Post-Quantum Cryptography: A Comprehensive Guide

cnlab security AG, Simran Tinani, Urs Wagner

Version	Description	Date
1.0	Final Version	2025-04-16

cnlab

cnlab security AG Obere Bahnhofstrasse 32b CH-8640 Rapperswil-Jona www.cnlab.ch +41 55 214 33 40

Contents

List of Figures

List of Tables

Objective and Scope

This paper is intended to support organizations in understanding and addressing the long-term risks that quantum computers pose to existing cryptographic systems. It provides the context and insight needed to assess the implications of quantum computing, prioritize and plan a strategic response, and take informed steps toward adopting quantum-secure cryptographic solutions. It spans foundational concepts in classical and post-quantum cryptography, the current state of quantum computing, the core principles of post-quantum cryptography (PQC), and the status of global standardization efforts. As these standards mature and the quantum threat becomes more tangible, this paper helps technical and strategic stakeholders prepare for a secure and timely transition.

1_ Introduction

Cryptography is a cornerstone of IT security, crucial for protecting the confidentiality, integrity, authenticity, and non-repudiation of data. Public-key cryptosystems are integral to various security protocols, enabling encryption, signatures, key exchange, and other critical functions. Classical public-key cryptosystems are based on the Diffie-Hellman and RSA protocols, which rely on the computational difficulty of specific mathematical problems: discrete logarithm computation and integer factorization, respectively. These protocols are foundational to internet security and have demonstrated resilience to attacks based on classical algorithms.

However, the advent of quantum computing poses a serious threat to these systems. Mathematician Peter Shor's quantum algorithm offers a highly efficient method to solve the underlying mathematical problems of both Diffie-Hellman and RSA, effectively breaking their security. At present, this threat remains theoretical, as no quantum computer capable of executing Shor's algorithm at a practical scale has been developed yet.

The term **quantum computer** refers to a computing machine that leverages the principles of quantum mechanics to perform computations in fundamentally different ways than classical computers, which rely on the laws of classical physics. A **cryptographically relevant quantum computer (CRQC)** is a quantum computer capable of breaking classical cryptographic algorithms by implementing quantum algorithms like Shor's algorithm. Although practical CRQCs do not yet exist, experts predict that they could be realized within a decade or two.

Under the field of **Post-Quantum Cryptography (PQC),** cryptographic systems designed to withstand attacks from both quantum and classical algorithms are developed. Over the past decade, PQC has evolved from a predominantly academic field into an area of high practical importance for industries and governments. Currently, the NIST Post-Quantum Cryptography Standardization Program, launched in 2017, is nearing completion. Standards for some PQC algorithms have been published and these are being integrated into leading technological provisions. Organizations must therefore understand and act on the risks posed by quantum computers, the importance of PQC algorithms, and the urgency of transitioning to quantum-secure systems.

This article is a comprehensive, in-depth guide to the foundations of post-quantum cryptography, written with security practitioners in mind. It is structured as follows.

- Chapter 2_ introduces the basic notation and terminology used in the article.
- Chapter 3_describes some widely used **classical cryptographic protocols** and the best-known **attacks** on them.
- Chapter 4_ introduces the basic principles of **quantum information theory** and describes Shor's algorithm, its effect on classical cryptography, and some implementational aspects.
- Chapter 5_ explores the fundamental **differences between quantum and classical computers** and discusses some of the major obstacles in developing a CRQC.
- Chapter 6_ provides an overview of **post-quantum cryptography (PQC)**, outlining the mathematical foundations, standardization efforts, and the leading cryptographic schemes.
- Chapter 7_ addresses the **implications** of quantum threats today, **current implementational steps** towards quantum-secure cryptography, and offers **recommendations** for organizations and manufacturers for the transition to quantum-secure IT systems.
  Each chapter begins with a short summary capturing its main takeaways.

2_ Notation and Terminology

While the technical and mathematical details in this article have been kept to a minimum, understanding some notations and terminology is crucial to grasp the context and idea behind most cryptographic protocols and their security aspects.

2.1 Mathematical Operations

In cryptography, various mathematical operations are used to transform data. Of these, the operations that are important for this article are as follows.

- **Exponentiation**: raises a number to a power. In the expression a^b, a is the base, and b is the exponent; a^b means a multiplied by itself b times. For instance, 2^3 means $2 \times 2 \times 2 = 8$.
- **Modulo**: finds the remainder when one number is divided by another. For example, $13\ (mod\ 5)$ means the remainder when 13 is divided by 5, which is 3, $20\ (mod\ 5)$ is 0.
- **GCD** (Greatest Common Divisor): finds the largest number that divides two or more numbers without leaving a remainder. For example, the GCD of 12 and 15 is 3, the GCD of 10 and 7 is 1.

2.2 Terminology: Time Complexity and Practical Running Time

2.2.1 Bit Security Level

In cryptography, **security level** (or bit security level) quantifies the computational effort required to break a cryptographic system, thus representing the system's resistance against attacks. Typically, it is measured in bits. For a system with an n-bit security level, the best-known

attack algorithm on the system requires approximately 2^n elementary operations (e.g., encryptions or finite field multiplications) to succeed: the number of operations is exponential in the security parameter n. As of 2024, a bit security level of approximately 120 bits is considered adequately secure. This is precisely the security level recommended for all cryptographic mechanisms by the German Federal Office for Information Security (BSI, Cryptographic Mechanisms: Recommendations and Key Lengths, 2024).

2.2.2 Time Complexity: Polynomial, Exponential, and Subexponential time

Time complexity is a theoretical measure used to describe how the running time of an algorithm scales with the size of its input, based on the number of elementary steps involved. The terms polynomial, exponential, and subexponential time are used to classify the time complexity of cryptographic algorithms.

A **polynomial-time algorithm** has a running time that increases relatively slowly with the size of its input, making it practical and time-efficient for real-world applications. In contrast, an **exponential-time algorithm** experiences a rapid growth in running time as input size increases, rendering it impractical for large inputs. A **subexponential algorithm** falls between these two extremes; its running time depends on the specific parameters and the size of the input.

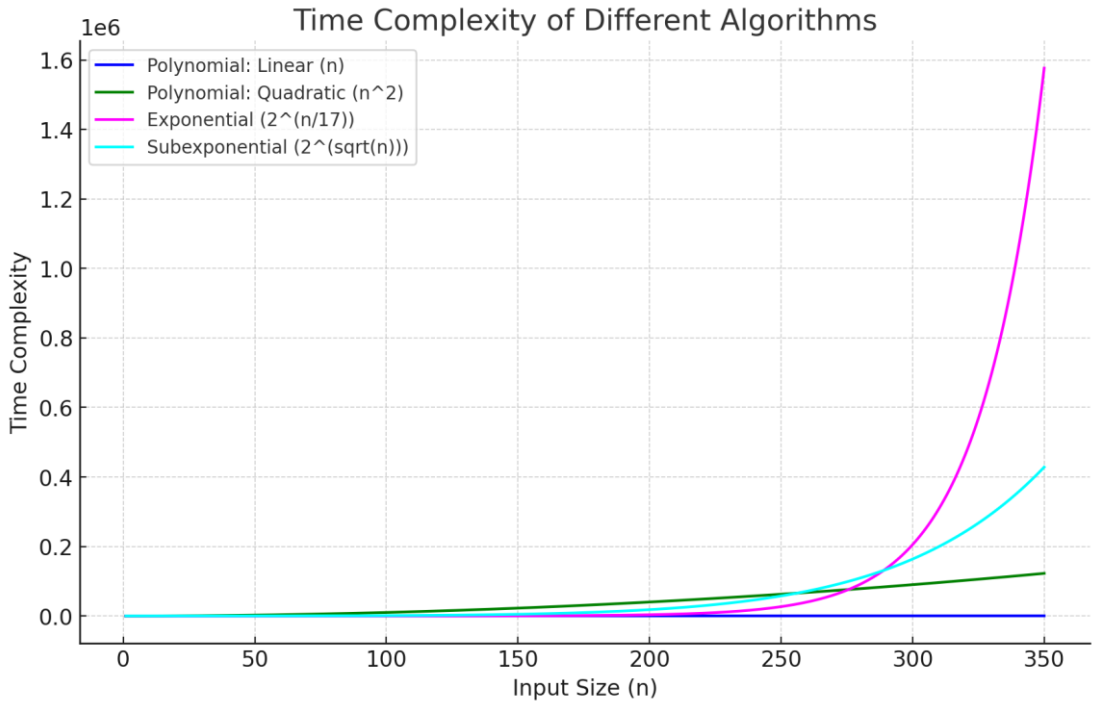

Figure 1: Illustration of time complexities

2.3 Computational Power and Key Sizes

While the theoretical time complexity is a function only of the algorithm, the actual time required to run an algorithm in practice depends on the processing power of the machine used. A computer's processing power is defined in terms of metrics that describe its performance in terms of speed, efficiency, and capacity.

A computer's general processing power is determined by a combination of factors, including hardware design (e.g., number of CPU cores, transistor count, floating-point units (FPUs), memory bandwidth, and cache architecture), the Instruction Set Architecture (ISA, such as x86 or ARM), cache performance, software optimizations, and the nature of the workload. Some common general-purpose metrics associated with a CPU are its clock speed, Instructions per Cycle (IPC), and Millions of Instructions per Second (MIPS).

One key factor determining computational power is the number of transistors on a microchip, which is the core of a computer's processing unit. As technology advances over time, the number of transistors on a microchip increases. In fact, **Moore's Law**, an empirical observation made in 1975 (Moore, 1975), predicts that this number doubles approximately every two years. Since 1975, Moore's Law has largely held true, though technological advancements have slowed slightly below the predicted pace since 2010 (Hennessy & Patterson, 2018).

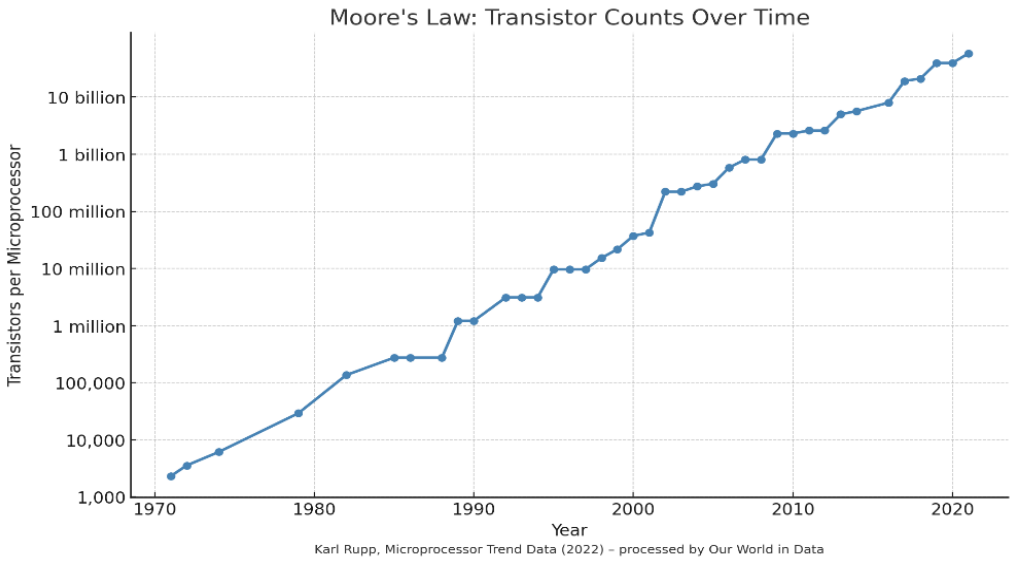

Figure 2: Transistors on a microprocessor chip (data source: Karl Rupp (Rupp, 2022), processed by Our World in Data)

Even though cryptographic performance itself does not scale directly with transistor density, Moore's law serves as a good overall approximation. Moore's Law implies that to maintain the same level of security (i.e. computational effort for an attack), cryptographic key sizes must also increase over time.

For instance, if the best-known attack on a cryptosystem has an exponential time complexity (e.g., 2^n) in the bit size n of the key, it suffices to increase the key size by one bit every two years to maintain security. In practice, key sizes are chosen with sufficient margins to avoid frequent adjustments.

However, if the best-known attack has polynomial time complexity (e.g. n^2) in the bit size n of the key, the key size must be increased by a factor of $\sqrt{2}$ every two years. In that case, the key sizes required for a secure system are often tremendously large and grow rapidly with time, rendering them infeasible. If a polynomial-time attack is discovered for a cryptosystem, it is henceforth usually considered broken. This is precisely the case for Shor's quantum algorithm for classical public-key cryptography. For this reason, the rest of this article emphasizes theoretical time complexity over practical running times.

For subexponential attacks, as known e.g. for RSA, the necessary increase in the key size is feasible, but not satisfactory in the long run. Key sizes must thus be updated on a regular basis.

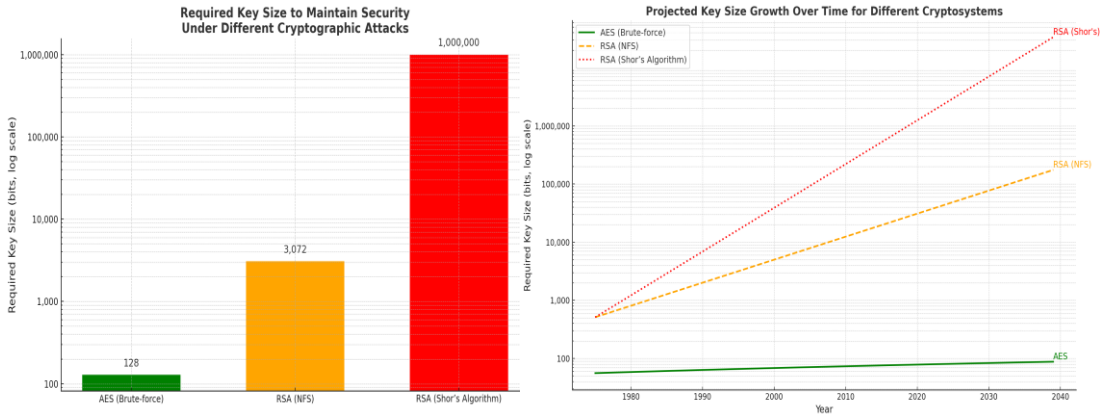

Figure 3: Key sizes and their projected necessary growths under different attack complexities

3_ Classical Cryptography: Algorithms & Attacks

Section summary: Cryptographic systems can broadly be classified as public-key (asymmetric) or secret-key (symmetric). In secret-key systems, two parties use a pre-shared common secret to secure their communication. In public-key systems, communication can be protected without any pre-shared secrets, i.e. without having ever exchanged any information through a separate, private channel.
For state-of-the-art secret-key systems and hash functions, no attacks (classical or quantum) substantially faster than brute-force are known. These systems are believed to remain secure also under the advent of quantum computers, provided reasonably scaled parameter sizes are used.
Things look different for public-key systems. While subexponential time classical algorithms exist to attack the most common representatives such as Diffie-Hellman and RSA, they are still considered secure in the face of classical attacks given large enough key sizes.
However, Shor's polynomial-time quantum algorithm efficiently breaks these systems, without any scope for practical mitigation through parameter adjustment. For this reason, public-key systems are the central theme in this article and in the general area of post-quantum cryptography.

3.1 Secret-Key Algorithms and Hash Functions

3.1.1 Secret-Key Algorithms

Secret-key cryptography, (or private-key or symmetric cryptography), is the branch of cryptography under which the communicating parties encrypt and decrypt their messages using a single shared secret key that the adversary does not possess. A prerequisite for this is that the communicating parties have a private medium to exchange secret keys beforehand.

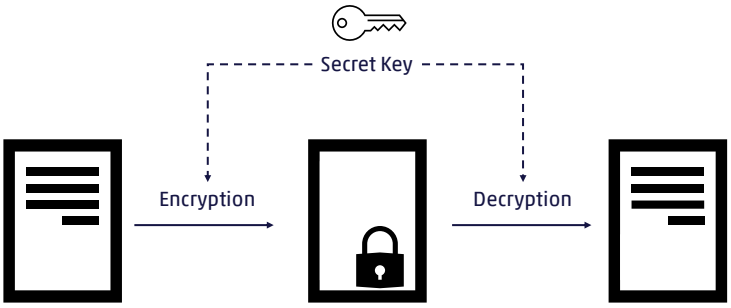

Figure 4: Symmetric encryption

Secret-key cryptosystems are used extensively in conjunction with public-key cryptosystems. In fact, due to their superior efficiency, they are typically the preferred method for message encryption, after a public-key cryptosystem has been used for key exchange.

Most modern secret-key cryptosystems employ repeated bit-level operations, combined so that they are difficult to reverse without the private key. Notable examples include AES, ChaCha20, and Twofish, with AES being the standard, commonly using 128, 192, or 256-bit keys.

Attacks

The fastest known classical attack on AES is brute force, where all possible keys are tested until the correct one is found. Unlike public-key cryptography, which can be efficiently broken by quantum computers using Shor's algorithm, no exponentially faster quantum algorithm exists for attacking symmetric-key systems like AES. However, quantum computers can still perform a more efficient brute-force attack using Grover's algorithm (Grover, 1996).

For example, AES-128 (128-bit keys), which requires around 2^{128} steps in a classical attack, would require 2^{64} operations under Grover's algorithm. This quantum advantage can be controlled by using reasonably larger key sizes, such as AES-256. The same principle applies to other secure secret-key encryption algorithms. **For this reason, quantum computers are not considered a threat to secret-key cryptography.**

3.1.2 Hash Functions

A hash function is a function that compresses an arbitrary-length message to a fixed length digest. A cryptographic hash function is a hash function that exhibits certain additional cryptographically relevant properties, such as collision resistance (computing two different messages that map to the same digest (output) requires an exponential-time algorithm) and preimage

resistance (given a digest, computing a message that maps to it requires an exponential-time algorithm).

Hash functions have several applications in cryptography. They are often used in password storage, for verifying the integrity of files transmitted over the internet, and as components of digital signatures to produce a digest of the message prior to generating the signature. The standard and most widely used family of cryptographic hash functions are SHA-2 and SHA-3 from the series of Secure Hash Algorithms (NIST, Secure Hash Standard, 2015). SHA-2 and SHA-3 differ in their construction principle, leading to slightly different security properties. It is, for example, easier to construct a message authentication codes (MAC) from SHA-3 due to its resistance to length-extension attacks.

Attacks

The fastest known classical attack on SHA-2 and SHA-3 are brute-force attacks. For an n-bit hash function (output length n bits), finding a preimage hence requires approximately 2^n classical operations and finding a collision requires approximately $2^{n/2}$ operations (using a "birthday attack").

Similar to attacks on private-key systems, there is no exponentially superior quantum algorithm known for this task. Quantum computers also attack hash functions using brute-force, albeit a more efficient version of it, applying a speedup using Grover's algorithm (Grover, 1996). Therefore, the security level in the quantum setting can be preserved by scaling the digest size linearly. **For this reason, quantum computers are not considered a threat to hash functions**.

3.2 Public-Key Algorithms

As explained above, secret-key cryptosystems rely on a pre-shared secret key accessible exclusively to the communicating parties. Establishing such a shared key over the internet is challenging, as no inherently private channel exists for secure key exchange. Public-key cryptography addresses this limitation by enabling secure communication over insecure channels without requiring pre-shared secret information. It not only simplifies key distribution but also facilitates advanced security features, such as digital signatures and secure multi-party computations, making it indispensable for modern secure communication.

The three major types of public-key protocols are **key exchange protocols**, which establish shared, secret encryption keys based on exchanged public-key material, **encryption protocols**, in which publicly available key material is directly used to encrypt messages, and **digital signature** protocols, which are used to verify the authenticity of messages and their origins. In contrast to secret-key cryptosystems, the effect of quantum computers on the presently used public-key cryptosystems is devastating. This vulnerability makes public-key cryptography a primary focus in the development of post-quantum cryptography.

Today, most widely used public-key protocols employ the Diffie-Hellman and the RSA algorithms. These are constructed based on the respective computational difficulties of the discrete logarithm problem and the factoring of large integers. Below, we describe these in more detail.

3.2.1 Encryption

In public-key encryption (or asymmetric encryption) schemes, the communicating parties exchange encrypted messages without having to share a secret key. To achieve this, the sender and receiver use a different key to encrypt and decrypt the message (hence, the term "asymmetric"). A receiving party publishes a public key, which senders can use to encrypt their messages to them. The corresponding private key to this public key is retained secretly with the receiver only and is used to decrypt the received encrypted messages. Any other party without possession of the secret key cannot decrypt the exchanged messages.

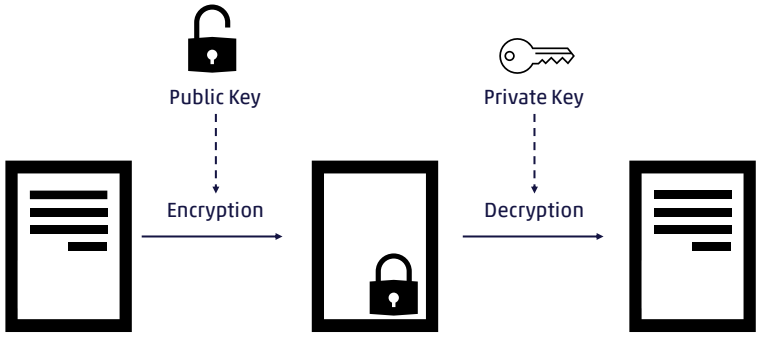

Figure 5: Asymmetric encryption

One of the most important examples of a public key encryption scheme is the RSA protocol.

Loosely, it proceeds as follows. A receiving party selects two large secret primes p and q and computes $N = pq$. They choose encryption exponent e and publish (N, e) as the public key. A sender encrypts their message m by exponentiating by the encryption exponent e. The ciphertext is then $c = m^e \pmod N$.

The receiving party decrypts the message by exponentiating the ciphertext with the decryption exponent d, which is chosen to give back $m = c^d \pmod N$. It is assumed, based on decades of research (though not proven), that without knowledge of the factors p and q, d cannot be efficiently computed. This is why the parameters p, q, and d are referred to as the private portion of the key.

The RSA algorithm can also be easily modified for digital signatures.

3.2.2 Key Exchange and Key Encapsulation Mechanisms

As the previous subsection outlines, asymmetric encryption schemes enable the exchange of encrypted messages without the use of a shared secret key. However, they are far less efficient compared to symmetric encryption schemes. Thus, in practice, instead of using public key algorithms for directly encrypting messages, they are used for the secure transport of shared symmetric keys, which are then subsequently used for encrypting messages. The two major ways of achieving this are key exchange algorithms and key encapsulation mechanisms.

In a Key Encapsulation Mechanism (KEM), a public-key encryption algorithm is used by two communicating parties to securely encrypt symmetric keys before they are transported to the peer.

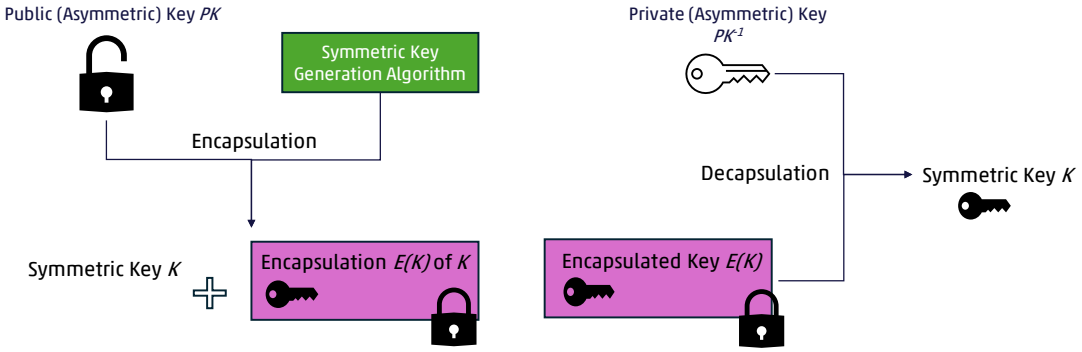

Figure 6: Key encapsulation and decapsulation

Another method to transport symmetric secret keys is by means of a Key Exchange algorithm, the most prominent of which is the Diffie-Hellman (DH) key exchange protocol.

The security of the Diffie-Hellman protocol relies on the computational difficulty of the Discrete Logarithm Problem (DLP) in a suitable finite field or elliptic curve group. In its simplest form, it can be described as follows. Let p be a prime, g be a number that is not a multiple of p, x be some number, and $h = g^x \pmod p$ be obtained by taking the remainder of the x^{th} power of g when divided by p. The DLP is the problem of finding the exponent x when given the values of g, h and p.

A key exchange protocol based on the DLP is constructed as follows. Two parties exchange the values g^x and g^y over a public channel, keeping their chosen integers x and y secret. Using the public share of the other party and their own secret numbers, they both calculate a shared secret g^{xy}. However, the adversary, who can only see g^x and g^y, must solve the DLP to calculate one of x and y, to subsequently get g^{xy}. While it is not proven that the adversary's task is equivalently computationally hard as the DLP, there exists strong heuristic evidence in support of this.

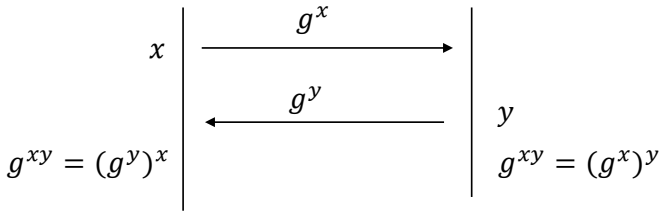

Figure 7: Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange works analogously over elliptic curves. The numbers above are replaced by elements defined by an elliptic curve and appropriate calculation rules. Various extensions and modifications of the Diffie-Hellman protocol also construct direct encryption and digital signature schemes based on the discrete logarithm problem. Some examples are the ElGamal, Schnorr and DSA schemes.

It is noteworthy that the Diffie-Hellman key exchange algorithm offers the property of **perfect forward secrecy (PFS),** which means that messages exchanged in previous sessions remain

protected even when a long-term key is compromised. This property arises from the fact that Diffie-Hellman does not use long-term values for key exchange (except, in addition, to ensure authenticity of the keys). KEMs do not provide PFS by default but can be modified to do so by using only ephemeral key pairs per session.

3.2.3 Digital Signatures

The first two subsections in this chapter describe the exchange of messages and symmetric keys over a public, insecure channel. However, an equally important goal in cryptography is to be able to verify the authenticity of messages, i.e. their origination at the source that they claim to be from. Without a way to prove authenticity, any party can impersonate another.

For example, in a man-in-the-middle attack, an attacker impersonates (at least) one of the communicating parties and sends their own public key in place of the legitimate one and subsequently decrypts and/or modifies the rest of the communication. Such attacks can be prevented with a sound authentication mechanism combined with a system for identity attestation such as a public key infrastructure (PKI).

A digital signature is an algorithm used to prove and verify the authenticity and integrity of electronic messages and the identity of the signing party. In addition, digital signatures can offer non-repudiation.

Under a digital signature protocol, the sender uses a signature algorithm to generate a signature from the message and the private key. Typically, a signature algorithm involves the calculation of a hash of the message on which the signing function using the secret key is applied. The signature can be verified by the receiver using the sender's public key. The primary security property required from a signature scheme is "existential unforgeability", which means that a party without access to the private key is not able to forge a valid signature on a message.

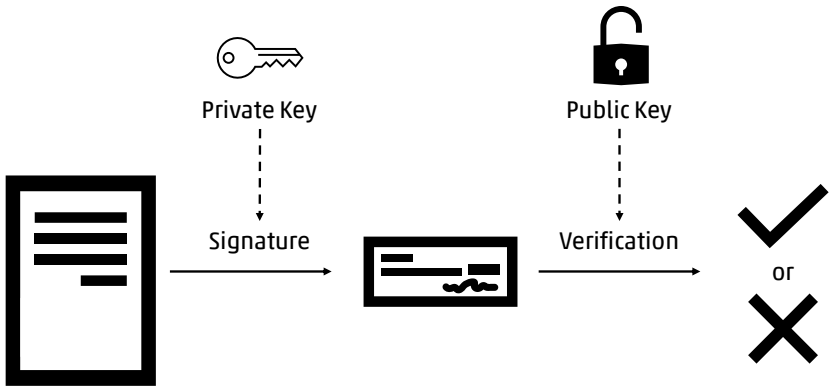

Figure 8: Digital signatures

A primitive example of a signature algorithm is the RSA signature algorithm, which also uses the difficulty of factorization as its security basis, similarly to the encryption variant. Loosely, it proceeds as follows. A signing party selects two large secret primes p and q and computes $N = pq$. They choose a secret signing key d and a matching verification exponent e and publish (N, e) as

the public key. To produce a signature on a message m, the signing party exponentiates it by the secret signing exponent d. The signature is then $s = m^d \pmod N$.

The receiver verifies the signature by exponentiating it with the public verification exponent e, which is chosen to give back the message $m = s^e \pmod N$, which is also sent separately. With this, the receiver has confirmation that the message was received untampered from the intended sender. As for the encryption algorithm, it may be assumed that without knowledge of the factors p and q, d cannot be efficiently computed. This is why the parameters p, q, and d are referred to as the private portion of the key.

In practice, the most used digital signature algorithms are the RSA Signature scheme, the Digital Signature Algorithm (DSA) based on the discrete log problem in a finite field, and the Elliptic Curve Digital Signature Algorithm (ECDSA) based on the discrete log problem in an elliptic curve.

A digital signature algorithm serves as proof of ownership of the claimed private key and ensures this proof is linked to the transmitted message. The corresponding public key is often associated to the identity of the sending party via a digital certificate, which is issued by a trusted authority. This process is part of Public Key Infrastructure (PKI), which manages the issuance and validation of digital certificates. PKI relies on a hierarchy of Certificate Authorities (CAs) to verify identities and ensure the integrity and authenticity of cryptographic keys.

3.2.4 Best-Known Attacks

As discussed above, the discrete logarithm and integer factorization problems form the basis of most public-key cryptosystems of today. Both these problems are believed, with strong supporting evidence, to have no polynomial time solution implementable on a classical computer. This means that the best-known classical algorithms for their solutions are not efficient enough to solve them with the available computational power, as long as the parameters such as key sizes are chosen appropriately.

Brute force

Consider first the factorization problem. Let $N = pq$ be a product of two large primes. The naïve way to factorize N would be to iterate through all numbers smaller than N (actually, it suffices to only go up to the square-root of N) and check if each number is a divisor of N.

Similarly, the discrete logarithm problem can be solved by brute force: one may simply try every value of x below some bound and check if $g^x = h$. A more efficient approach for a general group uses collision-based algorithms, finding matches between lists of elements. Both the crude brute force and the collision-based improvement are exponential-time algorithms, and therefore become impractical for large, cryptographic-scale parameter sizes. However, there are more efficient classical algorithms than brute force for both these problems.

General Number Field Sieve

The **general number field sieve (GNFS)** (Pomerance, 1996) is one of the most powerful known factorization methods and has been used to factor several RSA challenge numbers. It can also be used to solve the discrete logarithm problem in finite field groups. The GNFS is a subexponential time algorithm and reduces the bit security of RSA and DH in finite fields significantly.

The GNFS cannot be used in an elliptic curve group. Thus, for the DLP in well-chosen elliptic curve groups, exponential-time collision algorithms like **Pollard's Rho Algorithm** (Pollard, 1978) are the best-known approach. Therefore, the security of elliptic curve-based Diffie Hellman key exchange with a key size of 256 bits, is 128 bits. To achieve this same bit security, a key size of approx. 3072 bits must be used for DH over finite fields. Thus, elliptic curve-based systems achieve the same security level with far smaller key sizes and are therefore preferred for public-key cryptography.

The recommended key sizes for RSA in practice are selected based on the asymptotic estimate of the running time of the GNFS. At present, the RSA modulus N and the finite field Diffie-Hellman prime p are recommended by several central organizations like the German Federal Office for Information Security (BSI) (BSI, Cryptographic Mechanisms, 2024) to be no shorter than 3000 bits. With the currently available computational power, GNFS is impractical when these key sizes are used.

Algorithm	Purpose	Best Classical Attacks	Recommended Parameter Sizes (2025)
Diffie-Hellman	Key Exchange	▪ Finite Fields: GNFS (subexponential) ▪ Elliptic Curves: Pollard's Rho (exponential)	▪ Finite Fields: ≥ 3072-bit prime p ▪ Elliptic Curves: ≥ 256-bit prime
RSA Encryption	Message Encryption	GNFS (subexponential)	≥ 3072-bit modulus N
Digital Signatures (DH/RSA-based)	Authentication, integrity, non-repudiation	▪ RSA: GNFS (subexponential) ▪ Elliptic Curve: Pollard's Rho (exponential)	▪ RSA: ≥ 3072-bit modulus N ▪ Elliptic Curves: ≥ 256-bit prime

Table 1: Classical Algorithms and Attacks

3.3 Protocols

Secure communication over public networks like the internet is facilitated by a coalescence of secret-key encryption systems (e.g. AES), public-key systems for key exchange (e.g. DH) and signatures (e.g. RSA), and cryptographic hash functions (e.g. SHA-2). They are combined in carefully designed protocols for securing internet-based communication (e.g., TLS, OpenVPN, IPSec), DNS information (DNSSEC), network authentication (e.g., Kerberos), email communication (PGP, S/MIME), remote logins (e.g., SSH), and so on.

As mentioned before, a sufficiently large quantum computer would pose devastating consequences for existing public-key cryptosystems and the protocols using them. The study and implementation of new, post-quantum public-key algorithms is therefore paramount to preserving the security of the internet. For this, it is also important to understand the working of

quantum algorithms and their interplay with cryptography. In the upcoming chapters, we introduce quantum computing and explain Shor's algorithm and its effects on classical public-key systems.

4_ Quantum Information Theory

Section summary: While classical physics describes the behavior of macroscopic objects, quantum physics governs the behavior of microscopic particles like electrons, photons, and atoms. Quantum-scale particles exhibit novel and counterintuitive properties, unbeknownst to classical objects. For instance, particles like electrons can exist in multiple states simultaneously (superposition), or several particles share an inseparable state (entanglement).
A qubit is the basic unit of quantum information and embodies these principles. A quantum computer is a machine that leverages the principles of quantum mechanics, operating on qubits, and using quantum logic gates to implement quantum algorithms.
Quantum computing's relevance to cryptography stems from the existence of a quantum algorithm, Shor's algorithm, which can break classical public-key cryptosystems in polynomial time, an impossible feat for known classical algorithms.
It is important to recognize that quantum computers are not inherently faster or more powerful than classical computers; their advantage depends on the specific task they are used for.

Up to this point, we have discussed classical attacks on modern-day cryptosystems and have concluded that large enough parameters are enough to prevent these. However, in a 1994 paper (Shor, 1994), the mathematician Peter Shor demonstrated a groundbreaking polynomial-time quantum algorithm to break these cryptosystems. A large enough quantum computer that can implement this algorithm for current cryptographic parameter sizes (a cryptographically relevant quantum computer, CRQC) does not yet exist. It is thus, so far, a theoretical attack. However, the possibility of its practical implementation in the future is real.

In this section, we give a high-level overview of some key concepts in quantum information theory.

4.1 Classical and Quantum Physics

Classical physics studies the behavior of macroscopic objects—the ones we see and interact with in everyday life. It is grounded in Newton's laws of motion and classical mechanics, where objects have definite positions, velocities, trajectories, and momenta, all of which can be measured with precision and certainty. Quantum physics, in contrast, examines the behavior of microscopic entities like atoms, electrons, and photons, whose motion defies the laws of classical physics. To address their mechanics, quantum physics was developed in the early 20th century by pioneers such as Max Planck, Niels Bohr, and Albert Einstein.

Quantum-scale particles exhibit novel and counterintuitive properties, unbeknownst to classical objects. For instance, particles like electrons can exist in multiple states simultaneously (**superposition**), or several particles can share an inseparable state (**entanglement**).

4.2 Classical and Quantum Computers

A classical computer is a computing machine grounded in the principles of classical physics, operating on the Turing Machine model introduced by Alan Turing (Turing, 1937). Classical computers process information as bits, which represent either a 0 or a 1. In hardware, these bits are physically encoded using mechanisms such as high/low voltage, current on/off states, or up/down magnetic polarization.

Classical computers execute classical algorithms, which consist of sequences of elementary, deterministic instructions to perform specific tasks. Even in parallel computing systems, tasks are decomposed into sequential operations for individual processors. Classical logic gates, such as AND, OR, and NOT, form the building blocks of classical computation, producing predictable outputs based on inputs and the current system state.

Quantum computers, in contrast, operate on principles derived from quantum mechanics. They use quantum bits, or **qubits**, as their fundamental unit of information and manipulate them through **quantum gates**.

4.3 Qubits

Like classical bits, a qubit has two basis states. These are denoted by $|0\rangle$ and $|1\rangle$. However, unlike classical bits, which can only exist in one of the two base states, qubits can simultaneously exist in a **superposition** of both the base binary states, i.e. a probabilistic composite state between the two. Thus, a general state of a qubit looks like $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers, whose magnitudes, when squared, add up to 1.

This non-deterministic compound state persists until the qubit is "measured" or "observed", i.e. until a process tries to determine its state. When the qubit is measured, it collapses from its superposition state into one of the basis states. Thus, on measurement, the qubit always provides a binary value. The probability that a qubit collapses from the state $\alpha|0\rangle + \beta|1\rangle$ into the state $|0\rangle$ (resp. $|1\rangle$) is $|\alpha|^2$ (resp. $|\beta|^2$).

Another important property of qubits is **entanglement**, which is a phenomenon where the quantum states of two or more particles become correlated in such a way that the state of one particle cannot be described independently of the state of the other(s). Under entanglement, the state of one qubit can be dependent on the state of another, no matter how far apart they are. The measurement of one qubit influences the value of all other qubits that are entangled with it. An example of an entangled two-qubit system is given by $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. When measured, both qubits yield the same value 0 or 1 with probability $\frac{1}{2}$.

Entanglement is a unique feature of quantum mechanics that allows for correlations between quantum systems, which cannot be explained by classical physics and cannot be replicated by classical systems.

4.4 Quantum Algorithms

Quantum algorithms are usually represented as circuits of **quantum gates**, analogous to classical logic gates. Quantum computers implement quantum gates to manipulate qubits by leveraging quantum phenomena such as superposition and entanglement. These gates form the foundation of quantum computation, enabling capabilities beyond those of classical logic gates.

Unlike classical systems, where computations yield deterministic outputs, quantum outputs are probabilistic, sometimes necessitating repeated executions to refine confidence in the outcome. Notable examples of quantum algorithms include Shor's algorithm for integer factorization and Grover's algorithm for unstructured search.

4.5 Theoretical Advantage and Limitations

Quantum and classical computers differ fundamentally in how they process information, but they are widely believed to be equivalent in terms of what they can compute. This idea is formalized by the Church-Turing thesis, which posits that any function computable by a physical machine can, in principle, be computed by a classical Turing machine. In other words, quantum computers do not extend the scope of what is computationally possible—they can solve the same problems as classical computers, though sometimes far more efficiently.

Further, quantum advantage is not universal – it applies only to specific classes of problems. Quantum computers are not inherently faster than classical computers for all tasks. In fact, even a basic task like parity checking (determining if the number of ones in the binary representation of a number is odd oreven) has been proven to not allow any asymptotic improvement when performed on a quantum computer (Beals, Buhrman, Cleve, Mosca, & de Wolf, 2001).

Nevertheless, quantum computers exhibit significant advantages for problems that exploit quantum phenomena. A prime example is **period-finding**, the core subroutine in Shor's algorithm for factoring large integers. Quantum computers solve this problem efficiently with the Quantum Fourier Transform (QFT), which leverages the quantum phenomena of superposition and parallelism. While classical algorithms for period-finding often require numerous iterations and become inefficient for large inputs, the QFT renders quantum algorithms an exponential speedup. Thus, quantum computers can evaluate the periodicity of a function with far fewer computational steps compared to their classical counterparts.

4.6 Shor's Algorithm

Shor's algorithm (Shor, 1994) is a quantum algorithm developed in 1994 by the mathematician Peter Shor. It provides an efficient method to solve the discrete logarithm problem, and can be adapted to find the prime factors of an integer, and, in fact, to solve the more general Hidden Subgroup Problem (of which discrete log and factorization are a special form) in an abelian group (Ettinger, Høyer, & Knill, 2004). It thus invalidates the security assumptions of both RSA and Diffie-Hellman cryptosystems in the quantum framework.

Consider the problem of integer factorization that forms the basis of the RSA public-key cryptosystems for encryption and signatures. These cryptosystems rely on the fact that factorizing the product of two large, soundly chosen primes is a difficult problem for a classical computer. The problem can be stated more formally as follows: given (the numeric value of) $N = pq$, a product of two primes p and q of large enough size, find the factors p and q.

The first phase of Shor's algorithm involves reduction of the factorization problem to a period-finding problem. The computations during this phase are all classical. The second phase involves using a quantum computer to compute the required period.

4.6.1 Reduction Phase

One chooses a number g randomly (smaller than N) and calculates $d = \gcd(g, N)$. If $d \neq 1$, then d gives a non-trivial factor of N. If $d = 1$, then it is known that there must exist some integer p such that N divides $g^p - 1$. The value p may be taken to be the smallest such exponent. If p is even this can be rewritten as $N|\left(g^{p/2} - 1\right)\left(g^{p/2} + 1\right) = g^p - 1$. Since p was the smallest such exponent, it cannot happen that $N|\left(g^{p/2} - 1\right)$. Thus, computing $\gcd\left(g^{p/2} + 1, N\right) \neq 1$ gives a factor of N. This may still be trivial, if it is equal to N. In this case, or if the p obtained is odd, one tries again with a different g. Within a few steps, one finds a factor successfully.

4.6.2 Period Computation Phase

The question remains as to the value of p is computed. Without elaborating on the details, it is important to remark here that the number p can mathematically be viewed as a period of the modular exponentiation function modulo the number N with respect to the "base" g. Thus, computing p is a matter of finding a period of a function. From here on, Shor's algorithm leverages the Quantum Fourier Transform (QFT) to compute the period of this function. Thus, Shor's algorithm reduces factoring to period-finding and then exploits the periodic behavior of the modular exponentiation function for quick period computation.

4.6.3 Practical Aspects

Shor's quantum algorithm is a polynomial time algorithm; so, in theory, it performs cryptographic-scale factorizations and discrete logarithm computations very efficiently. However, as mentioned earlier, a practical CRQC does not yet exist.

This is due to the various challenges involved in implementing a quantum computer. Chapter 5_ describes these challenges in greater detail and discusses the capabilities of today's quantum computers regarding the implementation of Shor's Algorithm in Section 5.5.

It is important to note the caveats present in known demonstrations of quantum factorization. Some methods demonstrated implement algorithms different from Shor's, which do not provide an exponential advantage over classical algorithms and often use classical methods for some steps. The known implementations of Shor's algorithm (e.g., (Vandersypen, et al., 2001), (Skosana & Tame, 2021)) do not implement the algorithm in its pure form, and rely on various shortcuts, simplifications, and classical assistance. For example, many demonstrations have relied on pre-selecting a valid base a for modular exponentiation, and some implementations have optimized circuit depth by replacing quantum modular exponentiation with classical precomputations. Despite this, the numbers known to be factored using Shor's algorithm on a quantum computer are merely a few digits long and therefore far off from cryptographic scale.

5_ Quantum Computers

Section summary: The development of practical quantum computers has advanced significantly since the theoretical foundations of quantum computing were established in the late 20th century. The capabilities of a quantum computer are measured across several dimensions, with the number of physical qubits being a primary factor. While quantum computers with a few hundred physical qubits have been built, a cryptographic relevant quantum computer is estimated to require millions of physical qubits.
Scaling quantum systems to this level presents substantial engineering challenges. As the number of qubits increases, maintaining precise control over quantum states becomes increasingly complex. These difficulties contribute to high error rates and instability, limiting the size and reliability of current quantum hardware. The timeline for the realization of a practical, cryptographically relevant quantum computer is a matter of large debate and speculation. While predictions vary, many experts consider a timeframe of one to two decades to be a realistic projection based on current trends.

Over the last few decades, the materialization of practical quantum computers has been a subject of intense research and speculation. The early theory of quantum computing emerged about four decades ago, with the foundations laid by researchers like Richard Feynman, David Deutsch, Peter Shor, Lov Grover, and Paul Benioff, and Gills Brassard.

Shor's algorithm was proposed in 1994. The early 2000s saw experimental progress in the implementation of basic quantum operations and algorithms. This was done using small-scale quantum systems, such as trapped ions and superconducting qubits. During this era, quantum error correction techniques were also developed to mitigate errors caused by noise and decoherence.

In the mid-2010s, major advancements were made by companies like IBM, Google, and Rigetti in building larger and more stable quantum hardware. Currently, several efforts are underway to scale up quantum systems and to improve hardware reliability and fault-tolerance.

5.1 Architecture and Implementations

The hardware architecture of quantum computers is fundamentally different from that of classical computers. At the core is the Quantum Processor Unit (QPU), composed of qubits implemented using advanced quantum technologies. To implement quantum algorithms on quantum computers, a specialized **quantum assembly language (QASM)** that defines operations on qubits is used. Higher-level quantum programming frameworks, such as IBM's Qiskit or Google's Cirq, translate quantum algorithms written in Python-like syntax into QASM.

The implementation of qubits differs between different systems. Superconducting qubits, employed by IBM (IBM Quantum, 2024) and in Google (Google Quantum AI, 2023), use circuits cooled to near absolute zero to maintain quantum states. Semiconductor qubits, used by Intel (Intel Corporation, 2024), employ silicon to trap and control electron spins. Trapped ion systems, like those developed by IonQ, hold single ions in electromagnetic fields and manipulate them with lasers. Other approaches include photonic qubits, which use particles of light, and topological qubits, which rely on exotic particles for increased error resistance. The physical constraints for the different implementations of qubits requires advanced manufacturing and engineering capabilities.

Quantum systems also lack persistent memory for qubits; instead, they store results in classical systems after measuring the qubit states, as quantum states are fragile and short-lived.

5.2 Physical and Logical Qubits, Quantum Error Correction

Quantum circuits (see Section 4.4) are designed to act on a certain number of **(logical) qubits**. To reliably run such a quantum circuit, today's quantum computers need many times more **physical qubits**.

This is because physical qubits are susceptible to various noise sources and errors in control operations, which cause errors in quantum computations. Achieving high-fidelity operations on physical qubits is a major challenge in quantum computing.

To address these challenges, the information of a logical qubit is spread into several physical qubits. Fault-tolerant quantum operations are then enabled by protecting quantum states from errors through **quantum error correction** (QEC).

QEC allows errors in quantum states to be both detected and corrected using a process called syndrome measurement, under which measurements are made on certain subsets of qubits rather than on the whole system to reveal the type and location of errors in the state. Error correction operations are then applied to revert the quantum state to its desired encoded form.

5.3 Measuring the Capabilities of a Quantum Computer

The scale or computational power of a quantum computer is measured in terms of various factors. While the number of qubits is an important parameter, it does not by itself sufficiently describe its capabilities and performance. Below, we describe some of the most important parameters in measuring the quality of a quantum computer.

- **Number of qubits** represents the amount of information that a quantum computer can describe, and thus, typically, the complexity of possible calculations. With more qubits, a quantum computer has the potential to tackle larger computational problems, especially those that involve many variables or states. Note that this number commonly refers to the number of physical qubits and does not easily translate to the number of logical qubits. The implementation of a logical qubit may (depending on the architecture) comprise up to 1000 physical qubits.
- **Connectivity** refers to the ability of qubits to interact with each other during gate operations. When qubits are fully connected, they can execute algorithms more efficiently, solving problems with fewer steps. Connectivity is a prerequisite for creating entanglement between qubits. The choice of physical qubit architecture and qubit coupling mechanisms determines the connectivity of a quantum computer.
- **Gate fidelity** refers to the accuracy and precision with which quantum gates perform their intended operations. Gate fidelity is reduced by noise and decoherence incurred during gate operations. The fidelity measures the divergence between the final ideal state (the formal result of the mathematical operations) and the real state after the application of a sequence of gate operations. Lower gate fidelities increase the number of errors, necessitating more frequent error correction and potentially more complex quantum error correction (QEC) codes.
- **Coherence time:** A quantum system is said to be coherent when it exists in a well-defined superposition of states, meaning it simultaneously occupies multiple states with specific probabilities. Decoherence is the loss of coherence in a quantum system due to interactions with its environment. In decoherence, superpositions of states collapse into classical-like states, and the quantum behaviour breaks down. Quantum coherence is essential for implementing quantum algorithms, since it allows for the manipulation and control of quantum states. Coherence time is the duration over which a qubit retains its quantum state before decoherence.

5.4 Challenges

Scaling a quantum computer to a large number of qubits while maintaining sufficiently high gate fidelity, connectivity, coherence time, error correction rates, and reliability, has proven a major challenge. As the number of qubits increases, it becomes more complex to control and manipulate them, leading to increased susceptibility to errors, and lowering the gate fidelity and coherence. Quantum error correction requires additional qubits and gates, further straining coherence and fidelity.

Maintaining the right balance of connectivity and coherence time is a demanding task. An optimal qubit interaction designs allow for efficient quantum operations and entanglement and yet

also minimize noise and decoherence. Without sufficient isolation from its environment, a qubit suffers from quantum decoherence, which introduces noise into calculations. However, perfect isolation is also undesirable because some degree of qubit connectivity is needed during quantum computations. Extending coherence times involves addressing multi-fold material properties, temperature control, and isolation from external noise. Thus, maintaining sufficient qubit connectivity simultaneously with low error rates as the number of qubits increases is a demanding problem.

Quantum computing platforms also have physical constraints, like the choice of qubit technology (e.g., superconducting qubits, trapped ions) and the need to maintain low temperatures with cryogenic cooling. Maintaining high yields and reproducibility across large numbers of qubits, which is a requirement for building large-scale quantum computers, requires advanced manufacturing and engineering capabilities.

5.5 Present Day and Outlook

As of March 2025, quantum computers with tens to hundreds of physical qubits have been demonstrated by various research groups and companies. Table 2 lists some of today's quantum computers and the number of physical qubits for illustration purposes. Note though, that as we have seen in Section 5.3, the number of physical qubits alone is not sufficient to compare the performance of the different quantum computers.

Quantum Computer	Manufacturer	Year	Physical Qubits
IBM Quantum System One	IBM	2019	20
IBM Eagle	IBM	2021	127
IBM Osprey	IBM	2022	433
IBM Condor	IBM	2023	1121
Sycamore	Google	2019	54
Willow	Google	2024	105
Aspen-M	Rigetti	2021	80
H1 Series	Honeywell (Quantinuum)	2020	10
Zuchongzhi 2.1	University of Science and Technology of China (USTC)	2021	66
Zuchongzhi 3	University of Science and Technology of China (USTC)	2025	105

Table 2: Scale of selected existing quantum computers

These computers can perform simple quantum simulations, basic optimization problems, and factoring small integers. However, existing quantum computers are far off from the scale needed of a CRQC. While existing claims of quantum supremacy have drawn significant attention to the discipline, these are demonstrated on niche, contrived tasks, for which near-term practical use cases are limited.

To implement Shor's algorithm with the parameters used in practical cryptosystems, quantum computers with **thousands of stable, error-corrected logical qubits are estimated to be required** (Campbell, Terhal, & Vuillot, 2017) (Kudelski Security, 2021). Using common error-correction codes like the surface code (~1000 physical qubits per logical qubit), **this amounts to millions of physical qubits** (Campbell, Terhal, & Vuillot, 2017).

In a 2021 paper (Gidney & Ekerå, 2021), the authors estimate the use of 14586 logical qubits, each of which covers 1568 physical qubits, thus yielding a total of approximately 23 million physical qubits to factor an RSA-2048 number. For this, they assume a physical gate error rate of 10^{-3}, which is typically considered acceptable and achievable in the present. They estimate a running time of about 5 hours for a single run of factorization, and a probability of 99% that this run results in success.

There is a wide variation between expert opinions on when practical quantum computers will be realized. Some researchers and industry leaders believe that large-scale, fault-tolerant practical quantum computers will emerge within the next decade or two. Yet, some others are less optimistic, foreseeing significant challenges, leading to a much longer timeline. In 2024, the German Federal Office for Information Security (BSI) estimated that the conservative end was in 16 years (BSI, Entwicklungsstand Quantencomputer, 2024). In 2022, the BSI formulated recommendations for high-security requirements based on the worst-case hypothesis that cryptographically relevant quantum computers will become available in the early 2030s (BSI, Quantum-Safe Cryptography, 2022).

6_ Post-Quantum Cryptography

Section summary: Post-quantum cryptography is the field of study of cryptosystems that are resistant to attacks by both classical and quantum algorithms. Post-quantum cryptographic schemes base their security on mathematical problems for which all known classical and quantum algorithms have no efficient solution, e.g. lattice problems, the difficulty of decoding general linear codes, or the one-wayness of cryptographic hash functions.
In 2016, NIST (National Institute of Standards and Technology) launched the Post-Quantum Cryptography Standardization program (NIST, 2017) to standardize quantum-secure cryptographic schemes for key encapsulation and digital signatures. This resulted in three standards so far: FIPS 203 for key encapsulation (lattice-based), and FIPS 204 (lattice-based) and FIPS 205 (hash-based) for digital signatures. The HQC KEM scheme has also been selected for standardization. Further schemes will presumably be standardized.
In lattice-based cryptography, key encapsulation and digital signature schemes are built relying on the difficulty of solving mathematical problems in high-dimensional lattices. Some prominent examples of such problems are the LWE (Learning with Errors) problem and Shortest Vector Problem (SVP). In hash-based cryptography, hash functions can be used to construct digital signature schemes whose security relies on the collision resistance and preimage resistance of the underlying hash functions.

In general, to achieve comparable bit security levels to classical algorithms, post-quantum schemes require larger public and private key sizes and generate larger outputs.

Post-quantum cryptography is the study of (public-key) cryptosystems that are resistant to attacks by both classical and quantum algorithms. Under this field, alternative constructions for key encapsulation mechanisms (KEM) and signature schemes are researched, which base their security on entirely different mathematical problems whose solution does not have any known quantum speed-up.

Several different mathematical frameworks have been explored to construct quantum-resistant cryptographic systems: some of these are lattice-based cryptography, code-based cryptography, hash-based cryptography, multivariate cryptography, and isogeny-based cryptography. Most of these theoretical frameworks existed before the context of cryptography and were studied as topics in mathematics and computer science. However, they gained major traction after their pertinence to cryptography was realized.

6.1 NIST Post-Quantum Cryptography Standardization program

In 2016, NIST (National Institute of Standards and Technology) launched the Post-Quantum Cryptography Standardization program (NIST, 2017) to standardize potentially quantum-secure cryptographic primitives. Under this program, public submissions could be made for candidate protocols, and all material on the candidates was published online, open to analysis and attack attempts by other researchers.

Many candidates were broken, e.g. the multivariate and isogeny-based submissions, or discarded for other reasons such as efficiency or practical implementation. Lattice-based and hash-based systems have emerged to standardization as primary algorithms, with a code-based primitive selected as an alternative.

In August 2024, after three rounds of evaluation, NIST released the final versions of three Federal Information Processing Standard (FIPS) documents—FIPS 203 (NIST, FIPS 203, 2024), 204 (NIST, FIPS 204, 2024), and 205 (NIST, FIPS 205, 2024)—which represent the first set of post-quantum encryption standards. In March 2025, the standardization of HQC KEM was announced. While additional digital signature schemes are still being evaluated for standardization, the selection for KEMs has been concluded (NIST, 2025)

The following table summarizes the most important post-quantum key encapsulation (KEM) algorithms, evaluated by the NIST PQC standardization process.

Algorithm	**Use Cases**	**Framework**	**Status**
CRYSTALS-KYBER (ML-KEM)	Primary algorithm for most use cases	Module-Lattices	Standardized in FIPS 203 under the name ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism)

Algorithm	Use Cases	Framework	Status
BIKE	Avoid reliance only on lattices	Code-based	Not selected for standardization by NIST
HQC	Avoid reliance only on lattices; backup for ML-KEM	Code-based	Selected for standardization; Initial version of the standard being drafted
Classic McEliece	Avoid reliance only on lattices	Code-based	Not selected for standardization by NIST

Table 3: NIST finalists for KEM

The following table summarizes the most important post-quantum digital signature algorithms, evaluated by the NIST PQC standardization process.

Algorithm	Use Cases	Framework	Status
CRYSTALS-Dilithium (ML-DSA)	Primary algorithm for most use cases	Module-Lattices	Standardized in FIPS 204 under the name ML-DSA (Module-Lattice-Based Digital Signature Algorithm)
SPHINCS+ (SLH-DSA)	Avoid reliance only on lattices	Stateless Hash	Standardized in FIPS 205 under the name SLH-DSA (Stateless Hash-Based Digital Signature Algorithm). Disadvantages: large signature size, performance.
FALCON	Cases where Dilithium signatures are too large	NTRU lattices	Selected for standardization, Initial version FIPS 206 being drafted

Table 4: NIST finalists for signatures

It is noteworthy that some potential candidates that did not advance to the final rounds of the NIST competition were excluded due to considerations like efficiency, practical implementation, and emphasis on diversity of algorithm frameworks, rather than security concerns. For example, NTRU Prime is a lattice-based KEM candidate that was not selected for the fourth round but remains free from any practical attack. It has been chosen for implementation in OpenSSH 9.0 (Friedl, Mojzis, & Josefsson, 2023) (OpenSSH Project, 2024).

Similarly, FrodoKEM was ruled out of the NIST standardization program based on performance and efficiency. FrodoKEM-976 and FrodoKEM-1344 have been recommended by the German Federal Office for Information Security (BSI) as cryptographically suitable for long-term confidentiality (BSI, Cryptographic Mechanisms: Recommendations and Key Lengths, 2024). Further, the Crypto Forum Research Group within the Internet Engineering Task Force has standardized two other stateful hash-based signature schemes (XMSS (IRTF, XMSS RFC, 2018) and LMS/HSS (IRTF, LMS RFC, 2019)).

6.2 Lattice-Based Cryptography

In lattice-based cryptography, public-key cryptosystems are built relying on the difficulty of solving mathematical problems in high-dimensional lattices. Lattices are geometric structures formed by a set of points arranged in a regular, infinitely repeating pattern in a multi-dimensional space. Figure 9 shows an example on how points are arranged in a three-dimensional lattice.

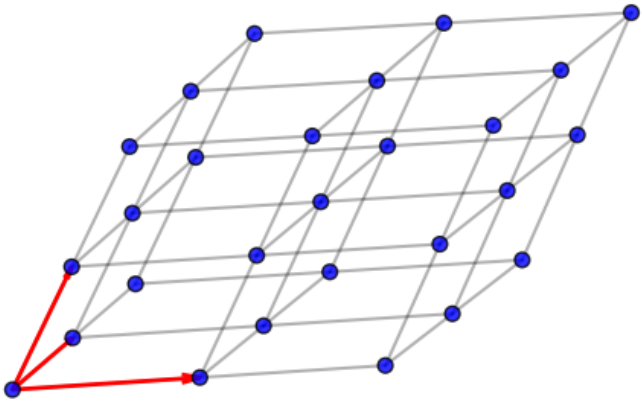

Figure 9: A three-dimensional lattice (It extends in all directions. Red are basis vectors.)

Although a lattice contains infinitely many points, it can be described by a finite representation called a "basis" (plural bases), which is a minimal set of points (called basis vectors) that can be combined in different ways to get every point on the lattice. Every lattice (of dimension ≥ 2) has infinitely many representations in terms of bases.

Lattices provide a rich mathematical structure which offers several algorithmic problems with strong hardness proofs. Additionally, they are believed to be resistant to quantum algorithms. This means that presently known quantum algorithms, such as Shor's algorithm, do not offer substantial advantages in their solution.

Two prominent examples of such problems are the **Shortest Vector Problem (SVP)**, which involves finding the shortest non-zero vector in a lattice and the **Closest Vector Problem (CVP)** which asks for the closest lattice point to a given arbitrary point in the multidimensional state.

The first attempts to construct lattice-based public-key encryption involved hiding the plaintext as the solution to a lattice problem (Goldwasser, 1997). The idea was that only a recipient knowing the optimal basis (the so-called "trapdoor") was able to solve the problems. Only a "bad" basis of the lattice was made public.

Many of the proposed algorithms in the NIST competition (e.g. ML-KEM, ML-DSA) are based on **LWE (Learning with Errors)** problems, another important class of lattice-based problems. LWE involves solving noisy linear equations to recover the underlying secret variables. Itis defined as follows: Given a matrix A, a vector s, a vector e with small, random errors, and the equation $\mathrm{b} = \mathrm{As} + \mathrm{e}$, all over a finite field, find s given the values of A and b. LWE can be formulated as a lattice problem by defining a lattice whose basis is given by rows of the matrix A.

The security of the NIST candidates NTRUEncrypt (Hoffstein, Pipher, & Silverman, 1998) and its refinement NTRU Prime (Bernstein, et al., 2017) relies on the hardness of finding short vectors in lattices formed by polynomial rings.

Lattice-based problems are closely connected to *NP-hard* problems, which is a class of problems strongly believed to be computationally difficult and to have no general polynomial-time solution. In general, the hardness conjectures for lattice and coding problems are much stronger than those for factoring and discrete logarithms, making them a strong foundation for cryptography.

6.3 Code-based Cryptography

The branch of code-based cryptography utilizes the mathematical structure of error-correcting codes, which were originally developed to facilitate reliable data transmission in noisy communication channels. Error-correcting codes encode information in a way that allows the receiver to detect and correct errors introduced during transmission, ensuring the accurate reconstruction of the original message. The security assumptions of code-based cryptography are based on the hardness of decoding general linear codes, which is proven to be an NP-hard problem.

Three code-based KEM schemes were in the fourth evaluation round of the NIST standardization program: BIKE (Aragon, et al., 2017), HQC (Aguilar Melchor, et al., 2017), and Classic McEliece (Albrecht, et al., 2022). Of these, HQC has been selected for standardization (NIST, 2025).

It is noteworthy that the McEliece cryptosystem was presented already in 1978 but rarely used to due to large key sizes. Its main idea is to encrypt the plaintext by encoding it using an easily decodable code and adding some error. The structure of the code is kept secret, so that the resulting message is hard to decode for anyone without the knowledge of the decoding algorithm. Encryption can be done by anyone, while decryption is possible only with knowledge of the secret structure of the code.

The early lattice-based cryptosystems mentioned above were inspired by this idea. As with lattice problems, no quantum algorithm is known to give a substantial advantage to breaking code-based cryptosystems.

6.4 Hash-based Cryptography

Hash functions can be used to construct digital signature schemes. The idea behind these schemes is simple. First assume that one wants to sign a single bit (0 or 1). The private key consists of two random strings, r_0, corresponding to the message 0, and r_1, corresponding to the message 1. The public key is the pair of hash values $\mathrm{H}(\mathrm{r}_0)$ and $\mathrm{H}(\mathrm{r}_1)$, which are published along with their correspondence with the bits. To sign the message bit b (0 or 1), the signer reveals the random string corresponding to it. The signature can be verified by computing the hash value of the random string and comparing it with the published value.

To sign a longer bitstring, this principle can be extended by signing each bit individually, or in small blocks of bits. A challenge with this approach is that the public key size grows rapidly with

the message size. To address this, hierarchical structures called Merkle Trees are used to efficiently store the hashes. The root hash, representing the entire tree, is published as the public key.

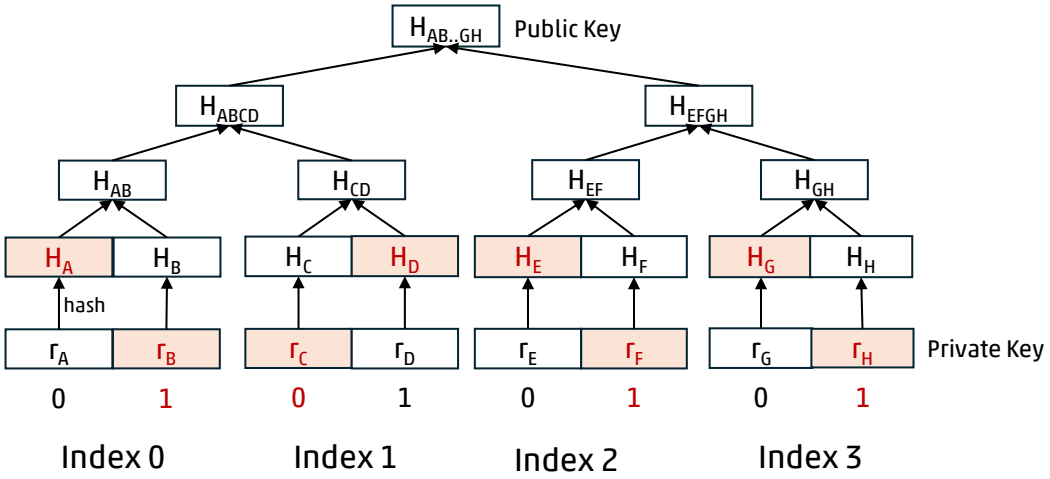

Figure 10: Merkle Tree for hash-based signature scheme. The signature of 1011 is given by the red boxes. It can be verified by computing the root of the tree and comparing it to the public key.

One differentiates between **stateful** and **stateless** hash-based signature schemes.

- In stateful signature schemes, the private key consists of multiple pairs of random strings, each used for a single bit or block of a message (as in the example above). Each pair can only be used once. The signer must carefully track the "state", i.e. which private key strings have already been used to ensure no part is reused. If the same pair is reused in signing multiple messages, an attacker can observe the revealed strings, reconstruct parts of the private key, and potentially forge signatures for unauthorized messages. Since stateful schemes require careful tracking of each key usage, they are suitable for controlled environments (e.g., a central server) that can maintain the state. Examples of stateful signature schemes include XMSS (eXtended Merkle Signature Scheme) (Hülsing, 2018) and LMS (Leighton-Micali Signature Scheme) (McGrew, 2019).
- In stateless schemes, there is no need to track private key usage. These schemes circumvent the problem of reusing private key parts by using few-time signature (FTS) schemes instead of one-time signatures (OTS) like the example discussed above. In FTS, each private key part is used only a small number of times, and the key selection is suitably randomized for signature generation, thereby minimizing the probability of reusing the same part. The standardized SLH-DSA scheme is an example of such a signature scheme.

The security of hash-based signatures relies on the collision resistance and preimage resistance of the underlying hash functions. These schemes are considered quantum-resistant because no efficient quantum algorithms are known to break modern cryptographic hash functions.

6.5 Security Levels and Key Sizes

As part of PQC Standardization process, NIST has defined a framework of security levels to measure the strength of cryptographic algorithms against both classical and quantum attacks. These levels are inspired by the security provided by symmetric encryption schemes like AES and are designed to benchmark the computational difficulty of breaking an algorithm. NIST specifies five security levels, with Level 1 (128-bit security) offering protection equivalent to AES-128, requiring 2^{128} operations to break. Level 3 (192-bit security) and Level 5 (256-bit security) correspond to AES-192 and AES-256, respectively. Intermediate levels, such as Level 2 (equivalent to collision search on SHA256) and Level 4 (equivalent to collision search on SHA384), are less commonly used but provide additional flexibility for specific applications (NIST, Security (Evaluation Criteria), 2017).

These security levels may be used to evaluate and compare both post-quantum and classical cryptographic algorithms. For example, in the face of classical attacks, RSA-2048 offers ≈112-bit security, RSA-3072 provides ≈128-bit security, and the P-256 elliptic curve gives ≈128 bits of security for ECDH/ECDSA. Post-quantum algorithms, such as Kyber (Avanzi, et al., 2021) and Dilithium (Ducas, et al., 2017) are designed with variants targeting Levels 1, 3, and 5, ensuring they match or exceed the strength of classical cryptosystems.

The following tables list the approximate security levels, public and private key sizes, and the size of the outputs (e.g., ciphertexts or signatures) for some of the discussed classical and post-quantum cryptographic schemes. The values for the PQC schemes have been taken from the Open Quantum Safe website (Open Quantum Safe Project, 2023).

6.5.1 KEMs

Algorithm	Bit Security (approx.)	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext size (bytes)
RSA-3072	128	384	384	384
RSA-15360	256	1920	1920	1920
ML-KEM-512	128	800	1632	768
ML-KEM-1024	256	1568	3168	1568
HQC-128	128	2249	2305	4433
HQC-256	256	7245	7317	14421
Classic McEliece-348864	128	261120	6492	96

Table 5: Parameter sizes for KEMs. The ciphertext encapsulates a secret-key of 32 bytes.

One can see that the newly standardized lattice-based KEMs (ML-KEM) have parameter sizes comparable to RSA, while the size of the public key for Classic McEliece is much larger (see also Figure 11). This size is very practically relevant, since the public key needs to be sent during protocols. For example, during the TLS handshake, it is sent by the server within its certificate.

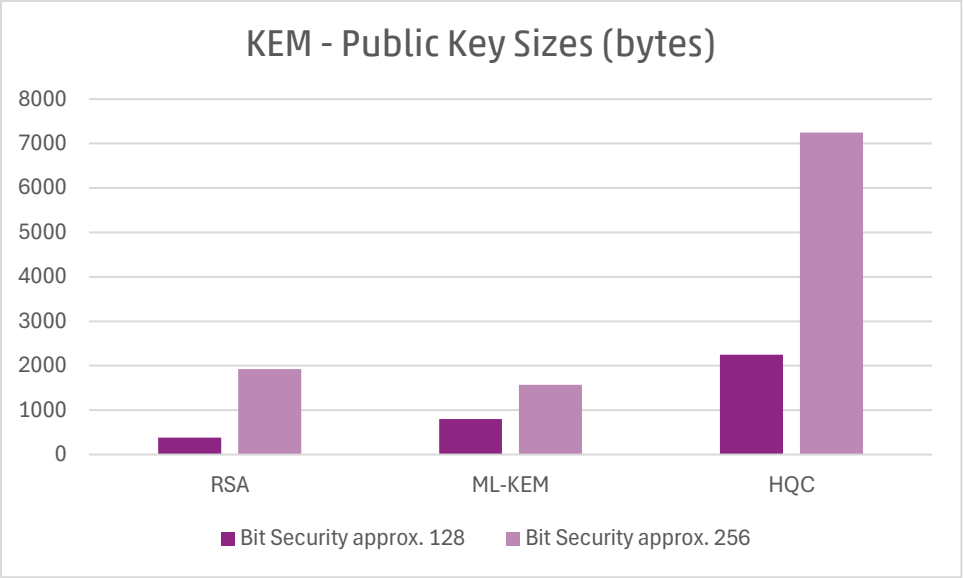

Figure 11: Public key sizes for KEMs

6.5.2 Signature Schemes

Algorithm	Bit Security (approx.)	Public Key Size (bytes)	Private Key Size (bytes)	Signature Size (bytes)
RSA-3072	128	384	384	384
RSA-15360	256	1920	1920	1920
ECDSA (P-256)	128	64	32	64
ECDSA (P-521)	256	131	66	132
ML-DSA-44	128	1312	2560	2420
ML-DSA-87	256	4896	4896	4627
SLH-DSA-SHA2-128s	128	32	64	7856
SLH-DSA-SHA2-256s	256	64	128	29792
Falcon-512	128	897	1281	752

Table 6: Parameter sizes for signature schemes

The table indicates that the newly standardized signature schemes generate larger signatures than classical schemes (see also Figure 12), with hash-based schemes producing the largest signatures. Additionally, the public key size of ML-DSA is significantly larger (see also Figure 13), which in turn increases the size of the corresponding certificate. In protocols like TLS, larger certificates and signatures increase transmission overhead and affect performance.

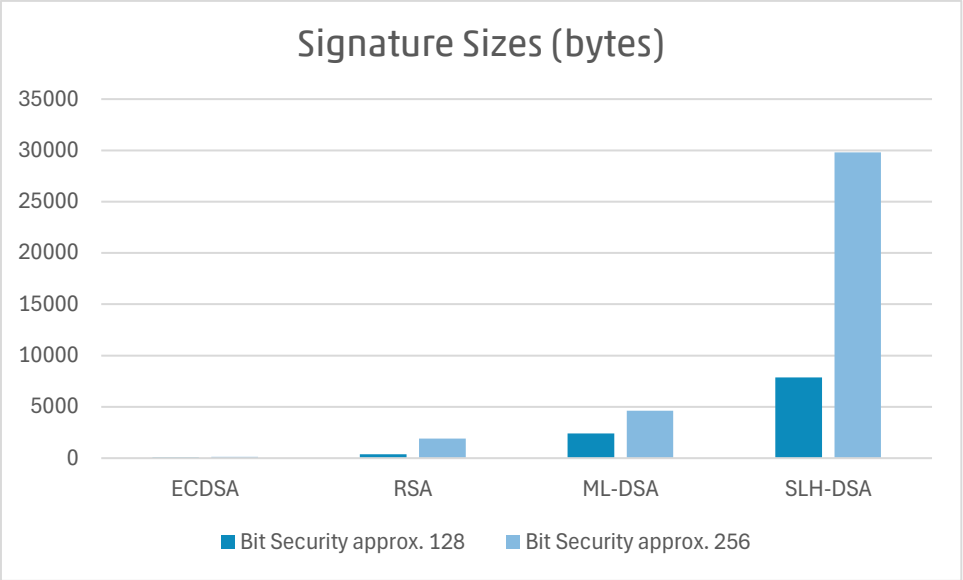

Figure 12: Comparison of signature sizes for different signature schemes

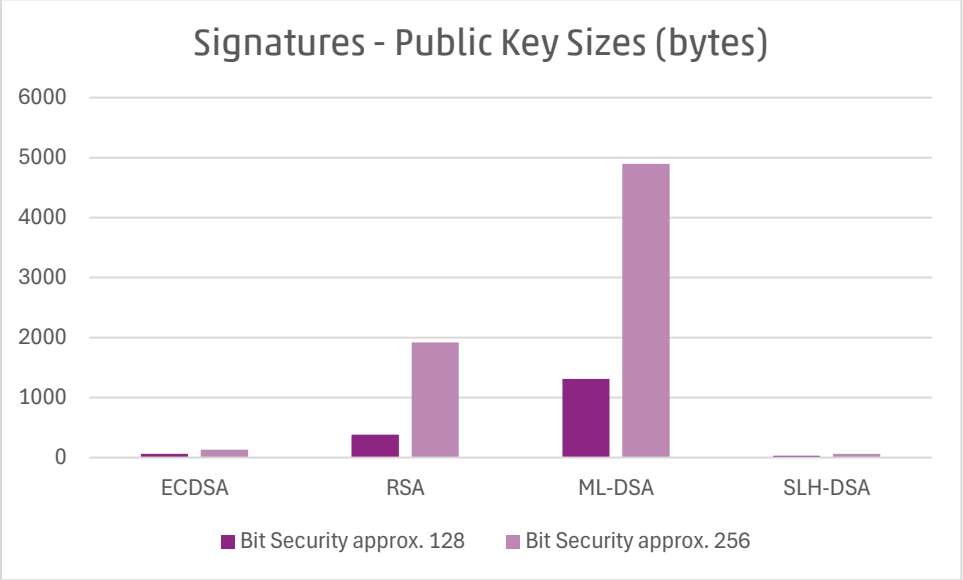

Figure 13: Comparison of public key sizes for different signature schemes

7_ Present-day Consequences for IT Security

Section summary: Although a cryptographically relevant quantum computer (CRQC) may still be a decade or more away, its impact on information security is already unfolding. The "store-now-decrypt-later" threat means that sensitive data requiring long-term confidentiality must *immediately* be protected using quantum-resistant encryption.
Not all use cases require immediate migration—some applications, such as user authentication, or electronic signatures may transition later. However, systems that cannot be easily updated risk becoming insecure when quantum threats materialize and must already be designed with post-quantum security in mind
Industry leaders such as Google, Cloudflare, AWS, and IBM have already begun integrating PQC into their products, marking the first wave of early adopters. Governments and cybersecurity agencies, including NIST, ANSSI, BSI, and ETSI, are actively promoting PQC migration strategies. Hybrid cryptographic schemes, which combine classical and post-quantum algorithms, are emerging as a bridge in the migration from the classical to the post-quantum cryptography.
To mitigate the quantum threat, organizations must understand the risks, assess and inventory cryptographic assets, classify them based on urgency, and subsequently develop an organization-specific post-quantum strategy to prepare for a full transition. Cryptographic implementers, including software and infrastructure providers, must integrate PQC and hybrid algorithms into security protocols like TLS, SSH, and VPNs and upgrade PKI infrastructures.

7.1 Relevance of the Quantum Threat

Public-key cryptographic algorithms are used extensively in the various communication protocols and trust frameworks across the internet. They protect data confidentiality and integrity, identity authentication, and several public trust-based frameworks like PKI, DNSSEC, and BGPSEC. They are also a crucial component in the verifiability of genuine hardware and software.

7.1.1 Confidentiality

Cryptography protects the confidentiality of data at rest and in transit. Although CRQCs may still be a decade or more away, their potential impact on key exchange and data encryption is already relevant today. This is due to the risk of **"store-now-decrypt-later"** attacks—where encrypted data is stored now to be decrypted later with quantum technology. The theft of encrypted data can take place during transport or at rest, but in general, data is more at-risk during transport. One example is diverting internet traffic through a border gateway protocol hijack. It has been speculated that some actors have already initiated harvesting mass amounts of encrypted sensitive data with the intention to mount this attack in the future.

Consequently, data that requires long-term confidentiality (beyond the advent of a CRQC) must already be encrypted using quantum-resistant algorithms today. As the advent of a CRQC approaches, the need for post-quantum encryption grows to more and more classes of data.

Mosca's theorem (Mosca, 2018) quantifies the relationship between the required security shelf-life (how long the data must stay confidential, say x), the collapse time (time until a quantum computer breaks the cryptography, say z), and the migration time (the duration to deploy a quantum-safe framework, say y. It states that if $x + y > z$, then the data is at serious risk.

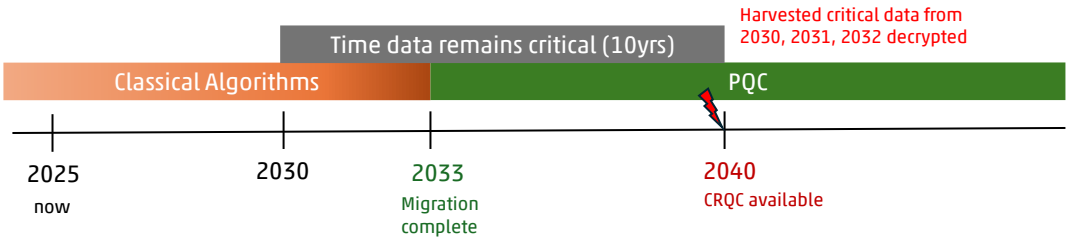

Figure 14: An illustration of Mosca's Theorem

For example (see Figure 14), assume that a quantum computer arrives in 2040, 15 years from now, ($z = 15$), that keys need to be secure for 10 years ($x = 10$), and the migration time is 8 years ($y = 8$). The migration will therefore be complete only in 2033. Now, keys generated in 2032 must be secure until 2042. The adversary may store the encrypted data in 2032 and decrypt and compromise it in 2040.

7.1.2 Authentication and Integrity

Various authentication mechanisms make use of public-key cryptography. These include live user or machine authentication protocols, like challenge-response protocols, as well as static, long-term digital signatures on critical data like firmware updates, certificates, DNS data (DNSSEC) and BGP IP prefixes. Digital signatures, in general, require an infrastructure (e.g., digital certificates) that ties entities to their public keys. Often, this association is valid for long periods of time, in-built into various systems, and difficult to update or replace.

In general, authentication systems are not prone to store-now-decrypt-later attacks. Instead, their vulnerability to quantum computers comes from their reliance on classical signature algorithms and classical public key infrastructure. When a CRQC becomes available, an adversary can forge signatures corresponding to any classical public key and therefore impersonate any entity. Thus, it is crucial that classical signatures stop being trusted when a CRQC is available.

This is already an issue to be addressed today, because several applications integrate public keys in immutable and long-term ways. One example of this is hard-coded certificates identifying root certificate authorities (CAs), which are often valid for a decade or more.

Another good example of this challenge is firmware updates. Firmware is a special kind of software embedded into hardware devices to help them operate effectively by abstracting their functionality. Hardware manufacturers regularly release firmware updates, so their devices remain secure and compatible with new media. Attacks on firmware can end up compromising the entire connected hardware device.

To ensure that firmware installation and updates only come from the trusted vendor, firmware signatures are used. The software vendor digitally signs the firmware image using a private key,

whose public key is somehow embedded into the hardware device or protected from tampering in some other way. Prior to installation, the firmware signature is cryptographically verified to ensure authenticity.

Since such public keys are often embedded ("burnt ") into the hardware and unchangeable, migrating firmware signatures to use post-quantum algorithms poses a challenge that must be tackled immediately. In other words, newly produced devices must already be embedded with post-quantum public keys for signatures.

Apart from digital signatures, data authenticity and integrity are facilitated through hash functions, symmetric-key MACs and authenticated encryption algorithms like AES-GCM. As for symmetric encryption, the key, tag, and digest sizes for these must already be adjusted for long-term use cases. To maintain the same bit security levels in the face of Grover's algorithm, they must be doubled.

7.1.3 Non-repudiation

Non-repudiation is a security property that ensures a party cannot deny the authenticity of their signature on a document, the origin of a transaction, or the commitment to an agreement. In the digital realm, non-repudiation is typically achieved through **electronic signatures**, which are implemented using digital signature algorithms that rely on public-key cryptography. Electronic signatures are legally recognized in many countries. Many electronic signatures are tied to critical legal or financial documents, such as contracts or wills, which must remain valid for decades.

However, once a CRQC becomes available, an attacker can use it to forge digital signatures tied to classical public keys, rendering these signatures invalid and undermining their non-repudiation guarantees. This creates a significant legal and operational challenge, as signatories may newly deny having signed a document or authorized a transaction, on the pretext that the signature was forged with a CRQC.

One mitigation strategy is the use of quantum-resistant attestations including a timestamp. By applying such an attestation to a signature before the advent of CRQCs, organizations can establish a verifiable record of the signature's creation time. This protects the signature's integrity and non-repudiation, as an attacker would need both to forge the signature and the attestation.

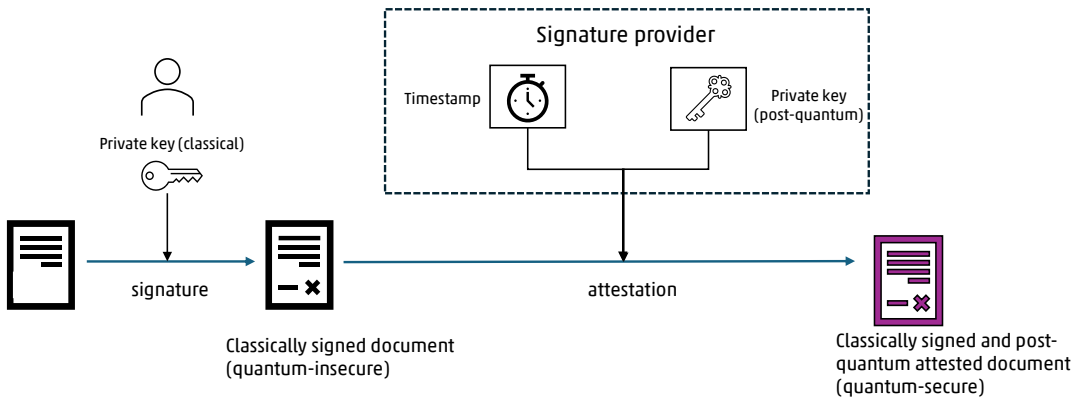

Figure 15: Quantum-secure attestation of electronic signatures

While quantum-resistant timestamp-based attestation provides a solution to port existing signatures into the quantum era, it does not eliminate the need for a broader migration to quantum-safe electronic signatures. To safeguard the long-term validity of electronic signatures, organizations must begin adopting post-quantum digital signature schemes.

7.2 Deployment of new PQC Standards

NIST has emphasized that the three new cryptography standards should be adopted immediately for most applications, as they represent the primary defense against future quantum threats. The adoption of quantum-safe cryptographic algorithms is not a simple, one-time replacement of current systems. It is a multi-layer, step-by-step, and gradual process. Organizations must assess risks, catalogue cryptographic assets, and classify data to identify long-term security needs. The security of public key infrastructures must be re-engineered to support quantum-resistant primitives.

7.2.1 Hybrid Schemes

While the NIST standardization program has built a good deal of trust in the security of the newly standardized protocols, it is important to account for the fact that these have been around for far less time than their classical counterparts. While the new schemes have been subjected to thorough analysis by researchers, they are yet to prove resilience to practical and potentially implementation-based attacks. Hybrid cryptography, which involves combining classical and post-quantum cryptographic schemes, is meant to serve as a bridge in the migration from the classical to the post-quantum cryptography.

The goal of hybrid schemes is to remain secure as long as one of the underlying schemes remains secure. The hybrid system is thus expected to remain secure from both quantum and classical adversaries. Even if a novel classical attack on the post-quantum component were to arise, the system remains secure from classical adversaries.

The combination of the two schemes is done in a relatively straightforward and intuitive way. For key encapsulation mechanisms, the outputs from the two algorithms are fed into a key derivation function to produce the final key which is used for symmetric encryption.

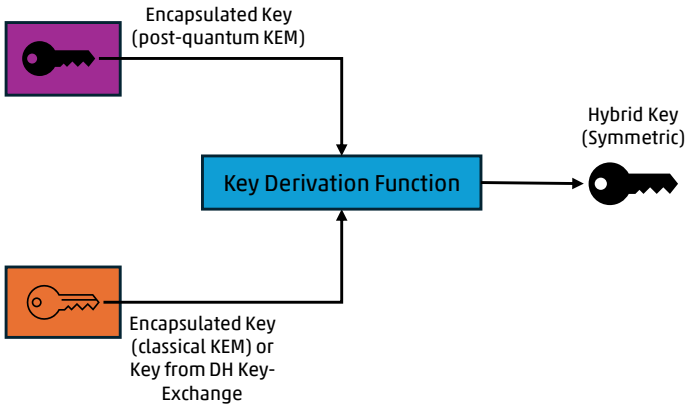

Figure 16: Hybrid scheme for key encapsulation

For signatures, messages are signed twice, once with the classical signature algorithm and once with the post-quantum signature algorithm. The verifier only accepts the message if both signatures are valid. Hybrid certificates may be implemented to build a hybrid public key infrastructure.

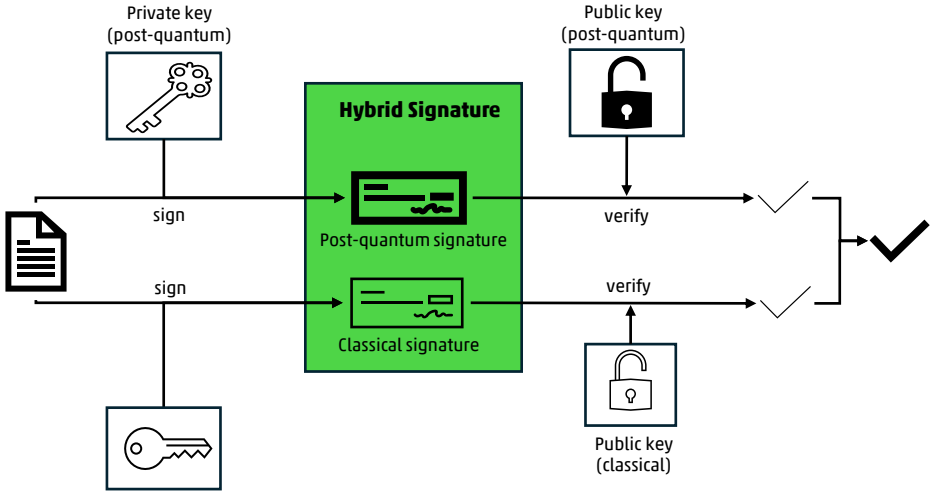

Figure 17: Hybrid signature scheme

Several authorities and agencies are accommodating (NIST, IR 8547 (IPD), 2024) and even requiring the use of hybrid techniques to combine classical and post-quantum cryptographic schemes. France's ANSSI, an organization involved in editing the French national cryptographic guidelines, requires the deployment of hybrid solutions during the transition to quantum-resistant systems (Jérôme Plût, 2023). The BSI, its German counterpart, also encourages the adoption of hybrid approaches to ensure smooth migration to post-quantum cryptographic standards. ETSI (European Telecommunications Standards Institute), a key global player in telecommunications standards, also promotes hybrid approaches during the transition to quantum-safe cryptography.

7.2.2 Early Adopters

Technology companies around the world have been working on concrete implementations of post-quantum schemes in their products and standards for several years. Some selected outcomes of these efforts are listed below. Interestingly many of them implement hybrid schemes first.

- Google Chrome initially introduced the hybrid scheme X25519Kyber768 for establishing symmetric secrets in TLS, starting in Chrome 116, with an experimental flag available in Chrome 115 (Chromium Blog, 2023). Chrome replaced X25519Kyber768 with ML-KEM in version 131, released on November 6, 2024 (Google Security Blog, 2024).
- Google introduced the first quantum-resilient FIDO2 security key implementation as part of OpenSK, their open-source security key firmware. It uses a hybrid signature scheme consisting based on ECC and Dilithium (Google, Toward Quantum-Resilient Security Keys, 2023)
- Cloudflare has integrated Kyber alongside other PQC algorithms into CIRCL, the Cloudflare Interoperable, Reusable Cryptographic Library. (Cloudflare, 2019)
- Amazon Web Services (AWS) supports hybrid modes involving Kyber in their AWS Key Management Service (KMS). (Amazon Web Services, 2024)
- IBM has introduced in 2019 the "World's First Quantum Computing Safe Tape Drive" using Kyber and Dilithium. (IBM Research, 2019).
- OpenSSH, a tool widely used for remote access and data transfer, introduced the first hybrid PQC key exchange based on Streamlined NTRU Prime and X25519 with release 9.0 (April 2022) (OpenSSH Project, 2022). This key exchange is already used by default in current versions of Linux distributions, MacOS and BSD Unix. In release 9.9 (September 2024), OpenSSH added an additional PQC key exchange based on ML-KEM and X25519 (OpenSSH Project, 2024).
- Open Quantum Safe (OQS) is an open-source initiative to support the development and integration of post-quantum cryptographic algorithms into various tools and applications. For this purpose, it develops the *liboqs* (Open Quantum Safe Project, 2023) is a C library that provides implementations of quantum-safe cryptographic algorithms and is designed to be easily integrated into protocols and applications, including widely used libraries like OpenSSL.

7.3 Recommendations for Organizations and Manufacturers

In this section, we outline general guidelines for different types of entities involved in cryptographic usage and implementation. We categorize these entities into three groups based on their level of cryptographic involvement:

- **Cryptographic Developers**: These entities design and develop the basic building blocks of cryptography, such as encryption methods, security protocols, and software libraries. They provide the foundational technologies that others build upon. Examples include organizations developing cryptographic standards (e.g., NIST, ETSI), open-source cryptographic libraries (e.g., OpenSSL, liboqs, BoringSSL).
- **Cryptographic Implementers**: These entities integrate and apply cryptographic functions *at a fundamental level* within security-focused software, hardware, networks, or infrastructure.

They do not develop new cryptographic primitives but use existing standards and cryptographic libraries to protect digital systems. Examples include developers of hardware tokens, VPNs, password managers, operating system security modules, web browsers, cloud-based key management services, manufacturers of cryptographic hardware such as hardware security modules (HSMs) and trusted platform modules (TPMs), and general-purpose software that directly implements cryptographic protocols, such as web servers.

- **Cryptographic Consumers**: These entities utilize existing cryptographic technologies without needing to modify or extend the underlying technology. They rely on cryptographic libraries and security software to protect their systems and data. Examples include banks, insurance companies, and organizations building software that integrates encryption but does not modify cryptographic operations at a core level.

Many organizations may fall into multiple categories depending on their role in cryptographic operations. For example, a company that develops cryptographic libraries (developer) might also implement them in its own security products (implementer) while relying on encryption software for internal data protection (consumer). Similarly, a bank primarily classified as a consumer may also develop custom cryptographic solutions for secure transactions, making it an implementer in certain contexts. These categories serve as general guidelines rather than rigid classifications, recognizing that organizations often engage with cryptography at multiple levels depending on their needs and responsibilities.

7.3.1 Recommendations: Cryptographic Developers

- **Validation and Testing:** Post-quantum cryptographic algorithms must undergo rigorous testing and validation by standardization bodies such as NIST, ETSI, and ISO.
- **Implementation Security:** Secure coding practices and formal verification methods should be applied to prevent vulnerabilities in PQC implementations.
- **Performance Optimization:** Algorithms should be optimized for efficiency across diverse computing environments, including cloud infrastructures, embedded systems, and IoT devices. Optimization should consider memory efficiency, computational overhead, and real-world deployment constraints. Developers must address the larger key sizes and output sizes (e.g., ciphertexts and signatures) of quantum-resistant algorithms compared to classical systems.
- **Mitigating Side-Channel Attacks:** Implementations must incorporate countermeasures to reduce vulnerabilities from side-channel attacks, ensuring resilience against power analysis, timing attacks, and fault injection. Post-quantum cryptographic implementations in CPUs and HSMs must be designed with hardened security to mitigate side-channel exploitation.
- **Maintaining Cryptographic Libraries:** Established cryptographic libraries (e.g., OpenSSL, liboqs, BoringSSL, Botan) must be updated to integrate post-quantum and hybrid algorithms. Libraries and APIs should be designed for seamless integration across diverse software ecosystems and hardware architectures.
- **Collaboration and Knowledge Sharing:** Developers must collaborate with organizations such as NIST, IETF, ETSI, and ISO to align with global PQC adoption strategies. Coordination with academic institutions, government agencies, and industry stakeholders is essential to accelerate

secure PQC deployment. Contributions to open-source projects should be encouraged to enhance transparency and trust in cryptographic implementations.

- **Long-Term Research and Planning:** To ensure a seamless transition to PQC, proactive planning and investment in research and hybrid cryptographic system development will be essential. Future-proofing security infrastructure must be a key consideration in cryptographic research efforts.

7.3.2 Recommendations: Cryptographic Implementers

- **Upgrading Cryptographic Libraries:** Cryptographic libraries (e.g., OpenSSL-PQC, liboqs) and APIs must be updated to integrate post-quantum and hybrid algorithms while maintaining backward compatibility with classical primitives. Operating System (OS) vendors must integrate PQC into system-level cryptographic libraries, encryption APIs (e.g., Windows CryptoAPI, Apple Security Framework), and secure boot mechanisms.
- **Upgrading Transport Security Protocols:** TLS, SSH, and VPN encryption mechanisms must be updated to support hybrid and post-quantum key exchanges. Content Delivery Networks (CDNs) and large-scale web infrastructure must integrate PQC-ready TLS to ensure long-term transport security. Internet Service Providers (ISPs) must upgrade their infrastructure to support PQC-secure encrypted communications and routing authentication mechanisms. Firewall software must enforce PQC-compliant encryption standards and ensure TLS inspection is compatible with quantum-resistant certificates.
- **Integration into hardware:** Cryptographic implementations in dedicated hardware hardware modules and **secure cryptoprocessor chips must support quantum-safe algorithms.** PQC algorithms must be efficiently integrated into hardware security modules (HSMs), trusted platform modules (TPMs), and secure processors. IoT devices, which have limited computational power and bandwidth, will require lightweight quantum-resistant solutions to maintain efficiency. CPU manufacturers must develop post-quantum cryptographic instruction sets and hardware acceleration to improve PQC performance in high-speed computing environments.
- **Developing Key Management Infrastructure:** Key management solutions must support post-quantum cryptographic keys, ensuring secure storage and rotation practices. Cloud security providers must integrate PQC-ready cryptographic services within their Key Management Services (KMS), encryption APIs, and TLS termination points.
- **Updating Digital Signatures:** Digital signature implementations must support quantum-safe algorithms. Operators of DNSSEC (Domain Name System Security Extensions) and BGPSec (Border Gateway Protocol Security) must transition to post-quantum digital signatures to ensure continued authenticity of domain name resolutions and routing security. Code-signing authorities must adopt PQC-ready digital signatures for software verification, CI/CD pipeline security, and software update integrity. Tools for electronic signatures must integrate quantum-safe timestamping services.
- **Updating PKI Systems:** PKI infrastructures must be enhanced to issue certificates using post-quantum signature schemes as they become available. This includes DNS resolvers, domain registrars, and secure routing infrastructure that relies on cryptographic trust mechanisms. Certificate Authorities (CAs) must transition certificate issuance, validation, and revocation

processes to post-quantum secure digital signatures, including updating OCSP and CRLs for PQC compatibility.

- **Updating network components:** Communication endpoints like web clients, proxies, application gateways, and web servers must support post-quantum and hybrid encryption and signature algorithms, and the capability to validate PQC certificates. Secure email and messaging systems (e.g., S/MIME, PGP) must integrate PQC-based encryption and signature schemes to maintain confidentiality and authenticity.
- **Mitigating Side-Channel Attacks:** Cryptographic implementations must be optimized to reduce potential vulnerabilities to side-channel attacks in post-quantum cryptographic systems.
- **Ensuring Crypto Agility:** Systems must be designed to allow future cryptographic upgrades and migration paths for PQC adoption. In particular, hardware manufacturers, must ensure modular security architectures that facilitate future updates and PQC firmware migration.
- **Supporting Cryptographic Consumers:** Guidance and technical assistance should be provided to cryptographic consumers in implementing PQC-based solutions, ensuring secure configurations, best practices, and ongoing security updates.
- **Integrating PQC into Blockchain Systems:** As blockchain security depends on asymmetric cryptography, blockchain networks must transition to post-quantum-secure digital signatures and key exchange mechanisms to ensure long-term security.

7.3.3 Recommendations: Cryptographic Consumers

Confidentiality

The protection of confidentiality of data must proceed in five stages.

- **Identification of relevant data:** The organization must identify its critical data that will remain sensitive beyond 10 years. Other factors such as storage location, data size, transport method, and security mechanisms must also be considered.
- **Analysis of cryptographic mechanisms:** The organization must inventory the existing systems and mechanisms for encrypting data. This includes identifying the cryptographic algorithms in use, key sizes, encryption modes, nonce generation, and overall security dependencies.
- **Modification of cryptographic mechanisms:** Legacy cryptographic algorithms need to be replaced with quantum-resistant algorithms. This may require decrypting and re-encrypting stored data. Cryptographic libraries used in internal systems may need to be updated or replaced to ensure compatibility with post-quantum standards.
- **Encrypting data with a quantum-resistant algorithm:** For data at rest, which is typically encrypted with symmetric cryptography, this may involve increasing key sizes. For transported data, this requires replacing transport protocols with ones that support quantum-resistant encryption schemes or hybrid cryptographic approaches.
- **Secure deletion of legacy encrypted data:** All previous copies, including decrypted caches and the original encrypted versions, must be securely deleted to prevent future vulnerabilities.

Authenticity, Integrity, Non-repudiation

- **Transitioning to PQC-compliant certificates:** Organizations must ensure internal PKI systems and certificate-based authentication methods are upgraded to quantum resistant digital

signatures. Internal certificate authorities must be updated to issue quantum-secure certificates for validity periods exceeding 10 years.

- **Updating authentication and authorization systems:** Identity verification mechanisms must integrate PQC-ready cryptographic authentication solutions or be prepared for updates in this aspect.
- **Ensuring long-term electronic signature validity:** Long-term electronic signatures accepted and issued by the organization must incorporate quantum-secure timestamps to prevent legal nullification in the future.

General Preventative Measures and Best Practices

- **Using hybrid cryptographic schemes:** During the transition period, hybrid cryptographic approaches should be used to maintain security. Exclusive reliance on post-quantum algorithms in the early stages is discouraged until further validation.
- **Using strong symmetric encryption:** Symmetric encryption keys should be at least 256 bits to mitigate vulnerabilities from Grover's algorithm.
- Where feasible, **replace or combine asymmetric encryption with pre-shared symmetric keys** in scenarios where key exchange occurs out of band (e.g., exchanging physical media).
- **Ensuring endpoint security and cryptographic monitoring:** Organizations must enforce PQC security policies on end-user devices, VPNs, and cloud applications, working with vendors and service providers to enable a smooth transition.
- **Adopting crypto agility:** Systems should be designed to enable flexible adaptation to evolving post-quantum standards and seamless updates during the migration process.

8_ References

Aguilar Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., . . . Zémor, G. (2017, November). Hamming Quasi-Cyclic (HQC). *Hamming Quasi-Cyclic (HQC)*, https://pqc-hqc.org/. Retrieved from https://hal.science/hal-01946880

Albrecht, M. R., Bernstein, D. J., Chou, T., Cid, C., Gilcher, J., Lange, T., . . . Wang, W. (2022, October). Classic McEliece: conservative code-based cryptography. *Classic McEliece: conservative code-based cryptography*. Retrieved from https://inria.hal.science/hal-04288769

Amazon Web Services. (2024). Post-Quantum TLS (PQTLS). *Post-Quantum TLS (PQTLS)*.

Aragon, N., Barreto, P. S., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., . . . Zémor, G. (2017, December). BIKE: Bit Flipping Key Encapsulation. *BIKE: Bit Flipping Key Encapsulation*. Retrieved from https://hal.science/hal-01671903

Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., . . . Stehlé, D. (2021). Crystals-kyber (version 3.02)-submission to round 3 of the NIST post-quantum project (2021). *Crystals-kyber (version 3.02)-submission to round 3 of the NIST post-quantum project (2021)*.

Barker, E. (. (2020). *Recommendation for Key Management: Part 1 – General.* National Institute of Standards and Technology. Retrieved from https://doi.org/10.6028/NIST.SP.800-57pt1r5

Beals, R., Buhrman, H., Cleve, R., Mosca, M., & de Wolf, R. (2001, July). Quantum lower bounds by polynomials. *J. ACM, 48*, 778–797. doi:10.1145/502090.502097

Bernstein, D. J., Brumley, B. B., Chen, M.-S., Chuengsatiansup, C., Lange, T., Marotzke, A., . . . Yang, B.-Y. (2017). NTRU Prime. *NTRU Prime*.

Bernstein, D. J., Dobraunig, C., Eichlseder, M., Fluhrer, S. R., Gazdag, S.-L., Hülsing, A., . . . Schwabe, P. (2017). SPHINCS + Submission to the NIST post-quantum project. Retrieved from https://api.semanticscholar.org/CorpusID:111381974

BSI. (2022, May). *Quantum-Safe Cryptography.* Retrieved from https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf

BSI. (2024, February). *Cryptographic Mechanisms.* Retrieved from https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=7

BSI. (2024). *Cryptographic Mechanisms: Recommendations and Key Lengths.* Retrieved from www.bsi.bund.de: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=10

BSI. (2024). *Entwicklungsstand Quantencomputer.* Retrieved from https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Quantencomputer/Entwicklungstand_QC_V_2_1.pdf?__blob=publicationFile&v=3

Callas, J., Donnerhacke, L., Finney, H., Shaw, D., & Thayer, R. (2007). *OpenPGP Message Format.* Tech. rep., Internet Engineering Task Force. Retrieved from https://datatracker.ietf.org/doc/html/rfc4880

Campbell, E. T., Terhal, B. M., & Vuillot, C. (2017). Roads towards fault-tolerant universal quantum computation. *Nature, 549*, 172–179. doi:10.1038/nature23460

Chromium Blog. (2023, August). *Protecting Chrome Traffic with Hybrid Kyber KEM.* Retrieved from https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html

Cloudflare. (2019). Introducing CIRCL: Cloudflare Interoperable, Reusable Cryptographic Library. *Introducing CIRCL: Cloudflare Interoperable, Reusable Cryptographic Library*.

Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D. (2017). CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation. Retrieved from https://api.semanticscholar.org/CorpusID:198994007

Ettinger, M., Høyer, P., & Knill, E. (2004). The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters, 91*, 43-48. doi:https://doi.org/10.1016/j.ipl.2004.01.024

Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., . . . Zhang, Z. (2019). Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Retrieved from https://api.semanticscholar.org/CorpusID:231637439

Friedl, M., Mojzis, J., & Josefsson, S. (2023). NTRU Prime for SSH. *NTRU Prime for SSH*.

Gidney, C., & Ekerå, M. (2021, April). How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum, 5*, 433. doi:10.22331/q-2021-04-15-433

Goldwasser, S. G. (1997). Public-key cryptosystems from lattice reduction problems. *Proceedings of the 17th Annual International Cryptology Conference (CRYPTO '97)*, 112-131.

Google. (2023, August). Protecting Chrome traffic with hybrid post-quantum and classic encryption. *Protecting Chrome traffic with hybrid post-quantum and classic encryption*.

Google. (2023, August). Toward Quantum-Resilient Security Keys. *Toward Quantum-Resilient Security Keys*.

Google Quantum AI. (2023). Weber: A 20 qubit programmable processor. *Weber: A 20 qubit programmable processor*.

Google Security Blog. (2024, September). *A new path for Kyber on the web.* Retrieved from https://security.googleblog.com/2024/09/a-new-path-for-kyber-on-web.html

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (pp. 212–219).

New York, NY, USA: Association for Computing Machinery. doi:10.1145/237814.237866

Grumbling, E., & Horowitz, M. (2019). *Quantum Computing: Progress and Prospects.* (E. Grumbling, & M. Horowitz, Eds.) Washington, DC: The National Academies Press . doi:10.17226/25196

Hennessy, J., & Patterson, D. (2018). A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development. *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, (pp. 27-29). doi:10.1109/ISCA.2018.00011

Hoffstein, J., Pipher, J., & Silverman, J. H. (1998). NTRU: A ring-based public key cryptosystem. In J. P. Buhler (Ed.), *Algorithmic Number Theory* (pp. 267–288). Berlin: Springer Berlin Heidelberg.

Hülsing, A. B. (2018). XMSS: eXtended Merkle Signature Scheme (RFC 8391). *Internet Engineering Task Force (IETF)*.

IBM Quantum. (2024). *IBM's Big Bet on the Quantum-Centric Supercomputer.* Retrieved from https://spectrum.ieee.org/ibm-quantum-computer-2668978269

IBM Research. (2019). CRYSTALS: Advancing Quantum-Safe Cryptography. *CRYSTALS: Advancing Quantum-Safe Cryptography*.

Intel Corporation. (2024). Intel Demonstrates High-Spin Qubit Wafer Fidelity and Uniformity. *Intel Demonstrates High-Spin Qubit Wafer Fidelity and Uniformity*.

IRTF. (2018). *XMSS RFC.* Retrieved from https://www.rfc-editor.org/rfc/rfc8391.html

IRTF. (2019). *LMS RFC.* Retrieved from https://www.rfc-editor.org/rfc/rfc8554.html

Jérôme Plût, A. (2023, November). Retrieved from ANSSI plan for post-quantum transition: https://pkic.org/events/2023/pqc-conference-amsterdam-nl/pkic-pqcc_jerome-plut_anssi_anssi-plan-for-post-quantum-transition.pdf

Kent, S., & Seo, K. (2005). *Security Architecture for the Internet Protocol.* Tech. rep., Internet Engineering Task Force. Retrieved from https://datatracker.ietf.org/doc/html/rfc4301

Kudelski Security, S. (2021, August). Quantum Attack Resource Estimate: Using Shor's Algorithm to Break RSA vs DH/DSA VS ECC. *Quantum Attack Resource Estimate: Using Shor's Algorithm to Break RSA vs DH/DSA VS ECC*.

McGrew, D. C. (2019). Leighton-Micali Hash-Based Signatures (RFC 8554). . *Internet Engineering Task Force (IETF).* .

Moore, G. E. (1975). Progress in digital integrated electronics. Retrieved from https://api.semanticscholar.org/CorpusID:110637516

Mosca, M. (2018). Cybersecurity in an Era with Quantum Computers: Will We Be Ready? *IEEE Security & Privacy, 16*, 38-41. doi:10.1109/MSP.2018.3761723

NIST. (2015, July). *Secure Hash Standard.* Tech. rep. doi:10.6028/nist.fips.180-4

NIST. (2017, January). Post-Quantum Cryptography Standardization. *Post-Quantum Cryptography Standardization*.

NIST. (2017). *Security (Evaluation Criteria).* Retrieved from https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-%28evaluation-criteria%29

NIST. (2024). *FIPS 203.* Retrieved from https://csrc.nist.gov/pubs/fips/203/final.

NIST. (2024). *FIPS 204.* Retrieved from https://csrc.nist.gov/pubs/fips/204/final.

NIST. (2024). *FIPS 205.* Retrieved from https://csrc.nist.gov/pubs/fips/205/final.

NIST. (2024, November). *IR 8547 (IPD).* Retrieved from Transition to Post-Quantum Cryptography Standards : https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf

NIST. (2025, March). *Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process.* Retrieved from https://nvlpubs.nist.gov/nistpubs/ir/2025/NIST.IR.8545.pdf

Open Quantum Safe Project. (n.d.).

Open Quantum Safe Project, l. (2023). Retrieved from liboqs: A C library for quantum-safe cryptography.: https://github.com/open-quantum-safe/liboqs

OpenSSH Project. (2022, August). OpenSSH Release Notes. *OpenSSH Release Notes*.

OpenSSH Project. (2024). OpenSSH Release Notes. *OpenSSH Release Notes*.

Pfaendler, S. M., Konson, K., & Greinert, F. (2024). Advancements in Quantum Computing–Viewpoint: Building Adoption and Competency in Industry. *Datenbank-Spektrum, 24*, 5–20. doi:10.1007/s13222-024-00467-4

Pollard, J. M. (1978). Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation, 32*, 918–924. doi:10.1090/S0025-5718-1978-0503523-8

Pomerance, C. (1996). A Tale of Two Sieves. *Notices of the American Mathematical Society, 43*, 1473–1485. Retrieved from https://www.ams.org/notices/199612/pomerance.pdf

Rupp, K. (2022). *Microprocessor Trend Data.* Retrieved from https://github.com/karlrupp/microprocessor-trend-data/blob/master/LICENSE.txt

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, (pp. 124-134). doi:10.1109/SFCS.1994.365700

Skosana, U., & Tame, M. (2021). Demonstration of Shor's factoring algorithm for N =21 on IBM quantum processors. *Scientific Reports, 11*, 16599. doi:10.1038/s41598-021-95973-w

Turing, A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, s2-42*, 230-265. doi:https://doi.org/10.1112/plms/s2-42.1.230

Vandersypen, L. M., Steffen, M., Breyta, G., Yannoni, C. S., Sherwood, M. H., & Chuang, I. L. (2001). Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature, 414*, 883-887. Retrieved from https://api.semanticscholar.org/CorpusID:4400832