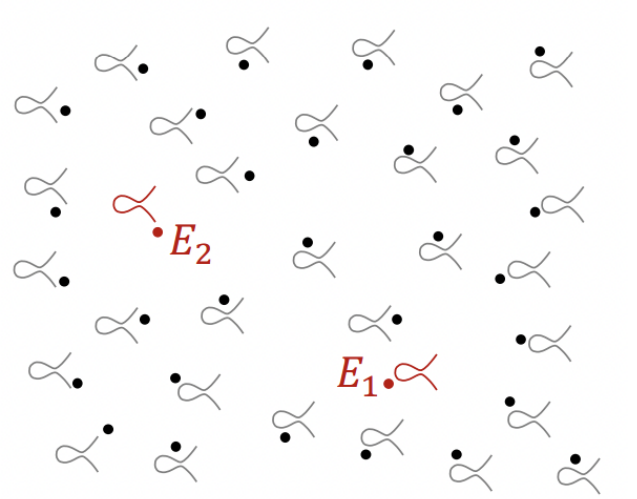

Figure 5: Illustration of an isogeny in the elliptic curve space.

1.7 Future Work

Even though NIST is expected to finalise the post-quantum cryptography standard by 2024, current schemes still have a number of limitations. For example, digital signature schemes selected for standardisation are slow or have large sizes, so one direction for future work is to design better post-quantum digital signature schemes. Another major milestone to achieve is to integrate post-quantum cryptography into the TLS layer for future quantum-safe communications. Finally, more post-quantum cryptographic algorithms may need to be designed, e.g., for authenticated key exchange.

1.8 Summary

In summary, advances in quantum computing pose a significant threat to existing cryptosystems, rendering them insecure in the future. Therefore, measures must be taken to design post-quantum cryptographic standards for quantum-safe communications. NIST has undertaken major steps towards standardisation with promising solutions based on quantum hard problems related to lattices, error-correction codes, etc., but still, more work is needed to integrate post-quantum cryptography into existing systems and make it more efficient.

2 Fully Homomorphic Encryption

The approach of encryption is a crucial mechanism to preserve the privacy of sensitive information/data. Even if the encrypted data is stored in / or accessible by an untrusted third party, such as the cloud server, the data is still secure.

It will be very convenient if the could server could operate on the encrypted data directly. In this way, the user does not need to download the data, operate on the data, encrypt the data, and at last upload the encrypted data to the server again. What we need is the so-called homomorphic encryption scheme. Homomorphic Encryption (HE), which is a special kind of

encryption scheme, can address this problem as it allows any third party to operate on the encrypted data without decrypting it.

The term homomorphism on encrypted data was used for the first time in 1978 by Rivest et al. [RAD+78] as a solution to compute without decrypting problems. This had led to a lot of work from all over the world to design homomorphic encryption with a large set of or unlimited operations. It was until 2009 that Craig Gentry [Gen09] proposed the first fully homomorphic encryption based on lattice. Before Gentry's work, all the attempts allowed either one type of operation or a limited number of operations on the encrypted data. All the attempts can be categorized into three types: 1) partially homomorphic encryption which only supports one type of operations with unlimited times; 2) somewhat homomorphic encryption that allows some operations with limited times; and 3) fully homomorphic encryption which allows all the operations with an unlimited number of times.

We will introduce what is fully homomorphic encryption (FHE), start from partially homomorphic encryption, discuss somewhat homomorphic encryption and show how to lift it to FHE via the bootstrapping technique.

2.1 What is fully homomorphic encryption (FHE)

Informally, FHE allows for arbitrary computations on encrypted data. Computing on encrypted data means that if $c_1, c_2, \cdots, c_n$ are encryptions of $m_1, m_2, \cdots, m_n$ and user wants to obtain the encryption of $f(m_1, \cdots, m_n)$ for some function f, it is possible to compute on $c_1, c_2, \cdots, c_n$ obtaining a ciphertext which decrypts to $f(m_1, \cdots, m_n)$.

We formally define the syntax of fully homomorphic encryption scheme. A HE encryption scheme HE consists of PPT algorithms $(\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ such that the followings hold.

$\mathsf{HE.KeyGen}(1^\lambda)$ On input 1^λ, generate secret key sk, public key pk, and evaluation key evk, respectively.

$\mathsf{HE.Enc}(\mathsf{pk}, m)$ This is a probabilistic or deterministic algorithm. On input public key pk and message m, generate and output c as the ciphertext.

$\mathsf{HE.Dec}(\mathsf{sk}, c)$ This is a deterministic algorithm. On input secret key sk and the ciphertext c, return the corresponding message m or $\perp$.

$\mathsf{HE.Eval}(\mathsf{pk}, f, \mathsf{evk}, c_1, \cdots, c_n)$ On input public key pk, function f from a function family S, evaluation key evk, and ciphertexts $c_1, \cdots, c_n$ encrypting $m_1, \cdots, m_n$ respectively, return a ciphertext c_f.

Definition 2.1 (Correctness). *We say that HE correctly evaluates a function family S, if for all $f \in S$ and $c_1, \cdots, c_n$ encrypting $m_1, \cdots, m_n$ respectively, we have*

$$\Pr[\mathsf{HE.Dec}(\mathsf{sk}, c_f) \neq f(m_1, \cdots, m_n)] = \mathsf{negl},$$

where $c_f = \mathsf{HE.Eval}(\mathsf{pk}, f, \mathsf{evk}, c_1, \cdots, c_n)$, and the probability is taken over the randomness used in the computation.

If function family S only contains one operation (such as addition, multiplication), the HE scheme is a partially homomorphic encryption scheme, including RSA [RSA78b] and Paillier encryption [Pai99] schemes and so on.

If S contains several operations with limited times, the HE scheme is a somewhat homomorphic encryption scheme.

If S contains all the operations with unlimited times, the HE scheme is a fully homomorphic encryption scheme.

2.2 Partially homomorphic encryption

RSA [RSA78b] and Paillier encryption [Pai99] support the homomorphic operations of addition and multiplication with unlimited times, respectively.

Let $N = pq$ be the multiplication of two primes p, q. Let $\phi(N) = (p-1)(q-1)$ be the Euler function of N.

2.2.1 RSA

RSA encryption RSA has been discussed in Lecture 3, and consists of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{RSA.KeyGen}(1^\lambda)$ On input 1^λ, select two primes p, q, compute $N = pq$ and set $\phi(N) = (p-1)(q-1)$. Choose $e \in Z^*_{\phi(N)}$ such that $\gcd(e, \phi(N)) = 1$ and compute $d = e^{-1} \mod \phi(N)$. Let

$$\mathsf{sk} = (N, d), \mathsf{pk} = \mathsf{evk} = (N, e).$$

$\mathsf{RSA.Enc}(\mathsf{pk} := (N, e), m)$ On input public key (N, e) and message $m \in \mathbb{Z}_N$, generate and output

$$c = m^e \mod N$$

as the ciphertext.

$\mathsf{RSA.Dec}(\mathsf{sk}, c)$ On input secret key $\mathsf{sk} := (N, d)$ and the ciphertext c, return the corresponding message

$$m = c^d \mod N.$$

The RSA encryption supports the homomorphic operation of multiplication. Let c_1, c_2 be the RSA encryption of m_1, m_2 respectively. We have

$$\begin{aligned} c_1 \cdot c_2 \mod N &= m_1^e \cdot m_2^e \mod N \\ &= (m_1 m_2)^e \mod N \end{aligned} \tag{2}$$

Thus, $c_1 \cdot c_2 \mod N$ is the encryption of $m_1 m_2 \mod N$.

2.2.2 Paillier

Paillier encryption [Pai99] Pai is an encryption with message space $\mathbb{Z}_N$ and ciphertext space $\mathbb{Z}_{N^2}$, and consists of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Pai.KeyGen}(1^\lambda)$ On input 1^λ, select two primes p, q, compute $N = pq$ and set $\phi(N) = (p-1)(q-1)$. Compute $\lambda(N) = \mathsf{lcm}(p-1, q-1)$ where lcm represents the least common multiple of two numbers. Let

$$\mathsf{sk} = (N, \lambda(N)), \mathsf{pk} = \mathsf{evk} = N.$$

$\mathsf{Pai.Enc}(\mathsf{pk} := N, m)$ On input public key N and message $m \in \mathbb{Z}_N$, choose a randomness $r \in \mathbb{Z}_N^*$ generate and output

$$c = (1+N)^m r^N \mod N^2$$

as the ciphertext.

$\mathsf{Pai.Dec}(\mathsf{sk}, c)$ On input secret key $\mathsf{sk} := (N, \lambda(N))$ and the ciphertext c, compute $c^{\lambda(N)} \mod N^2$. We have

$$1 + m \cdot N\lambda(N) = c^{\lambda(N)} \mod N^2. \tag{3}$$

according to Lemma 2. We can recover m from equation 3, i.e, recover $m\lambda(N) \mod N$ at first and then compute compute m with $\lambda(N)$.

Since $|\mathbb{Z}_{N^2}^*| = N\phi(N)$, we have the lemma directly.

Lemma 1. *For any $r \in \mathbb{Z}_N$, we have $r^{N\lambda(N)} = 1 \mod N^2$.*

Lemma 2.

$$(1+N)^x \mod N^2 = 1 + x \cdot N \mod N^2$$

Remark 1. *Due to Lemma 1 and 2, we have*

$$\begin{aligned} c^{\lambda(N)} \mod N^2 &= (1+N)^{m\lambda(N)} r^{N\lambda(N)} \mod N^2 \\ &= (1+N)^{m\lambda(N)} \mod N^2 \\ &= 1 + m \cdot N\lambda(N) \mod N^2. \end{aligned} \tag{4}$$

The Paillier encryption supports the homomorphic operation of addition. Let c_1, c_2 be the RSA encryption of m_1, m_2 respectively. We have

$$\begin{aligned} c_1 \cdot c_2 \mod N^2 &= (1+N)^{m_1} r_1^N \mod N^2 \cdot (1+N)^{m_2} r_2^N \mod N^2 \\ &= (1+N)^{m_1+m_2} (r_1 r_2)^N \mod N^2 \end{aligned} \tag{5}$$

Thus, $c_1 \cdot c_2 \mod N^2$ is the encryption of $m_1 + m_2 \mod N$.

2.3 Somewhat homomorphic encryption from lattice

In this subsection we present a somewhat homomorphic encryption scheme from learning with errors (LWE) assumption.

Let $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ be a random chosen matrix, and

$$\mathbf{b} = \mathrm{LWE}_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{A}\mathbf{s} + \mathbf{e} \mod q,$$

where $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ (the error term) are chosen from some distributions. (Generally, $\mathbf{e}$ is a short vector) $\text{LWE}_{\mathbf{A}}(\cdot, \cdot)$ is called the LWE function.

According to the work of Regev [Reg09], LWE function is one way and $\mathbf{b}$ is indistinguishable with uniform distribution over $\mathbb{Z}_q^m$, assume the SIVP and GapSVP problem are hard against quantum computer.

We can use the randomness of $\mathbf{b}$ in LWE function to design a symmetric key encryption scheme. Informally, let $\mathbf{s}$ be the secret key. To encrypt a message $\mathbf{m} \in \mathbb{Z}^m$, we sample $\mathbf{A}$, choose $\mathbf{e}$ according to the LWE function, and let

$$\mathsf{Enc}_{\mathbf{s}}(\mathbf{m}, \mathbf{e}) = (\mathbf{A}, \mathbf{b} + \mathbf{m} \mod q)$$

be the ciphertext, where $\mathbf{b} = \mathbf{As} + \mathbf{e} \mod q$. With secret key $\mathbf{s}$, we intend to recover $\mathbf{m}$ by computing $(\mathbf{b} + \mathbf{m}) - \mathbf{As} \mod q = \mathbf{m} + \mathbf{e} \mod q$. Obviously, the least significant bits of $\mathbf{m}$ is corrupted by the error $\mathbf{e}$. We handle this problem by only use the most significant bits, i.e., to encrypt $\mathbf{m} \in \mathbb{Z}_p^m$, let

$$c = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}, \mathbf{e}) = (\mathbf{A}, \mathbf{b} + \lfloor \frac{q}{p} \rfloor \mathbf{m} \mod q),$$

for some small number q (e.g. $q = 2$).

Then the decryption algorithm $\mathsf{Dec}_{\mathbf{s}_1}(c)$ computes $(\mathbf{b} + \lfloor \frac{q}{p} \rfloor \mathbf{m}) - \mathbf{As} \mod q = \lfloor \frac{q}{p} \rfloor \mathbf{m} + \mathbf{e} \mod q$ and recovers the most significant bits of the message.

In this case, the encryption supports several linear operations. For example,

$$\mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_1, \mathbf{e}_1) + \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_2, \mathbf{e}_2),$$

is the encryption of $\mathbf{m}_1 + \mathbf{m}_2$ while doubling the error term to $\mathbf{e}_1 + \mathbf{e}_2$; $-\mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_1, \mathbf{e}_1)$ is the encryption of $-\mathbf{m}_1$, and $c * \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_1, \mathbf{e}_1)$ is the encryption of $c * \mathbf{m}_1$ for some constant c (the error term is also multiplied by the constant c).

By using further technique, we could modify the scheme such that the operations of several multiplication and addition could be supported.

We can see that the operations generally would add the size of error term (or noise). However, if the somewhat encryption scheme supports the operation of decrytion circuit, Gentry [Gen09] showed a novel technique to get a new ciphertext with small noise (error term) from a ciphertext with large noise while both of them encrypt the same message.

2.4 Bootstrapping

Bootstrapping, proposed by Gentry [Gen09], is a technique to refresh a ciphertext with large noise to a new ciphertext encryption the same message with small noise. We use the notions of Section 2.3 to illustrate how it works. We assume that a somewhat encryption as in Section 2.3 and its decryption function belongs to the function family S (the scheme we have shown does not satisfy this, more work should be done to achieve this).

Let $\mathbf{s}_1$ be the secret key, and $c_1 = \mathsf{Enc}_{\mathbf{s}_1}(\mathbf{m}, \mathbf{e}_1)$ is the encryption of $\mathbf{m}$ with a large noise $\mathbf{e}_1$.

let $c = \mathsf{Enc}_{\mathbf{s}_2}(\mathbf{s}_1, \mathbf{e}_2)$ let the encryption of $\mathbf{s}_1$ under another secret key $\mathbf{s}_2$ with a small noise $\mathbf{e}_2$). Let c be the public parameter related to the encryption scheme. We could transfer c_1 to a new ciphertext of $\mathbf{m}$ under secret key $\mathbf{s}_2$ with a small noise $\mathbf{e}_2$).

Denote function $f_{c_1} : \mathbf{s}_1 \to \mathsf{Dec}_{\mathbf{s}_1}(c_1)$. As said before, we assume $f_{c_1} \in S$. Thus, we could apply the homomorphic operations to c and get

$$\begin{aligned} c_2 =& \mathsf{Enc}_{\mathbf{s}_2}(f_{c_1}(\mathbf{s}_1), \mathbf{e}_2) \\ =& \mathsf{Enc}_{\mathbf{s}_2}(\mathsf{Dec}_{\mathbf{s}_1}(c_1), \mathbf{e}_2) \\ =& \mathsf{Enc}_{\mathbf{s}_2}(\mathbf{m}, \mathbf{e}_2), \end{aligned} \tag{6}$$

which is the encryption of $\mathbf{m}$ under secret key $\mathbf{s}_2$ with a small noise $\mathbf{e}_2$).

Now, the new ciphetext is like just encrypted (under another secret key). Further homomorphic operations could be applied until the noise is larger than a threshold again. Then, we applied the bootstrapping to reduce the noise to a small one.

2.5 Four generations of FHE

Note[1]

The first generation of FHE In 2009, Gentry [Gen09] introduced the first homomorphic encryption scheme based on ideal lattices. Its security is based on three mathematical problems and an additional security assumption: the Sparse Subset Sum Problem, the Bounded Distance Decoding Problem ,the Ideal Coset Problem and circular security assumption. Gentry designed a blueprint for the construction of homomorphic encryption scheme: First, construct a somewhat homomorphic encryption scheme, which can homomorphically evaluate circuits of a certain depth; Then the decryption circuit is squashed to make it homomorphically evaluate its own augmented decryption circuit; Finally, Bootstrapping is performed to obtain a fully homomorphic encryption scheme that can homomorphically evaluate arbitrary circuits. Bootstrapping is basically "recrypting" procedure to get a fresh ciphertext corresponding to the same plaintext.

In 2010, Van Dijk et al. [vDGHV10] proposed a fully homomorphic encryption scheme on integers based on the blueprint designed by Gentry. The security of the scheme is based on SSSP assumption and AGCD assumption. The first generation of homomorphic encryption schemes represented by [Gen09] and [vDGHV10] have significant shortcomings in efficiency and security.

The second generation of FHE In 2011, Brakerski and Vaikuntanathan introduced two FHE schemes based on the LWE [BV11a] and the RLWE [BV11b], and the circular security assumption. The first scheme proposed two new techniques: re-linearization and dimension-modulus switching. The re-linearization technique is used to construct a SHE scheme and the dimension-modulus switching technique is used to reduce the complexity of the decryption circuit. Although there is no need to squash the decryption circuit, the public key still needs to include the encryption of its private key information. Compared with the first generation scheme, the efficiency and security of these schemes have been greatly improved. However, the encryption of its private key information is still required in homomorphic evaluation, resulting in relatively large key sizes that limit the efficiency of the schemes.

[1] this subsection is done by Xiaohan Wan

The third generation of FHE The third generation of FHE started with GSW scheme [GSW13].The scheme proposed a technique called approximate eigenvector method, avoided the use of re-linearization. In this scheme, homomorphic addition and multiplication correspond directly to matrix addition and multiplication, making it conceptually simpler and more efficient. In 2014, Brakerski and Vaikuntanathan [BV14] improved the bootstrapping process using the GSW scheme and Barrington's theorem [Bar86], resulting in shorter computation time and slower error growth, while maintaining the security level comparable to that of a regular public key encryption based on the LWE problem. Ducas and Micciancio [DM15] proposed a new bootstrapping method which can homomorphically compute the NAND of two standard LWE ciphertexts. Overall,the third generation FHE had further improved in terms of security and efficiency.

The fourth generation of FHE In 2017, Cheon, Kim, Kim and Song [CKKS17] introduced a new generation of FHE. Its security is based on the RLWE problem. Unlike the previous generations of homomorphic encryption schemes, the core of the CKKS scheme lies in treating the encryption noise as part of the approximate evaluation error, with the decrypted result being directly viewed as an approximate value of the original plaintext message. In 2018, Cheon et al. [CHK+18] proposed a bootstrapping algorithm for the CKKS scheme to achieve fully homomorphism. An interesting feature of CKKS is the capability to homomorphically operate over approximations of real numbers, which makes it a suitable scheme to work with floating-point arithmetic. Therefore, it is suitable for scenarios such as machine learning that do not require precise results.

References

[AAAS+19] Gorjan Alagic, Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the first round of the nist post-quantum cryptography standardization process. 2019.

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

[ABSS97] Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

[App16] Benny Applebaum. Cryptographic hardness of random local functions: Survey. *Computational complexity*, 25:667–722, 2016.

[Bal21] Philip Ball. First 100-qubit quantum computer enters crowded race. *Nature*, 599:542, 2021.

[Bar86] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5, 1986.

[Ber05] Daniel J Bernstein. The poly1305-aes message-authentication code. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers 12*, pages 32–49. Springer, 2005.

[BGG+20] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann. Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II 40*, pages 62–91. Springer, 2020.

[BL17] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[Bry12] PG John Bryson. Secure hash standard (shs)(federal information processing standards publication 180-4), 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 505–524. Springer, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 1–12, 2014.

[CCJ+16] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray A Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology . . . , 2016.

[CD22] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). *Cryptology ePrint Archive*, 2022.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I 37, pages 360–384. Springer, 2018.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. Csidh: an efficient post-quantum commutative group action. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*, pages 395–427. Springer, 2018.

[DK22] Bartosz Drzazga and Łukasz Krzywiecki. Review of chosen isogeny-based cryptographic schemes. *Cryptography*, 6(2), 2022.

[DM15] Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34*, pages 617–640. Springer, 2015.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[Flu17] Scott R. Fluhrer. Reassessing grover's algorithm. *IACR Cryptol. ePrint Arch.*, 2017:811, 2017.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[Hel76] Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, 1998.

[ISA19] ISARA. Isogeny-based cryptography tutorial, 2019.

[JDF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4*, pages 19–34. Springer, 2011.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1:36–63, 2001.

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, October 1979. This paper was published by IEEE in the Proceedings of HICSS-43 in January, 2010.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 738–755. Springer, 2012.

[MAA+20] Dustin Moody, Gorjan Alagic, Daniel C Apon, David A Cooper, Quynh H Dang, John M Kelsey, Yi-Kai Liu, Carl A Miller, Rene C Peralta, Ray A Perlner, et al. Status report on the second round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020.

[Mer01] Ralph C Merkle. A certified digital signature. In *Advances in cryptology—CRYPTO'89 proceedings*, pages 218–238. Springer, 2001.

[Mic01] Daniele Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

[MNR+20] Miralem Mehic, Marcin Niemiec, Stefan Rass, Jiajun Ma, Momtchil Peev, Alejandro Aguado, Vicente Martin, Stefan Schauer, Andreas Poppe, Christoph Pacher, and Miroslav Voznak. Quantum key distribution: A networking perspective. *ACM Comput. Surv.*, 53(5), sep 2020.

[MP22] Michele Mosca and Marco Piani. 2021 quantum threat timeline report, 2022.

[MR22] Dustin Moody and Angela Robinson. Cryptographic standards in the post-quantum era. *IEEE Security & Privacy*, 20(6):66–72, 2022.

[MV04] David A McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In *Indocrypt*, volume 3348, pages 343–355. Springer, 2004.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 223–238. Springer, 1999.

[Pre22] Richard H. Preston. Applying grover's algorithm to hash functions: a software perspective. *IEEE Transactions on Quantum Engineering*, 3:1–10, 2022.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[RSA78a] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[RSA78b] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[ST21] Unathi Skosana and Mark Simon Tame. Demonstration of shor's factoring algorithm for n = 21 on ibm quantum processors. *Scientific Reports*, 11 1:16599, 2021.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 24–43. Springer, 2010.