POLITECNICO DI TORINO

Master Degree course in Computer Engineering

Master Degree Thesis

Temporal Resource Comparison Between Classical Asymmetric Cryptosystems and Post-Quantum Alternatives.

Supervisors
Prof. Danilo Bazzanella

Candidate
Pietro Mazza

Company Supervisor
Giulia Bracco

Academic Year 2024-2025

Acknowledgements

First, I would like to thank my family, who have always been by my side and supported me in my choices and throughout my growth both academically and personally: my parents, Laura and Angelo, and my brothers, Paolo and Michele. I also thank my grandparents and uncles, who have always made every effort so that I could continue to grow and learn.

I warmly thank all my friends, those I've known my whole life and those I've met more recently, both in academic settings and outside the university. Each of them has helped push me to give my best and to improve.

I would also like to thank Professor Danilo Bazzanella for supervising me during the writing of this thesis.

I am grateful to all my colleagues, particularly to Andrea and Giulia, at Spike Reply for giving me the opportunity to do my internship and for the warm welcome I received.

Finally, I want to give a special thanks to my grandfather Franco, for having taught me so much and for passing on the ideals that make our family united and always ready to support each other.

Abstract

This thesis presents a quantitative investigation into the temporal performance of asymmetric cryptographic schemes, addressing the urgent need to migrate from classical algorithms to post-quantum alternatives in the face of emerging quantum computing threats. The core of this research is a comparative evaluation of the key-pair generation latency between established cryptosystems (RSA, ECDSA, EdDSA) and selected post-quantum candidates standardized by NIST, specifically lattice-based (ML-DSA, ML-KEM) and hash-based (XMSS, LMS) algorithms.

The study employs a rigorous experimental methodology conducted across two distinct environments: a production-grade Thales Luna Hardware Security Module (HSM) and a corresponding Utimaco software simulator. By systematically measuring execution times for classical algorithms on both platforms, a performance ratio was established to account for hardware acceleration and network overhead. This ratio was subsequently used to project the expected real-world performance of the post-quantum algorithms, which were benchmarked on the simulator due to the lack of native HSM support at the time of research.

The results reveal a significant performance dichotomy between the post-quantum families. The lattice-based schemes, ML-DSA and ML-KEM, demonstrate remarkable efficiency, with key generation times comparable to, or in some cases faster than, their classical elliptic curve counterparts. In stark contrast, the stateful hash-based signature schemes, XMSS and LMS, exhibit substantial computational overhead. Their key generation latency scales dramatically with security parameters, ranging from seconds to several hours, highlighting a critical implementation trade-off between signature capacity and pre-computation time.

In conclusion, this work provides a crucial quantitative baseline for organizations planning the transition to quantum-resistant cryptography. While the performance of lattice-based algorithms suggests a pathway for a seamless migration for many applications, the high latency of hash-based key generation necessitates strategic deployment models, such as offloading and pre-computation. These findings underscore the importance of balancing security imperatives with operational performance constraints to ensure a robust and practical defense against future quantum threats.

Contents

List of Figures

List of Tables

Chapter 1

Public encryption as it stands

Public-key cryptography represents a pivotal advancement in securing digital communications, enabling confidential information exchange over public channels without the need for a shared secret key. This paradigm shift is deeply rooted in number theory, particularly in the properties of prime numbers and modular arithmetic. A cornerstone of public key cryptography is the concept of one-way functions, mathematical functions that are computationally easy to evaluate in one direction but infeasible to reverse without specific information. Two prominent examples are the integer factorization problem and the discrete logarithm problem.

The RSA cryptosystem exemplifies the application of the integer factorization problem. In RSA, a user generates a public key by selecting two large prime numbers, p and q, and computing their product, $n = p \times q$. The security of RSA hinges on the difficulty of factoring the large composite number n back into its prime factors, a task that becomes computationally prohibitive as the size of n increases. Similarly, the Diffie-Hellman key exchange protocol relies on the discrete logarithm problem. This problem involves finding the exponent x in the equation $g^x \equiv h \pmod{p}$, where g and h are elements of a finite cyclic group, and p is a prime modulus. While exponentiation in such groups is computationally efficient, the reverse operation-determining x given g, h, and p is believed to be infeasible for sufficiently large p.

These cryptographic systems leverage modular arithmetic, where numbers "wrap around" upon reaching a certain modulus. In the context of RSA, operations are performed modulo n, the product of two primes, while Diffie-Hellman operates within the multiplicative group of integers modulo a prime p. The mathematical structures of these groups, particularly their cyclic nature and the distribution of prime numbers, are critical in ensuring the computational hardness that underpins the security of these systems. Public Key Cryptography (PKC) is founded on this complex mathematical principles, with its security derived from the intractability of specific number-theoretic problems.

1.1 Mathematical Aspects

1.1.1 Modular Arithmetic

Modular arithmetic, often referred to as "clock arithmetic," is a fundamental component of public key cryptography, providing the mathematical framework for many encryption and decryption processes. In this system, numbers wrap around upon reaching a specified modulus, n, leading to the concept of congruence: two integers a and b are congruent modulo n if $a \equiv b \pmod{n}$, meaning n divides the difference $a-b$. This structure exhibits properties such as closure, associativity, commutativity, and distributivity, and allows for the existence of multiplicative inverses when two numbers are coprime. Key theorems in modular arithmetic underpin the security of cryptographic algorithms. Fermat's Little Theorem states that if p is a prime number and a is an integer not divisible by p, then $a^{p-1} \equiv 1 \pmod{p}$. This theorem is instrumental in simplifying computations involving large exponents in modular systems. Euler's Theorem generalizes this result: for any integer a coprime to n then $a^{\phi(n)} \equiv 1 \pmod{n}$, and $\phi(n)$ denotes Euler's totient function, representing the count of integers up to n that are coprime with n. These theorems facilitate efficient computation of modular inverses and exponentiation, which are critical operations in cryptographic protocols.

1.1.2 Prime Numbers

Prime numbers play a fundamental role in number theory and cryptography due to their unique arithmetic properties. The Fundamental Theorem of Arithmetic establishes that every integer greater than one has a unique prime factorization, up to the order of factors, making primes the building blocks of the integer set. However, their distribution among natural numbers is irregular, as described by the Prime Number Theorem, which approximates the number of primes less than or equal to n as $\frac{n}{\ln(n)}$, indicating their decreasing density as n grows. This property is crucial in cryptographic applications, particularly in public-key cryptosystems such as RSA, where security is based on the computational difficulty of factoring large composite numbers into their prime components. To generate cryptographic keys, efficient methods for producing large prime numbers are essential. Since the probability of a randomly chosen n-bit integer being prime is roughly $1/(3n)$, selecting primes involves generating random numbers and verifying their primality using specialized tests. Primality testing algorithms are divided into deterministic and probabilistic methods. The AKS (Agrawal-Kayal-Saxena) primality test, discovered in 2002, provides a polynomial-time deterministic solution, but its computational overhead renders it impractical for cryptographic applications. Instead, probabilistic algorithms such as the Miller-Rabin test are commonly used due to their efficiency. The Miller-Rabin test is a strong pseudoprime test that takes advantage of modular arithmetic and Fermat's theorem. By checking for nontrivial square roots of unity, the test efficiently detects compositeness, ensuring that numbers identified as prime have an extremely high probability of being so. Given an efficient mean of prime number generation, cryptographic security assumptions can be established. The integer factorization problem assumes that no polynomial-time algorithm can efficiently factor a product of two large primes, if this

assumption were broken, an adversary could recover private keys by factoring the public modulus $N = pq$, thereby compromising the security of encrypted communications. Consequently, the robustness of public-key cryptography hinges on the mathematical intractability of prime factorization and the careful selection of cryptographic primes using rigorous number-theoretic techniques.

1.1.3 Group Theory

In public key cryptography, group theory provides a rigorous mathematical framework essential for constructing secure and efficient cryptographic protocols. A group is an algebraic structure consisting of a set G equipped with a binary operation $*$ that satisfies four fundamental properties: closure (for any $a, b \in G$, $a * b \in G$), associativity (for any $a, b, c \in G$, $(a * b) * c = a * (b * c)$), the existence of an identity element e (such that for any $a \in G$, $a * e = e * a = a$), and the existence of inverse elements (for each $a \in G$, there exists an $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$). Groups can be finite or infinite, but in cryptographic applications finite groups are predominantly utilized due to their well-defined structure and computational manageability. A particularly significant class of groups in cryptography are cyclic groups. A group G is cyclic if there exists an element $g \in G$ such that every element $h \in G$ can be expressed as $h = g^k$ for some integer k. The order of a cyclic group is the number of distinct elements it contains, and for any element h in a finite group of order m, it holds that $h^m = e$, where e is the identity element. This property is fundamental in many cryptographic algorithms that rely on exponentiation within groups.

One of the cornerstone problems in public key cryptography is the Discrete Logarithm Problem (DLP), which involves finding the integer x given g and h in a cyclic group G such that $g^x = h$. The computational difficulty of solving the DLP in appropriately chosen groups underpins the security of various cryptographic protocols, including the Diffie-Hellman key exchange and the ElGamal encryption scheme. Closely related is the Computational Diffie-Hellman (CDH) problem, which entails computing g^{xy} given g^x and g^y without knowledge of x or y. An even stronger assumption is the Decisional Diffie-Hellman (DDH) problem, which posits that distinguishing between the tuple (g^x, g^y, g^{xy}) and (g^x, g^y, g^z) (where z is a random element) is computationally infeasible. The hardness of these problems is contingent upon the algebraic structure of the underlying group and the absence of efficient algorithms to solve them.

In practice, groups of prime order are favored in cryptographic applications because they mitigate vulnerabilities associated with the factorization of the group order. The Pohlig-Hellman algorithm, for instance, demonstrates that the difficulty of the DLP is significantly reduced in groups whose orders have small prime factors. Therefore, selecting groups with a large prime order enhances security by ensuring that the DLP remains computationally intractable. A common choice is the multiplicative group of integers modulo a prime p, denoted $\mathbb{Z}_p^*$, which consists of all integers from 1 to $p - 1$ under multiplication modulo p. This group is cyclic with order $p - 1$, and by selecting a prime p such that $p - 1$ has a large prime factor, one can construct a subgroup of prime order q within $\mathbb{Z}_p^*$, thereby providing a suitable setting for cryptographic operations.

Elliptic Curve Cryptography (ECC) introduces an alternative group structure based

on the set of points on an elliptic curve over a finite field. An elliptic curve is defined by an equation of the form $y^2 = x^3+ax+b$ over a finite field $\mathbb{F}_p$, and the group operation is point addition, which has a geometric interpretation. The group formed by the points on an elliptic curve, together with a point at infinity serving as the identity element, is abelian and can be made cyclic by appropriate selection of the curve parameters. The primary advantage of ECC is that it offers higher security per bit of key length compared to traditional groups like $\mathbb{Z}_p^*$. This efficiency arises because the best-known algorithms for solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) have exponential complexity, making ECC particularly attractive for environments with constrained computational resources. The security of public key cryptographic systems hinges on the careful selection of group parameters to ensure the intractability of underlying hard problems such as the DLP and ECDLP. Advancements in mathematical research and computational capabilities necessitate ongoing evaluation of these parameters to maintain the robustness of cryptographic protocols. By leveraging this structures cryptographers can design systems that achieve both high security and operational efficiency.

1.1.4 Use in cryptography

At its core, PKC utilizes asymmetric key pairs, public keys for encryption and private keys for decryption, to ensure confidentiality, authenticity, and integrity of data. The security of these systems is predicated on computational problems that are easy to perform in one direction but infeasible to reverse without specific knowledge. The Diffie-Hellman key exchange protocol, one of the pioneering public-key cryptosystems, leverages the hardness of the DLP in the multiplicative group of integers modulo a prime p. In this protocol, two parties agree on a large prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$. Each party selects a private key, computes the corresponding public key by exponentiating g to their private key, and exchanges public keys. The shared secret is then derived by raising the received public key to the power of their own private key. The security of this exchange relies on the difficulty of solving the DLP, ensuring that an eavesdropper cannot feasibly compute the shared secret from the public information alone. Elliptic Curve Cryptography (ECC) represents a modern advancement in PKC, offering equivalent security with smaller key sizes by exploiting the algebraic structure of elliptic curves over finite fields. The Elliptic Curve Discrete Logarithm Problem (ECDLP), finding an integer k such that $Q = kP$, given points P and Q on the curve, is believed to be even more intractable than the classical DLP. This increased difficulty allows ECC to achieve comparable security to RSA and Diffie-Hellman with significantly smaller parameters, resulting in faster computations and reduced storage requirements. Digital signature algorithms also heavily rely on the principles of modular arithmetic and group theory. The Digital Signature Algorithm (DSA) and its elliptic curve variant, the Elliptic Curve Digital Signature Algorithm (ECDSA), utilize the hardness of the DLP and ECDLP, respectively, to ensure that signatures cannot be forged without access to the private key. In these schemes, a message is signed by generating a value derived from the private key and the message hash, and verification is performed using the corresponding public key, providing assurance of the message's authenticity and integrity.

Furthermore, prime numbers and modular arithmetic are employed in the design of

cryptographic hash functions and pseudorandom number generators. For example, the Secure Hash Algorithm (SHA) family utilizes modular addition and bitwise operations to produce fixed-size hash values from arbitrary-length input data, ensuring data integrity and authenticity. Pseudorandom number generators like the Mersenne Twister rely on properties of prime numbers to produce sequences of numbers with long periods and good statistical properties, essential for cryptographic applications such as key generation and nonce creation.

1.1.5 Hardness Of The Mathematical Problems

The security of widely used cryptographic systems depends on the computational infeasibility of solving these mathematical problems with existing technology. However, real-world data highlights the growing difficulty of maintaining security as computational power advances. For example, in December 2019 researchers factored a 795-bit (240-digit) RSA modulus using the general number field sieve (GNFS) algorithm, a process that required approximately 4,000 core-years of computation on a cluster of CPUs [35]. Extrapolating from this, factoring a 1024-bit RSA key, still used in some legacy systems, would take roughly 100,000 core-years, and a 2048-bit key, the current standard, remains infeasible with classical methods. However, advancements in specialized hardware and algorithms threaten this security assumption.

Similarly for the discrete logarithm problem that underpins Diffie-Hellman key exchange and ElGamal encryption. In 2016, researchers solved a 768-bit discrete logarithm problem in a prime field, requiring an estimated 5,000 CPU-years [36]. Given current progress in computational techniques, a 1024-bit key could be broken within decades, prompting a shift towards 2048-bit or larger key sizes. Elliptic curve cryptography (ECC) provides a more efficient alternative. A 256-bit ECC key offers security comparable to a 3072-bit RSA key. The largest successfully solved ECDLP instance involved a 114-bit key, taking 10,000 CPU-years, suggesting that breaking a standard 256-bit key remains far beyond current capabilities. These real-world benchmarks illustrate the practical difficulty of breaking cryptographic schemes today.

Despite extensive study, these hardness assumptions remain unproven; no mathematical proof currently confirms that integer factorization, DLP, or ECDLP are intractable problems. Consequently, the security of PKC systems is based on the belief that no efficient (polynomial-time) algorithms exist for these problems. This belief is supported by decades of cryptanalytic efforts failing to find such algorithms.

1.2 Key Management and Diffie-Hellman

Key distribution is a pivotal aspect of cryptography, fundamentally influencing the secure exchange of information between parties. In symmetric key cryptography, a single secret key is employed for both encryption and decryption processes, necessitating a secure method for key distribution to maintain system integrity. The primary challenge lies in transmitting this key over potentially insecure communication channels; if an adversary intercepts the key during transmission, the confidentiality of the encrypted data is compromised, as the attacker would then be able to decrypt any messages encrypted

with that key. This issue becomes increasingly complex in expansive networks, where the number of participants and required secure channels escalates. Asymmetric cryptography, also known as public-key cryptography, offers an alternative by utilizing a pair of keys: a public key for encryption and a private key for decryption. This approach simplifies key distribution since the public key can be openly shared without compromising security. However, symmetric key cryptography remains prevalent due to its computational efficiency and speed advantages over asymmetric methods. Nonetheless, the challenges associated with key distribution in symmetric systems are significant, particularly concerning scalability, confidentiality, and integrity within large, distributed networks. As the number of participants increases, the number of unique keys required for secure communication between all parties grows exponentially, following the formula $\frac{n(n-1)}{2}$ for n participants. This exponential growth results in substantial overhead, as each participant must securely exchange a key with every other participant. In dynamic networks, where users frequently join or leave, the logistical challenges of generating, distributing, and managing keys are further amplified. To mitigate these issues, approaches such as Key Distribution Centers (KDCs) have been developed. KDC acts as a trusted third party responsible for generating and distributing keys to participants, reducing the number of keys each participant must manage. However, this introduces a single point of failure; if the KDC is compromised or becomes unavailable, the entire system's security is jeopardized. Additionally, the KDC must be inherently trustworthy and secure, which can be a significant risk in distributed networks. Group key management protocols offer another solution by managing keys for groups of participants, enabling secure communication within a group without necessitating unique keys for every pair of participants. As participants join or leave the group, the group key is updated and distributed securely. While this reduces the number of keys that need to be exchanged, it introduces challenges in efficiently updating and distributing group keys, especially in large and dynamic groups. Moreover, ensuring that group keys are updated securely without allowing unauthorized participants to access them can be a difficult problem. Logistical challenges in key management include not only secure distribution but also regular key rotation to mitigate the risk of key compromise. Keys must be periodically replaced to ensure that an attacker who has gained access to a key cannot continue to decrypt communications indefinitely. However, key rotation introduces its own set of challenges, particularly when dealing with large numbers of users and devices. Coordinating key rotation across all devices while ensuring that the new keys are securely transmitted can be a complex task. Additionally, key storage and revocation mechanisms must be implemented to prevent unauthorized access to old or expired keys. The process of securely revoking compromised keys and ensuring that they are no longer used for decryption is another critical challenge, particularly in environments with high mobility or frequent changes in network topology. Backward compatibility poses another challenge; as cryptographic algorithms evolve and key lengths increase to address advancements in computing power, older systems may no longer support new key exchange protocols or key lengths. This creates a dilemma for organizations that must maintain compatibility between legacy and newer systems while ensuring all systems remain secure. Transitioning from one encryption scheme to another, particularly in large-scale networks, can be a slow and resource-intensive process,

and during the transition period, vulnerabilities may arise as both old and new systems coexist. The advent of wireless communication and the Internet of Things (IoT) further complicates key distribution in large, dynamic networks. In these environments, devices are often added or removed from the network frequently, and the communication channels themselves may be more vulnerable to interception or eavesdropping. The challenge of securely distributing keys to a large number of low-power, resource-constrained devices, each with varying security capabilities, is an ongoing area of research. Solutions that work in traditional wired networks may not be applicable in the context of IoT, where devices may have limited processing power and memory and may operate in environments with high levels of interference or physical security risks. In conclusion, key distribution is a critical challenge for symmetric key systems, particularly in large, distributed, or dynamic networks. The need for secure channels to exchange keys, the exponential growth of required keys as the network scales, and the complexities of key management all pose significant obstacles to ensuring the security of encrypted communications. While various techniques such as key distribution centers, group key management, and secure key exchange protocols help mitigate these challenges, they introduce their own risks and complexities. The evolution of cryptographic protocols must address these problems while also adapting to the increasing demands of modern networks, which are characterized by high scalability, dynamic membership, and varying levels of security. For all these reasons, the most widely used solution is the Diffie-Hellman key exchange protocol. Diffie-Hellman enables two parties to establish a shared secret over an insecure channel without prior knowledge of each other, effectively addressing the key distribution problem inherent in symmetric key systems. This method allows for secure key exchange even in large and dynamic networks, reducing the complexities associated with key management and distribution. Consequently, Diffie-Hellman has become a cornerstone in modern cryptographic practices, facilitating secure communications across various platforms and applications.

1.2.1 Diffie-Hellman

The Diffie-Hellman key exchange protocol, introduced by Whitfield Diffie and Martin Hellman in 1976, revolutionized secure communications by enabling two parties to establish a shared secret over an insecure channel without prior shared secrets. This protocol laid the foundation for public-key cryptography, a paradigm shift from traditional symmetric encryption methods that required secure key distribution channels. At its core, the Diffie-Hellman protocol leverages the mathematical principles of modular arithmetic and the computational difficulty of the discrete logarithm problem. The protocol operates within a cyclic group G of prime order p, generated by a generator g. Both p and g are publicly known parameters. The security of the protocol hinges on the difficulty of solving the discrete logarithm problem in this group for appropriately chosen parameters. The protocol proceeds as follows:

1. **Parameter Selection**:

   Alice and Bob agree on a large prime number p and a generator g of the multiplicative group of integers modulo p, denoted as $\mathbb{Z}_p^*$. These parameters are public and

can be known to potential adversaries without compromising security.

2. **Private Key Generation**:

   Alice selects a private key a randomly from the set $\{1, 2, ..., p-2\}$.

   Bob selects a private key b independently from the same set.

3. **Public Key Computation**:

   Alice computes her public key $A = g^a \mod p$.

   Bob computes his public key $B = g^b \mod p$.

4. **Public Key Exchange**:

   Alice and Bob exchange their public keys A and B over the insecure channel.

5. **Shared Secret Computation**:

   Alice computes the shared secret $S_A = B^a \mod p$.

   Bob computes the shared secret $S_B = A^b \mod p$.

Due to the properties of modular exponentiation, both computations yield the same result:

$$_A = (g^b \mod p)^a \mod p = g^{ba} \mod p = g^{ab} \mod p$$

$$_B = (g^a \mod p)^b \mod p = g^{ab} \mod p$$

Thus, $S_A = S_B = g^{ab} \mod p$, establishing a mutual shared secret S between Alice and Bob.

The security of this exchange is predicated on the computational difficulty of the discrete logarithm problem for a sufficiently large value of p. This infeasibility ensures that an eavesdropper, who has access to g, p, A, and B, cannot feasibly compute the shared secret S without knowledge of the private keys a or b.

However, the basic Diffie-Hellman protocol is susceptible to man-in-the-middle (MITM) attacks. In such an attack, an adversary intercepts the public keys exchanged between Alice and Bob, substituting them with their own public keys. Consequently, the adversary establishes separate shared secrets with both Alice and Bob, enabling them to decrypt and potentially alter the messages exchanged. To mitigate this vulnerability, the protocol is often augmented with authentication mechanisms, such as digital signatures or certificates, to verify the identities of the communicating parties and ensure the integrity of the key exchange process. Furthermore, the standard Diffie-Hellman protocol does not inherently provide perfect forward secrecy (PFS). If an adversary later compromises the private keys a or b, they can retroactively compute the shared secret S for past communications. To address this limitation, the ephemeral Diffie-Hellman (EDH) variant is employed, wherein ephemeral (temporary) private and public keys are generated for each session and discarded afterward. This approach ensures that the compromise of long-term private keys does not jeopardize the confidentiality of past sessions.

In practical applications, the Diffie-Hellman key exchange is widely used in protocols such as Transport Layer Security (TLS) to secure internet communications. Diffie-Hellman key exchange protocol represents a seminal advancement in cryptography, enabling secure key establishment over insecure channels.

1.3 Public Key Encryption

In a typical public-key encryption scheme, the receiver generates a key pair consisting of a public key (pk) and a private key (sk). The public key is disseminated widely, enabling any sender to encrypt messages intended for the receiver. However, only the receiver, possessing the corresponding private key, can decrypt these messages. The security of this system hinges on the confidentiality of the private key; if an adversary gains access to it, the security of the entire communication is compromised. Public-key encryption offers several advantages over symmetric encryption. Foremost, it resolves the key distribution problem, as parties can establish secure communication without needing to share a secret key beforehand. This is particularly advantageous in scenarios where multiple senders need to communicate securely with a single receiver, such as in e-commerce transactions where an online merchant must process sensitive information from numerous customers. The merchant can maintain a single private key while enabling secure communication with all potential customers. Additionally, the sender's identity need not be predetermined at the time of key generation, enhancing the system's flexibility. However, public-key encryption is computationally more intensive than symmetric encryption, often being several orders of magnitude slower, which can make it impractical for resource-constrained devices or applications requiring high throughput. Consequently, symmetric encryption is generally preferred when feasible due to its superior efficiency. A critical aspect of public-key cryptography is the secure distribution of public keys. For this reason Public Key Infrastructures (PKIs) are employed, which involve a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption. A certificate authority (CA) within the PKI vouches for the identities assigned to specific private keys by producing a digital certificate. However, the security of PKIs depends on the trustworthiness of the CAs; if a CA is compromised, an attacker could issue fraudulent certificates, facilitating man-in-the-middle attacks. Therefore, the integrity of the certification hierarchy is crucial in deploying public-key systems.

Normally, a public-key encryption scheme consists of three algorithms: a key-generation algorithm (Gen), an encryption algorithm (Enc), and a decryption algorithm (Dec). The key-generation algorithm produces a pair of keys (pk, sk) based on a given security parameter. The encryption algorithm uses the public key to encrypt a message, producing a ciphertext, while the deterministic decryption algorithm utilizes the private key to recover the original message. Correctness requires that decryption should successfully recover the original message with overwhelming probability, except in cases where the encryption process introduces negligible errors. A fundamental security criterion for public-key encryption is security against chosen-plaintext attacks (CPA-security). In this model, an adversary is given access to the public key and allowed to choose two messages of equal length, one of which is encrypted and provided as a challenge ciphertext. The adversary's task is to determine which message was encrypted. A scheme is considered CPA-secure if no probabilistic polynomial-time adversary can succeed in this task with probability significantly better than random guessing. This definition encapsulates the notion of indistinguishable encryptions, meaning that an adversary cannot differentiate

between ciphertexts corresponding to different plaintexts. Importantly, this level of security inherently implies that public-key encryption must be probabilistic, as deterministic schemes would allow an adversary to infer the original message simply by encrypting candidate messages and comparing them to the observed ciphertext. An extension of CPA-security is security under multiple encryptions, which ensures that even when multiple messages are encrypted using the same public key, an adversary remains unable to distinguish them. In the public-key setting, single-message security suffices to guarantee multi-message security due to the inherent asymmetry of key usage. This result simplifies security analysis, as proving CPA-security for a single message automatically extends to multiple messages. Despite CPA-security providing a robust foundation for public-key encryption, additional security considerations arise when adversaries can manipulate ciphertexts. Chosen-ciphertext attacks (CCA-security) model scenarios where an adversary has access to a decryption oracle, allowing them to decrypt arbitrary ciphertexts except for the challenge ciphertext. This security notion is particularly relevant in real-world applications where an attacker might intercept and modify encrypted messages. CCA-security ensures that even in such scenarios, an adversary cannot gain meaningful information about the original message. This level of security is crucial for preventing attacks such as padding-oracle attacks, malleability exploits, and adaptive chosen-ciphertext attacks.

1.3.1 RSA

The RSA encryption algorithm, named after its inventors Rivest, Shamir, and Adleman, is a cornerstone of asymmetric cryptography, leveraging the mathematical properties of large prime numbers to facilitate secure data transmission. The algorithm's security is predicated on the RSA assumption, which posits that factoring the product of two large prime numbers is computationally infeasible.

Key Generation :

1. Select two distinct large prime numbers, p and q.
2. Compute their product to obtain the modulus: $N = p \times q$.
3. Calculate the totient function: $\phi(N) = (p-1) \times (q-1)$.
4. Choose a public exponent e such that $1 < e < \phi(N)$ and $\gcd(e, \phi(N)) = 1$, ensuring e and $\phi(N)$ are coprime.
5. Determine the private exponent d as the modular multiplicative inverse of e modulo $\phi(N)$, satisfying $d \times e \equiv 1 \pmod{\phi(N)}$.
6. The public key is (N, e); the private key is (N, d).

Encryption :

1. Represent the plaintext message m as an integer in the range $0 \leq m < N$.
2. Compute the ciphertext c using the public key: $c = m^e \pmod{N}$.

Decryption :

1. Compute the original message m using the private key: $m = c^d \pmod{N}$.
2. This decryption works because $(m^e)^d \equiv m \pmod{N}$, a property derived from Euler's theorem.

Despite its theoretical elegance, plain RSA encryption is deterministic and thus vulnerable to various attacks. For instance, an adversary can perform a chosen-plaintext attack by encrypting potential plaintexts and comparing the resulting ciphertexts to the target ciphertext. Additionally, if a small public exponent e, such as 3, is used, and the plaintext m is small enough that $m^e < N$, the ciphertext c equals m^e, allowing the adversary to recover m by taking the integer e-th root of c. Coppersmith's attack further exploits scenarios where parts of the plaintext are known or have specific structures, enabling efficient recovery of the remaining plaintext. Moreover, when the same message is encrypted under multiple public keys sharing the same small exponent, the Chinese Remainder Theorem can be applied to recover the plaintext, a vulnerability known as the "broadcast attack."

To mitigate these vulnerabilities, padding schemes have been developed to introduce randomness into the plaintext before encryption, thereby achieving probabilistic encryption. The PKCS #1 v1.5 padding scheme, for example, prepends a structured padding string to the plaintext, ensuring that the input to the RSA function is sufficiently large and randomized. However, PKCS #1 v1.5 has been shown to be susceptible to adaptive chosen-ciphertext attacks, notably Bleichenbacher's attack [8], which exploits the padding structure to progressively decrypt a ciphertext. To address these shortcomings, the Optimal Asymmetric Encryption Padding (OAEP) scheme was introduced. OAEP employs a Feistel-like construction using hash functions to generate a padded message that is both randomized and resistant to chosen-ciphertext attacks. In OAEP, the plaintext is first padded with zeros and then processed through a series of masking operations involving hash functions and random seeds, resulting in a ciphertext that is indistinguishable from random values under the RSA assumption in the random oracle model.

Beyond padding schemes, RSA-based Key Encapsulation Mechanisms (KEMs) have been developed to securely transmit symmetric keys. In RSA-KEM, a random integer r is selected uniformly from the range $[0, N-1]$ and encrypted using the recipient's public key to produce the ciphertext $c = r^e \pmod{N}$. A symmetric key k is then derived from r using a key derivation function H, such that $k = H(r)$. The ciphertext c is transmitted to the recipient, who decrypts it using their private key to recover r and subsequently derives the symmetric key k. This method decouples the encryption of data from the encryption of keys, providing semantic security under the RSA assumption.

In practical implementations, the efficiency of RSA decryption can be enhanced using the Chinese Remainder Theorem (CRT). By computing separate exponentiations modulo p and q, and then combining the results using CRT, the computational load is significantly reduced. However, this approach necessitates of careful implementation to prevent side-channel attacks, such as fault attacks, where induced errors during computation can leak information about the prime factors of N. Ensuring the use of high-quality

randomness during key generation is also critical, as poor entropy can lead to the generation of weak keys with shared factors, making them susceptible to factorization attacks. Despite its widespread adoption, RSA is increasingly being supplanted by cryptosystems based on the Computational Diffie-Hellman (CDH) and Decisional Diffie-Hellman (DDH) assumptions, primarily due to the larger key sizes required for RSA to maintain security against advancing computational capabilities. Elliptic Curve Cryptography (ECC), for instance, offers comparable security with significantly smaller key sizes, resulting in improved performance and reduced resource consumption.

1.3.2 CDH-DDH

The ElGamal encryption scheme, introduced by Taher ElGamal in 1985, is a seminal asymmetric key encryption algorithm that extends the Diffie-Hellman key exchange protocol to enable secure message transmission. Operating over a cyclic group G of prime order q with a generator g, the scheme's security is predicated on the computational hardness of the Decisional Diffie-Hellman (DDH) problem within G. The DDH assumption posits that distinguishing between the tuple (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b, c are randomly chosen exponents, is computationally infeasible.

Key Generation : Alice, the recipient, selects a private key x uniformly at random from the set $\{1, \ldots, q-1\}$ and computes the corresponding public key $h = g^x$. The public key is the tuple (G, q, g, h), while x remains confidential as the private key.

Encryption : To encrypt a message m (which must be an element of G) for Alice, the sender, Bob, performs the following steps:

1. **Ephemeral Key Generation**: Selects a random ephemeral key y from $\{1, \ldots, q-1\}$.
2. **Ciphertext Computation**: Calculates $c_1 = g^y$ and $c_2 = m \cdot h^y$.

The resulting ciphertext is the pair (c_1, c_2).

Decryption : Upon receiving the ciphertext (c_1, c_2), Alice decrypts it as follows:

1. **Shared Secret Computation**: Computes the shared secret $s = c_1^x = (g^y)^x = g^{xy}$.
2. **Message Recovery**: Derives the original message by computing $m = c_2/s = c_2/g^{xy}$.

The security of ElGamal encryption is underpinned by the DDH assumption, ensuring that the ciphertext components c_1 and c_2 do not reveal information about the plaintext m without knowledge of the private key x. However, practical deployment of ElGamal encryption necessitates addressing certain limitations:

Message Space Restriction : The scheme requires that messages be elements of the group G. To encrypt arbitrary bit strings, messages must be mapped to G, which

can be inefficient. A common solution is to employ hybrid encryption, where ElGamal encrypts a randomly chosen symmetric key, and this symmetric key is then used to encrypt the actual message using a symmetric cipher.

Ciphertext Expansion : ElGamal encryption results in ciphertexts that are twice the size of the plaintext, due to the pair (c_1, c_2). This expansion can be mitigated by using a Key Encapsulation Mechanism (KEM), where the shared secret is used to derive a symmetric key for encrypting the message, reducing the ciphertext overhead.

Malleability : The scheme is inherently malleable; an adversary can alter a ciphertext (c_1, c_2) to $(c_1, \alpha \cdot c_2)$, resulting in the decryption of $\alpha \cdot m$. This vulnerability renders ElGamal encryption insecure against chosen-ciphertext attacks (CCA).

To enhance security against CCA, the Diffie-Hellman Integrated Encryption Scheme (DHIES) and its elliptic curve variant, the Elliptic Curve Integrated Encryption Scheme (ECIES), have been developed. These schemes integrate ElGamal-like encryption with symmetric encryption and Message Authentication Codes (MACs) to provide authenticated encryption.

DHIES/ECIES Encryption Process :

1. **Ephemeral Key Generation**: The sender generates a random ephemeral private key r and computes the corresponding public key $R = rG$, where G is the generator of the elliptic curve group.
2. **Shared Secret Computation**: Computes the shared secret $S = P_x$, where $P = r \cdot K_B$ and K_B is the recipient's public key.
3. **Key Derivation**: Derives symmetric encryption and MAC keys from S using a Key Derivation Function (KDF): $k_E \| k_M = \text{KDF}(S \| S_1)$, where S_1 is optional shared information.
4. **Message Encryption**: Encrypts the plaintext message m using a symmetric encryption algorithm E with key k_E: $c = E(k_E; m)$.
5. **MAC Computation**: Computes the MAC $d = \text{MAC}(k_M; c \| S_2)$, where S_2 is additional optional shared information.
6. **Ciphertext Formation**: The final ciphertext is the concatenation of R, c, and d.

Decryption Process :

1. **Shared Secret Reconstruction**: The recipient computes the shared secret $S = P_x$ using their private key k_B and the sender's ephemeral public key R: $P = k_B \cdot R$.
2. **Key Derivation**: Derives k_E and k_M from $S \| S_1$ using the same KDF.
3. **MAC Verification**: Verifies the MAC d; if invalid, decryption fails.
4. **Message Decryption**: Decrypts c.

1.4 Digital Signature

Digital signature schemes are fundamental cryptographic constructs that ensure the authenticity, integrity, and non-repudiation of digital communications within public-key infrastructures. Unlike public-key encryption, which aims to maintain the confidentiality of messages, digital signatures provide assurances that a message originates from a specific sender and remains unaltered during transmission. This capability is crucial across various applications, including secure software distribution, financial transactions, and legal document verification. A digital signature scheme is formally defined by a triplet of probabilistic polynomial-time algorithms: key generation (Gen), signing (Sign), and verification (Vrfy). The key generation algorithm Gen, upon input of a security parameter 1^k, outputs a pair of keys: a public verification key pk and a private signing key sk. The signing algorithm Sign takes as input the private key sk and a message m, producing a signature σ. The verification algorithm Vrfy, given the public key pk, a message m, and a signature σ, outputs a boolean value indicating the validity of the signature. For correctness, it is required that for all valid key pairs generated by Gen and for all messages m, the verification algorithm accepts the signature produced by the signing algorithm:

$$\Pr[\text{Vrfy}(pk, m, \text{Sign}(sk, m)) = \text{accept}] = 1$$

The security of a digital signature scheme is characterized by its resistance to existential forgery under adaptive chosen-message attacks. This means that even if an adversary can obtain signatures for messages of their choosing, it remains computationally infeasible to forge a valid signature for any new message not previously signed by the legitimate signer. This property is formalized as:

$$\Pr[\text{Vrfy}(pk, m', \sigma') = \text{accept} \mid (m', \sigma') \notin Q] < \text{negl}(k)$$

here Q denotes the set of message-signature pairs queried during the attack, and $\text{negl}(k)$ represents a negligible function in the security parameter k.

Digital signatures offer several advantages over symmetric-key-based message authentication codes (MACs). Notably, they provide public verifiability, allowing any party to confirm the authenticity of a signed message using the public key, and non-repudiation, ensuring that a signer cannot plausibly deny having signed a message. These properties are essential in scenarios where trust and accountability are paramount, such as in legal agreements and digital certificates. A prevalent method to enhance the efficiency of digital signatures is the hash-and-sign paradigm.

Directly signing long messages can be computationally intensive; therefore, a cryptographic hash function H is employed to compute a fixed-size digest of the message, $h = H(m)$, which is then signed. This approach not only improves performance but also abstracts the signing process from the message length. The security of this paradigm hinges on the collision resistance of the hash function, ensuring that it is computationally infeasible to find two distinct messages m and m' such that $H(m) = H(m')$.

Various digital signature schemes have been developed based on different hard mathematical problems. In the RSA-based signature scheme, for instance, a message digest h is signed by computing $\sigma = h^d \mod N$, where d is the private exponent, and verified by

checking whether $h \equiv \sigma^e \mod N$, with e being the public exponent. However, plain RSA signatures are vulnerable to existential forgery attacks, as an adversary can potentially compute valid signatures for arbitrary messages. To mitigate this, schemes like RSA Full-Domain Hash (RSA-FDH) have been proposed, where the message is first hashed into an element of the appropriate domain before signing. DSA-FDH has been proven secure in the random oracle model under the assumption that the RSA problem is hard. Another class of signature schemes is based on the hardness of the discrete logarithm problem. The Digital Signature Algorithm (DSA), standardized by NIST in the Digital Signature Standard (DSS), operates over finite fields and involves parameters such as a prime modulus p, a subgroup order q, and a generator g. The private key is a randomly selected integer x, and the public key is $y = g^x \mod p$. Signing a message involves generating a per-message secret value k, computing the signature components r and s, and ensuring that k remains secret to prevent private key recovery. Elliptic Curve Digital Signature Algorithm (ECDSA) is a variant of DSA that operates over elliptic curve groups, offering equivalent security with smaller key sizes, which is advantageous for performance and storage efficiency.

Implementing digital signature schemes requires careful attention to key management, parameter selection, and protection against side-channel attacks. The security of the scheme is contingent upon the secrecy of the private key and the integrity of the signing process.

1.4.1 Signatures from the Discrete-Logarithm Problem

A foundational example is the ElGamal signature scheme, introduced by Taher ElGamal in 1985. In this scheme, the global parameters consist of a large prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$. The signer selects a private key x uniformly at random from the set $\{1, 2, \ldots, p-2\}$ and computes the corresponding public key $y = g^x \mod p$. To sign a message m, the signer chooses a random ephemeral key k from $\{1, 2, \ldots, p-2\}$ such that $\gcd(k, p-1) = 1$, ensuring that k is coprime with $p-1$. The signer then computes $r = g^k \mod p$ and $s = k^{-1}(H(m) - xr) \mod (p-1)$, where H denotes a cryptographic hash function, and k^{-1} represents the modular inverse of k modulo $p-1$. The signature is the pair (r, s). Verification involves checking that $0 < r < p$, $0 < s < p-1$, and that $g^{H(m)} \equiv y^r r^s \mod p$. The security of the ElGamal signature scheme is predicated on the intractability of the DLP and the computational Diffie-Hellman problem within the multiplicative group of integers modulo p.

Building upon the ElGamal framework, the Digital Signature Algorithm (DSA) was standardized by the U.S. National Institute of Standards and Technology (NIST) in 1994 as part of the Digital Signature Standard (DSS). DSA operates within a subgroup of $\mathbb{Z}_p^*$ of prime order q, where q divides $p-1$. The parameters include a prime p, a prime divisor q of $p-1$, and a generator $g = h^{(p-1)/q} \mod p$, where h is any integer satisfying $1 < h < p-1$ and $h^{(p-1)/q} \not\equiv 1 \mod p$. The private key x is selected uniformly from $\{1, 2, \ldots, q-1\}$, and the public key is $y = g^x \mod p$. To sign a message m, the signer picks a random k from $\{1, 2, \ldots, q-1\}$, computes $r = (g^k \mod p) \mod q$, and $s = k^{-1}(H(m) + xr) \mod q$. The signature is the pair (r, s). Verification entails ensuring $0 < r < q$, $0 < s < q$, and computing $w = s^{-1} \mod q$, $u_1 = H(m)w \mod q$, $u_2 = rw \mod q$, and checking

that $(g^{u_1}y^{u_2} \mod p) \mod q = r$. The security of DSA is contingent upon the hardness of the DLP in the chosen subgroup.

An advancement in efficiency is exemplified by the Schnorr signature scheme, proposed by Claus Schnorr in 1989. This scheme operates in a subgroup of $\mathbb{Z}_p^*$ with prime order q. The signer selects a private key x from $\{0, 1, \ldots, q-1\}$ and computes the public key $y = g^x \mod p$. To sign a message m, the signer chooses a random k from $\{1, 2, \ldots, q-1\}$, computes $r = g^k \mod p$, and derives the challenge $e = H(m \parallel r) \mod q$, where $\parallel$ denotes concatenation. The response is $s = (k - xe) \mod q$. The signature is the pair (s, e). Verification involves computing $r' = g^s y^e \mod p$ and checking that $e = H(m \parallel r') \mod q$. The Schnorr scheme is lauded for its simplicity and efficiency, producing shorter signatures and requiring fewer computational resources compared to DSA. Its security is based on the hardness of the DLP and is proven in the random oracle model. The Elliptic Curve Digital Signature Algorithm (ECDSA) is an adaptation of DSA that operates over elliptic curve groups, offering equivalent security with smaller key sizes. The domain parameters include an elliptic curve E defined over a finite field $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$, a base point G of prime order n, and associated field parameters. The private key d is a randomly selected integer in $\{1, 2, \ldots, n-1\}$, and the public key is the point $Q = dG$. To sign a message m, the signer selects a random integer k from $\{1, 2, \ldots, n-1\}$, computes the elliptic curve point $R = kG$, and derives $r = R_x \mod n$, where R_x is the x-coordinate of R. If $r = 0$, the process is repeated with a new k. The signer then computes $s = k^{-1}(H(m) + dr) \mod n$, where H is a cryptographic hash function, and k^{-1} is the modular inverse of k modulo n. The signature is the pair (r, s). Verification involves ensuring $0 < r < n$, $0 < s < n$, and computing $w = s^{-1} \mod n$, $u_1 = H(m)w \mod n$, and $u_2 = rw \mod n$. The verifier then computes the elliptic curve point $R' = u_1G + u_2Q$ and checks that $r = R'_x \mod n$, where R'_x is the x-coordinate of R'. ECDSA's security relies on the hardness of the elliptic curve discrete logarithm problem (ECDLP), which is believed to be significantly more challenging than the DLP in finite fields, allowing for shorter key sizes without compromising security. This efficiency makes ECDSA particularly well-suited for resource-constrained environments, such as embedded systems and mobile devices, where computational power and storage are limited.

1.4.2 Signature From Hash Functions

A particularly compelling class of these schemes is hash-based digital signatures, which leverage the properties of cryptographic hash functions to provide security without relying on number-theoretic assumptions, making them resistant to quantum attacks. These schemes encompass various constructions, including one-time signatures, stateful, and stateless schemes, each with distinct mechanisms and security considerations.

One-time signature (OTS) schemes form the foundational layer of hash-based signatures. Introduced by Leslie Lamport in 1979, the Lamport OTS is a seminal example that utilizes cryptographic hash functions. In this scheme, the private key comprises pairs of random values, and the public key consists of the corresponding hash values. To sign a message, each bit of the message determines which value from each pair is revealed; for a bit '0', the first value is disclosed, and for a bit '1', the second. Verification involves hashing the revealed values and comparing them to the public key. The security of the

Lamport OTS hinges on the preimage resistance of the hash function, ensuring that an adversary cannot feasibly invert the hash to forge a signature. However, a significant limitation is that each key pair is secure for only one single message; reusing keys compromises security since previously revealed parts of the private key can aid in forging new signatures. To extend the applicability beyond single-use keys, stateful hash-based signature schemes have been developed. These schemes require the signer to maintain state information to ensure each one-time key pair is used only once. A prominent example is the Merkle Signature Scheme (MSS), proposed by Ralph Merkle. In MSS, a large number of OTS key pairs are generated, and their public keys are organized into a binary tree structure known as a Merkle tree. The root of this tree serves as the global public key. To sign a message, an unused OTS key pair is selected, and along with the OTS signature, an authentication path is provided. This path consists of the sibling nodes required to reconstruct the root of the Merkle tree, allowing the verifier to confirm the validity of the OTS public key used. While MSS enables multiple signatures, it necessitates careful state management to prevent key reuse, which can be complex and error-prone.

Addressing the challenges of state management, stateless hash-based signature schemes have been introduced. A notable example is SPHINCS (Stateless Practical Hash-based Incredibly Nice Cryptographic Signature), which eliminates the need for state tracking by employing a hierarchical structure of multiple layers of Merkle trees. SPHINCS uses a few-time signature (FTS) scheme at the lowest layer, allowing each key pair to sign a limited number of messages securely. The upper layers of Merkle trees certify the public keys of the layers below. To sign a message, SPHINCS selects a random subset of OTS key pairs, reducing the risk of key reuse and distributing security uniformly. The security of SPHINCS relies on the underlying hash function's properties, particularly collision resistance and pseudorandomness. While SPHINCS offers the advantage of being stateless, it comes with trade-offs in terms of larger signature sizes and increased computational overhead compared to stateful schemes.

1.5 Factorization and Discrete Logarithm Computation

1.5.1 Factorization

The security of RSA encryption is fundamentally anchored in the computational difficulty of factoring large composite integers, a problem that has been extensively studied within number theory and cryptography. In RSA, the public key comprises a modulus N, which is the product of two large prime numbers p and q. The security of the system hinges on the infeasibility of deducing these prime factors from N. Over the years, several algorithms have been developed to tackle the integer factorization problem, each with varying degrees of efficiency and applicability depending on the size and nature of N.

One of the earliest and simplest methods is trial division, which involves dividing N by successive integers to find factors. While straightforward, this approach is computationally prohibitive for large N, as its time complexity is exponential relative to the number of digits in N. Consequently, trial division is impractical for the large integers typically used in cryptographic applications. To improve upon this, Pollard's rho algorithm [28],

introduced by John Pollard in 1975, offers a probabilistic method that is particularly effective for numbers with small prime factors. The algorithm generates a pseudo-random sequence of numbers modulo N and employs cycle detection techniques, such as Floyd's cycle-finding algorithm, to identify a non-trivial factor. Its expected time complexity is $O(N^{1/4})$, making it more efficient than trial division but still inadequate for the large semiprimes used in RSA keys. For larger integers, the quadratic sieve algorithm [29], developed in the early 1980s, represented a significant advancement. This algorithm searches for numbers x such that $x^2 \mod N$ is a smooth number (i.e., a number with only small prime factors). By collecting enough of these smooth numbers, one can use linear algebra techniques over finite fields to find a non-trivial factor of N. The quadratic sieve has a sub-exponential time complexity of $O\left(\exp\left((\log N)^{1/2}(\log\log N)^{1/2}\right)\right)$, making it effective for numbers up to about 100 digits. However, its efficiency diminishes for larger integers. The most powerful classical algorithm for factoring large integers is the general number field sieve (GNFS), which has become the standard for factoring large semiprimes, such as those used in RSA encryption. The GNFS operates by selecting a polynomial with a root modulo N and constructing two number fields. It then searches for relations between these fields that lead to a congruence of squares modulo N. The algorithm involves several complex steps, including sieving to find smooth relations, matrix reduction to find dependencies, and square root extraction in the number field. Its heuristic time complexity is $O\left(\exp\left((64/9)^{1/3}(\log N)^{1/3}(\log\log N)^{2/3}\right)\right)$, making it the most efficient known algorithm for factoring large integers. Notably, in 2020, a 250-digit (829-bit) RSA number, known as RSA-250, was factored using the GNFS, requiring approximately 2,700 core-years of computation.

Despite these advancements, the GNFS and other classical algorithms remain computationally intensive for factoring integers of the size commonly used in RSA keys today. To maintain security against these evolving factorization methods, cryptographic standards have progressively increased recommended key sizes. As of recent guidelines, a minimum key size of 2048 bits is recommended for RSA, providing an estimated security strength of 112 bits. For applications requiring higher security, 3072-bit keys are advised, corresponding to a security strength of 128 bits. These recommendations aim to balance security with computational efficiency, acknowledging that larger key sizes offer greater security margins but also incur higher computational costs.

1.5.2 Discrete Logarithm

The discrete logarithm problem (DLP) is a cornerstone of modern cryptography, underpinning the security of numerous protocols such as Diffie-Hellman key exchange, ElGamal encryption, and various elliptic curve cryptosystems. Normally, given a finite cyclic group G with a generator g and an element h in G, the DLP entails finding an integer x such that $g^x \equiv h \mod p$, where p is a prime number and g is a primitive root modulo p. The presumed computational intractability of solving the DLP for large p ensures the security of these cryptographic systems. Over time, several algorithms have been developed to address the DLP, each with varying degrees of efficiency depending on the group structure and size. Among the foundational methods are the Pohlig-Hellman and

baby-step/giant-step algorithms.

The Pohlig-Hellman algorithm [27], introduced by Steven Pohlig and Martin Hellman in 1978, exploits the factorization of the group order to decompose the DLP into smaller, more manageable subproblems. Specifically, if the order of the group $|G|$ factors into small prime powers, the algorithm reduces the original DLP to multiple DLPs in these smaller subgroups. By solving each subproblem and combining the results using the Chinese Remainder Theorem, the overall problem becomes more tractable. The efficiency of this algorithm is contingent upon the factorization of $|G|$; it is particularly effective when $|G|$ has small prime factors but offers no advantage when $|G|$ is a large prime, a common choice in cryptographic applications to mitigate this very vulnerability.

In contrast, the baby-step/giant-step algorithm, proposed by Daniel Shanks in 1971, provides a general solution to the DLP without relying on specific group factorizations. This algorithm employs a meet-in-the-middle strategy to achieve a time complexity of $O(\sqrt{n})$, where n is the order of the group. The method involves precomputing a table of "baby steps" (successive powers of the generator g) and then searching for matches using "giant steps" (multiples of the group order). While more efficient than brute-force approaches, the baby-step/giant-step algorithm requires significant computational resources and storage, rendering it impractical for groups with large orders typically used in cryptographic settings.

Building upon these foundational algorithms, more advanced methods have been developed to tackle the DLP in specific contexts. The index calculus algorithm, for instance, offers sub-exponential running times for certain cyclic groups by leveraging the representation of group elements and factoring techniques. This method involves generating a system of linear equations involving logarithms of small prime factors and solving for the unknowns using linear algebra. The preprocessing step of generating these equations can be reused across multiple DLP instances within the same group, enhancing efficiency. The index calculus algorithm is particularly effective in groups like $\mathbb{Z}_p^*$, the multiplicative group of integers modulo p, where it serves as the basis for the number field sieve-the most efficient known method for computing discrete logarithms in large prime fields. Recent advancements have further refined these approaches. For example, the **double index calculus algorithm** has been proposed to solve the DLP in finite prime fields more efficiently. Additionally, significant progress has been made in computing discrete logarithms in finite fields of small characteristic, raising concerns about the security of cryptographic schemes that rely on such groups.

In contrast, the elliptic curve discrete logarithm problem (ECDLP) remains resistant to sub-exponential attacks, making elliptic curve cryptography (ECC) an attractive option for secure communications. ECC offers comparable security to traditional systems like RSA but with significantly smaller key sizes, leading to improved performance and reduced computational overhead. For instance, a 256-bit key in ECC is considered to provide a security level equivalent to a 3072-bit key in RSA. The selection of appropriate cryptographic key lengths is crucial to maintaining security against evolving computational capabilities. Organizations such as the National Institute of Standards and Technology (NIST) provide guidelines on key sizes to ensure robust security. For example, NIST recommends a minimum of 2048-bit keys for RSA and 256-bit keys for ECC

to achieve an adequate security level. These recommendations are periodically updated to reflect advancements in algorithmic techniques and increases in computational power.

1.6 PKC exploitation

While current public key cryptography (PKC) systems remain secure against classical algorithmic attacks, they are not impervious to other forms of exploitation. Side-channel attacks, for instance, leverage unintended information leakage, such as timing discrepancies, power consumption variations, or electromagnetic emissions during cryptographic operations-to extract secret keys or sensitive data. Man-in-the-middle (MITM) attacks present another significant threat. In these scenarios, attackers intercept and potentially alter communications between two parties who believe they are directly communicating with each other. This can compromise the integrity and confidentiality of the exchanged information. MITM attacks can be particularly effective during the key exchange process, where the attacker can negotiate cryptographic parameters with both parties, thereby gaining unauthorized access to the communication channel.

To mitigate these risks, it is essential to implement robust security measures. This includes employing constant-time algorithms to prevent timing attacks, integrating hardware countermeasures against power analysis, and ensuring proper authentication protocols to thwart MITM attempts. Regularly updating cryptographic libraries and staying informed about emerging vulnerabilities are also crucial steps in maintaining the security of PKC systems. By proactively addressing these challenges, we can enhance the resilience of public key cryptography against non-algorithmic threats.

Chapter 2

Quantum computers

Quantum computing represents a groundbreaking advancement in computation, harnessing the principles of quantum mechanics to process and store information in fundamentally different ways compared to classical computers. While classical computing relies on bits, which are confined to binary values of '0' or '1', quantum computing uses qubits, which can exist simultaneously in both states, thanks to quantum superposition. This allows quantum computers to access a vastly larger computational space, offering the potential to solve certain problems exponentially faster than classical computers. Quantum properties like superposition, interference, and entanglement enable quantum computers to explore many computational paths at once, dramatically increasing efficiency for specific tasks, such as prime number factorization, quantum simulations, and data search. The ability of quantum computers to outperform classical systems is exemplified by Shor's algorithm, which can factor large numbers much faster than traditional methods, posing a significant challenge to current cryptographic protocols that rely on the difficulty of such tasks. This has led to widespread interest in the field, particularly from cryptography experts concerned about the future of secure communication.

The development of quantum computing has been heavily influenced by the work of early pioneers such as Richard Feynman, who first proposed the idea of a quantum computer in 1981, and David Deutsch, who in 1985 formalized the concept of a universal quantum computer. In the 1990s, the field gained significant momentum with the development of algorithms like Shor's and Grover's, which demonstrated the potential for quantum computers to solve certain problems more efficiently than classical systems. Shor also introduced Quantum Error Correction (QEC) in 1995, showing that information from one logical qubit can be encoded onto multiple physical qubits, protecting it from errors. Shor's work demonstrated the possibility of executing quantum computations reliably with noisy quantum hardware. Further research has revealed that noise and quantum scaling are related, so if errors and noise are below a certain threshold then it's theoretically possible to scale up quantum computers to larger sizes. These breakthroughs sparked a wave of interest from both academia and industry, with major companies like IBM, Google, and Microsoft investing heavily in quantum research. In 2019 Google demonstrated quantum supremacy, where a quantum computer performed a specific task faster than a classical supercomputer [13]. Moreover, there has been

rapid progress in quantum algorithms, with hybrid quantum-classical approaches like the Variational Quantum Eigensolver (VQE) [26] gaining traction. These algorithms, which combine the strengths of both classical and quantum computing, have shown promise in areas like quantum chemistry simulations and quantum artificial intelligence. In fact, quantum computing has the potential to significantly accelerate machine learning tasks, though it remains to be seen whether quantum algorithms will ultimately outperform classical techniques in all areas. The field also saw progress in quantum cryptography, with the development of protocols like BB84 [19] and advances in entanglement-based secure communication. As of 2025, the field continues to grow rapidly, with quantum computing becoming one of the most discussed areas of research. A wide range of companies are working on developing quantum hardware and software. In addition to hardware advancements, several quantum programming languages and libraries, such as Qiskit, Cirq, and PennyLane, are available to facilitate the development of quantum algorithms. These tools allow researchers to create quantum circuits and test algorithms on available quantum devices.

Despite its potential, quantum computing is still in its early stages, currently entering the Noisy Intermediate-Scale Quantum (NISQ) era. At this point, quantum computers are not yet fault-tolerant and are subject to noise and imperfections, which limit their practical capabilities nonetheless they are already sufficiently reliable for solving certain problems more efficiently than classical computers. Researchers are working on error correction techniques and hardware improvements to make quantum devices more reliable. The NISQ era provides valuable opportunities to refine quantum algorithms, explore error correction, and test quantum hardware, paving the way for scalable, fault-tolerant quantum computers in the future. Progress has been made with various quantum computing platforms, including superconducting qubits, trapped ions, and topological qubits, each offering unique advantages and challenges. However, there is no clear consensus on the best approach, and the diversity of research reflects the exploratory nature of the field. Current quantum devices are primarily limited by issues like decoherence, where qubits lose their quantum coherence when interacting with the environment, making it a major obstacle for large-scale quantum computing. NISQ devices are designed to cope with these imperfections, but developing error correction procedures and reducing decoherence rates remain key research goals.

The field of quantum computing is diverse, with several paradigms being explored. The most popular are measurement-based quantum computing, adiabatic quantum computing (implemented as quantum annealing), and gate-based quantum computing, which uses the quantum circuit model. The quantum circuit model, in particular, is flexible and programmable, making it a feasible approach for many quantum computing applications. In this model, quantum algorithms are expressed as sequences of quantum gates, which manipulate qubits to perform computations. These gates can be combined to form quantum circuits that solve complex problems.

2.1 Theoretical aspects

2.1.1 Mathematical concepts

Quantum mechanics often feels daunting at first, not only because of its counterintuitive principles but also due to its reliance on unfamiliar notation and abstract linear algebraic structures. These mathematical tools, however, form the bedrock of the theory, enabling precise descriptions of quantum states and their evolution. At its core, the framework builds on vector spaces, particularly $\mathbb{C}^n$, the space of complex n-tuples where vectors can be combined through addition or scaled by complex numbers. The zero vector, for instance, acts as a neutral element in this algebraic system. Within these spaces, vector subspaces, subsets that themselves satisfy the axioms of a vector space allow us to isolate smaller, self-contained regions of interest.

Dirac notation streamlines these ideas with elegant symbolism: a vector is written as $|\psi\rangle$ (a *ket*), its dual (conjugate transpose) as $\langle\psi|$ (a *bra*), and their inner product as $\langle\varphi|\psi\rangle$, which quantifies their "overlap" in the space. This notation extends naturally to operations like the tensor product (e.g., $|\varphi\rangle \otimes |\psi\rangle$), which combines vectors from separate spaces into composite states. For matrices, the Hermitian conjugate ($A^\dagger$), a combination of complex conjugation and transposition, generalizes the notion of transposition to complex-valued operators, ensuring compatibility with quantum mechanics' probabilistic structure.

A vector space's structure is deeply tied to its spanning sets and bases. A spanning set acts as a "generator" for the space: any vector can be expressed as a linear combination of its elements. However, redundancy often exists in such sets. A basis eliminates this redundancy by being a minimal spanning set of linearly independent vectors, vectors that cannot be expressed as combinations of one another. The number of basis vectors defines the space's dimension, a measure of its complexity. Operators, represented by matrices, act linearly on these vectors.

Inner products elevate vector spaces to Hilbert spaces, a complete inner product space, by assigning complex numbers to vector pairs, encoding geometric relationships like angles and lengths. Two vectors are orthogonal if their inner product vanishes ($\langle\varphi|\psi\rangle = 0$), and an orthonormal basis ensures every basis vector is a unit vector and mutually orthogonal. The Gram-Schmidt procedure systematically transforms any basis into such an orthonormal set, simplifying computations and ensuring stability in quantum algorithms.

Key operator properties emerge through their interactions with vectors and other operators. The outer product $|w\rangle\langle v|$, for instance, maps vectors via linear transformations, acting as a "building block" for more complex operators. The completeness relation ($\sum_i |i\rangle\langle i| = I$) formalizes the idea that an orthonormal basis fully spans the space: any vector can be reconstructed from its projections onto the basis. Meanwhile, inequalities like Cauchy-Schwarz ($|\langle v|w\rangle|^2 \leq \langle v|v\rangle\langle w|w\rangle$) set fundamental limits on correlations between states, reflecting quantum mechanics' probabilistic nature.

Eigenvalues and eigenvectors reveal an operator's scalar invariants: an eigenvector $|v\rangle$ satisfies $A|v\rangle = \lambda|v\rangle$, where λ is the eigenvalue. Solving the characteristic equation ($\det(A - \lambda I) = 0$) uncovers these eigenvalues, while diagonalization decomposes operators into their spectral form ($A = \sum_i \lambda_i |i\rangle\langle i|$). This is possible for normal operators

($AA^\dagger = A^\dagger A$), a class that includes Hermitian operators ($A^\dagger = A$), which model physical observables with real eigenvalues, and unitary operators ($U^\dagger U = I$), which preserve inner products and describe reversible quantum evolution.

Composite quantum systems rely on tensor products to merge spaces ($V \otimes W$) and operators ($A \otimes B$). For example, if A acts on qubit 1 and B on qubit 2, their tensor product $A \otimes B$ acts on the combined system, with $(A \otimes B)(|v\rangle \otimes |w\rangle) = A|v\rangle \otimes B|w\rangle$. The Kronecker product mirrors this abstract operation for matrices: for $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and B, $A \otimes B = \begin{bmatrix} aB & bB \\ cB & dB \end{bmatrix}$, scaling each element of A by the entire matrix B. This block structure is indispensable for modeling multi-qubit gates in quantum circuits.

Operators also decompose into simpler, interpretable forms. The polar decomposition ($A = UJ$ or $A = KU$) splits an operator into a unitary U (representing rotation) and a positive operator J/K (representing stretching). The singular value decomposition ($A = UDV$) factors matrices into two unitaries (U, V) and a diagonal matrix D of non-negative singular values, offering geometric insight into linear transformations.

Finally, the trace (sum of diagonal elements) acts as a scalar invariant of operators, unaffected by basis changes, while commutators ($[A, B] = AB - BA$) and anti-commutators ($\{A, B\} = AB + BA$) quantify operator relationships. Commuting Hermitian operators ($[A, B] = 0$) share eigenbases, enabling simultaneous diagonalization, a property critical for measuring compatible observables like position and momentum in classical physics, but restricted in quantum theory by the uncertainty principle.

Together, these concepts weave a mathematical tapestry that underpins quantum mechanics, transforming abstract definitions into tools for modeling superposition, entanglement, and measurement. By grounding quantum phenomena in linear algebra's rigor, we gain both computational power and conceptual clarity.

2.1.2 Quantum mechanics

Quantum mechanics provides a fundamental mathematical framework for understanding physical systems, defining the principles that govern quantum states, evolution, and measurement.

The first postulate establishes that any isolated physical system is associated with a complex vector space endowed with an inner product where the system's state is represented by a unit vector. The simplest quantum system, the qubit, has a two-dimensional state space spanned by orthonormal basis vectors $\{|0\rangle, |1\rangle\}$, with an arbitrary state vector expressed as $|\psi\rangle = a|0\rangle + b|1\rangle$, where a and b are complex numbers satisfying the normalization condition $|a|^2 + |b|^2 = 1$. This superposition principle differentiates qubits from classical bits, allowing them to exist in states that are not definitively $|0\rangle$ or $|1\rangle$ but rather a combination of both.

The second postulate that a closed quantum system changes in time by applying a special kind of "perfectly reversible" transformation U that always keeps the total probability equal to one, which means $U^\dagger U = I$. Mathematically, this is captured by the Schrödinger

equation $i\hbar \frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle$, where the Hamiltonian H is a Hermitian operator representing the system's total energy and ensuring real energy values. When H doesn't change with time, you can solve this equation exactly to get $U(t_2, t_1) = \exp(-\frac{i}{\hbar}H\,(t_2-t_1))$, which, if you break H into its energy eigenstates $H = \sum_E E\,|E\rangle\langle E|$, shows that each energy state simply picks up a phase factor $e^{-iE(t_2-t_1)/\hbar}$. In everyday terms, the system's state vector "rotates" in a way that conserves probability, can be reversed by running time backward, and makes any observable that commutes with H remain unchanged over time.

The third postulate of quantum mechanics formalizes the inherently probabilistic nature of quantum measurement. At its heart, this postulate asserts that measuring a quantum system is governed by a set of mathematical entities called measurement operators $\{M_m\}$, which act on the system's state space. Suppose the system is initially in a state $|\psi\rangle$. When a measurement is performed, each possible outcome m is tied to a specific operator M_m. The probability of observing outcome m is calculated as $p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$, reflecting how the operator M_m "overlaps" with the state. After the measurement, the system's state collapses to $\frac{M_m|\psi\rangle}{\sqrt{p(m)}}$, a process that highlights the abrupt, discontinuous nature of quantum observation. To ensure consistency with probability theory, these operators must satisfy the completeness relation $\sum_m M_m^\dagger M_m = I$, where I is the identity operator. This equation guarantees that the probabilities of all possible outcomes sum to one, preserving the probabilistic framework of quantum theory. A pivotal special case arises with projective measurements, where the operators $\{M_m\}$ are replaced by orthogonal projectors $\{P_m\}$. These projectors satisfy $P_m P_n = \delta_{mn} P_m$, meaning they partition the state space into mutually exclusive, non-overlapping subspaces. Each projector P_m corresponds to a distinct eigenvalue of a Hermitian observable M, which represents a physical quantity like energy or spin. The observable can be decomposed as $M = \sum_m mP_m$, linking measurement outcomes to eigenvalues m. The expectation value $\langle M\rangle = \langle\psi|M|\psi\rangle$ quantifies the average outcome of many measurements on identically prepared states, while the uncertainty $\Delta M = \sqrt{\langle M^2\rangle - \langle M\rangle^2}$ measures the statistical spread of results. These concepts underpin the famous Heisenberg uncertainty principle, which states that for two incompatible observables A and B, the product of their uncertainties is bounded by $\Delta A \Delta B \geq \frac{1}{2}|\langle[A, B]\rangle|$, where $[A, B] = AB - BA$ is their commutator. This principle encodes the fundamental trade-off in precision when measuring non-commuting quantities, such as position and momentum.

The fourth postulate addresses composite quantum systems, stating that the state space of a composite system is the tensor product of the state spaces of its component systems. If system i is in state $|\psi_i\rangle$, the joint state of the composite system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$, a crucial principle for understanding entanglement. An entangled state cannot be written as a product of individual states, as exemplified by $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$, which underpins quantum computation and communication protocols such as quantum teleportation and superdense coding. The measurement postulate extends to general measurements described by Positive Operator-Valued Measures (POVMs), a set of positive operators $\{E_m\}$ satisfying $\sum_m E_m = I$, which generalizes projective measurements and is particularly useful for optimizing state discrimination in quantum information processing. The concept of phase in quantum mechanics plays a crucial role in interference effects, where a quantum state $|\psi\rangle$ can acquire a global phase factor $e^{i\theta}$ without affecting

measurement outcomes, while relative phase differences between superposition components lead to physically observable effects. The inability to directly observe the state vector, the invasive nature of quantum measurement, and the probabilistic collapse of the wavefunction highlight the departure of quantum mechanics from classical intuition. These principles form the theoretical foundation for quantum computing and quantum communication.

2.2 Main practical concepts

The following presents some important concepts commonly used in the field of quantum computing.

2.2.1 Qbits

In classical computing, a bit is analogous to a binary light switch, capable of assuming only two discrete states, 0 or 1. In contrast, a qubit functions more like a dimmer switch, existing in an intermediate state as a linear combination of 0 and 1, weighted by probability amplitudes. A qubit's wave-like nature follows the Schrödinger equation, theoretically existing in an infinite-dimensional Hilbert space. However, for practical purposes, it is often represented in a two-dimensional Hilbert space using Dirac's braket notation. In an N-dimensional Hilbert space a quantum system's state is expressed as a linear combination of basis vectors:

$$\psi = (c_0, c_1, \ldots, c_{N-1})^\top = c_0|0\rangle + c_1|1\rangle + \ldots + c_{N-1}|N-1\rangle,$$

where the basis vectors such as $|0\rangle$, $|1\rangle$, $|2\rangle$, ..., $|N-1\rangle$ represent distinct quantum states. These basis states are orthonormal, satisfying $\langle i|j\rangle = 0$ if $i \neq j$ and $\langle i|j\rangle = 1$ if $i = j$, where $|j\rangle$ is its Hermitian conjugate is the complex conjugate transpose of $\langle j|$ and $\langle i|j\rangle$ represents the inner product (or scalar product) between two quantum states. The inner product gives a scalar value that quantifies the similarity between the two quantum states. The state vector must be normalized so that the sum of the squared magnitudes of its coefficients equals one:

$$\langle\psi|\psi\rangle = |c_0|^2 + |c_1|^2 + \ldots + |c_{N-1}|^2 = 1$$

, this encapsulates quantum superposition. The probability of measuring the system in a particular state $|i\rangle$ is given by $|c_i|^2$, ensuring that the total probability sums to one. This concept is famously illustrated by Schrödinger's cat thought experiment, where a cat is considered to be in a superposition of being both alive and dead until an observation collapses the state. The qubit is therefore mathematically described as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers satisfying the normalization condition $|\alpha|^2+|\beta|^2 = 1$. Physical implementations of qubits include the polarization states of photons, spin states of electrons, and energy levels of atoms, allowing controlled manipulation for quantum information processing.

A two-qubit system has computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, with a general two-qubit state expressed as

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

where the coefficients determine probability amplitudes. Generally, an n-qubit system is described as a superposition of 2^n computational basis states, requiring 2^n complex numbers to fully specify its state, highlighting the exponential scaling of quantum state space and its potential computational power over classical systems. However, harnessing this power necessitates precise control over quantum operations and measurements, with quantum algorithms and error correction methods playing a crucial role.

In February 2025, physicists unveiled an eight-qubit topological quantum processor, marking a substantial leap forward in the field. Additionally, IBM's recent unveiling of the Condor processor, with its 1,121 superconducting qubits, showcases the progress in scaling quantum processors.

Superposition Quantum superposition is a fundamental principle of quantum mechanics stating that a quantum system can exist in multiple states simultaneously until it is measured. Quantum superposition allows qubits to encode and process multiple states at once, enabling quantum computers to explore a vast computational space exponentially larger than that of classical systems. This exponential growth in computational power arises because each additional qubit doubles the number of possible states the system can represent. However, the power of superposition comes with a critical caveat: when a qubit is measured, its superposition collapses to one of the basis states, $|0\rangle$ or $|1\rangle$, with probabilities determined by the squared magnitudes of the coefficients α and β. This collapse is a fundamental aspect of quantum mechanics and underscores the probabilistic nature of quantum systems.

Entanglement In classical computing, the state of one bit is independent of another, but in quantum systems, entanglement creates dependencies between qubits. In a multi-qubit quantum system, entanglement occurs when the state of the composite system cannot be expressed as a product of individual qubit states. For example, consider a two-qubit system whose state is described by

$$|\psi\rangle = c_0\,|00\rangle + c_1\,|01\rangle + c_2\,|10\rangle + c_3\,|11\rangle\,.$$

If this state can be factored into the product of two single-qubit states, such as

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = (a_1\,|0\rangle + b_1\,|1\rangle) \otimes (a_2\,|0\rangle + b_2\,|1\rangle),$$

then it is not entangled. However, if such a factorization is impossible, the state is entangled. Examples of unentangled states include $|00\rangle$ and $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$, while entangled states include $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ and $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. These entangled states, often referred to as Bell states or EPR (Einstein-Podolsky-Rosen) pairs, exhibit correlations that are stronger than any classical system can achieve. For instance, the Bell state

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

cannot be separated into individual qubit states, meaning the measurement of one qubit instantaneously determines the state of the other, regardless of the distance between them. This non-local correlation, famously highlighted by Einstein, Podolsky, and Rosen, challenges classical intuitions and has profound implications for quantum communication and computation.

This property has many practical application, for example in quantum communication protocols like quantum key distribution (QKD), where entanglement ensures secure information transfer by detecting eavesdropping attempts. Entanglement also enable "quantum parallelism," where multiple computations can be performed simultaneously. For example, applying a Hadamard gate to three qubits initially in a superposition of $|0\rangle$ and $|1\rangle$ creates a state that represents all eight possible combinations of three-bit values (000, 001, 010, ..., 111) at once. This state cannot be decomposed and any change in one qubit's state immediately affects the other. This exponential scaling of the state space allows quantum computers to process vast amounts of information in parallel, providing a significant computational advantage over classical systems. Unlike classical bits, where combining two bits increases information linearly, entangling qubits multiplies their dimensions, exponentially expanding the computational possibilities. This property is harnessed in quantum algorithms like Shor's algorithm for factoring large numbers and Grover's algorithm for unstructured search. Entanglement also plays a central role in quantum error correction, where information from one logical qubit is encoded across multiple physical qubits to protect against errors. For example, Shor's error-correcting code uses entanglement to distribute quantum information redundantly, allowing errors to be detected and corrected without directly measuring the qubits. This is crucial for building fault-tolerant quantum computers capable of performing reliable computations despite noise and decoherence.

Interference Quantum interference is a fundamental phenomenon that arises from the wave-like nature of quantum particles, such as electrons or photons. While qubits are often represented mathematically using braket notation or visualized on the Bloch sphere, these are merely abstract representations of their underlying quantum states. In reality, a qubit's state is described by a quantum wavefunction, a mathematical entity that satisfies the Schrödinger equation and encapsulates the complex probability amplitudes of the system. These probability amplitudes determine the likelihood of the qubit being in a particular state upon measurement. When multiple qubits are involved, their individual wavefunctions combine to form an overall wavefunction that describes the resultant states of the quantum system. This process of combining wavefunctions is known as quantum interference. Interference occurs when two or more quantum wavefunctions overlap, leading to constructive or destructive interference. Constructive interference amplifies the probability amplitudes of certain states, while destructive interference diminishes or cancels them out. This alteration of the resultant wavefunction directly affects the probability distribution of the quantum system's states, enabling quantum computers to perform computations in ways that classical systems cannot.

Interference is exploited to manipulate probability amplitudes and achieve computational advantages. For example, in the quantum Fourier transform, a key component

of Shor's algorithm, interference is used to extract periodicity information from quantum states. In Grover's algorithm interference is harnessed to amplify the probability amplitudes of correct solutions while suppressing those of incorrect ones.

Noise Quantum noise refers to the uncertainty and fluctuations that arise in quantum systems due to the probabilistic nature of quantum mechanics. It poses significant challenges in quantum computing, even at low temperatures. Unlike classical noise, which is often associated with random variations in signals or disturbances caused by external factors, quantum noise can have more complex and detrimental effects. In classical systems, noise typically adds random errors to a signal. In contrast, when a quantum system is in a superposition state, its outcome upon measurement is not deterministic but is determined by the probability distribution of the quantum states. Noise and errors can affect the outcome due to the quantum system's interaction with the external environment. This interaction can lead to the loss of quantum properties-such as superposition, entanglement, and interference-over time, thereby affecting the performance of quantum circuits. Quantum noise manifests in several ways within quantum systems, impacting various aspects of quantum computing. Common examples include:

- Measurement Noise: When measuring a quantum system, the act of measurement can cause the system to lose its quantum superposition, collapsing the quantum state into one of its possible outcomes. This process introduces uncertainty due to the probabilistic nature of quantum measurements.

- Decoherence: Interactions with the environment can cause quantum systems to lose coherence, leading to the loss of superposition, entanglement, and interference. Decoherence significantly affects the performance of quantum algorithms and is a major obstacle to developing practical quantum computers.

To address the challenges posed by quantum noise, researchers have been developing quantum error correction (QEC) techniques. These techniques are essential for preserving quantum states against the detrimental effects of measurement noise and decoherence. Unlike classical error correction, which often employs redundancy, QEC involves encoding quantum information across multiple physical qubits to detect and correct errors without directly measuring the quantum information, thus avoiding further disturbance to the system. Implementing effective QEC is crucial for achieving fault-tolerant quantum computing, where computations can proceed reliably despite the presence of noise.

2.2.2 Bloch sphere

The Bloch sphere is a geometric representation that provides an intuitive and powerful way to visualize the state of a single qubit, the fundamental unit of quantum information. Any pure quantum state of a qubit can be expressed in the form:

$$|\psi\rangle = e^{i\gamma}\left[\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right]$$

where γ is a global phase factor, θ and ϕ are angles that define the state's orientation, and $|0\rangle$ and $|1\rangle$ are the computational basis states. The global phase $e^{i\gamma}$ has no observable effect on the system, so it can be neglected, simplifying the state to:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

This state can be represented as a unit vector on the surface of the Bloch sphere, a three-dimensional sphere with radius one, where the vector's coordinates are given by:

$$(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$$

The angles θ and ϕ determine the direction of this unit vector, with θ representing the polar angle measured from the positive z-axis and ϕ representing the azimuthal angle in the xy-plane.

On the Bloch sphere, specific points correspond to well-known quantum states. For example, the unit vector pointing in the positive z-direction represents the $|0\rangle$ state, while the vector pointing in the negative z-direction represents the $|1\rangle$ state. Similarly, the vector pointing in the positive x-direction corresponds to the state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, and the vector pointing in the positive y-direction corresponds to $\frac{|0\rangle+i|1\rangle}{\sqrt{2}}$. The difference between these two states lies in their relative phase ϕ, which is $0°$ for $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $90°$ for $\frac{|0\rangle+i|1\rangle}{\sqrt{2}}$. This relative phase is a crucial aspect of quantum states, as it influences interference effects and the outcomes of quantum operations.

While pure states lie on the surface of the Bloch sphere, mixed states, which represent statistical mixtures of pure states, lie inside the sphere. A mixed state is described by a density matrix and can be visualized as a point within the sphere, with its position determined by the probabilities and phases of the constituent pure states.

The Bloch sphere is not only a valuable tool for visualizing single-qubit states but also for understanding and analyzing single-qubit operations. Quantum gates, such as the Pauli-X, Pauli-Y, and Pauli-Z gates, can be interpreted as rotations on the Bloch sphere. For instance, the Pauli-X gate performs a $180°$ rotation around the x-axis, flipping the state $|0\rangle$ to $|1\rangle$ and vice versa. Similarly, the Hadamard gate, which creates superpositions, can be visualized as a rotation that maps states between the z-axis and the x-axis. These rotations are described by the following operators:

$$R_x(\theta) = e^{\left(-i\frac{\theta}{2}X\right)} = \cos\left(\frac{\theta}{2}\right)I - i\sin\left(\frac{\theta}{2}\right)X$$

$$R_y(\theta) = e^{\left(-i\frac{\theta}{2}Y\right)} = \cos\left(\frac{\theta}{2}\right)I - i\sin\left(\frac{\theta}{2}\right)Y$$

$$R_z(\theta) = e^{\left(-i\frac{\theta}{2}Z\right)} = \cos\left(\frac{\theta}{2}\right)I - i\sin\left(\frac{\theta}{2}\right)Z$$

The Bloch sphere also aids in understanding relaxation and decoherence processes, where a qubit's state evolves from a pure state on the surface of the sphere toward a mixed state inside the sphere due to interactions with the environment. This visualization helps

researchers design error mitigation strategies and develop quantum error correction techniques to preserve the coherence of qubits. It is also used to design and optimize quantum algorithms, as it provides a clear geometric interpretation of quantum states and operations. Furthermore, the Bloch sphere is instrumental in quantum state tomography, a technique used to reconstruct the density matrix of an unknown quantum state by performing measurements along different axes of the sphere. This process is essential for characterizing and calibrating quantum devices, ensuring their reliability and performance. The Bloch sphere is limited to representing single-qubit states and cannot directly visualize multi-qubit systems, which require more complex mathematical frameworks.

2.2.3 Density matrix

The density matrix is used to describe both pure and mixed quantum states, providing a comprehensive representation of a quantum system's statistical properties. For a pure state $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$, the density matrix is defined as $\rho = |\psi\rangle\langle\psi|$, where $\langle\psi| = \begin{bmatrix} a^* & b^* \end{bmatrix}$ is the conjugate transpose of $|\psi\rangle$. In matrix form, this becomes

$$\rho = \begin{bmatrix} |a|^2 & ab^* \\ a^*b & |b|^2 \end{bmatrix},$$

where the diagonal elements $|a|^2$ and $|b|^2$ represent the probabilities of measuring the system in the $|0\rangle$ and $|1\rangle$ states, respectively, while the off-diagonal elements ab^* and a^*b capture the coherence between these states. For mixed states, which arise when the exact state of the system is unknown or when the system interacts with its environment, the density matrix generalizes to

$$\rho = \sum_i p_i |\phi_i\rangle\langle\phi_i|,$$

where $p_i \geq 0$ and $\sum_i p_i = 1$. Here, p_i represents the probability of the system being in the pure state $|\phi_i\rangle$. The density matrix is Hermitian, meaning $\rho = \rho^\dagger$, and its trace (the sum of its diagonal elements) is always 1, reflecting the normalization of probabilities. A pure state satisfies $\mathrm{Tr}(\rho^2) = 1$, while a mixed state satisfies $\mathrm{Tr}(\rho^2) < 1$, indicating the presence of statistical mixtures. In practice, mixed states are more common due to decoherence, where interactions with the environment cause the off-diagonal elements of the density matrix to decay, transforming a pure state into a mixed one. This process underscores the importance of the density matrix in describing real-world quantum systems, where environmental interactions and noise are inevitable. By capturing both the probabilistic nature and coherence of quantum states, the density matrix is an essential tool for analyzing quantum systems and designing quantum algorithms.

2.2.4 Quantum state initialization

Quantum state initialization is a critical first step in quantum computing, ensuring that qubits start in a well-defined and known state before any computation begins. Similar to classical computing, where bits are typically initialized to a known value (often 0),

quantum computing commonly starts with qubits in the $|0\rangle$ state. For a single qubit, this initial state corresponds to the density matrix $\rho = |0\rangle\langle 0| = \frac{1}{2}I + \frac{1}{2}\sigma_z$, where I is the identity matrix and σ_z is the Pauli-Z operator. The polarization, or $\langle\sigma_z\rangle$, of this state is 1, representing a fully polarized state, as the maximum possible value of $|\langle\sigma_z\rangle|$ is 1. The goal of quantum initialization is to maximize the purity or polarization of the system, ideally achieving a value of 1, which corresponds to a pure state. However, achieving this ideal initialization can be challenging and often depends on the specific physical system used to implement qubits. For example, in liquid-state nuclear magnetic resonance (NMR) systems, the thermal equilibrium state at room temperature has extremely low polarization, making it difficult to achieve true initialization. As a result, these systems often rely on pseudo-pure states, which mimic the behavior of pure states for computational purposes. In diamond color center systems, initialization is frequently accomplished through fluorescence techniques, which prepare the qubit in the ground state $|0\rangle$. In superconducting qubit systems, one common initialization method involves performing a projective measurement, which collapses the qubit into either the $|0\rangle$ or $|1\rangle$ state. The $|0\rangle$ state is then selected and further processed for quantum computation. Each physical platform has its own unique challenges and methods for initialization, but the overarching goal remains the same: to prepare qubits in a well-defined, high-purity state to ensure the reliability and accuracy of subsequent quantum operations.

2.2.5 Quantum-gate

Quantum gates are fundamental building blocks in quantum computing, serving as operations that transform one quantum state into another by manipulating qubits. These gates are analogous to classical logic gates but leverage the principles of quantum mechanics, such as superposition and entanglement, to perform computations. The evolution of quantum states is governed by the Schrödinger equation:

$$i\hbar\frac{d}{dt}|\psi\rangle = H|\psi\rangle$$

where H is the system Hamiltonian. By controlling H and the evolution time, quantum gates can be implemented as unitary matrices U, which map an initial state $|\psi_1\rangle$ to a final state $|\psi_2\rangle$ through $U|\psi_1\rangle = |\psi_2\rangle$. For density matrices, this transformation is expressed as $U\rho_1 U^\dagger = \rho_2$. The unitary matrix U is derived from the time-ordered exponential:

$$U = T\exp\left(-\frac{i}{\hbar}\int_{t_1}^{t_2} H(t)\,dt\right)$$

which accounts for both time-independent and time-dependent Hamiltonians. In an n-qubit system U is a $2^n \times 2^n$ matrix, reflecting the exponential scaling of quantum state space.

Quantum gates can be categorized into single-qubit and multi-qubit gates. Single-qubit gates, such as the Hadamard gate (H), the phase gate ($P(\pi/4)$), and rotation gates (R), manipulate individual qubits. The Hadamard gate, represented by:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

creates equal superpositions of the basis states $|0\rangle$ and $|1\rangle$, transforming $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The phase gate $P(\pi/4)$, represented by:

$$P(\pi/4) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

introduces a phase shift of $\pi/4$ to the $|1\rangle$ state without altering the probability distribution. Rotation gates, defined as:

$$R = \exp\left(-i\frac{\theta}{2}(n_x\sigma_x + n_y\sigma_y + n_z\sigma_z)\right)$$

perform rotations on the Bloch sphere around an axis (n_x, n_y, n_z) by an angle θ. These gates are essential for creating and manipulating superpositions and phases, which are critical for quantum algorithms.

Multi-qubit gates, such as the controlled-NOT (CNOT) gate and the SWAP gate, operate on multiple qubits and are crucial for creating entanglement and performing complex computations. The CNOT gate, represented by:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

flips the target qubit if the control qubit is in the $|1\rangle$ state, leaving it unchanged otherwise. This gate is fundamental for entangling qubits and is a key component in constructing universal quantum circuits. The SWAP gate, which exchanges the states of two qubits, can be implemented using three CNOT gates and is represented by:

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Another important multi-qubit gate is the Toffoli (CCNOT) gate, a three-qubit gate that flips the target qubit only if both control qubits are in the $|1\rangle$ state. This gate is represented by an 8×8 matrix and is involutory.

Quantum gates also include phase shift gates, such as the T gate ($P(\pi/4)$), the S gate ($P(\pi/2)$), and the Pauli-Z gate ($P(\pi)$), which introduce specific phase shifts to qubit states. The Pauli-X (X), Pauli-Y (Y), and Pauli-Z (Z) gates, based on the Pauli matrices, perform rotations around the x, y, and z axes of the Bloch sphere, respectively. The Pauli-X gate, also known as the NOT gate, flips $|0\rangle$ to $|1\rangle$ and vice versa, while the Pauli-Z gate introduces a phase flip, leaving $|0\rangle$ unchanged and flipping the phase of $|1\rangle$. These gates are also involutory, meaning applying them twice returns the qubit to its original state. The identity gate:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

leaves the qubit unchanged and is used primarily for mathematical completeness. Controlled gates, such as the controlled-U gate, generalize the concept of the CNOT gate by applying an arbitrary unitary operation U (Unitary operator is a linear transformation acting on a Hilbert space that preserves the norm of quantum states, so leaving the state's probability amplitude intact. Mathematically:

$$U^\dagger U = UU^\dagger = I,$$

) to the target qubit if the control qubit is in the $|1\rangle$ state. These gates are essential for constructing complex quantum circuits and algorithms, as they enable conditional operations based on the state of other qubits.

2.2.6 Quantum measurement

Quantum measurements differ from classical measurements in several ways. In quantum mechanics, any measurement corresponds to a Hermitian operator, and the eigenstates of this operator can be used as a set of basis vectors in the quantum state space. When a measurement is made on a quantum state, the state of the system will randomly collapse to one of the eigenstates of the measurement operator, and the measurement result will be the corresponding eigenvalue. This probabilistic nature of quantum measurements is a key feature of quantum mechanics. Quantum systems also have angular momenta, which are quantized and can take only discrete values. Spin is a type of angular momentum, and nuclear spin is usually expressed by the symbol I. Considering nuclei with $I = 1/2$, they can have $(2I + 1) = 2$ discrete values along a specified axis. These two discrete values correspond to two states of a spin, i.e., parallel or anti-parallel along a specified axis. Without loss of generality, the two states with spin components parallel and antiparallel to the z-axis are denoted as $|0\rangle$ and $|1\rangle$. Any pure state of the qubit can be expressed as a vector in the state space spanned by $|0\rangle$ and $|1\rangle$. The matrix expressions of spin operators I_x, I_y, and I_z, which are the x, y, and z components of the spin angular momentum, are given as

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

with $I_i = (\hbar/2)\sigma_i$ for $i = x, y, z$. These are known as Pauli matrices. For the state $a|0\rangle + b|1\rangle$, measuring the z component of the spin operator, I_z, yields the eigenstate $|0\rangle$ with probability $|a|^2$ and measurement result $\hbar/2$, while the eigenstate $|1\rangle$ appears with probability $|b|^2$ and measurement result $-\hbar/2$. Both $|0\rangle$ and $|1\rangle$ are eigenstates of I_z, with corresponding eigenvalues of $\hbar/2$ and $-\hbar/2$, respectively. Measuring I_z multiple times on many copies of the state $a|0\rangle + b|1\rangle$ results in the expectation value

$$\langle I_z \rangle = |a|^2(\hbar/2) - |b|^2(\hbar/2) = \mathrm{Tr}(I_z\rho),$$

where ρ is the density matrix. This expectation value can be obtained using the quantum mechanical trace rule, which states that the expectation value of a measurement is the trace of the product of the density matrix and the measurement operator. In quantum computing, measurement is a fundamental operation that provides a way to extract information from a quantum system. When a quantum measurement is performed on a

qubit, it yields a classical result, collapsing the superposition states of the qubit into one definite state, either 0 or 1, based on the probability distribution of the qubit states. This collapse, known as the measurement problem in quantum mechanics, converts multiple probabilistic states into a single definite state, eliminating the existence of other states and superpositions.

2.2.7 Fidelity

The concept of distance in state space is useful for expressing the degree of similarity between quantum states, particularly when evaluating the closeness between the final state obtained after applying a quantum gate and the theoretically expected target state.

A commonly used metric for this purpose is quantum state fidelity, which quantifies the overlap between two quantum states. If the two states are identical, the overlap reaches its maximum value of 1, whereas if they are completely different, such as $|0\rangle$ and $|1\rangle$, the fidelity reaches its minimum value of 0. The mathematical definition of fidelity depends on the nature of the quantum states being compared. For two pure states $|\phi\rangle$ and $|\varphi\rangle$, the fidelity is given by:

$$F(|\phi\rangle, |\varphi\rangle) = |\langle\phi|\varphi\rangle|^2$$

When comparing a pure state $|\phi\rangle$ with a mixed state ρ, the fidelity is expressed as:

$$F(|\phi\rangle, \rho) = \langle\phi|\rho|\phi\rangle$$

The most general formulation, applicable to two mixed states ρ and σ, is given by:

$$F(\rho, \sigma) = \left(\mathrm{Tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2$$

which, in special cases, simplifies to the previous equations. This measure plays a crucial role in assessing the accuracy of quantum operations and the reliability of quantum information processing.

2.2.8 Lifetime

In an ideal scenario where a qubit is completely isolated from the environment, and spontaneous emission is not considered, its state is entirely determined by the initial state, Hamiltonian, and evolution time. If the Hamiltonian is the identity matrix, the state of the qubit will remain unchanged. However, in practice, qubits cannot be entirely isolated from the environment; thus, the state of a qubit will always change due to its interaction with external factors, leading to a finite quantum state lifetime. This change is typically a relaxation process determined by the environment. There are two common types of relaxation: transverse and longitudinal.

As an example, consider a single-qubit NMR system. Suppose the static magnetic field is along the z direction. In the thermal equilibrium state, the nuclear spins align along the direction of the magnetic field due to their interaction with it. The probability distribution of the different energy levels of the qubit follows the Boltzmann distribution.

In this case, the expectation value of the z component of the spin operator is nonzero, but the x and y components are zero, i.e., $\langle\sigma_z\rangle \neq 0$, $\langle\sigma_x\rangle = 0$, $\langle\sigma_y\rangle = 0$. If we apply pulses to the system and rotate the spin operator to the xy-plane, we have $\langle\sigma_z\rangle = 0$, $\langle\sigma_x\rangle \neq 0$, $\langle\sigma_y\rangle \neq 0$. In this situation, transverse relaxation will result in both $\langle\sigma_x\rangle$ and $\langle\sigma_y\rangle$ approaching 0, while longitudinal relaxation will cause a change in $\langle\sigma_z\rangle$ such that the system approaches the thermal equilibrium state determined by the temperature of the environment.

Both relaxation processes have characteristic times, denoted by T_2 and T_1, respectively. T_2 is the time it takes for $\langle\sigma_x\rangle$ and $\langle\sigma_y\rangle$ to decay to $1/e$ of their initial values, and T_1 is the time it takes for $\langle\sigma_z\rangle$ to recover to $(1-1/e)$ of its equilibrium value. Notably, T_1 relaxation involves energy exchange with the environment (spin-lattice relaxation), while T_2 relaxation is associated with loss of phase coherence among spins (spin-spin relaxation). Typically, T_1 is longer than or equal to T_2 in most systems. Transverse relaxation is also known as decoherence and is detrimental to quantum computing as it transforms a pure state into a mixed state, leading to information loss and computational errors. On the other hand, longitudinal relaxation is often utilized to initialize qubits by increasing their polarization to the maximum allowed by the environment, i.e., by maximizing $|\langle\sigma_z\rangle|$ to prepare for subsequent quantum computing tasks.

2.3 Today's main algorithms

In the early 20th century, mathematical logicians such as Hilbert, Turing, and Gödel formalized the concept of algorithms, laying the foundation for modern computing. With the advent of electronic computers, algorithm design became crucial. At its core, an algorithm is a step-by-step process for completing a task within a finite amount of time. A computing machine starts with an input and follows a well-defined sequence of steps, constrained by the rules and states permitted by its design. Mechanical and electronic computing rely on classical physics laws, which dictate the allowed states and transformations. However, quantum computing based on quantum mechanics has the potential to radically change how we process information, providing new algorithmic opportunities. Quantum computing takes advantage of quantum mechanical principles such as superposition and entanglement, offering a fundamentally different approach to computation. Quantum algorithms, having evolved from quantum physics models to contemporary computer science applications, now hold the promise of solving problems that classical computers struggle with. A significant step toward practical quantum computing is the development of industrial-scale quantum computers, which could transform various sectors, including cybersecurity. Recent advancements have been made in this area. For instance, Microsoft's Azure Quantum team has reported the creation of a quantum chip powered by a topological core architecture, utilizing Majorana quasiparticles. This development brings the prospect of utility-scale quantum computers closer to reality.

Daniel Simon introduced the first quantum method to outperform classical algorithms in terms of efficiency. This laid the groundwork for several foundational quantum algorithms, including the Deutsch-Josza, Bernstein-Vazirani and Shor's algorithms. Numerous quantum algorithms, including those for quantum machine learning, simulation and

search, leverage amplitude amplification, a technique that boosts the probability of correct results to solve complex challenges more effectively than traditional approaches. In addition, recent developments have introduced quantum approximate optimization algorithms aimed at solving graph theory problems, often using hybrid quantum-classical systems. These advancements highlight the diversity and breadth of quantum computing applications.

Fundamentally, quantum computing models fall into two main categories: the quantum gate model and quantum annealing. The quantum gate model operates by applying quantum gates to qubits, which manipulate them in ways similar to classical logic gates, but with the added advantage of exploiting quantum phenomena. This universal model supports a broad range of applications,however challenges like decoherence and the need for error correction are significant obstacles in implementing this model effectively. In contrast, quantum annealing uses an adiabatic approach to approximate solutions to optimization problems, relying on the natural tendency of quantum systems to find low-energy states. It is more resilient to certain computational errors compared to the gate model, as it takes advantage of the quantum system's inherent properties, such as tunneling, to arrive at solutions. Despite being less sensitive to errors, quantum annealing still requires maintaining coherence throughout the process to function effectively.

2.3.1 Main category

Quantum algorithms primarily fall into three categories that leverage unique quantum mechanical phenomena. The first category, rooted in the quantum Fourier transform (QFT). The QFT, analogous to the classical discrete Fourier transform (DFT) but exponentially faster on quantum hardware, operates by transforming quantum state superpositions into frequency-domain representations, enabling the detection of periodic patterns critical for solving problems like the hidden subgroup problem. While the classical fast Fourier transform (FFT) requires $O(N \log N)$ operations for N elements, the QFT accomplishes this in $O((\log N)^2)$ steps, a speedup that becomes transformative for large-scale problems. However, extracting meaningful results from the QFT's output remains challenging due to quantum measurement constraints. This limitation underscores a recurring theme in quantum computation: the interplay between algorithmic speedups and the inherent fragility of quantum information. The second category, quantum search algorithms, is exemplified by Grover's algorithm. By employing amplitude amplification to iteratively enhance the probability of locating a target state, Grover's algorithm reduces the classical $O(N)$ complexity to $O(\sqrt{N})$, offering significant advantages for database queries, optimization tasks, and machine learning applications. Though less dramatic than exponential speedups, this quadratic improvement becomes increasingly valuable as problem scales grow, particularly in domains like cryptography and combinatorial search where brute-force classical approaches are impractical. The third category, quantum simulation algorithms, exploits quantum systems' natural ability to model other quantum phenomena efficiently, a task that strains classical resources due to the exponential growth of state space with particle count. Where classical simulations require $O(c^n)$ bits to describe n-particle systems, quantum computers achieve this with $O(n)$ qubits, enabling accurate modeling of molecular interactions, material properties, and quantum chemical processes

that defy classical computation. This capability holds profound implications for drug discovery, materials science, and fundamental physics, though practical challenges persist in designing measurement protocols that extract actionable insights without collapsing delicate quantum states.

The computational power of these algorithms is contextualized through quantum complexity theory, which introduces classes like BQP (bounded-error quantum polynomial time) to describe problems efficiently solvable by quantum computers. While BQP is known to encompass classical polynomial-time problems (P) and reside within polynomial-space bounds ($PSPACE$), its relationship to NP, the class of classically hard verification problems, remains unresolved. Quantum algorithms like Shor's demonstrate that certain NP problems (e.g., factoring) can be solved efficiently, but whether this extends to all NP-complete problems remains unknown. This theoretical landscape highlights quantum computing's nuanced potential: while not a panacea for all computational challenges, it provides targeted exponential or polynomial speedups for problems with specific structural features, such as periodicity (exploited by QFT) or unstructured search spaces (addressed by amplitude amplification). The field's evolution is tightly coupled with hardware advancements, as error correction, qubit coherence times, and scalable architectures determine when theoretical speedups translate to practical advantages. Current research focuses not only on developing new algorithms but also on hybrid approaches that integrate quantum and classical processing, particularly in near-term noisy intermediate-scale quantum (NISQ) devices. As hardware matures, quantum computing's impact is anticipated to grow across cryptography, optimization, and scientific simulation, though fundamental questions persist about the ultimate limits of quantum advantage and the physical realizability of fault-tolerant systems.

2.3.2 Quantum Fourier Transform

The Quantum Fourier Transform is a quantum analogue of the classical discrete Fourier transform (DFT). For an orthonormal basis $|0\rangle, \ldots, |N-1\rangle$, the QFT acts as

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle,$$

transforming a quantum state $\sum_{j=0}^{N-1} x_j |j\rangle$ into $\sum_{k=0}^{N-1} y_k |k\rangle$, where y_k are the DFT coefficients of x_j. The QFT can be efficiently implemented on a quantum computer using $O(n^2)$ gates for n qubits, compared to the classical fast Fourier transform (FFT), which requires $O(n2^n)$ operations. However, the QFT does not directly speed up classical DFT computations because of the impossibility of direct measurement; instead, its power lies in enabling quantum algorithms like phase estimation, order-finding, and factoring.

Phase estimation is a key application of the QFT, allowing the estimation of the eigenvalue $e^{2\pi i\varphi}$ of a unitary operator U given an eigenstate $|u\rangle$. The algorithm uses two registers: the first stores t qubits initialized to $|0\rangle$, and the second stores $|u\rangle$. The procedure involves creating a superposition in the first register, applying controlled-U^{2^j} operations, and then performing an inverse QFT to estimate φ. The accuracy of the estimate depends on t, with $t = n + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits ensuring an n-bit approximation

with success probability at least $1-\epsilon$. Phase estimation is a subroutine in many quantum algorithms.

Order-finding involves determining the smallest positive integer r such that $x^r \equiv 1 \mod N$ for integers x and N. This problem is classically hard but can be solved efficiently on a quantum computer using phase estimation. The algorithm prepares a superposition of states $|j\rangle|x^j \mod N\rangle$, applies the QFT, and measures the first register to obtain an estimate of s/r, where s is an integer. The continued fractions algorithm is then used to extract r from this estimate. The success probability is high because, for randomly chosen s, s and r are likely coprime. The algorithm requires $O(L^3)$ operations for an L-bit number N, making it exponentially faster than the best-known classical algorithms. Factoring is reduced to order-finding by leveraging the relationship between the order of an element x modulo N and the prime factors of N. If x is chosen randomly and coprime to N, and its order r is even with $x^{r/2} \neq -1 \mod N$, then $\gcd(x^{r/2} \pm 1, N)$ yields non-trivial factors of N.

The QFT also underpins algorithms for solving the hidden subgroup problem (HSP), a generalization of period-finding and order-finding. In the HSP, a function f maps a group G to a finite set X and is constant on cosets of a hidden subgroup $K \subseteq G$. The goal is to find a generating set for K. For finite Abelian groups, the HSP can be solved efficiently using quantum algorithms that generalize the QFT. The algorithm involves creating a superposition over group elements, applying the function f, and performing a QFT to extract information about K. The success of the algorithm relies on the ability to decompose G into cyclic groups of prime power order, which can be done efficiently using quantum techniques.

Other applications of the QFT include period-finding and discrete logarithms. Period-finding involves determining the period r of a function $f(x+r) = f(x)$, while the discrete logarithm problem involves finding s such that $a^s = b$ for given a and b. Both problems can be reduced to instances of the HSP and solved using quantum algorithms that leverage the QFT. The discrete logarithm algorithm, for example, uses a quantum black box to compute $f(x_1, x_2) = b^{x_1}a^{x_2} \mod N$ and applies the QFT to estimate s. The hidden subgroup problem provides a unifying framework for many quantum algorithms, including Deutsch's problem, Simon's problem, and graph isomorphism.

Recent advancements have led to the development of hybrid quantum-classical algorithms that combine the flexibility of digital quantum computation with the robustness of analog quantum simulation. These digital-analog quantum algorithms offer potential improvements in implementing the QFT, especially under noisy intermediate-scale quantum (NISQ) conditions.

Deutsch-Jozsa algorithm Deutsch's algorithm addresses a fundamental problem in computational complexity by determining whether a binary function is constant or balanced with unparalleled efficiency. The problem, formalized as Deutsch's problem, challenges a computer to classify a function $f : \{0,1\}^n \to \{0,1\}$ as either constant (producing identical outputs for all inputs) or balanced (yielding 0 and 1 equally often) using minimal evaluations. For the simplest case of a single-bit input ($n = 1$), classical computation requires two function evaluations to compare $f(0)$ and $f(1)$, but Deutsch's

quantum algorithm achieves this with a single evaluation through the strategic use of superposition, quantum parallelism, and interference. The process begins by initializing two qubits in the state $|01\rangle$, which undergoes transformation via Hadamard gates into the superposition $\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$. Applying the function-specific unitary operation U_f entangles the qubits, yielding $(-1)^{f(x)}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$, where the phase factor $(-1)^{f(x)}$ encodes the function's output. A final Hadamard gate on the first qubit collapses the state to $\pm|f(0)\oplus f(1)\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$, allowing measurement of the first qubit to directly reveal $f(0)\oplus f(1)$. If the result is 0, f is constant; if 1, f is balanced. This single-step evaluation starkly contrasts with classical methods, demonstrating quantum advantage through the exploitation of superposition and interference.

The algorithm's significance extends beyond its immediate application through its generalization in the Deutsch-Jozsa algorithm, which scales the problem to n-bit inputs. Here, the initial state $|0\rangle^{\otimes n}|1\rangle$ is transformed via Hadamard gates into a uniform superposition $\sum_{x\in\{0,1\}^n}\frac{|x\rangle}{\sqrt{2^n}}\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$. After applying U_f, the state becomes $\sum_x\frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}}\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$, with phases reflecting f's behavior. A subsequent Hadamard transform on the query register produces interference patterns in the state $\sum_z\sum_x\frac{(-1)^{x\cdot z+f(x)}|z\rangle}{2^n}\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$, where $x\cdot z$ denotes the bitwise inner product modulo 2. Measurement collapses this to either all zeros (constant function) or a non-zero result (balanced function), resolving the problem in one quantum evaluation versus the classical worst-case requirement of $2^n/2+1$ evaluations. While the problem itself lacks direct practical relevance, the algorithm's theoretical implications are profound: it illustrates quantum computing's ability to extract global function properties through interference, a capability absent in classical systems.

Critically, the quantum advantage in these algorithms is nuanced. Classical probabilistic approaches can achieve high-confidence solutions with a small number of random evaluations, diminishing the perceived gap in practical settings. However, the deterministic certainty of quantum results-achieved without sampling-highlights a qualitative distinction. The core innovation lies in quantum parallelism, where a single U_f operation processes all possible inputs simultaneously, and interference effects distill the relevant global property. For example, in the single-qubit case, the Hadamard gates create superpositions that allow U_f to encode both $f(0)$ and $f(1)$ into a single state. The final Hadamard gate then induces constructive or destructive interference depending on f's parity, collapsing the system to a definitive answer. This mechanism, while elementary, underpins more advanced quantum protocols, showcasing how quantum systems can manipulate information in ways that transcend classical limitations. Despite its simplicity, Deutsch's algorithm remains a pedagogical cornerstone, crystallizing the principles of superposition, entanglement, and interference that define quantum computational advantage.

Bernstein-Vazirani algorithm The Bernstein-Vazirani algorithm is a quantum algorithm designed to determine an unknown binary sequence efficiently using parity checks. In classical algorithms, discovering an n-bit binary number requires n queries, whereas the Bernstein-Vazirani quantum algorithm accomplishes this task with just one query by leveraging quantum superposition. Unlike some quantum algorithms, this variant

does not require entanglement, yet it still demonstrates a significant quantum speedup. The core problem involves a hidden n-bit binary number $a \in \{0,1\}^n$, where each bit a_i represents part of the sequence. A query function $f(x)$ returns the modulo-2 sum of the bits where both a and the input binary number x are 1, expressed mathematically as $f(x) = a \cdot x \pmod 2 = \sum_{i=1}^{n} a_i x_i \pmod 2$. Classically, determining the secret string a requires n function evaluations with carefully chosen inputs x to isolate each bit of a. In contrast, the quantum approach encodes the query function into a quantum state by preparing a uniform superposition of all possible inputs. The algorithm initializes an n-qubit system in the state $|0\rangle^{\otimes n}$ and applies the Hadamard transformation $H^{\otimes n}$, producing an equal superposition of all possible computational basis states: $|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$. Applying the oracle function U_f, which imparts a phase based on $f(x)$, transforms the state into: $\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{a\cdot x} |x\rangle$. Since the Hadamard transform is its own inverse, applying $H^{\otimes n}$ again yields: $H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{a\cdot x} |x\rangle \right) = |a\rangle$, allowing for the direct measurement of a in a single query. The unitary transformation U_f can be decomposed into the direct product of n single-qubit gates, where $U_i = I$ if $a_i = 0$ and $U_i = \sigma_z$ if $a_i = 1$, with σ_z being the Pauli-Z matrix. This decomposition facilitates efficient implementation of U_f without entanglement, relying solely on quantum superposition to achieve the speedup over classical methods. The Hadamard transformation itself is also a product of single-qubit gates, enhancing the algorithm's scalability.

Beyond its fundamental implications for quantum speedup, the Bernstein-Vazirani algorithm has practical applications in cryptography and computational optimization. Although it does not offer an exponential advantage like Shor's algorithm, its efficiency highlights the power of quantum mechanics in computational tasks, particularly those involving structured data retrieval. The absence of entanglement in this variant makes it an excellent candidate for near-term quantum devices, where limited qubit connectivity and noise remain challenges. Recent developments have further expanded the algorithm's applications. For instance, MicroAlgo Inc. has designed a quantum register based on the Bernstein-Vazirani algorithm, enabling efficient storage and processing of qubits, and implementing FULL adder functionality through quantum gate operations [34].

Simon's algorithm

Simon's algorithm is a quantum protocol that solves the problem of identifying a hidden bitstring defining a periodic function with exponential speedup over classical methods. The problem involves a function $f : \{0,1\}^n \to \{0,1\}^n$, promised to satisfy $f(x) = f(y)$ if and only if $y = x \oplus s$, where $s \in \{0,1\}^n$ is a secret string and $\oplus$ denotes bitwise XOR. Classically, determining s requires up to $O(2^{n/2})$ queries, but Simon's quantum algorithm achieves this in $O(n)$ queries by exploiting superposition and interference. The algorithm initializes two n-qubit registers to $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$, applies Hadamard gates to the first register, creating the state $\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|0\rangle$, and evaluates $f(x)$ via an oracle U_f, yielding $\frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle$. Measuring the second register collapses the first register to a superposition of x and $x \oplus s$: $\frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle)$. Applying Hadamard gates to the first register produces interference patterns dependent on s, resulting in a state

$\frac{1}{\sqrt{2^{n-1}}} \sum_{y \cdot s=0} (-1)^{x \cdot y} |y\rangle$, where $y \cdot s = 0$ denotes the bitwise inner product modulo 2. Repeated measurements yield linearly independent equations constraining s, which is then determined via Gaussian elimination. The algorithm's quantum advantage arises from its ability to extract global periodicity through interference, bypassing the classical necessity of exponentially many queries.

Simon's algorithm has been employed to analyze the security of symmetric cryptographic primitives, such as Feistel networks, revealing vulnerabilities in certain configurations. The algorithm also inspired Shor's factoring algorithm, which adapts its period-finding framework to solve integer factorization. Unlike the Bernstein-Vazirani algorithm, which avoids entanglement, Simon's protocol inherently leverages entanglement between registers during the oracle evaluation phase. This distinction underscores the versatility of quantum resources in solving diverse computational problems, even as both algorithms exemplify the power of structured quantum interference.

Shor's algorithm

Shor's algorithm [31] is a landmark quantum protocol that achieves an exponential speedup for integer factorization by casting the problem into period finding. Given a composite integer N, one picks a random base a coprime to N and considers the function $f(x) = a^x \bmod N$, which is guaranteed to be periodic with some unknown period r. Classically, determining r requires super-polynomial time, but Shor's quantum algorithm finds r in polynomial time $O((\log N)^2 \log\log N \log\log\log N)$ by harnessing superposition, entanglement, and the quantum Fourier transform (QFT). The procedure initializes two registers-one of $m \approx 2\lceil \log N \rceil$ qubits in $|0\rangle^{\otimes m}$ and a second of $\lceil \log N \rceil$ qubits in $|0\rangle^{\otimes \lceil \log N \rceil}$, applies Hadamard gates to the first to create $\frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle|0\rangle$, and computes $f(x)$ via a modular-exponentiation oracle U_f, yielding $\frac{1}{\sqrt{2^m}} \sum_x |x\rangle|a^x \bmod N\rangle$. Measuring the second register collapses the state of the first to a superposition of all x consistent with a single function value, i.e. $x \equiv x_0 \pmod r$, and applying the QFT to the first register produces interference peaks at integer multiples of $2^m/r$. A final measurement returns a value y from which one extracts a candidate fraction $y/2^m \approx k/r$; classical continued-fractions analysis then recovers r. After a few repetitions to ensure independence and co-prime checks, one obtains nontrivial factors of N via $\gcd(a^{r/2} \pm 1, N)$. Shor's algorithm thus transforms the intractable classical task of factoring into a sequence of efficiently solvable quantum subroutines, modular exponentiation, QFT, and continued-fractions extraction, demonstrating how structured quantum interference and entanglement can collapse exponential search spaces into polynomial-time solutions.

2.3.3 Quantum search - Grover's algorithm

Grover's algorithm, introduced by Lov Grover in 1996, addresses the fundamental challenge of unstructured search in computer science, where a specific item must be located within an unsorted database of N elements. Classical algorithms require $O(N)$ operations in the worst case, as each entry must be checked sequentially, but Grover's quantum algorithm achieves this with $O(\sqrt{N})$ operations, a quadratic speedup that becomes profoundly significant for large datasets. This efficiency stems from quantum superposition

and amplitude amplification, which enable parallel evaluation of multiple states. The algorithm begins by initializing an n-qubit system (where $n = \log_2 N$) into an equal superposition of all possible states via the Hadamard transform, ensuring each index is represented with equal probability. The core of the algorithm lies in the iterative application of the Grover iteration, a quantum subroutine comprising two critical operations: the oracle and the diffusion operator. The oracle, a unitary operator, marks the target state by inverting its phase, mathematically expressed as $|x\rangle \to (-1)^{F(x)}|x\rangle$, where $F(x) = 1$ if x is a solution. This phase shift does not alter probabilities directly but prepares the system for interference. The diffusion operator then amplifies the marked state's amplitude through an inversion about the mean, represented as $R_2 = 2|\varphi\rangle\langle\varphi| - I$, where $|\varphi\rangle$ is the uniform superposition state. Geometrically, this process rotates the quantum state vector toward the solution subspace, with each iteration increasing the probability of measuring the target. After approximately $O(\sqrt{N/M})$ iterations (where M is the number of solutions), the probability of collapsing to a solution state nears certainty. This optimality is intrinsic: no quantum algorithm can solve the unstructured search problem with fewer than $\Omega(\sqrt{N})$ oracle calls, as demonstrated by bounding the deviation caused by the oracle and analyzing the polynomial relationships between classical and quantum query complexities. While the quadratic speedup does not resolve NP-complete problems in polynomial time, it enables significant efficiency gains; For example, reducing the complexity of brute-force searches for problems like Hamiltonian cycles from $O(2^{n \log n})$ classically to $O(2^{n \log n/2})$ quantumly. Beyond basic search, Grover's framework extends to applications like quantum counting, which estimates the number of solutions M using phase estimation techniques. Experimental validations of the algorithm highlight its practicality, including a recent milestone achieving 98.9% accuracy on a four-qubit silicon processor without error correction, as detailed in a 2025 Nature Nanotechnology study [37]. As a benchmark for quantum hardware, Grover's algorithm underscores the transformative potential of quantum mechanics in computation, balancing theoretical limits with tangible advancements in solving classically intractable problems.

2.4 Quantum Computing's Impact on Cybersecurity

The emergence of quantum computing represents a paradigm shift with profound implications for cybersecurity, challenging the foundations of classical cryptographic systems rooted in computational intractability. At the heart of this disruption lies Shor's algorithm, which exploits quantum parallelism and the quantum Fourier transform to solve integer factorization and discrete logarithm problems in polynomial time, a capability that threatens public-key cryptosystems like RSA and ECC. Since its discovery in 1994, the theoretical potential of Shor's algorithm has driven both algorithmic refinements and sobering realizations about practical implementation challenges. Early estimates projected staggering resource demands, with breaking RSA-2048 initially requiring $\sim$1 billion physical qubits and months of runtime due to rudimentary error-correction models and limited gate fidelities. These projections have evolved dramatically through decades of research, with modern estimates stabilizing at $\sim$15 million physical qubits and 8-hour runtimes for the same task, a hundredfold improvement enabled by surface code error

correction, optimized modular arithmetic, and parallelized circuit designs [11] [18].

The trajectory of Grover's algorithm follows a distinct but complementary arc. Offering a quadratic speedup for unstructured search problems, it reduces the effective security of symmetric cryptographic standards like AES-128 to 64 bits, necessitating doubled key lengths for quantum resistance. Initially, hardware platforms faced prohibitive noise and scalability challenges (e.g., requiring $\sim 10^6$ qubits for useful database tasks). However, recent six-qubit Grover experiments on trapped-ion systems, with randomized compiling and error-detection mitigation, have attained single- and two-qubit fidelities around 99%, demonstrating significant algorithm-level improvements [2] [5]. Yet the algorithm's logarithmic scaling of qubits relative to search space size underscores a persistent challenge: maintaining coherence and gate precision across thousands of qubits remains beyond current noisy, intermediate-scale quantum (NISQ) devices, relegating practical threats to symmetric cryptography to the medium-term horizon.

Resource estimation trends reveal a field maturing from theoretical abstraction to engineering pragmatism. Initial projections in the 1990s, such as factoring 2048-bit RSA moduli with 10^4 logical qubits and 10^7 physical qubits, reflected idealized assumptions about quantum parallelism and nascent error-correction schemes. The formalization of surface codes in the 2010s marked a turning point, enabling fault-tolerant computation with reduced physical-to-logical qubit ratios. Subsequent breakthroughs in windowed arithmetic, coset representations, and lattice surgery techniques further halved logical qubit requirements, while tools like Microsoft's Azure Quantum Resource Estimator introduced granular analysis of gate counts and routing overhead. Recent resource estimations for factoring 2048-bit RSA have fallen to the range of $\sim$4,000-20,000 logical qubits and $\sim$20 million physical qubits, thanks to co-design approaches that integrate surface-code error correction, modular-arithmetic circuit reductions, and hardware compilation [7].

Three interconnected drivers underpin this progress: algorithmic optimizations, error-correction advancements, and realistic hardware modeling. Modular exponentiation subroutines, once naively implemented, now leverage oblivious carry runways and Karatsuba multiplication to reduce circuit depths by 30-50%. Concurrently, surface code thresholds achieving 99.99% gate fidelity and explorations of qLDPC codes have redefined fault-tolerance benchmarks, while superconducting and trapped-ion platforms deliver qubits with $< 0.1\%$ error rates and second-long coherence times. These innovations feed into iterative refinements of the "quantum resource stack", where each layer, from logical qubit design to photonic interconnects, contributes to shrinking the space-time volume of computations.

The cybersecurity implications are urgent yet paradoxical. While current quantum processors remain far from factoring even 128-bit integers, the downward trend in Shor's resource estimates has expedited PQC standardization. Algorithms like CRYSTALS-Kyber (standardized as FIPS 203 in August 2024) and Falcon are prompting widespread adoption of hybrid encryption systems. Additionally, twin-field QKD has achieved secure transmission over 500-1000 km, although achievable key rates are in the order of bits per second, not megabits-restrictions largely due to photon loss and infrastructure demands, making QKD viable only in high-security and specialized deployments.

Hardware diversification further shapes the landscape. Superconducting qubits lead in

scalability, exemplified by IBM's 1,121-qubit Condor processor (2023), while trapped ions excel in error resilience with 99.9% gate fidelities. Photonic and neutral-atom platforms hint at future distributed quantum networks but face developmental hurdles. Across all modalities, the million-qubit threshold for cryptographically relevant computations looms as both a technical and economic challenge, necessitating breakthroughs in qubit multiplexing, 3D integration, and classical control systems capable of exascale error decoding.

As the field transitions from laboratory experiments to utility-scale engineering, the interplay between algorithmic innovation and hardware progress continues to redefine timelines. Projections suggest fault-tolerant quantum computers capable of breaking RSA-2048 may emerge by the late 2030s, driven by qubit factories and photonic interconnects. Meanwhile, quantum machine learning and hybrid variational algorithms demonstrate near-term potential for tasks like intrusion detection, albeit constrained by NISQ-era noise. This duality, between existential risks and incremental opportunities, underscores the need for cryptographic agility, sustained investment in PQC migration, and international collaboration on quantum security standards. The evolution of quantum computing from theoretical curiosity to applied technology demands a proactive reimagining of global cybersecurity infrastructures, ensuring resilience against a future where Shor's and Grover's algorithms transition from academic marvels to operational tools.

Chapter 3

Quantum threats and Mitigations

3.1 Today's performances

The quantum computing landscape has witnessed remarkable advancements across multiple technological platforms, each pushing the boundaries of computational capability through distinct approaches. Leading this charge, Google Quantum AI achieved a pivotal milestone with its Willow processor, demonstrating below-threshold error correction and quantum supremacy by solving a specialized problem in minutes-a task theoretically requiring millennia on classical supercomputers [4]. This dual focus on raw power and reliability underscores the industry's pursuit of fault-tolerant systems. IBM Quantum complements this progress through sheer scale, breaking the 1,000-qubit barrier with its Condor processor while introducing the Heron generation, which triples performance via tunable couplers and crosstalk elimination. Their Quantum System Two architecture envisions modular, scalable quantum-centric supercomputing, marrying increased qubit counts with enhanced connectivity [9]. Quantinuum's trapped-ion H2-1 system exemplifies precision, executing 4,600 two-qubit gates across 56 qubits without noise-induced failure, while its planned 100-logical-qubit system by 2027 signals aggressive strides toward fault tolerance. Microsoft's topological Majorana 1 chip introduces hardware-level error resistance via Majorana particles, aiming for million-qubit chips within years rather than decades, a potential paradigm shift in scalability [1]. Diverse qubit architectures each address unique challenges. Superconducting qubits, exemplified by IBM and Google, leverage semiconductor-like fabrication for rapid gate operations (99.9% single-qubit fidelity in China's Zuchongzhi-3 [10]) but face coherence limitations (72 microseconds) and cryogenic demands. Trapped ions, championed by Quantinuum and IonQ, excel in coherence (seconds) and all-to-all connectivity, though slower gates complicate scaling. Photonic systems, advanced by PsiQuantum and Xanadu, thrive at room temperature with minimal decoherence, yet grapple with photon control precision. Neutral atoms, as seen in Microsoft-Atom Computing collaborations, balance scalability (hundreds of qubits in 2D arrays) and fidelity (99.6% two-qubit gates) [12], while topological and diamond qubits explore error-resistant and portable designs.

This architectural diversity reflects strategic trade-offs: superconducting qubits prioritize industrial scalability, trapped ions favor algorithmic depth, photonics target quantum

networking, and neutral atoms blend modularity with precision. Algorithmic innovations amplify hardware progress. Shor's algorithm epitomizes quantum threat to cryptography, with IBM factoring 48-bit numbers via superconducting qubits, a precursor to breaching RSA-2048 (In collaborazione con un gruppo di ricerca cinese, IBM ha fattorizzato numeri piú piccoli), theoretically feasible with 4,000 error-corrected logical qubits [11]. Grover's quadratic speedup, demonstrated at 98.9% accuracy on silicon processors, pressures symmetric cryptography, effectively halving AES-256's security. Hybrid algorithms like VQE and QAOA bridge NISQ-era limitations, enabling practical applications in chemistry and optimization. Error correction breakthroughs, from Google's surface codes [3] to Microsoft's logical qubit entanglement, are critical enablers: Willow's distance-5 code reduced logical errors, while Quantinuu's H2 maintained coherence across deep circuits. These software strides, coupled with metrics like quantum volume (exponentially outpacing classical scaling) and coherence gain (0.7 Log10 robustness), signal accelerating maturity. Performance benchmarks reveal both promise and hurdles. While trapped ions lead in gate fidelity (99.9% single-qubit, 99.65% two-qubit) and coherence, superconducting platforms dominate qubit counts (IBM's 1,121 vs. Quantinuum's 56). Photonic advances, like Quandela's 100,000-fold component reduction for fault tolerance, hint at disruptive scalability. Neutral atoms, though nascent, showcase error-corrected logical qubits and 99.9% single-gate fidelity. Yet universal fault tolerance remains elusive: error rates necessitate millions of physical qubits per logical unit, and photonic/phononic interconnects are vital for modular scaling. Cybersecurity implications loom large, while current hardware cracks only trivial RSA keys, projections suggest 2048-bit vulnerabilities within decades, urging cryptographic migration. Coherence times, though improving to seconds in ions, still lag behind algorithmic demands, emphasizing the need for dynamic decoupling and materials innovation. The field's trajectory hinges on converging architectures, algorithms, and error mitigation. As superconducting systems scale, trapped ions refine deep-circuit reliability, photonics enable quantum networks, and topological qubits reimagine fault tolerance. Algorithm-architecture co-design, such as tailoring Shor's to ion-trap coherence or optimizing QAOA for superconducting speed, will drive near-term utility. With quantum volume doubling annually and error rates plummeting, the transition from laboratory curiosities to industry-grade tools accelerates, reshaping fields from drug discovery to cryptography. Yet the true inflection point awaits error-corrected, modular systems blending qubit types, where photonic links merge superconducting speed with ion-trap precision. This symbiotic evolution, rather than any singular technology, will unlock quantum computing's transformative potential, heralding an era where problems once deemed intractable succumb to coordinated quantum-classical orchestration.

3.2 Prediction for quantum hardware evolution

Quantum decoherence remains the most persistent barrier, as qubits, highly sensitive to thermal fluctuations, electromagnetic interference, and vibrational noise, lose their quantum states within microseconds to milliseconds, curtailing computation depth and reliability.

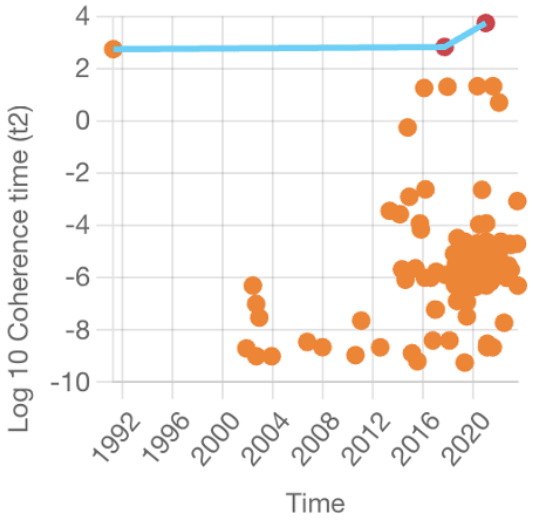

Figure 3.1. Evolution in coherence time in quantum hardware since 1992

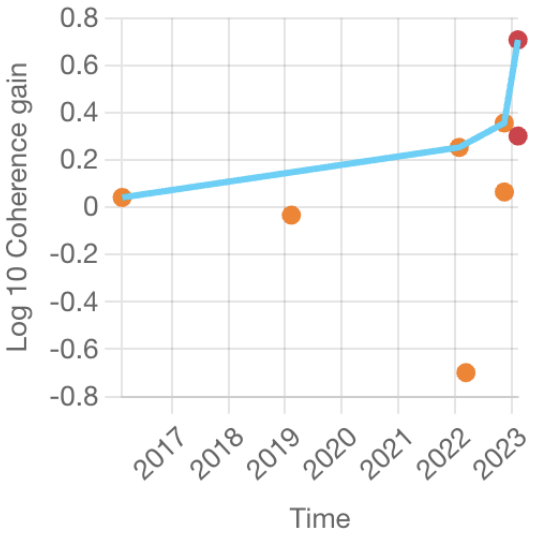

Figure 3.2. Evolution in coherence gain in quantum hardware since 2017

This fragility demands extreme environmental controls: superconducting qubits operate near absolute zero in multi million dollar dilution refrigerators, while trapped ions require ultra-high vacuums and laser precision, creating resource-intensive systems largely

confined to research facilities.

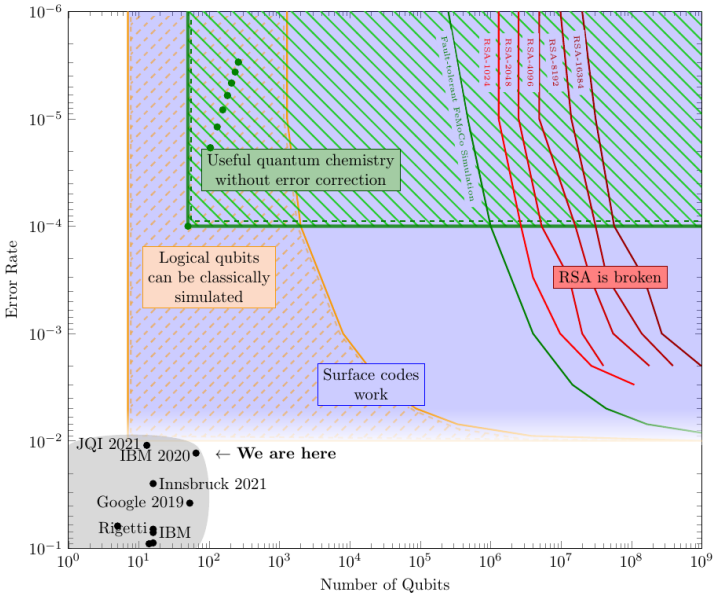

Figure 3.3. Projection of resources requirement for various cryptographic application

Scalability compounds these issues, with increasing qubit counts exacerbating crosstalk and control complexity. Current architectures struggle to balance growth with performance, IBM's Condor processor reached 1,121 superconducting qubits, yet error rates ($\sim$ 1% per gate) and limited connectivity hinder practical utility. Error correction introduces further overhead, requiring hundreds or thousands of physical qubits per logical qubit to achieve fault tolerance, a threshold even industry leaders like Google and Quantinuum have only preliminarily demonstrated with small-scale implementations. Despite these hurdles, incremental progress suggests a path forward. Materials innovation aims to extend coherence times, with diamond-based qubits showing promise for room-temperature operation, potentially reducing reliance on cryogenics. IBM's roadmap emphasizes hardware-software co-development, introducing metrics like gate operations to track control fidelity improvements alongside qubit count. Early commercial applications hint at niche viability: IBM's quantum-enhanced fraud detection reduced false negatives by 5%, while QC Ware and Goldman Sachs achieved quantum advantage in Monte Carlo simulations for financial modeling. Such proofs-of-concept, though laboratory-bound, validate hybrid quantum-classical approaches that leverage current noisy intermediate-scale quantum (NISQ) devices. Industry timelines remain contentious, IBM projects mainstream adoption within 6-10 years, but McKinsey surveys reveal skepticism, with 28% of experts anticipating fault-tolerant systems only post-2040. Gartner offers a middle ground, predicting that 25% of enterprises will gain competitive edges through quantum-inspired algorithms by 2027, even without full-scale quantum supremacy. Standardization

and talent gaps complicate this trajectory. The absence of unified programming frameworks and middleware forces organizations to hedge bets across competing platforms like Qiskit, Cirq, and PennyLane, inflating development costs. Meanwhile, cryogenic infrastructure demands pose environmental and economic challenges as systems scale-each dilution refrigerator consumes ~25 kW, rivaling data center energy footprints. Yet strategic investments persist: Crédit Agricole's quantum credit-scoring model matched classical accuracy with 96% fewer classifiers, demonstrating efficiency gains that could justify early adoption.

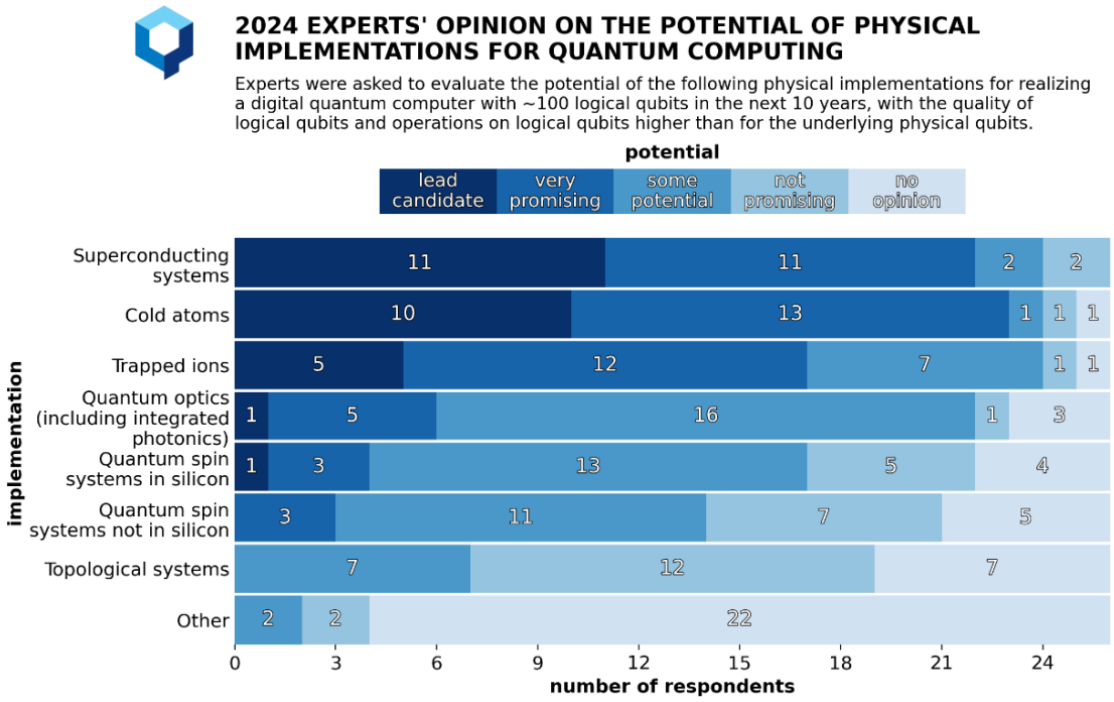

Figure 3.4. Potential of various physical implementation of quantum hardware

BCG posits 2025 as an inflection point for enterprise value generation, contingent on sustained hardware roadmaps and algorithmic refinements. Ultimately, quantum hardware evolution hinges on parallel advances-suppressing decoherence through topological qubits, optimizing error correction overhead, and developing modular architectures that distribute qubits across photonic networks or cryogenic modules. While universal fault tolerance remains distant, the convergence of these incremental breakthroughs suggests a gradual rather than revolutionary adoption curve, where quantum accelerators enhance specific workflows long before displacing classical infrastructure.

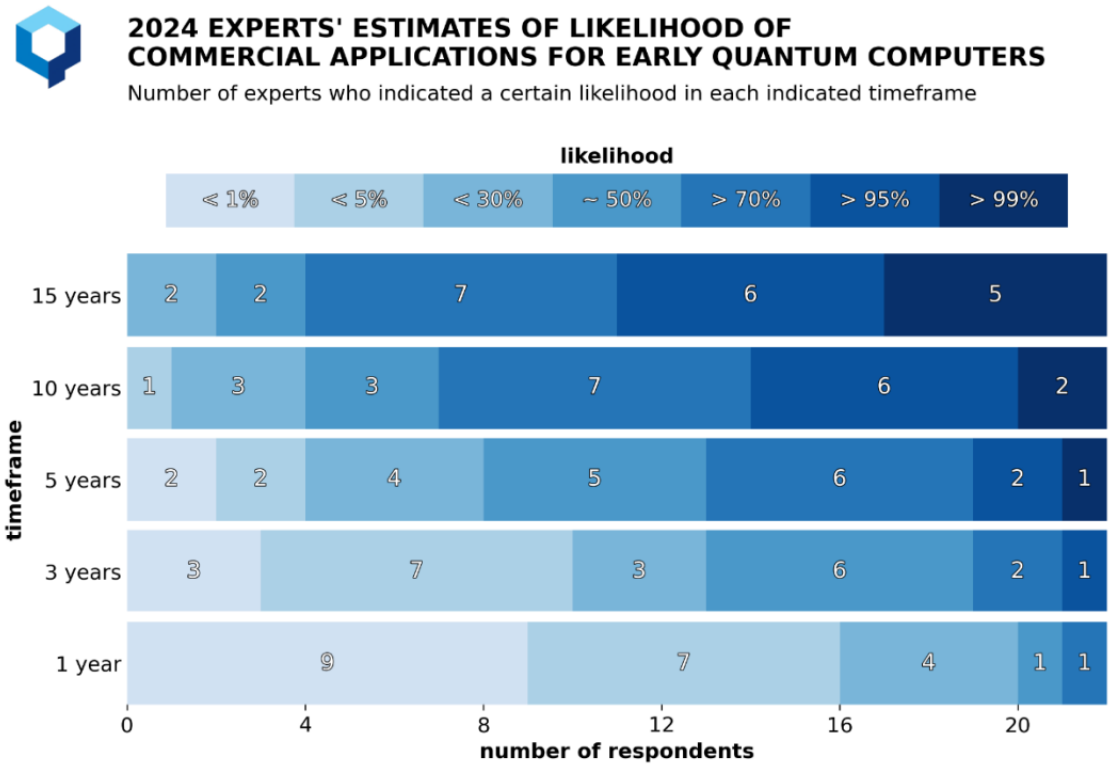

Figure 3.5. Expert's estimates of likelihood of commercial applications for quantum computers

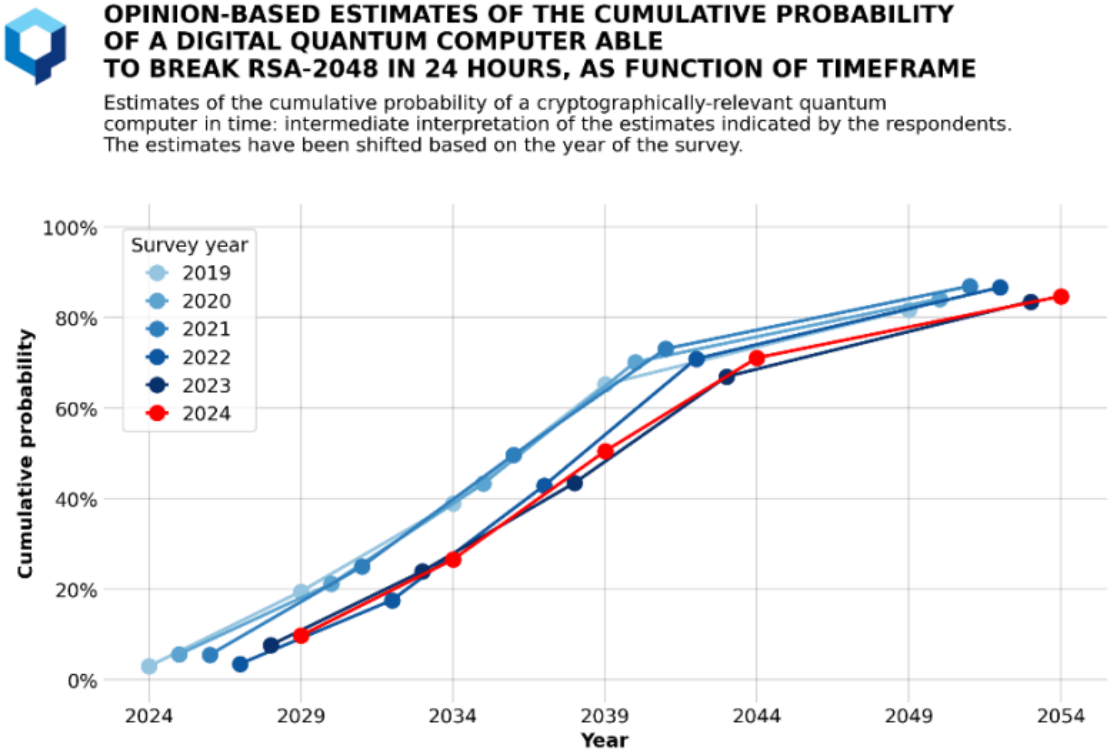

Figure 3.6. Prediction's evolution about quantum computer's probability to break RSA-2048 in 24 hours

3.3 Threats Brought by Quantum Computers

The emergence of quantum computing presents a profound and potentially existential threat to the cybersecurity landscape, fundamentally challenging the cryptographic underpinnings currently secure digital communications, data storage, and online transactions. While offering unprecedented computational power for solving complex problems, this capability directly targets the mathematical difficulties upon which contemporary cryptographic algorithms rely. The foundation of modern cybersecurity is built upon systems like the RSA algorithm, which depends on the intractability of factoring large prime numbers, and Elliptic Curve Cryptography (ECC), based on the algebraic structure of elliptic curves over finite fields. These methods are considered secure against classical computers because the computational effort required to break them is prohibitively vast. However, quantum computers, leveraging principles such as superposition and entanglement, can perform calculations at speeds far exceeding those of classical machines. This advancement poses a direct vulnerability, primarily through quantum algorithms like Shor's algorithm that render RSA and ECC effectively obsolete in a quantum-enabled future, and Grover's algorithm which, although less devastating than Shor's, it can effectively halve the key length of symmetric cryptographic systems, significantly reducing

their security margin. The threat extends beyond the immediate breaking of established encryption methods. A critical concern is the "Harvest Now, Decrypt Later" (HNDL) attack, where malicious actors are already collecting and storing large volumes of currently encrypted data. They anticipate that once powerful quantum computers become available in the future, they will be able to decrypt this sensitive information en masse. This strategy involves capturing encrypted network traffic or data at rest, often along with associated public keys and digital signatures, and waiting for the technological maturity of quantum systems. Upon the advent of sufficiently powerful quantum computers, these attackers could deploy algorithms like Shor's or Grover's to break the underlying cryptographic primitives and unlock the stored data. This poses an immense risk, particularly for sensitive information that retains its value over time, such as government secrets, intellectual property, financial records, and healthcare data. The implications are profound, highlighting that data encrypted today under classical assumptions could be exposed by future quantum capabilities, underscoring the urgency for developing and adopting quantum-resistant cryptographic methods. Beyond HNDL, quantum computing facilitates several other immediate cyber threats. Secure channel decryption becomes possible as encrypted network communications can be broken, allowing attackers to intercept and "listen in" on sensitive conversations in real-time. Signature impersonation is another risk, where quantum power enables attackers to forge signed digital certificates, leading to consequences like widespread malware distribution and sophisticated targeted phishing campaigns. Furthermore, the potential for yet unknown quantum algorithms to break existing cryptographic systems and the inherent difficulties associated with the global transition to quantum-resistant cryptography introduce new "Zero-Day" vulnerabilities.

Integrating quantum computing into cyber-attack strategies could result in more sophisticated malware and threats, exploiting vulnerabilities at unprecedented speeds. Specific areas of cybersecurity are particularly vulnerable.

Public-key cryptography, fundamental to secure online communication, digital signatures, and key exchange protocols like Diffie-Hellman, is directly threatened by Shor's algorithm, jeopardizing confidentiality and integrity. Symmetric-key algorithms like AES, while not directly broken by Shor's, face a reduced security margin due to Grover's algorithm, necessitating larger key sizes to maintain current security levels. Digital signatures, vital for verifying the authenticity and integrity of digital documents and transactions, could be forged by quantum computers targeting underlying algorithms like RSA and DSA, enabling impersonation. Blockchain technologies and cryptocurrencies, which rely heavily on cryptographic hash functions and public-key cryptography for security and wallet protection, face threats from quantum capabilities that could potentially find hash collisions or derive private keys from public ones, risking transaction manipulation, double-spending, and unauthorized access to digital assets.

Network authentication protocols used in VPNs, SSL/TLS, and Wi-Fi security (like WPA2) rely on public-key cryptography and could be compromised, leading to unauthorized access and man-in-the-middle attacks. The security of SSL/TLS certificates and the entire Public Key Infrastructure (PKI) system could also be undermined, enabling the forging of certificates.

The security of Internet of Things (IoT) devices, often utilizing lightweight cryptographic protocols due to resource constraints, is highly susceptible to quantum attacks, potentially compromising entire ecosystems. Even advanced privacy-preserving technologies like Homomorphic Encryption, which allows computations on encrypted data without decryption, and Zero-Knowledge Proofs (ZKPs), used for verifying information without revealing the data itself, could see their underlying cryptographic assumptions challenged by quantum attacks. Homomorphic encryption enables secure data processing in sensitive fields, while ZKPs, used in blockchain for privacy and scalability (e.g., zk-SNARKs in Zcash), maintain privacy in scenarios requiring trust without direct information sharing. Potential vulnerabilities, even in quantum key distribution (QKD) systems designed to enhance security, could be exploited, raising concerns about the overall reliability of quantum communication methods.

In recognition of these significant threats, organizations like the National Institute of Standards and Technology (NIST) are actively evaluating and standardizing post-quantum cryptographic algorithms specifically designed to resist quantum attacks. The transition to quantum-resistant cryptography is not merely a technical undertaking but a complex logistical and strategic imperative. It requires a comprehensive audit and update of current infrastructures, the global adoption of new protocols, and significant investment in research, development, and implementation to ensure a seamless shift to a post-quantum secure world and mitigate the pervasive risks posed by the advent of powerful quantum computers.

Crypto Type	Algorithms	Variants	Key Length (bits)	Classic Strength (bits)	Quantum Strength (bits)	Vulnerabilities	L	I	R	Recommended QC-Resistant Solutions
Asymetric	ECC [52]	ECC 256	256	128	0	Broken by Shor's Algorithm [19].	M	H	H	Algorithms presented in Table II.
		ECC 384	384	256	0		M	H	H	
		ECC 521	521	256	0		M	H	H	
	FFDHE [53]	DHE2048	2048	112	0	Broken by Shor's Algorithm [19].	M	H	H	
		DHE3072	3072	128	0		M	H	H	
	RSA [54]	RSA 1024	1024	80	0	Broken by Shor's Algorithm [19].	M	H	H	
		RSA 2048	2048	112	0		M	H	H	
		RSA 3072	3072	128	0		M	H	H	
Symmetric	AES [55]	AES 128	128	128	64	Weakened by Grover's Algorithm [56].	M	M	M	Larger key sizes are needed.
		AES 192	192	192	96		M	M	M	
		AES 256	256	256	128		M	L	L	
	SHA2 [57]	SHA 256	-	128	85	Weakened by Brassard et al.'s Algorithm [58].	M	M	M	Larger hash values are needed.
		SHA 384	-	192	128		M	L	L	
		SHA 512	-	256	170		M	L	L	
	SHA3 [57]	SHA3 256	-	128	85	Weakened by Brassard et al.'s Algorithm [58].	M	M	M	
		SHA3 384	-	192	128		M	L	L	
		SHA3 512	-	256	170		M	L	L	

Figure 3.7. Likelihood, Impact and Risk analysis of standard algorithm in a quantum-enabled future

3.4 Solutions to Quantum Threats

Addressing the existential risks posed by quantum computing to global cybersecurity requires a multi-layered defense strategy combining cryptographic innovation, infrastructure modernization, and proactive policy frameworks. At the core of this paradigm shift lies the development and deployment of post-quantum cryptography (PQC), mathematical systems designed to withstand both classical and quantum attacks through problems resistant to Shor's and Grover's algorithms. Transitioning to PQC demands more than algorithm substitution; it necessitates full-stack cryptographic agility, the capability to

dynamically update encryption protocols across hardware security modules, network protocols, and legacy systems without service disruption. This agility is being operationalized through hybrid systems that combine classical and quantum-resistant algorithms during the transition period, as demonstrated in Google's New Hope integration with TLS 1.3 [6], [39], which maintained backward compatibility while testing lattice-based key exchange. Complementing algorithmic upgrades, Quantum Key Distribution (QKD) introduces physics-based security to symmetric key exchange by encoding cryptographic keys in quantum states of photons. Through protocols like BB84 and E91 [19], QKD exploits the no-cloning theorem and quantum entanglement to detect eavesdropping attempts via perturbations in quantum bit error rates, any interception attempt inherently introduces detectable anomalies. Commercial deployments like China's 4,600 km Beijing-Shanghai quantum backbone [17] and the European Quantum Communication Infrastructure (EuroQCI) initiative demonstrate terrestrial QKD's viability for metropolitan-scale networks, though technical constraints persist. Fiber-based QKD faces distance limitations ($\sim$ 100 km without quantum repeaters) [40] and integration challenges with existing DWDM optical networks, while satellite-based systems like Micius achieve intercontinental ranges but suffer from limited transmission windows and atmospheric interference. To overcome these limitations, researchers are developing trusted-node networks and quantum repeaters using memory-based entanglement swapping, with recent breakthroughs achieving 50 km entanglement distribution through multiplexed quantum memories. Interim defense measures include cryptographic hygiene practices such as elongating symmetric key lengths, doubling AES from 256-bit to 512-bit effectively reduce the effects of Grover's quadratic speedup, and implementing crypto-periodic key rotation policies to limit exposure windows for harvested data. However, these stopgap solutions merely buy time; organizations must concurrently audit cryptographic inventories using tools like the NSA's Commercial Solutions for Classified (CSfC) program checklist, identifying vulnerable systems from PKI certificate authorities to IoT device authentication protocols. Critical infrastructure sectors are adopting migration frameworks like the NIST Cybersecurity Framework's Quantum Readiness profile, which prioritizes securing long-lived systems (e.g., power grid SCADA, financial transaction archives) through crypto-agile design patterns and quantum-safe pseudorandom number generators. Strategic policy initiatives amplify technical countermeasures. The U.S. National Security Memorandum-10 mandates federal agencies to achieve quantum-resistant cryptography by 2030, while the EU's Cyber Resilience Act imposes PQC compliance requirements on connected device manufacturers. Cross-industry consortia like the Quantum Safe Security Working Group are developing interoperability standards for PQC implementation across cloud platforms (AWS Hybrid Post-Quantum TLS, Azure Quantum-Safe), blockchain networks (Quantum Resistant Ledger), and 5G/6G infrastructures. Education and workforce development form another pillar, with academic programs integrating PQC into curricula and vendor certification schemes like Utimaco's Quantum-Safe HSM Engineer ensuring operational readiness.

3.4.1 PQC

The advent of quantum computing, with its potential to perform complex calculations exponentially faster than classical computers by harnessing phenomena like superposition and entanglement, necessitates a fundamental shift in cryptographic practices. The threat became particularly clear with mathematician Peter Shor's demonstration in the 1994s that a theoretical quantum computer could effortlessly break widely used public key encryption (PKE) algorithms, prompting a global investigation into new cryptographic systems. Post-quantum cryptography (PQC), also known as quantum-resistant encryption, refers to cryptographic algorithms specifically designed to be secure against attacks by quantum computers while typically remaining secure against classical computers as well [16] [20]. Unlike traditional methods susceptible to quantum algorithms like Shor's for factoring and discrete logarithms, and Grover's for database searching, PQC algorithms are based on mathematical problems believed to be hard for both classical and quantum computers. This field focuses on securing classical computer systems against the advanced computational capabilities of quantum computing. One of the most important steps to mitigate the quantum threat is thus to transition to these quantum-resistant cryptographic algorithms. This involves using encryption methods and digital signatures that remain secure even in the presence of powerful quantum computers. Organizations should plan for a seamless migration to post-quantum cryptographic standards as they are developed and standardized. Current research into PQC is pursuing various approaches based on such seemingly intractable mathematical problems. Among the leading candidates are lattice-based cryptography, hash-based cryptography, code-based cryptography and multivariate polynomial cryptography. These cryptographic systems offer a promising path towards maintaining confidentiality and integrity in the quantum era. Lattice-based cryptography (LBC) [30] [25] is currently one of the most promising and prominent areas of research in PQC, distinguished by its reliance on complex computational problems in mathematical lattices. A lattice is a discrete subgroup of an n-dimensional real vector space, meaning it's a set of points where you can add any two points and get another lattice point, and there's a minimum distance between any two distinct points, preventing them from being arbitrarily close. This structure resembles an infinite grid, becoming exceedingly complex to navigate in high dimensions (e.g., 400). The security of lattice-based schemes is based on the assumed difficulty of solving certain computational problems in these lattices, such as the Shortest Vector Problem (SVP), finding a non-zero lattice vector closest to the origin or the Learning With Errors (LWE) problem. While SVP is relatively easy in low dimensions, no efficient classical or quantum algorithms are known to solve it efficiently in higher dimensions for general lattices. Modern LBC schemes are often based on problems like LWE, where one is given a "noisy" linear system ($b = As + e \pmod{q}$) and must find the secret vector s, with e being a "short" error vector. The LWE problem can be shown to be asymptotically at least as hard as a variant of SVP given suitable parameters. The work of Ajtai is of fundamental theoretical importance in this area, proving "worst-case to average-case" reductions for certain lattice problems. Early lattice-based schemes included the relatively inefficient Ajtai-Dwork scheme. In search of practical solutions, schemes like NTRU (based on problems in structured lattices) and a cryptosystem inspired by Ajtai-Dwork (Goldreich,

Goldwasser, Halevi) were introduced. While the standard NTRU problem's hardness isn't definitively proven to be equivalent to worst-case lattice problems, NTRU-based methods have not been broken when parameters are chosen suitably. A major milestone was Regev's introduction of the LWE problem in 2005, forming the basis for many of today's lattice-based encryption and key agreement schemes. Variants like ring LWE and module LWE were developed to increase efficiency and reduce key sizes, based on the assumption that lattice problems remain hard even with additional algebraic structure. Structured lattices, while offering efficiency gains, also carry the risk of potentially providing further attack vectors, an important research question. Lattice-based cryptosystems have gained significant attention, notably since Google's experimental testing of "New Hope" in the Chrome browser, and they constitute a large part of the finalists in the current NIST standardization process for core cryptographic primitives like encryption, key encapsulation, and digital signature schemes, with examples including CRYSTALS-KYBER for encryption and CRYSTALS-Dilithium for digital signatures. LBC is a leading solution for post-quantum encryption and offers strong security guarantees while enabling functionalities like fully homomorphic encryption (FHE). Hash-based cryptography is a generic term for cryptographic constructions primarily focused on digital signatures whose security is based on the security properties of cryptographic hash functions [33]. A hash function is a compression function that maps input data of any length to a fixed-length output, known as a digest (typically 256-512 bits). Cryptographic hash functions possess security properties like one-way function (preimage resistance, hard to find input for a given output), weak collision resistance (second preimage resistance, hard to find a second input with the same output as a given input), and strong collision resistance (hard to find any two different inputs with the same output). Common constructions include Merkle-Damgård (SHA-2) and Sponge (SHA-3). Hash-based signature schemes are constructed using hash trees (Merkle trees) and one-time signature (OTS) schemes. OTS schemes, like Lamport-Diffie or Winternitz OTS, allow a private key pair to be used only once to generate a signature; using it multiple times compromises security and allows forgery. While highly unpractical for frequent signing in a multilateral environment due to the need for a large set of key pairs, Ralph Merkle solved this in 1979 with the invention of the hash tree. In a Merkle tree, public keys (or hashes of data blocks at the leaves) are compressed in pairs using a hash function recursively until a single value, the root, is reached. This root serves as the single "master" public key for all the individual private keys at the leaves. To sign a message, the corresponding private key (or keys for a binary encoded message) and the path in the tree needed to recompute the root from the leaf are revealed, allowing verification. Merkle signatures have very well-understood security properties and, in their current forms (LMS, XMSS), are considered mature quantum-safe signature schemes. A key drawback is their statefulness: the signer must meticulously track which one-time signature keys have been used, as any error leads to a loss of security and imposes high requirements on implementation and usage. Additionally, the number of possible signatures is limited, requiring a trade-off between signature size and the number of signatures during key generation. Despite this, stateful hash-based signatures like LMS and XMSS are particularly suitable for future-proof software update concepts where statefulness can

be managed and the maximum number of signatures estimated. They have been standardized by the IETF (RFC8554 for LMS, RFC8391 for XMSS) and adopted by NIST (SP 800-208), finding their way into formats like Cryptographic Message Syntax (CMS) and Concise Binary Object Representation (COSE) used in S/MIME and IoT. As a stateless variant, SPHINCS (based on Goldreich's design) has been developed. While it eliminates the need for state tracking, this statelessness results in efficiency disadvantages, such as larger signature sizes, compared to LMS and XMSS, requiring assessment in concrete application scenarios. Recent schemes like XMSS, LMS, SPHINCS, and BPQS offer improved performance, with most being stateful except SPHINCS. NIST's decision to standardize three digital signature algorithms for quantum-safe signatures includes the hash-based SPHINCS+ alongside CRYSTALS-Dilithium and FALCON, based on the well-established understanding and quantum safety of hash-based signatures. Code-based cryptography centers on developing cryptographic systems using error-correcting codes, which make it possible to detect and correct errors in stored or transmitted data and have a history going back to pioneers like Hamming and Shannon [38]. These codes are used widely in various communication and storage technologies. The general decoding problem, decoding a received message based on a random or unstructured code, is known to be NP-hard, forming the basis for the security of code-based schemes. The most prominent representative is the McEliece cryptosystem [32], an asymmetric encryption scheme introduced in 1978 by Robert McEliece. Its security relies on two assumptions: that the binary Goppa codes it uses are indistinguishable from random linear codes, and that random linear codes can only be decoded with exponential effort due to the General Decoding Problem, for both classical and quantum computers. After more than 40 years of research, and despite analyses considering modern computing power (like the 2008 attack on originally proposed parameters), no structural weakness has been found when binary Goppa codes are used, making McEliece one of the oldest unbroken quantum-safe proposals. A major disadvantage is its large public key size, which remains in the megabyte range for high-security applications, even with variants like the one described by Harald Niederreiter in 1986. However, code-based key agreement schemes offer very small ciphertexts (around 200 bytes), and encryption/decryption is much more efficient than RSA or ECC-based asymmetric schemes. Recent proposals using more structured codes aimed to reduce public key size, but some have faced successful attacks. Traditional code-based signature schemes have exhibited significant efficiency problems and are mostly of theoretical interest, with more efficient approaches based on coding theory still in early stages. Code-based cryptography aims to create algorithms that are rapid, secure, and efficient against quantum threats. Other important family is Multivariate Polynomial Cryptography, which involves systems of multivariate polynomial equations over finite fields whose solution is assumed to be hard, offering a different approach to secure communications. Given the complexity and ongoing development of PQC algorithms, a strategic approach involves hybrid encryption, combining traditional encryption methods with post-quantum algorithms. This layered approach provides enhanced security by leveraging the strengths of both systems. For example, a system might use RSA for immediate encryption needs while simultaneously employing a post-quantum algorithm to secure the same data for the long term. By combining algorithms, organizations can reduce the risk of a complete

security breach; if one algorithm is compromised, the other can still protect sensitive data. This allows organizations to implement post-quantum algorithms alongside existing systems, enabling a gradual transition rather than a complete overhaul. This approach also offers flexibility in adapting to various security needs and regulatory requirements, especially in industries with stringent compliance standards. Standardized PQC algorithms aim to be compatible with existing communication protocols and networks, facilitating this transition. Companies are investing in PQC to offer quantum-resistant solutions and safeguard systems from potential quantum computer-based attacks.

Chapter 4

Execution's time analyses of PQ and standard algorithms

In this comprehensive and methodically structured experimental investigation, the temporal performance characteristics associated with cryptographic key-pair generation for both well-established classical cryptographic schemes and a selection of emerging post-quantum algorithms were rigorously quantified, analyzed, and compared. The investigation was conducted across two distinct and technically diverse execution environments: a production-grade, network-connected hardware security module (HSM) and its corresponding locally hosted software-based emulator. The primary objective was to establish a robust and trustworthy baseline for key-generation performance using traditional algorithms, derive precise comparative metrics that highlight the efficiency differences between hardware-accelerated and purely software-based cryptographic operations, and ultimately project realistic and data-driven estimates of execution times for post-quantum primitives when implemented on physical HSM hardware, assuming future native support.

The principal hardware platform used throughout the study was the Thales Luna Network HSM 7, equipped with firmware version 7.9.0, securely interfaced via TCP/IP within the company's dedicated network to naturally incorporate authentic network round-trip latency, internal security policies, and hardware-accelerated cryptographic functionality. Complementing this setup, the Utimaco HSM Simulator was deployed on a virtualized Ubuntu 22.04 LTS environment provisioned with eight virtual CPUs and 4 GB of RAM; this simulator provides full compatibility with the PKCS#11 API specification implemented by the Thales Luna HSM but lacks hardware acceleration and network latency, offering a consistent and controllable software-based cryptographic testing environment ideal for development, debugging, and algorithmic performance assessment.

In each environment, namely the Thales Luna HSM and the Utimaco software simulator, an automated and reproducible Python-based testing framework invoked the standard 'C_GenerateKeyPair' interface defined by the PKCS#11 cryptographic token interface standard. A total of 100 discrete and independent key-generation operations were performed for each selected classical algorithm, including RSA with 2048- and 4096-bit key lengths and ECDSA over the NIST P-256, P-384, P-521, secp256k1, and Ed25519 curves, with high-resolution timing data captured via Python's 'time.time()' API, which

provides nanosecond-level precision and low overhead.

Immediately after each key-pair generation, the generated key material was securely destroyed and, where necessary, cryptographic sessions were explicitly closed and re-initialized to ensure consistent test conditions and avoid memory accumulation or execution-state interference between iterations. The raw timing data were systematically logged and exported, then processed using Python with the matplotlib and NumPy libraries to compute comprehensive statistical summaries, specifically, central tendency metrics including the arithmetic mean (μ) and the median (m), alongside measures of dispersion such as the standard deviation (σ), to provide a detailed view of both average performance and variability across runs.

To enable a direct and quantifiable comparison between hardware and software environments, a dimensionless acceleration ratio for each classical algorithm was calculated as $R = \mu_{HSM}/\mu_{Sim}$, where μ_{HSM} represents the average key-generation time on the Thales Luna HSM and μ_{Sim} represents the corresponding average time on the Utimaco simulator. This ratio quantifies the combined effects of hardware acceleration, cryptographic processing efficiency, and network-induced delays in the real HSM context relative to the locally hosted software environment, with higher values of R indicating a more significant time overhead when moving from simulator to hardware, a factor essential for extrapolating simulator-based measurements to real-world HSM usage scenarios.

Building upon this foundation of classical algorithm benchmarking and cross-platform comparison, the investigation was extended to include four representative post-quantum cryptographic algorithms reflecting different theoretical and structural paradigms: the eXtended Merkle Signature Scheme (XMSS) and the Leighton-Micali Signature Scheme (LMS), both stateful or stateless hash-based digital signature algorithms, and the Module-Lattice-based Digital Signature Algorithm (MLDSA) and the Module-Lattice-based Key-Encapsulation Mechanism (MLKEM), representing advanced lattice-based or hybrid cryptographic constructions. These post-quantum algorithms were evaluated exclusively within the Utimaco simulator, as the Thales Luna HSM does not yet natively support them; however, due to strict time constraints and the exceptionally high computational demands associated with some configurations of XMSS and LMS-requiring several hours to complete a single key-pair generation, only a single iteration per algorithm was executed to capture representative timing values. Although this adjustment limits the statistical robustness of the post-quantum performance data, it still offers meaningful insight into the order of magnitude of key-generation durations, with the standard high-resolution timing and logging methodology applied to ensure that even individual measurements remained as accurate and comparable as possible within the project timeline.

With the raw performance data for XMSS, LMS, MLDSA, and MLKEM obtained, simulated mean execution times for each post-quantum algorithm were scaled using the previously computed R_{RSA} and R_{ECDSA} ratios to produce a bounded range of estimated execution times reflecting the likely overhead imposed by a transition from a local emulator to a real hardware platform, accounting for potential hardware acceleration discrepancies as well as systemic delays related to secure network communication, concurrency handling, and operational integrity safeguards inherent in HSM architectures.

4.1 Standard algorithms

4.1.1 RSA

In FIPS 186-5 [21], the security parameter L specifies the bit length of the RSA modulus $n = p * q$, where p and q are large prime numbers, with standardized options of 2048, 3072, and 4096 bits. These values correspond approximately to NIST-defined security strengths of 112, 128, and 152 bits, respectively. This guidance aims to help implementers choose key sizes that maintain a practical balance between cryptographic robustness and system performance, especially in long-term deployments where future-proofing against evolving computational capabilities, including quantum threats, is a growing concern.

The RSA key-generation process begins by invoking a NIST-approved random bit generator to produce two large integers, each approximately $\lfloor L/2 \rfloor$ and $\lceil L/2 \rceil$ bits in length. These candidate integers are subjected to trial division against small primes, followed by multiple rounds of probabilistic primality testing, typically the Miller-Rabin test with sufficient random bases to ensure a negligible probability of error. This sequence is repeated until two strong primes p and q are identified, ideally differing significantly in magnitude to thwart certain factoring optimizations.

After generating p and q, the public exponent e is selected, often fixed to 65537 for its low Hamming weight and strong security history. This value must be verified to be co-prime with $\phi(n) = (p-1)*(q-1)$, the Euler totient of the modulus. Once verified, the private exponent $d = e^{-1} mod \phi(n)$ is computed using the extended Euclidean algorithm, ensuring correct decryption and signature capabilities.

The most computationally intensive part of RSA KeyGen lies in the repeated primality testing of large random integers, a process whose expected runtime increases superlinearly with L. This is primarily due to the decreasing density of prime numbers among larger integers and the higher cost of modular arithmetic with big integers.

Empirical key-generation benchmarks executed on the simulator, and so not running on specific accelerated hardware shows and execution time of several hundreds of milliseconds while the same algorithm when run on dedicated hardware show a $4x$, for RSA2048, and $6x$, for RSA4096, speed-up. Considering negligible the delay introduced by the network due to its small value compared with execution time.

Therefore, FIPS 186-5's specification of key lengths through L encapsulates a classic and fundamental trade-off between security and efficiency. Larger moduli dramatically increase the work factor for integer factorization attacks, thus strengthening RSA's resistance to both conventional cryptanalysis and hypothetical quantum threats posed by algorithms, even though in a quantum-enabled future this algorithm could be potentially broken by sufficiently capable quantum computers. However, this increased security comes at the price of slower key generation, greater computational load during encryption, decryption, signing, and verification operations, and increased memory and bandwidth usage.

	RSA2048	RSA4096
n (Bytes)	2048	4096
Signature (Bytes)	256	512
Time Simulator (ms)	664.362	7538.685
Time HSM (ms)	166.002	1118.986

Table 4.1. RSA parameters and execution time

4.1.2 ECDSA

In FIPS 186-5 [21], the Elliptic Curve Digital Signature Algorithm (ECDSA) derives its cryptographic strength from the selection of standardized elliptic curves defined over prime fields. The primary curves recommended by NIST for federal use are secp256r1 (also known as P-256), secp384r1 (P-384), and secp521r1 (P-521). These curves provide classical security levels of approximately 128, 192, and 256 bits respectively, and their selection effectively determines the security parameter for ECDSA operations.

The ECDSA key pair generation process starts by randomly selecting a private scalar d from the interval $[1, n-1]$, where n denotes the order of the curve's base point G. This randomness must be derived using a NIST-approved deterministic or true random bit generator to ensure sufficient entropy and compliance with federal standards. The associated public key Q is then calculated as $Q = d * G$, a scalar multiplication operation that forms the cornerstone of elliptic curve cryptography. Scalar multiplication is typically implemented using algorithms such as double-and-add, windowed non-adjacent form (wNAF), or fixed-window techniques, which optimize the number of required elliptic curve point operations. The time complexity of this operation scales roughly logarithmically with the size of n (i.e., $O(logn)$), primarily in terms of finite-field multiplications and additions.

During the signing phase, ECDSA relies on the generation of a fresh per-message ephemeral scalar k, which must also be sampled securely and uniquely for each message to prevent private key leakage. From this, a new curve point $R = k * G$ is derived, and the signature components (r, s) are computed through a series of finite-field operations, including modular inversion and multiplication within the underlying field $GF(p)$. This design introduces sensitivity to both randomness quality and arithmetic performance in constrained environments.

Empirical performance measurements that, as expected, the time increase with the order of the curve n. The times taken from the simulator show an execution time extremely short of few milliseconds. The same goes for the time required by the HSM even though in this case, working with time of the same order of magnitude as the network's delay, we see worst performances across all the curves due to dealy introduced by the TLS protocol as all the data that travel to and from the HSM have to be encrypted.

Compared to traditional public-key schemes like RSA, ECDSA provides a significant reduction in key sizes and computational burden for equivalent security levels, especially in bandwidth-sensitive and embedded environments. Its efficiency, coupled with strong security guarantees under widely studied mathematical assumptions, positions it as a

favored choice for digital signatures in modern cryptographic systems.

	P-256	P-384	P-521	secp256k1
Public Key (Bytes)	65	97	133	65
Private Key (Bytes)	32	48	66	32
Signature (Bytes)	64	96	132	64
Time Simulator (ms)	3.734	8.792	15.814	3.575
Time HSM (ms)	10.984	11.819	13.338	14.474

Table 4.2. ECDSA parameters and execution time

4.1.3 EdDSA

In NIST SP 800-186 [15], the Edwards-curve Digital Signature Algorithm (EdDSA) using curve 25519 with SHA-512, commonly referred to as Ed25519, defines its security parameter through the underlying field size and hash output length of 256 bits, yielding approximately 128 bits of classical collision resistance and 128-bit preimage resistance. Key pair generation involves sampling a 32-byte random seed via a NIST-approved random bit generator, which is then expanded with SHA-512 to derive a "clamped" scalar d by masking specific bits to enforce subgroup order and cofactor requirements. The public key A is obtained by computing the scalar multiplication $A = d * B$, where B is the standard base point on the Montgomery form of curve25519. This single scalar multiplication dominates the KeyGen cost and scales as $O(logl)$ finite-field operations, where $l = 2^{252} + 27742317777372353535851937790883648493$ is the group order.

Signature generation in Ed25519 is entirely deterministic: the message is first hashed with the secret-expanded seed to produce a per-message scalar r, and a second scalar multiplication $R = r * B$ is computed. The signature pair (R, S) is formed by computing $S = (r + H(R||A||M) * d)modl$, requiring only one additional scalar multiplication and a small number of modular additions. Verification similarly requires two scalar multiplications to check the equality $[S]B = R + H(R||A||M) * A$.

The experiment shows that, also in this case, working with time so short penalizes the HSM due to the delay introduced by the TLS protocol and by the network. We are working, indeed, with time in the order of few milliseconds and burden of the network represent a significant part of the total measured time.

	Ed25519
Public Key (Bytes)	32
Private Key (Bytes)	32
Signature (Bytes)	64
Time Simulator (ms)	5.371
Time HSM (ms)	13.698

Table 4.3. EdDSA paramenters and execution time

4.2 Post Quantum Algorithms

4.2.1 MLDSA

In FIPS 204 [22] the parameter $k \in \{2, 3, 5\}$ is the principal selector for one of three MLDSA security categories: Category 2 offering approximately 128-bit classical security and a conservative level of post-quantum resistance, Category 3 approximately 192-bit strength and Category 5 of about 256-bit strength. The parameter directly determines the variant names "MLDSA-44", "MLDSA-65" or "MLDSA-87" respectively. This selection not only denotes a target security level in line with NIST's post-quantum standardization goals, but also dictates the underlying module dimension parameters k, which expand from (4,4) for $k = 2$, through (6,5) for $k = 3$, to (8,7) for $k = 5$, thereby influencing the size and complexity of all subsequent arithmetic operations.

During the Key Generation algorithm, the implementation must generate a full $k * l$ matrix of secret polynomials and corresponding error distributions by invoking an extendable output function (XOF) to ensure sufficiently uniform randomness, followed by multiple Number-Theoretic Transform (NTT) and inverse-NTT operations to translate between coefficient and evaluation representations. As the module dimensions grow, the total number of coefficients subject to NTT processing increases from 16 in the smallest configuration to 30 and 56 in the larger ones, leading to an approximate $7x$ rise in transform counts from $k = 2$ to $k = 3$, and a $10x$ increase when moving to $k = 5$, with each transform costing approximately $O(NlogN)$ arithmetic steps over the ring modulus.

Empirical hardware benchmarks runned on the simulator demonstrate that this translated complexity results in a key-generation time slightly increased. Since the increasing in time from one algorithm to the other is in the order of hundreds of microseconds, we can assume that the majority of the time is introduced by our setup that though represent an actual work environment. Nevertheless we see that increasing the security parameter, increase the execution time of 400 microseconds from $k = 2$ to $k = 3$ and of 700 microseconds to $k = 5$.

Consequently, the choice of a higher k value in FIPS 204 produces a quantifiable trade-off: while elevating the cryptographic margin against both classical and quantum-threat models, it also incurs a significant, and roughly linear, penalty in key-generation latency that must be balanced against application performance requirements.

	ML-DSA-44	ML-DSA-65	ML-DSA-87
k	2	3	5
Public Key (Bytes)	1312	1952	2592
Private Key (Bytes)	2560	4032	4896
Signature (Bytes)	2420	3309	4627
Time (ms)	2.068	2.483	2.743

Table 4.4. ML-DSA parameters and execution time

4.2.2 MLKEM

In FIPS 203 [23] the security parameter $k \in \{2, 3, 4\}$ serves as the fundamental designator for the three standardized MLKEM variants: MLKEM-512 ($k = 2$), MLKEM-768 ($k = 3$) and MLKEM-1024 ($k = 4$), each corresponding to increasing levels of post-quantum security roughly equivalent to AES-128, AES-192, and AES-256 strength, respectively, while simultaneously fixing the key-encapsulation mechanism's internal module dimensions. This parameter is not merely symbolic, it directly influences the mathematical structure and computational demands of the cryptographic scheme, particularly in the key generation phase, which is a critical component in any secure communication protocol.

Specifically, the algorithm's KeyGen routine must sample a $k * k$ matrix $\hat{A}$ of polynomials from the ring $\mathbb{Z}_q[x]/(x^n+1)$, where $n = 256$ and $q = 3329$ is a small, carefully chosen prime modulus. Alongside this matrix, the algorithm generates secret vectors and error vectors, each of length n, adhering to noise distributions that preserve security guarantees. These polynomials are instantiated using a NIST-approved extendable-output function (XOF), such as SHAKE128 or SHAKE256, ensuring cryptographic randomness and determinism as required. Once the sampling is completed, the implementation must perform multiple forward and inverse Number-Theoretic Transforms (NTTs) to switch between the coefficient domain and the evaluation domain, facilitating efficient polynomial arithmetic operations essential to lattice-based cryptography.

Because each NTT of length n has computational complexity $O(n \log n)$, the overall workload imposed by these transforms increases substantially as k increases. More precisely, the number of transforms involved scales with k^2 for the matrix $\hat{A}$ and linearly with k for the associated vectors, resulting in a rapid escalation of computational cost. For example, when $k = 2$, as in MLKEM-512, the transform count is relatively modest, 4 forward and 1 inverse NTT, totaling 5. When $k = 3$ (MLKEM-768), the number of transforms increases to 7, and at $k = 5$ (MLKEM-1024), the count rises further to approximately 9 or more, depending on implementation optimizations and the structure of the arithmetic circuit. These counts imply increasing workload multipliers of approximately $1x$, $\approx 1x$, and $1.3x$ respectively, suggesting non-linear but manageable growth in complexity.

Practical benchmarks executed on the simulator reinforce these theoretical insights. For work environment the overhead introduced by the network and by all the layers that sits in between the actual HSM (or the Simulator in this case) and the final client make the difference in execution time almost negligible in fact in my testing the time difference between ML-KEM-512 and ML-KEM-768 is almost completely obscured by all the overhead. The computational difference is of greater impact in embedded system environment.

Hence, in FIPS 203, the choice of k encapsulates a classical cryptographic tradeoff between higher security and lower efficiency. While increasing k significantly boosts resilience against both classical and quantum attacks, particularly due to the enlarged module lattice dimension and increased entropy of the public and secret components, it also introduces latency during the KeyGen process. This latency, although acceptable in many contexts, becomes a critical factor in constrained environments such as embedded systems or high-frequency transaction settings. Implementers must therefore carefully

balance their choice of k based on the specific performance and security requirements of their target deployment scenario.

	ML-KEM-512	ML-KEM-768	ML-KEM-1024
k	2	3	5
Encapsulation Key (Bytes)	800	1184	1568
Decapsulation Key (Bytes)	1632	2400	3168
Cyphertext (Bytes)	768	1088	1568
Shared Secret Key (Bytes)	32	32	32
Time (ms)	1.881	1.887	2.163

Table 4.5. ML-KEM parameters and execution time

4.2.3 XMSS

In FIPS 205 [24], the security of the XMSS (eXtended Merkle Signature Scheme) algorithm is governed by a variety of parameters, among which the tree height h and the Winternitz parameter w are central in determining the trade-off between signature size, signing and verification time, and overall security margin. XMSS is a stateful hash-based signature scheme that relies on the Merkle tree construction, where each leaf represents a one-time signature key, and the internal nodes are computed using a cryptographic hash function in a hierarchical binary tree structure. The overall security level is primarily determined by the length of the output of the underlying hash function n, typically set to 256 or 512 bits, corresponding roughly to 128-bit and 256-bit security respectively, along with structural parameters that control how the tree is traversed and how many signatures are possible.

The height of the Merkle tree, denoted by h, determines the total number of one-time signatures (OTS) that can be issued from a single key pair, since the total number of leaves equals 2^h. For example, with $h = 10$, the scheme supports 1024 signatures; for $h = 16$, up to 65,536 signatures are possible and for $h = 20$ we reach 1,048,576 signature available. As h increases, the total number of usable signatures increases exponentially, but so does the size of the authentication path in each signature, which must include one hash value per level of the tree. This results in longer signatures and higher computational overhead for both signing and verifying, especially since XMSS must manage a secure state to avoid reusing OTS keys, which would compromise security.

Another critical parameter is the Winternitz parameter w, which is a trade-off lever between signature size and computational speed. A smaller w leads to shorter signatures but slower signing and verification operations, as it requires more hash function calls to encode the same amount of data. Conversely, a larger w reduces the number of hash operations needed, speeding up computation but producing larger signatures. Common choices include $w \in \{4, 16, 256\}$, with $w = 16$ offering a balanced compromise between signature size and performance. The selection of w also influences the structure of the Winternitz One-Time Signature (WOTS+) scheme used at each leaf of the Merkle tree,

affecting how efficiently messages are encoded and how many hash chains must be traversed. For the Utimaco implementation w is not a usable paramenter but its value is set to 16.

The KeyGen procedure in XMSS involves generating a large number of WOTS+ key pairs (one per tree leaf), calculating the corresponding public keys, and building the full Merkle tree by hashing upward from the leaves to compute the root, which serves as the public key. This process requires 2^h evaluations of the hash function just to compute the leaves, and an additional $2^h - 1$ evaluations for the internal nodes. As h increases, this cost becomes substantial, especially in constrained devices or systems where key generation needs to be repeated often. However, in most applications, key generation is performed infrequently and amortized over a large number of signatures, making it acceptable in many deployment scenarios.

Benchmark data taken form the simulator's execution shows that, for a typical setting of $n = 256$, $h = 16$ and $w = 16$, XMSS key generation can take several hundred milliseconds to several seconds on standard CPUs depending on optimization, while signature generation and verification each take tens of milliseconds per operation. With smaller h values, performance improves significantly, though at the cost of allowing fewer total signatures. These performance figures, however, remain within acceptable limits for many applications that require robust long-term security and can tolerate statefulness.

In conclusion, the parameters h and w in FIPS 205 are pivotal in balancing the efficiency, security, and practicality of XMSS deployments. Larger tree heights enable greater signature capacity and stronger security under forward-secrecy models, but incur greater computational cost and larger signatures. Meanwhile, the Winternitz parameter tunes the trade-off between signature compactness and the speed of cryptographic operations. Proper selection of these parameters must reflect the needs of the deployment context-whether that is high-throughput transaction processing, resource-constrained embedded devices, or long-term document signing, while always ensuring that the key state is securely managed to maintain XMSS's post-quantum security guarantees.

		n		
		192	256	512
h	10	1.73 s	3.37 s	14.72 s
	16	118.92 s	171.22 s	1040.65 s
	20	1891.26 s	2891.25 s	15591.25 s

Table 4.6. XMSS-SHA256 execution time

4.2.4 LMS

In FIPS 205 [24], the LMS (Leighton-Micali Signature) algorithm is presented alongside XMSS as a stateful hash-based signature scheme designed to provide strong post-quantum security guarantees. LMS operates by combining the Leighton-Micali One-Time Signature (LM-OTS) scheme with a Merkle tree of a specified height h, forming a hierarchical structure that enables multiple signature generations from a single public key. The key

		n	
		256	512
h	10	4.65 s	19.41 s
	16	341.25 s	1241.26 s
	20	5170.65 s	19431.25 s

Table 4.7. XMSS-SHAKE execution time

		n	
		192	256
h	10	3.53 s	4.76 s
	16	221.22 s	301.26 s
	20	3820.66 s	5171.24 s

Table 4.8. XMSS-SHAKE256 execution time

security parameters that influence both the performance and security of LMS are the tree height h, the hash function output length n, and the LM-OTS typecode, which encapsulates parameters like the Winternitz coefficient w and the length of the hash-based signature.

The parameter h, representing the height of the Merkle tree, determines the total number of allowable one-time signatures. Each LMS private key can generate up to 2^h signatures, since each leaf of the Merkle tree corresponds to a distinct LM-OTS key pair. For example, if $h = 10$, the LMS key pair supports 1024 signatures, whereas $h = 20$ enables 1,048,576 signatures. Larger h values allow more signatures but result in longer authentication paths in each signature and increased computational overhead during key generation and verification, as the Merkle tree must be traversed to compute or validate the path from the leaf to the root.

Key generation in LMS involves creating 2^h LM-OTS key pairs, each consisting of a set of n-byte strings that serve as the secret components, along with a public key derived through iterated hashing. These LM-OTS public keys are then hashed to form the leaves of the Merkle tree. The internal nodes of the tree are computed recursively by hashing pairs of child nodes until the root is obtained, which serves as the LMS public key. The cost of this operation scales linearly with the number of LM-OTS keys, i.e., with 2^h. Thus, increasing h directly affects the time required for key generation, making it a significant factor when assessing the feasibility of LMS in real-time or resource-constrained environments.

The LM-OTS parameters themselves also play a crucial role in determining the signature size and efficiency. The Winternitz parameter w controls the trade-off between the signature length and the number of hash computations required to sign or verify a message. Smaller w values lead to shorter signatures but involve more hash operations, whereas larger w values reduce the hash count but increase the size of the signature. Common values include $w = 1, 2, 4$ and 8, with $w = 4$ offering a balanced compromise.

For each message signed, the LM-OTS scheme must hash the message digest and associated checksum values a number of times dependent on w, meaning that the choice of w has a tangible impact on the speed of signing and verifying.

Benchmark runned on the simulator shows that LMS key generation time can range from milliseconds to several seconds depending on the chosen tree height h and the number of hash operations involved in generating and compressing the LM-OTS keys into the Merkle root. For example, using $h = 10$ with $w = 4$ and a 256-bit hash function, key generation can be completed in under a second, while $h = 20$ may require several minutes.

In the context of FIPS 205, LMS provides a secure and practical post-quantum signature mechanism, where the security and performance characteristics are primarily governed by the Merkle tree height h and the LM-OTS parameters encapsulated in the typecode. A larger h increases signature capacity but also computational and memory demands, while the Winternitz parameter w tunes the trade-off between signature size and processing efficiency. These parameters must be carefully selected based on the specific requirements of the target deployment, whether that involves embedded systems, software update signing, or document authentication, ensuring that LMS's statefulness and performance constraints are properly managed to maintain both usability and long-term security resilience against quantum adversaries.

		h				
		5	10	15	20	25
n,w	24,1	11.054 ms	244.858 ms	8.209 s	261.27 s	8231.285 s
	24,8	104.821 ms	3.096 s	98.986 s	3360.647 s	105381.244 s
	32,1	14.146 ms	329.693 ms	10.881 s	341.285 s	10911.245 s
	32,8	130.005 ms	3.947 s	123.938 s	4150.644 s	

Table 4.9. LMS-SHA256 execution time

		h				
		5	10	15	20	25
n,w	24,1	29.616 ms	851.712 ms	27.619 s	891.245 s	28231.257 s
	24,8	385.994 ms	12.518 s	372.187 s	12720.645 s	
	32,1	39.372 ms	1.119 s	41.462 s	1231.269 s	39271.253 s
	32,8	534.152 ms	16.798 s	511.245 s	17191.249 s	

Table 4.10. LMS-SHAKE256 execution time

Chapter 5

Conclusion

5.1 Time comparison and projections

In this master's thesis, I have undertaken a comprehensive quantitative analysis of the temporal resources required for the execution of asymmetric cryptographic algorithms, comparing classical schemes with their emerging post-quantum counterparts. By meticulously measuring and recording the execution times of established algorithms-specifically RSA, ECDSA, and EdDSA, I established a robust threshold value that serves as a reference benchmark. This benchmark enables a rigorous and meaningful performance comparison with post-quantum algorithms such as ML-DSA, ML-KEM, XMSS, and LMS, which differ substantially in mathematical structure and computational complexity.

The experimental investigation was carried out across two complementary execution environments. The first environment comprised a physical Luna Hardware Security Module (HSM) manufactured by Thales, deployed in the corporate infrastructure of Spike Reply, where I completed my internship and conducted the bulk of the measurements. The second environment was a software HSM simulator provided by Utimaco. The inclusion of a simulated environment was necessitated by the absence, at the time of data collection, of Thales firmware and software with native support for post-quantum algorithms. Consequently, I measured the execution times of classical algorithms in both the real and simulated HSM contexts, deriving performance ratios across different algorithmic complexity classes. Thereafter, I executed the post-quantum algorithms within the Utimaco simulator and, by integrating these measurements with the network and TLS-induced latency profile observed on the physical HSM, I extrapolated the expected execution times on dedicated HSM hardware once post-quantum support becomes available.

A key methodological consideration in this study pertains to the presence of end-to-end overheads introduced by network components, transport-layer protocols, and intermediate infrastructure between the client application and the HSM. Although these delays were inherently included in all timing measurements, their relative consistency throughout the experimentation phase allowed for a comparative analysis based on differential timing metrics. By focusing on relative performance deltas rather than absolute times, the influence of extraneous variables was effectively neutralized, ensuring that the comparative assessment reflects the intrinsic computational demands of each algorithmic

family.

The empirical results reveal that current post-quantum algorithms incur a substantially higher temporal overhead compared to classical asymmetric schemes. In enterprise environments equipped with dedicated HSMs, where scaling computing resources may be constrained by budgetary and administrative limitations, the execution time emerges as a critical performance indicator. Nonetheless, the transition to post-quantum cryptographic standards is indispensable to maintain data security in the advent of practical quantum computing capabilities.

Looking ahead, the development and release of official firmware and software updates for mainstream HSMs with built-in post-quantum algorithm support will be paramount. Such advancements will enable empirical validation of the projected performance estimates and facilitate the optimization of cryptographic offloading procedures. Future research should also expand the scope of resource analysis to include memory consumption, hash operation costs, and the impact on key management protocols. Moreover, it will be essential to investigate side-channel resistance and potential countermeasures in post-quantum implementations.

Only through an integrated, multidisciplinary approach, encompassing hardware engineering, protocol design, and security analysis, can the temporal overhead of post-quantum cryptography be mitigated to levels acceptable for high-performance operational settings. In conclusion, this thesis provides a clear quantitative framework for understanding the temporal challenges associated with migrating to post-quantum cryptographic solutions, thereby laying the groundwork for informed decision-making in the adoption of quantum-resistant data protection strategies.

Performances of standard algorithms

The execution of classical cryptographic algorithms exhibited a diverse spectrum of timing results, revealing the intricate balance between algorithmic complexity, hardware acceleration, and communication latency. For each algorithm, I collected a statistically significant number of samples ($n = 100$) to compute not only mean execution times but also standard deviations, and median values. This comprehensive statistical treatment ensures a robust characterization of performance under both typical and peak-load conditions.

RSA

RSA-2048: On the Luna HSM, signature generation and decryption operations showed an average latency of approximately 160ms (±15ms), with a standard deviation of ±100ms. The simulator, by comparison, yielded mean execution times around 650ms (±30ms) with a standard deviation of ±200ms and exhibited higher variance, with occasional peaks above 1200ms. These measurements translate into an HSM speed-up factor of roughly $4x$ for RSA-2048.

RSA-4096: Scaling up to a 4096-bit modulus amplified computational demands substantially: the HSM's mean operation time increased to around 1100ms (±85ms), while the simulator averaged close to 7500ms ±400 ms and a standard deviation of 4000 ms,

corresponding to a speed-up of about $6.5x$. The simulator's broader variance is indicative of its reliance on general-purpose CPU scheduling and memory management.

These results highlight the effectiveness of specialized cryptographic co-processors integrated into HSMs. Because the network and TLS handshake overhead (approximately 6ms) constitutes less than 3% of total RSA runtime, its impact on overall latency is minimal and can be safely amortized across batches of operations.

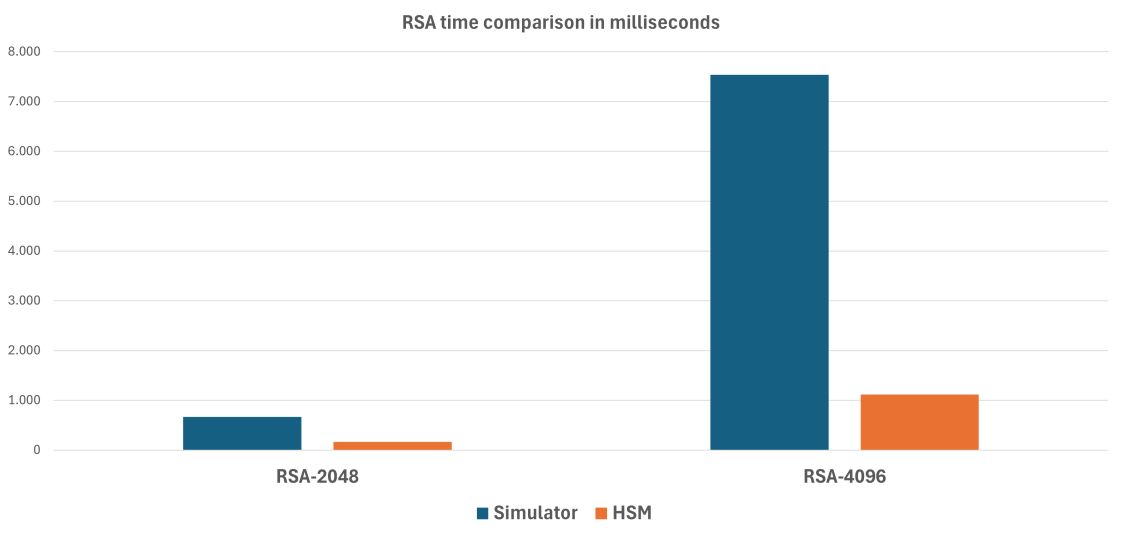

Figure 5.1. RSA time requirement measured both on the simulator and the HSM

ECDSA

The performance landscape shifts noticeably for elliptic-curve operations, where raw computational times often approach or fall below communication latency thresholds.

Curves P-256 and secp256k1: The Luna HSM exhibited end-to-end times averaging 10ms and 14ms (±0.2ms), while with the simulator 3.73 and 3.57 ms (±0.2ms).

Curves P-384 and P-521: For higher-security curves, the HSM's hardware accelerators gain prominence. Raw HSM latencies were measured at 11ms (±0.2ms) for P-384 and 13ms (±0.3ms) for P-521, compared to 9ms (±0.3ms) and 16ms (±0.4ms) on the simulator, respectively.

An important factor to consider while analyzing this result is their order of magnitude. With execution time so small we need to also consider the delay introduced by the network, that we measured being approximatively 6ms. Another time overhead seems to be introduced by some security function internal to the HSM. Even though this was not investigated in too much detail this value should be around 4ms. Combining this two values gives a total setup time of about 10ms. We can see, then, that the boost in raw computational performances is comparable to the one calculated for RSA and oscillates between 4 and 6.5 times.

EdDSA

Ed25519 demonstrates a unique profile characterized by uniform execution steps and simpler arithmetic:

End-to-end latency: The HSM recorded an average of 13ms (±0.1ms), while the simulator measured 5ms (±0.2ms). After removing the combined 10ms overhead introduced by network/TLS and HSM initialization, the adjusted HSM computation time show a significant increase in performances slightly inferior to the one calculated earlier of approximately 1.7 times.

These classical performance baselines serve as critical reference points for the subsequent analysis of post-quantum algorithms, enabling a nuanced assessment of the additional temporal investments required to achieve quantum-resistant security.

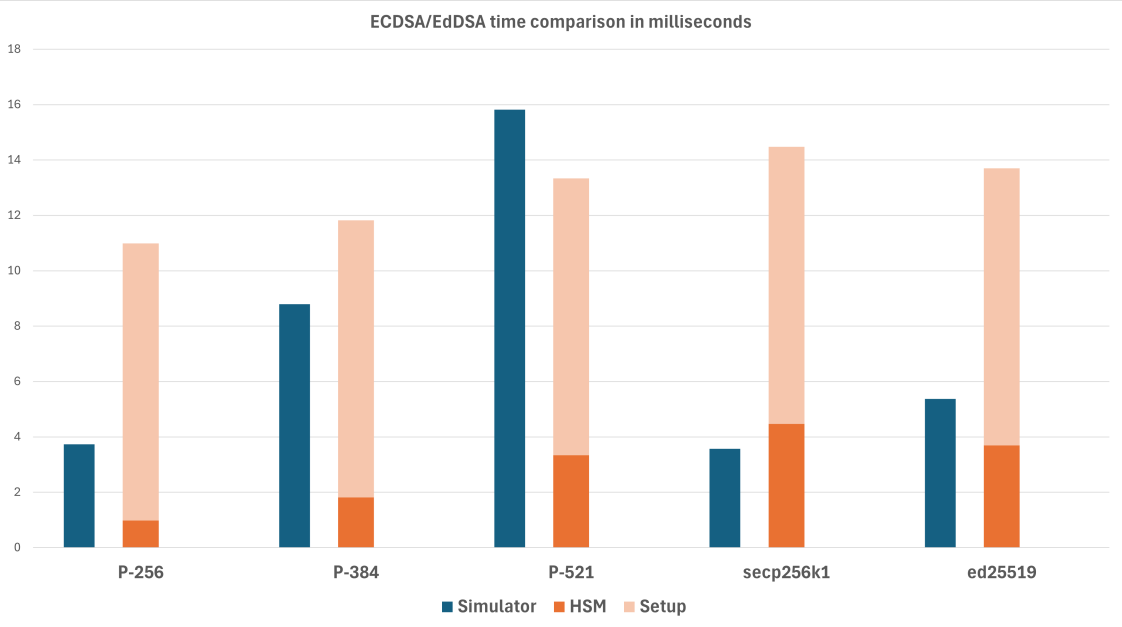

Figure 5.2. ECDSA and EdDSA time requirement measured both on the simulator and the HSM

Performances of PQ algorithms

Transitioning to post-quantum cryptographic schemes necessitates a thorough understanding of not only their theoretical security guarantees but also their operational performance characteristics in practical environments. In this study, we analyze four representative post-quantum algorithms: ML-DSA, ML-KEM, XMSS and LMS, using simulation data from Utimaco's HSM emulator and extrapolations to dedicated hardware. All projected end-to-end latencies incorporate an estimated network/TLS overhead (∼6ms) and hardware initialization cost (∼4ms) derived from classical ECDSA benchmarks.

ML-DSA

ML-DSA leverages structured lattice problems to achieve security against both classical and quantum adversaries. Its signature generation phase consists of sampling lattice vectors and performing a small number of polynomial multiplications:

The simulation results show key generation times of 2ms (k=2), 2.5ms (k=3) and 3ms (k=5), where k denotes the root-Hermite factor influencing security. Given this data we can extrapolate the time required by that HSM. Dividing by the classical ECDSA speed-up factor ($\sim 5x$ for small operations) suggests raw computation times of ∼0.5ms, ∼0.6ms, and ∼0.7ms, respectively. Including the 10ms combined latency, predicted total times are all very close to 11ms.

	ML-DSA-44	ML-DSA-65	ML-DSA-87
k	2	3	5
Time (ms)	10.517	10.621	10.686

Table 5.1. ML-DSA estimated execution time on HSM

ML-KEM

ML-KEM employs a variant of the bounded distance decoding problem for key encapsulation. Its performance exhibits minimal sensitivity to the security parameter k due to fixed-size polynomial operations:

Simulation stability: runtime variance under 0.1ms across k values 2, 3, and 5 (≈2ms baseline). With this data, even applying a more conservative $1.7x$ speed-up the obtained yield is ≈1.2ms of raw computation. After accounting for the netword ant setup overhead become very close to 11ms end-to-end latency.

	ML-KEM-512	ML-KEM-768	ML-KEM-1024
k	2	3	5
Time (ms)	10.47	10.472	10.541

Table 5.2. ML-KEM estimated execution time on HSM

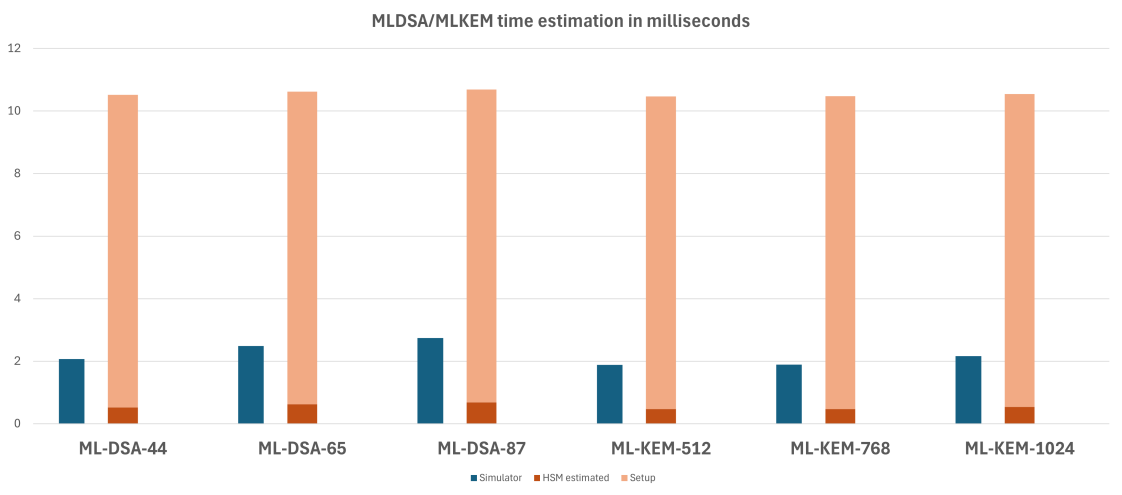

Figure 5.3. MLDSA and MLDSA time estimation obtained by projection of simulation results

XMSS

XMSS is a stateful hash-based signature system relying on Merkle trees to achieve forward security. Its key-generation phase requires computing 2^h leaf hashes and intermediate nodes, where h is the tree height. The impact of the height parameter gives us a 10 times increase at each step. For example for the XMSS-SHA256 algorithm with $n = 512$ changing h from 10 to 16 and to 20 gives execution times of ≈14 s, ≈1040 s and ≈ 15590 s (4 hours and 20 minutes). The same pattern can be seen with different hash functions and different values of the n parameter. On the other hand n makes the time change differently according to the hash function that is being used, and can vary a lot. For all this reasons, since all the measured times are well above the second, we can just ignore the contribution given by the network and setup overhead. For the ratio to use instead I applied the value of $6.5x$ given by RSA4096, since as for that case also here we have a many repeated computation and so the gain given by the dedicated hardware becomes significant. With this value we can see that despite still requiring a very long time to be executed, especially when compared with standard algorithm's time, XMSS becomes a little more manageable requiring ∼50 minutes in the worst case.

However, because XMSS is stateful and requires careful leaf index tracking, practical deployments often pre-generate key pairs during off-peak windows and store them for on-demand signing, effectively decoupling key-generation latency from real-time operational requirements.

		n		
		192	256	512
h	10	280 ms	530 ms	2.26 s
	16	18.3 s	26.34 s	160.1 s
	20	290.96 s	444 s	2398 s

Table 5.3. XMSS-SHA256 estimated execution time on HSM

		n	
		256	512
h	10	730 ms	2.99 s
	16	52.5 s	190.96 s
	20	795 s	2989 s

Table 5.4. XMSS-SHAKE estimated execution time on HSM

		n	
		192	256
h	10	540 ms	740 ms
	16	34.04 s	46.36 s
	20	587 s	795 s

Table 5.5. XMSS-SHAKE256 estimated execution time on HSM

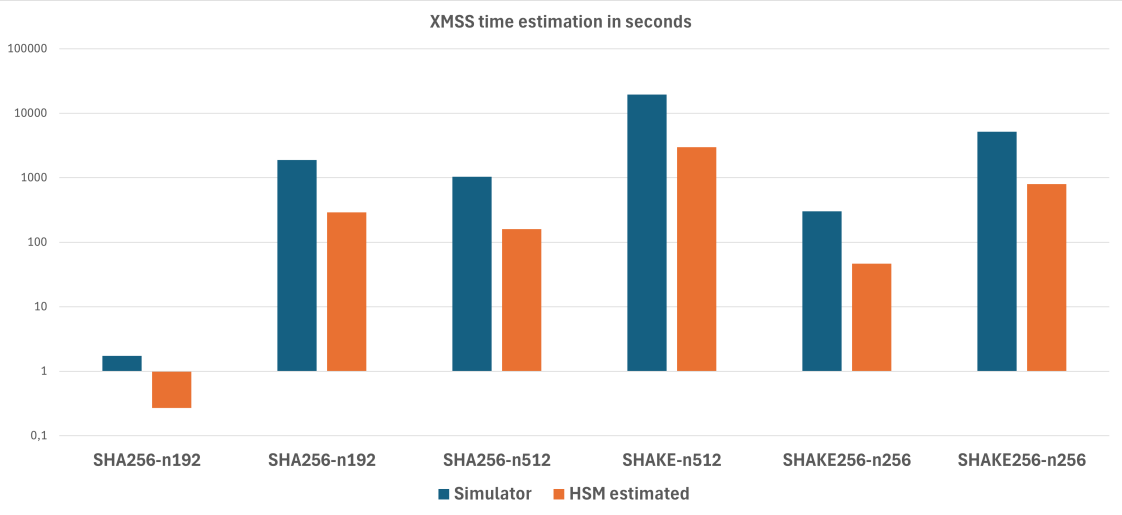

Figure 5.4. XMSS time estimation obtained by projection of simulation results

LMS

LMS is a stateful, hash-based scheme built on the Leighton-Micali one-time signature framework, whose key generation constructs a layered tree governed by parameters (h, n, w). In my most extreme test, using SHA-256 with $h = 25$, $n = 24$, and $w = 8$, building the tree required approximately 105300s (≈29hours), vividly illustrating the steep time cost of this approach. Unlike XMSS, LMS allows a wider choice of h values, producing execution times that range from tens of milliseconds ($approx$10ms at $h = 5$) up to hours at $h = 20$ or 25. The Winternitz parameter w doubles the number of chained hashes per leaf when increasing from 4 to 8, thereby halving signature size at the expense of computation, while the hash output length n scales runtime linearly. Extrapolating from these simulation results, and applying the same $6.5x$ speedup factor observed for XMSS on an HSM, reduces the worst-case key-generation time to about 4h30min. Although this remains orders of magnitude slower than conventional schemes, it supports over 33million signatures per key. To mitigate such latency in practice, parallel tree construction, hardware pipelining, and segmented key-generation jobs are recommended.

		h				
		5	10	15	20	25
n,w	24,1	11.701 ms	48 ms	1.27 s	40.2 s	1266 s
	24,8	26.126 ms	486 ms	15.24 s	517 s	16212 s
	32,1	12.176 ms	61 ms	1.68 s	50.5 s	1678 s
	32,8	30.001 ms	617 ms	19.08 s	638 s	

Table 5.6. LMS-SHA256 estimated execution time on HSM

		h				
		5	10	15	20	25
n,w	24,1	14.56 ms	141 ms	4.25 s	137 s	4343 s
	24,8	69.38 ms	1.94 s	57.27 s	1957 s	
	32,1	16.06 ms	182 ms	6.4 s	189 s	6041 s
	32,8	92.18 ms	2.59 s	78.66 s	2644 s	

Table 5.7. LMS-SHAKE256 estimated execution time on HSM

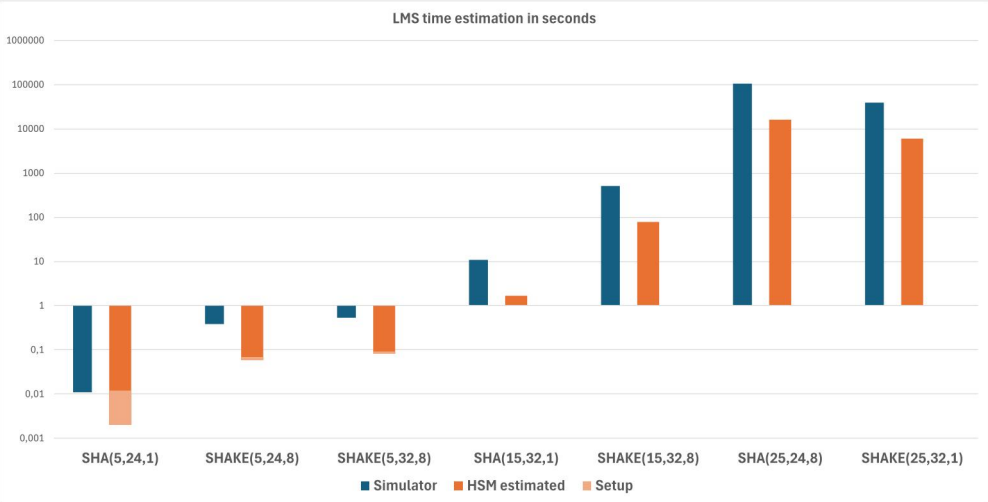

Figure 5.5. LMS time estimation obtained by projection of simulation results

In light of the temporal resource measurements presented in this work, it is clear that the four NIST-selected post-quantum schemes demand a use-case-driven balancing of security level against computational cost. Although our analysis has focused exclusively on key-generation latency in an environment unconstrained by processing power or memory, the literature indicates that both CRYSTALS-Dilithium (ML-DSA) and CRYSTALS-Kyber (ML-KEM) exhibit negligible signing and verification overhead, even on commodity hardware, and that tree-based schemes like XMSS and LMS similarly achieve sub-millisecond signature verification and low-millisecond signing times once their initial structures are built. Consequently, it is both feasible and advisable to offload the intensive key-generation phase, often involving millions of cryptographic hash invocations, to a high-performance host such as an HSM or dedicated server, and then provision the resulting keys to resource-limited endpoints.

However, for applications in tightly constrained environments (e.g., embedded systems, IoT devices, or edge sensors), even this offload model may prove inadequate. Our simulator-running on a modest laptop, required substantially longer to generate high-security parameter sets than the HSM, underscoring that vastly less capable embedded hardware would struggle to meet similar demands without extended latency or risking

thermal and power constraints. In the hash-based arena, although LMS and XMSS can incur hours of latency at maximum parameter selections, their one-to-many key-generation output (amassing upwards of 10^7 one-time keys per tree) means that system designers can tailor tree height (h) and Winternitz parameter (w) to match provisioning windows, trading off between precomputation time, signature size, and post-quantum security level. Moreover, hierarchical variants such as HSS further mitigate latency by enabling incremental key-generation across layered trees, allowing even compact devices to maintain an ample pool of precomputed certificates over extended operational periods.

Beyond discrete algorithmic performance, broader architectural strategies will be paramount. Network-wide key management frameworks must support seamless rotation and renewal of PQC keys, ideally integrating with existing PKI ecosystems to facilitate hybrid classical/quantum-resistant trust anchors. Parallelization of tree construction, hardware pipelining of modular-arithmetic units, and field-programmable gate array (FPGA) accelerators for polynomial arithmetic are promising avenues to compress critical path durations further. Comprehensive benchmarks on representative embedded platforms should be a priority for future work, ensuring that published performance claims translate reliably to target deployments.

Looking ahead, expert forecasting models for fault-tolerant quantum computers capable of undermining RSA-2048 continue to place their arrival in the 2040s, with median projections clustering around the mid-2040s under optimistic technology scaling scenarios. Yet, given the extended duration required to standardize, implement, and deploy post-quantum algorithms, compounded by their sharply increased resource footprints and integration complexities, organizations and governments cannot afford to defer migration. Early adoption of hybrid classical/PQC schemes, strategic offloading of key-generation workloads, and careful parameter selection will be essential to safeguard sensitive data against the quantum threat before it materializes. Only through proactive planning, rigorous performance validation, and adaptive security architectures can stakeholders navigate the transition to a post-quantum future with confidence and operational continuity.

Bibliography

[1] David Aasen and Morteza Aghaee. Roadmap to fault tolerant quantum computation using topological qubit arrays, 2025.

[2] Rajeev Acharya, Dmitry A. Abanin, and Laleh Aghababaie-Beni. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, December 2024.

[3] Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681, Feb 2023.

[4] Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, Feb 2025.

[5] Frank Arute, Kunal Arya, and Ryan Babbush. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[6] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, and Nicola Tuveri. Opensslntru: Faster post-quantum tls key exchange, 2021.

[7] Michael E. Beverland, Prakash Murali, Matthias Troyer, Krysta M. Svore, Torsten Hoefler, Vadym Kliuchnikov, Guang Hao Low, Mathias Soeken, Aarthi Sundaram, and Alexander Vaschillo. Assessing requirements to scale to practical quantum advantage, 2022.

[8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs#11.

[9] Castelvecchi Davide. Ibm releases first-ever 1,000-qubit quantum chip, 2023.

[10] Dongxin Gao and Daojin Fan. Establishing a new benchmark in quantum computational advantage with 105-qubit zuchongzhi 3.0 processor. *Phys. Rev. Lett.*, 134:090601, Mar 2025.

[11] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021.

[12] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020.

[13] Gil Kalai, Yosef Rinott, and Tomer Shoham. Google's quantum supremacy claim: Data, documentation, and discussion, 2023.

[14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.* Chapman & Hall/CRC, second edition, November 2014.

[15] Chen Lily, Moody Dustin, Regenscheid Andrew, Robinson Angela, and Randall Karen. Recommendations for discrete logarithm-based cryptography: Elliptic curve

domain parameters. Technical Report NIST SP 800-186, National Institute of Standards and Technology, 2023.

[16] Chen Lily, Jordan Stephen, Liu Yi-Kai, Moody Dustin, Peralta Rene, Perlner Ray, and Smith-Tone Daniel. Report on post-quantum cryptography. Technical Report NIST IR 8105, National Institute of Standards and Technology, 2016.

[17] XIN Ling. The world's first integrated quantum communication network, 2021.

[18] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, March 2019.

[19] SujayKumar Reddy M and Chandra Mohan B. Comprehensive analysis of bb84, a quantum key distribution protocol, 2023.

[20] NIST. Post-quantum cryptography.

[21] National Institute of Standards and Technology. Digital signature standard (dss). Technical Report FIPS 186-5, National Institute of Standards and Technology, 2023.

[22] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Technical Report FIPS 204, National Institute of Standards and Technology, 2024.

[23] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report FIPS 203, National Institute of Standards and Technology, 2024.

[24] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Technical Report FIPS 205, National Institute of Standards and Technology, 2024.

[25] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Paper 2015/939, 2015.

[26] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), July 2014.

[27] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms overgf(p)and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[28] J.M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 1975.

[29] Carl Pomerance. The quadratic sieve factoring algorithm. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *Advances in Cryptology — EUROCRYPT 84*, volume 209 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 1985.

[30] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography, 2024.

[31] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997.

[32] Harshdeep Singh. Code based cryptography: Classic mceliece, 2020.

[33] Vikas Srivastava, Anubhab Baksi, and Sumit Kumar Debnath. An overview of hash based signatures. Cryptology ePrint Archive, Paper 2023/411, 2023.

[34] Matt Swayne. Company claims quantum algorithm implements full adder operations on quantum gate computers, 2025.

[35] Emmanuel Thomé. 795-bit factoring and discrete logarithms, 2019.

[36] Kleinjung Thorsten, Diem Claus, Lenstra Arjen K., Priplata Christine, and Stahlke Colin. Computation of a 768-bit prime field discrete logarithm. *IACR*, 2017.

[37] I. Thorvaldson, D. Poulos, and C. M. Moehle. Grover's algorithm in a four-qubit silicon processor above the fault-tolerant threshold. *Nature Nanotechnology*, 20(4):472–477, Apr 2025.

[38] Violetta Weger, Niklas Gassner, and Joachim Rosenthal. A survey on code-based cryptography, 2024.

[39] Jieyu Zheng, Haoliang Zhu, Yifan Dong, Zhenyu Song, Zhenhao Zhang, Yafang Yang, and Yunlei Zhao. Faster post-quantum tls 1.3 based on ml-kem: Implementation and assessment, 2024.

[40] Yang Zhou, Jing-Yang Liu, Chun-Hui Zhang, Hua-Jian Ding, Xing-Yu Zhou, Jian Li, and Qin Wang. Experimental side-channel-secure quantum key distribution over 200 km, 2025.