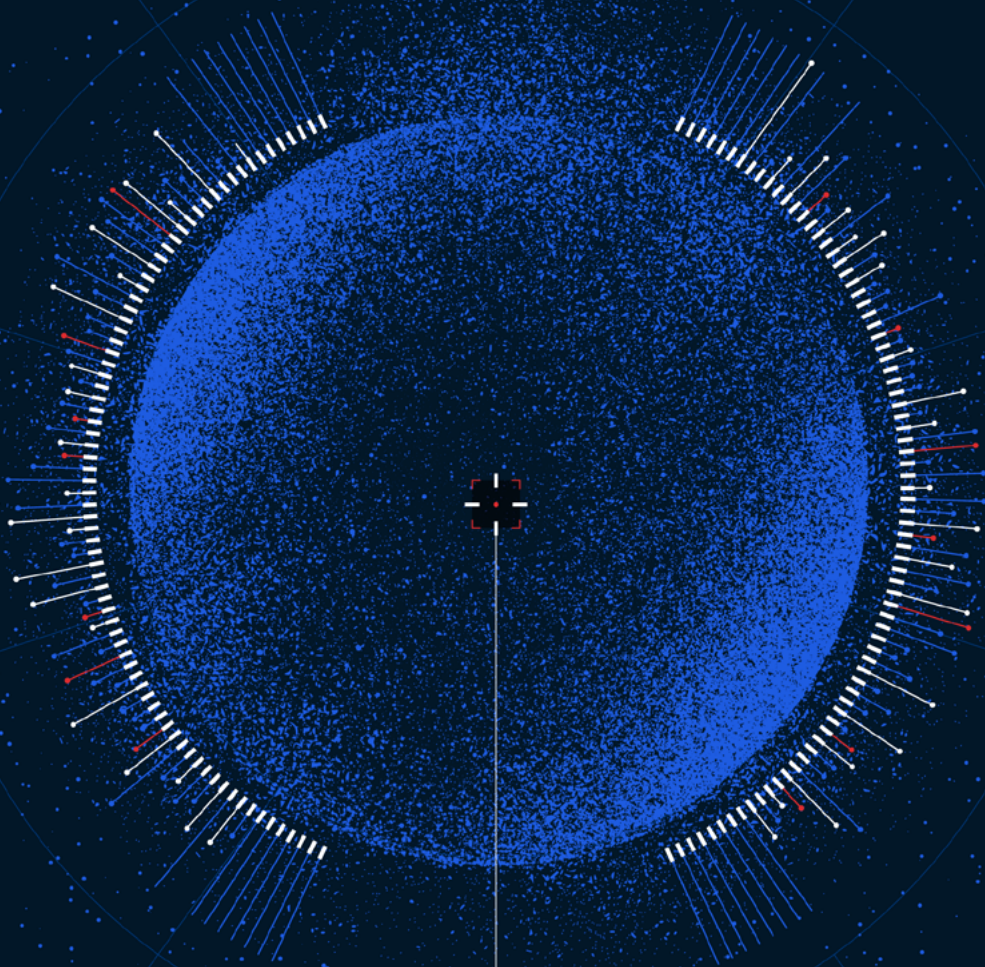




HARVEST

NOW,



DECRYPT

LATER

TABLE OF CONTENTS

3	Executive Summary	
	Key Findings — 4	
	Stakeholder Impact Assessment — 4	
	Recommendations — 4	
	Conclusion — 5	
6	1. Introduction: The impact of quantum computing in cryptography	
	1.1. The quantum threat — 8	
	1.2. When will quantum computing be able to break cryptography? — 9	
10	2. The HNDL threat	
	2.1. Steps involved in HNDL — 11	
	2.2. Attacker motivations and strategic advantages of HNDL — 13	
	2.3. Examples of real-world cases that could use the HNDL approach — 14	
15	3. Practical feasibility analysis of the HNDL threat	
	3.1. Secure communication protocols: TLS and SSH — 16	
	3.2. Storage with OpenPGP — 19	
	3.3. HNDL vulnerability analysis for TLS — 20	
	3.4. HNDL vulnerability analysis for SSH — 23	
	3.5. HNDL vulnerability analysis for storage — 25	
26	4. Mitigation strategies for the HNDL threat	
	4.1. Mitigation strategies at technical level — 27	
	4.2. Mitigation strategies at managerial level — 31	
	4.3. Hybrid key exchange — 32	
33	5. Conclusions	
	Mitigation and strategic response — 35	
36	Glossary	

EXECUTIVE SUMMARY

The advent of quantum computing poses a significant threat to current cryptographic systems, particularly those relying on asymmetric encryption, such as RSA and ECC. Quantum algorithms such as Shor's algorithm can efficiently break these systems, making encrypted data vulnerable to decryption in the future. This report explores the 'harvest now, decrypt later' (HNDL) threat, where encrypted data is collected today with the intent to decrypt it later when quantum capabilities become available.

This report, jointly developed by Europol and UC3M within the framework of the Advisory Group Research and Development, which implements Europol's strategy "Delivering security in partnership", assesses the technical feasibility and potential implications of the HNDL threat¹. It examines possible criminal motivations, identifies categories of data and stakeholders most likely to be affected, and outlines realistic scenarios in which law enforcement authorities may encounter this technique over time.

The report builds on Europol's existing work on quantum technologies, including *The Second Quantum Revolution: The Impact of Quantum Computing and Quantum Technologies on Law Enforcement*², the *Quantum-Safe Financial Forum – Call to Action*³, and the joint report on *Prioritising Post-Quantum Cryptography Migration*

*in Financial Services*⁴. While these publications have primarily focused on raising awareness, supporting preparedness and promoting the transition towards post-quantum cryptography (PQC), this report concentrates specifically on the criminal dimension most closely linked to quantum developments.

In particular, it aims to assess the extent to which the HNDL threat is plausible in the current landscape, and to identify stakeholders that may be more exposed based on the sensitivity and longevity of the data they manage. By doing so, the report seeks to support informed prioritisation of mitigation efforts and to contribute to a better understanding of emerging risks at the intersection of quantum technologies and security.

¹ Get to know more about the Advisory Group on Research and Development on <https://www.europol.europa.eu/about-europol/european-cybercrime-centre-ec3/ec3-partners>

² Europol, 'The Second Quantum Revolution: the impact of quantum computing and quantum technologies on law enforcement', 11 June 2024, <https://www.europol.europa.eu/publications-events/publications/second-quantum-revolution-impact-of-quantum-computing-and-quantum-technologies-law-enforcement>.

³ Europol, 'Quantum Safe Financial Forum - A call to action', 7 February 2025, <https://www.europol.europa.eu/publications-events/publications/quantum-safe-financial-forum-call-to-action>.

⁴ Europol, 'Prioritising Post-Quantum Cryptography Migration Activities in Financial Services', 21 January 2026, <https://www.europol.europa.eu/media-press/newsroom/news/joint-report-outlines-practical-approach-to-prioritising-post-quantum-cryptography-migration-in-financial-services>.

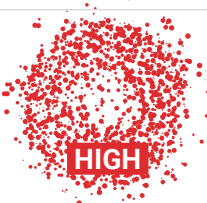
KEY FINDINGS

The HNDL threat is real. It involves collecting and storing encrypted data today, with the expectation of decrypting it in the future using advanced computational methods or quantum computers. This threat does not require the attacker to have a quantum computer today. This threat is particularly relevant for data that retains value over extended periods, such as government

communications, health records and intellectual property.

HNDL vulnerability assessment: The technical analysis confirms that standard security protocols like TLS, SSH and OpenPGP, currently in use across the internet, are susceptible to this threat, though risk levels vary based on configurations and key management practices.

STAKEHOLDER IMPACT ASSESSMENT

Stakeholder	Risk	Rationale
 Citizens & private individuals		While most personal data have limited long-term value, sensitive information such as financial records (especially for high-net-worth individuals), individual medical histories and personally identifiable information (PII) could become attractive targets for identity theft or blackmail in the future.
 SMEs & large corporations		Trade secrets, client lists, financial data and sensitive contracts are all high-value assets susceptible to corporate espionage via HNDL. A breach could result in significant financial loss, reputational damage or competitive disadvantage.
 Policymakers & government		National security communications (diplomatic cables, intelligence reports), law enforcement case files (ongoing investigations, informant identities), and critical infrastructure data are prime targets for state-sponsored actors utilising HNDL strategies. If compromised, this could jeopardise national security, diplomatic relations or public safety.

RECOMMENDATIONS

LOW-RISK STAKEHOLDERS:

1. Practise cybersecurity hygiene:

Use strong, unique passwords and enable multi-factor authentication. Regularly update devices and software to benefit from vendor security upgrades – including future PQC implementations offered through hybrid cryptographic solutions.

MEDIUM-RISK STAKEHOLDERS:

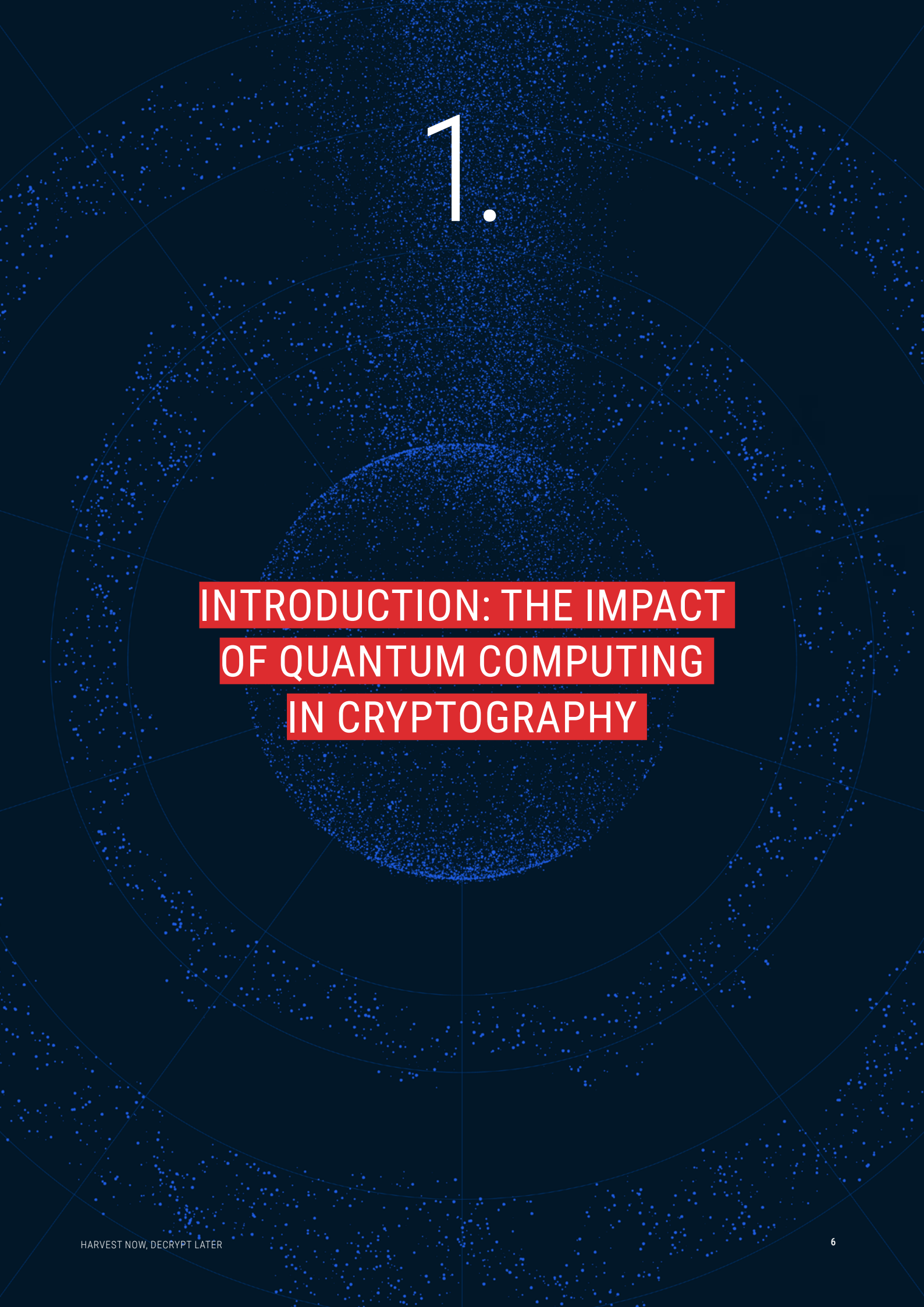
- 1. Prioritise immediate security enhancements:**
Upgrade to TLS 1.3 and SSH2, disable legacy protocols and enforce forward secrecy;
- 2. Implement data minimisation:**
Identify and securely delete unnecessary sensitive data. Limit long-term storage of confidential information;
- 3. Invest in long-term resilience** (as resources allow): Explore options for adopting PQC, including hybrid approaches, as it becomes more readily available through vendor solutions and standardised frameworks.

HIGH-RISK STAKEHOLDERS:

- 1. Deprecate legacy protocols:**
Immediately disable TLS 1.0/1.1 and other outdated cryptographic protocols and enforce forward secrecy;
- 2. Accelerate PQC transition:** Prioritise the integration of post-quantum cryptography into critical infrastructure and communication systems, starting with pilot projects, including hybrid cryptographic deployments, to gain experience and minimise disruption;
- 3. Develop a national PQC strategy:** Establish a coordinated plan for migrating to PQC across agencies, including funding, standards development and workforce training.

CONCLUSION

The HNDL threat is a forward-looking risk that requires proactive measures today to mitigate future vulnerabilities. By strengthening cryptographic agility and planning for the transition to PQC, organisations can protect sensitive data against the current HNDL threat and any other cryptographic threats that might emerge in the future.

The background of the slide features a dark blue field filled with numerous small, light blue dots. Overlaid on this are several concentric circles and radial lines, creating a grid-like pattern reminiscent of a radar screen or a stylized globe. The overall aesthetic is technical and futuristic.

1.

INTRODUCTION: THE IMPACT OF QUANTUM COMPUTING IN CRYPTOGRAPHY

Quantum computing represents a paradigm shift in information processing, leveraging principles from quantum mechanics, the physical domain that governs the behaviour of particles at atomic and subatomic scale.

Classic computers use binary digits or bits which exist in one or two states: 0 or 1. In contrast, quantum computers employ quantum bits, or qubits, which can exist in multiple states simultaneously due to a property known as superposition. This property enables quantum computers to process a vast number of possibilities concurrently, potentially leading to significant speed-ups for certain types of computations.

Another fundamental aspect of quantum computing is entanglement, a phenomenon where the state of one qubit becomes correlated with the state of another, regardless of the distance separating them. This property can be harnessed to perform complex calculations more efficiently than classical computers.

In essence, quantum computers will be able to break encryption by leveraging their superior computational power in integer factorisation, which is crucial for decrypting widely used cryptographic systems.

Cryptography, on the other hand, is the science of keeping information secret and secure. By using cryptography, a message can be scrambled or transformed into an unreadable form, known as encryption, and only people with the right key can unscramble or verify it through decryption. This fundamental science provides data confidentiality, integrity, authenticity and non-repudiation, allowing us to protect our messages, files, and even our identity when navigating the digital world.

In modern digital communication, we primarily use two types of cryptography (see *Figure 1.1*):

- **Symmetric cryptography** uses the same **secret key** to both encrypt and decrypt data, such as AES (advanced encryption standard), to provide confidentiality;
- **Asymmetric cryptography** (also known as public key cryptography) uses two different mathematically linked keys: a **public key**, which is accessible to everyone, and a **private key**, which is held exclusively by the owner. This key pair enables verification of identities online (through digital signatures) and the secure exchange of sensitive or confidential information (encryption with a public key), even between parties who have never met. Notable examples of public key cryptography algorithms include Rivest-Shamir-Adelman (RSA), El Gamal, Elliptic Curve Cryptography (ECC), and Digital Signature Algorithm ((EC)DSA).

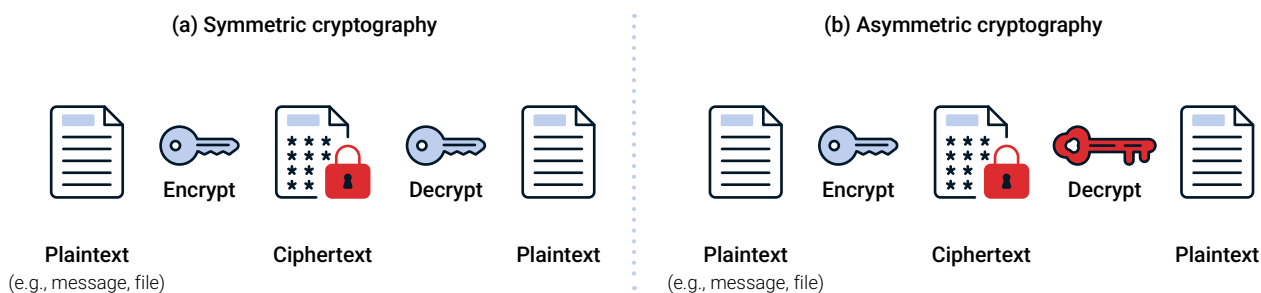


Figure 1.1. Symmetric cryptography (a) vs asymmetric cryptography (b)

In practice, security protocols combine symmetric and asymmetric encryption to secure files and communications. Asymmetric encryption is typically used to exchange a secret key, whereas symmetric encryption uses this secret key to

secure (encrypt) the data. The reason for this is efficiency. Depending on the implementation, symmetric encryption can be 1 000 to 10 000 times faster than asymmetric encryption.

1.1. THE QUANTUM THREAT

Generally speaking, cryptography is based on complex mathematical problems that are easy to solve when all required elements are available – typically a secret or private key – but infeasible to crack without it, and requiring thousands to millions of years using classical computers. Asymmetric cryptography security (e.g. RSA) relies on hard problems such as factoring large integers or solving discrete logarithms, while symmetric cryptography (e.g. AES) uses operations such as substitution and permutation to secure data. These foundational problems define the security of cryptographic systems.

However, this foundation is now threatened by the advent of **quantum computing**; quantum algorithms such as **Shor's algorithm** (1994) can solve the specific problems on which asymmetric cryptography relies, with significantly fewer steps than classical computers. In conclusion, current asymmetric cryptography relies on mathematical hurdles that quantum computers overcome far more efficiently, rendering many of today's standards insecure in a post-quantum world.

Unlike current asymmetric cryptography, symmetric cryptography is not fundamentally broken by quantum computers, though it is weakened. **Grover's algorithm** (1996) impacts symmetric algorithms by leveraging quantum computing to perform a brute-force search on an unstructured database with a quadratic speed-up, significantly reducing the time required to crack a key. For instance, an AES-128 key would require approximately 2^{128} attempts on a classical computer, but it would only require about 2^{64} attempts on a quantum computer. While this is still a massive number, it effectively halves the bit security strength of the algorithm. A similar principle applies to cryptographic hash functions from the SHA-2 and SHA-3 families, which ensure data integrity. Their collision and pre-image resistance are also impacted by quantum acceleration.

Essentially, quantum computing will be able to **break current asymmetric cryptography** and **weaken current symmetric cryptography**.

1.2. WHEN WILL QUANTUM COMPUTING BE ABLE TO BREAK CRYPTOGRAPHY?

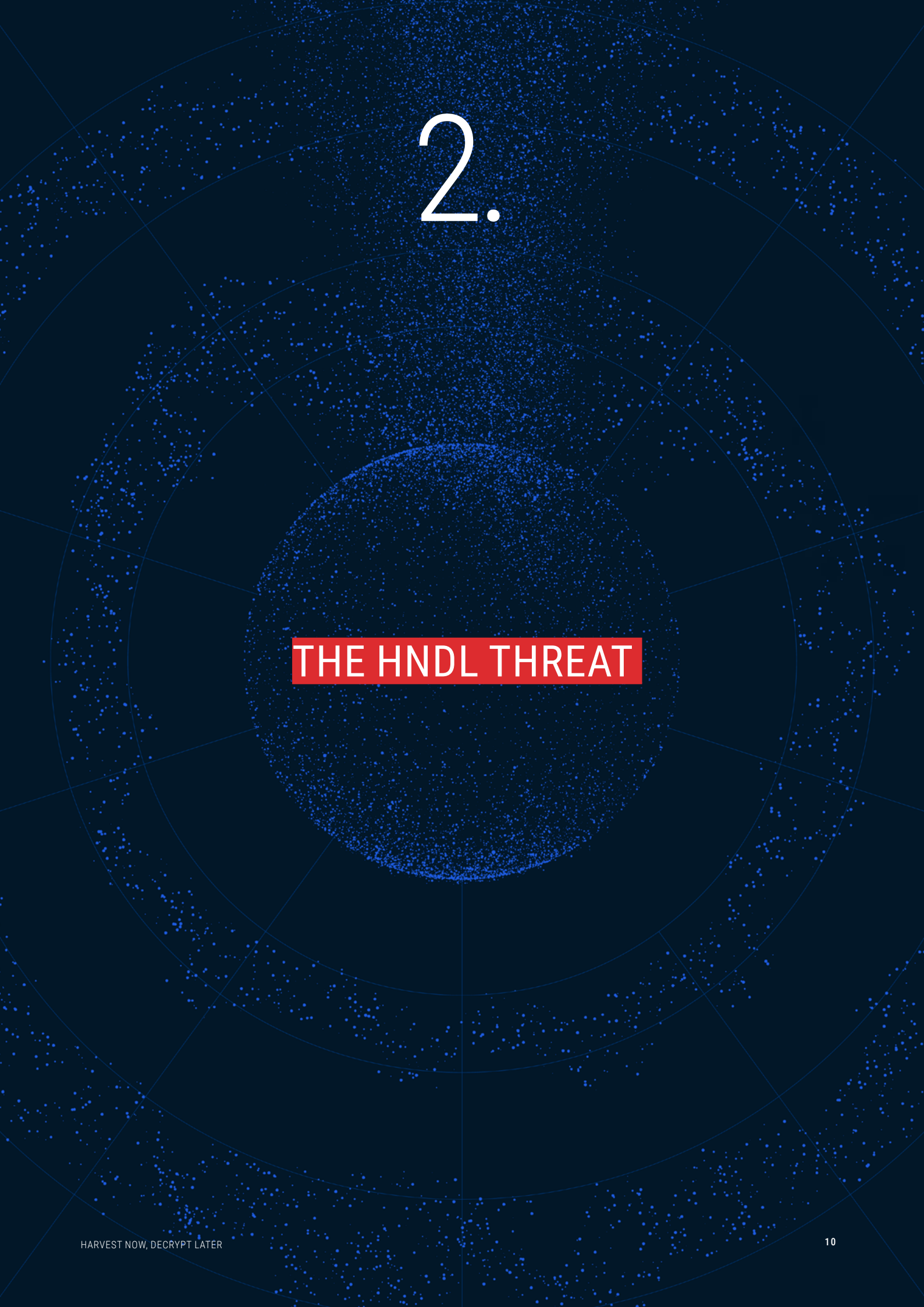
This is often the million-dollar question. However, a definite timeline is less important than it seems. Both cryptography and quantum computing are highly complex fields, and past predictions about their evolution have frequently failed to materialise.

Despite this uncertainty and ongoing scepticism, organisations must still ensure that their information and communication remain secure and interoperable by adopting robust security standards. For this reason, the US National Institute of Standards and Technology (NIST) has published the NIST IR 8547⁵ report, which states that classical public key cryptography should be phased out by 2035, with today's most common configurations deprecated by 2030. Similarly, the European Commission has issued a *Recommendation on a Coordinated Implementation Roadmap for the transition to Post-Quantum Cryptography*⁶, aimed at guiding the move towards more secure cryptographic solutions.

To address the **potential threat to asymmetric cryptography** posed by quantum computers, global security agencies and cryptography community are working on **post-quantum cryptography** (PQC). PQC relies on new algorithms based on mathematical problems that are believed to be difficult to solve for classical and quantum computers alike. Transitioning to quantum-resistant public key cryptography is essential and requires replacing current system such as RSA and ECC with PQC alternatives. For **symmetric encryption algorithms** and cryptographic **hash functions**, experts recommend precautionary measures such as **larger key sizes and longer hash outputs**, for example migrating from AES-128 to AES-256 or from SHA-256 to SHA-512 where possible.

⁵ NIST, 'Transition to Post-Quantum Cryptography Standards', 12 November 2024, <https://csrc.nist.gov/pubs/ir/8547/ipd>.

⁶ European Commission, 'A Coordinated Implementation Roadmap for the Transition to Post-Quantum Cryptography', 23 June 2025, <https://digital-strategy.ec.europa.eu/en/library/coordinated-implementation-roadmap-transition-post-quantum-cryptography>.

The background of the slide features a series of concentric circles centered on the page. These circles are composed of numerous small, light blue dots, creating a textured, starry effect. The overall color scheme is dark blue, with the text and title box providing contrast.

2.

THE HNDL THREAT

'Harvest now, decrypt later' (HNDL) is a cybersecurity threat that refers to the practice of collecting and storing encrypted data now, with the intention of decrypting it later when cryptographic protections may be weaker or broken. This strategy relies on the assumption that advancements in computing power or new cryptographic vulnerabilities will eventually make it possible to decrypt data that are currently considered secure (encrypted).

Think of a burglar who breaks into an office building and finds a large, heavy safe. The safe is locked, and they do not have the combination yet, so they cannot open it or profit from its contents right away. Instead of walking away, they haul the safe away and store it in a warehouse. Over time, they might learn the combination or acquire better tools to break in.

This raises an important question: will today's information still be valuable tomorrow? Some types of data will undoubtedly retain their value

over time, making them attractive targets now as well as in the future. The HNDL threat is a significant concern because it highlights the potential risks of data being compromised long before attackers can decrypt it. Moreover, classical cryptography faces a significant challenge from the development of quantum computers that are expected to break public key cryptography, as demonstrated by Shor's algorithm. In practical terms, this means that information considered secure today will be accessible in the future.

2.1. STEPS INVOLVED IN HNDL



1. DATA COLLECTION (HARVESTING)

Targeting encrypted data: Attackers or adversaries gather encrypted data, often in bulk, from various sources such as intercepted communications, data breaches, network traffic, or online repositories. These data can include personal information such as emails and financial data, but also sensitive corporate information or state secrets.



2. DATA STORAGE

Storing encrypted data: Once the data are harvested, attackers store it for future decryption. The encrypted data are typically stored in a hidden way to avoid detection. This can involve using stolen or throwaway accounts on cloud storage services, other digital services or compromised hosts, peer-to-peer networks or decentralised storage, physical storage devices, etc., to keep the data safe until they can be decrypted.



3. WAITING FOR DECRYPTION

Waiting for cryptographic advances: The attackers wait for new developments in cryptography or computational power that might allow them to decrypt the data. There are two main scenarios criminals might be waiting for:

3.1. ADVANCES IN CRYPTANALYSIS:

Cryptographic researchers might discover weaknesses in encryption algorithms, and new techniques or new vulnerabilities might emerge that make it feasible to break encryption;

3.2. QUANTUM COMPUTING:

One of the main reasons this threat is becoming more relevant is the rise of quantum computing. Quantum computers will break widely used encryption algorithms like RSA and ECC by leveraging quantum algorithms (such as Shor's algorithm) that could factor large numbers or solve discrete logarithm problems much faster than classical computers. Although practical quantum computers capable of breaking encryption may be still far off, adversaries may choose to gather encrypted data in the meantime.



4. DECRYPTION

Decrypting the data: Once attackers have the necessary computational power or have discovered new techniques or vulnerabilities, they can decrypt the harvested data. This may happen after years or even decades.

Once attackers decrypt the stolen data, they can exploit or sell the information for various malicious purposes. This can involve facilitating other crimes, such as financial fraud, identity theft, or corporate/government espionage. Additionally, the stolen data could be used to launch further cyberattacks, blackmail victims, or even influence political outcomes. Organisations, governments and individuals may not even realise their data has been compromised until it is too late.

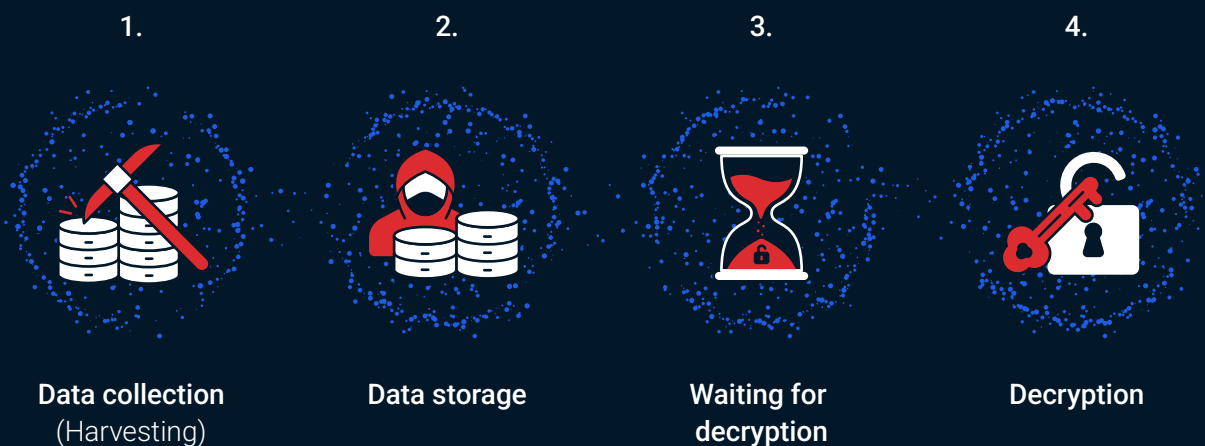


Figure 1.2: Harvest Now Decrypt Later process

2.2. ATTACKER MOTIVATIONS AND STRATEGIC ADVANTAGES OF HNDL

In most cases, cybercriminals aim to obtain an immediate profit from their actions. For example, the initial objective of ransomware activities is to encrypt files to block access to them, forcing companies to pay if they want to recover them and resume operations. Online fraudsters primarily target victims for financial gain, and some data breaches are carried out to sell the stolen information afterwards. But the HNDL threat presents cybercriminals with an alternative, sometimes complementary, strategic approach.

Criminals may resort to the HNDL tactic due to the potential long-term value of encrypted data and the advancement of technology that could eventually render today's cryptography obsolete. While this is a plausible threat, it is difficult to prove the long-term intent of current or potential future criminals. This section will therefore explore the motivations for why a cybercriminal would choose HNDL over (or in addition to) immediate exploitation.



Strategic data accumulation:

Some attackers may be focused on accumulating valuable data over time, even if it cannot be immediately

accessed or exploited. This is particularly true in espionage or large-scale cybercrime activities like ransomware, where stolen data may not be immediately useful, but could become an asset in the future. Attackers may target data related to intellectual property, personal information (for blackmail purposes) or national security, all of which could become valuable over time.

Think about how collectors operate. An antique or art collector invests in pieces not necessarily for immediate use, but for their potential appreciation in value. They research provenance, condition and market trends to identify pieces that will be worth more in the future. HNDL allows cyber criminals to act like collectors, identifying data with long-term value (like a rare antique) and preserving it for future profit or strategic advantage.



Reducing risk of immediate detection:

By passively harvesting encrypted data rather than actively attempting

to decrypt it, attackers can avoid triggering security alarms or alerting victims through ransom demands or extortion attempts. At the time of collection, security systems may fail to detect the full scope of the breach, and data might not be easily recognised as valuable, allowing attackers to remain under the radar. Imagine you are in a bustling café. If someone passively eavesdrops on your conversation, you might not even realise they are there, and neither would anyone else. However, if someone actively intercepts your conversation, it is like they are loudly repeating your words to someone else at a nearby table. You would likely notice the interruption, and so would others. Cybercriminals might prefer passive interception because it resembles the quiet listener in the café. They can gather information without alerting their targets or leaving obvious traces of their presence.

HNDL represents a calculated risk, allowing attackers to position themselves for future opportunities. The threat of quantum computing further amplifies the value of harvested data, making this strategic approach a potentially lucrative investment.

2.3. EXAMPLES OF REAL-WORLD CASES THAT COULD USE THE HNDL APPROACH

While there are no widely publicised, specific 'HNDL' cases, there are several real-world incidents where this tactic could be effective. Elements of this approach can be observed more broadly in cyber espionage and ransomware attacks, where attackers harvest and store data with the intention of exploiting it later.

CYBER ESPIONAGE AND DATA THEFT BY NATION STATES

Nation state actors often target intellectual property, sensitive communications, or strategic business or government data. They may not have an immediate use for all the data they steal but can choose to 'store' it for later use, especially as new vulnerabilities or cryptographic breakthroughs emerge.

USE OF RANSOMWARE

What happened: Ransomware groups typically encrypt data on victim systems and demand payment in exchange for a decryption key. In many cases, attackers also exfiltrate sensitive data before encryption and threaten to publish or sell it if the ransom is not paid. Stolen information may be retained for future exploitation, including resale on illicit markets or renewed extortion attempts. In some scenarios, attackers could store encrypted or sensitive data for long periods, potentially exploiting it later if new decryption capabilities emerge or if circumstances increase the likelihood that the victim will pay.

REAL CASE / HNDL ELEMENT

Groups like REvil and Clop have been observed performing triple extortion attacks, which involve:

1. Exfiltrating sensitive data before encrypting the victim's files;
2. Encrypting the victim's files and demanding a ransom for decryption;
3. Threatening to release or sell the exfiltrated data if the ransom is not paid.

The HNDL element comes into play when the attackers store the exfiltrated data, including the encrypted information, for potential future exploitation, rather than immediately monetising it. This approach allows ransomware groups to maximise long-term gain from the attack, combining immediate disruption with strategic future opportunities.

3.

PRACTICAL FEASIBILITY ANALYSIS OF THE HNDL THREAT

In this section, a feasibility analysis of the HNDL threat is conducted across widely used **secure communication protocols**, such as Transport Layer Security (TLS) versions 1.2 and 1.3 and Secure Shell (SSH), and **OpenPGP-based⁷ storage of information**. The analysis is based on the paper published, with technical details⁸. The specific conditions that must be met to exploit HNDL are outlined through certain representative use cases.

TLS is primarily used to secure web communications, email, databases or virtual private networks (VPNs). Similarly, SSH is mainly used for remote access and secure file transfer, making it a standard tool for IT

system administrators. Finally, PGP-based storage is commonly employed by password managers, backup systems and tools for encrypting files or email archives.

3.1. SECURE COMMUNICATION PROTOCOLS: TLS AND SSH

Although TLS and SSH are used for different purposes, both rely on asymmetric cryptography at the beginning of the communication to 'shake hands' and set security rules between client and server. This process is called the security negotiation phase, and it usually involves two main steps:

1. **Identity verification** (or authentication), using the client's and/or server's private keys (i.e. digital signatures or proof of possession of the private key associated with a public key certificate), without revealing the keys themselves;
2. **Secret key creation**, both sides create a shared secret, the 'session key', which will be used to encrypt the exchanged data.

Once the security negotiation phase finishes, client and server switch to fast symmetric encryption, ensuring applications run smoothly and quickly.

A standardised and widely used method of creating a shared secret key (or session key), over the open internet, is the Diffie–Hellman (DH)⁹ method. In DH, the key itself is not sent. Instead, each party generates a private secret key and corresponding public value. They exchange their public values, and each party combines the other party's public value with their own private secret key. When both sides perform this operation, they generate the same shared key without ever having transmitted their confidential information or the shared secret key itself, as shown in Figure 3.1.

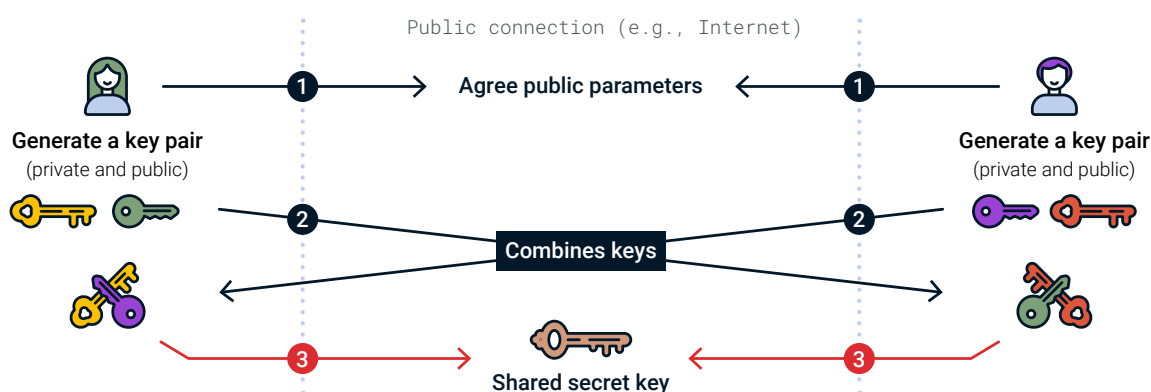


Figure 3.1. Diffie–Hellman Key Agreement

⁷ OpenPGP website, <https://www.openpgp.org/>.

⁸ Blanco-Romero, J., Mendoza, F.A., Rubio, C.G., Campo, C. and Sánchez, D.D. (Cornell University), 'On the Practical Feasibility of Harvest-Now, Decrypt-Later Attacks', 1 March 2026, <http://arxiv.org/abs/2603.01091>.

⁹ Rescorla, E., 'Diffie–Hellman Key Agreement Method', *IETF RFC 2631*, June 1999.

In TLS, this occurs during the **handshake protocol**, where client and server negotiate supported cipher suites (cryptographic algorithms), exchange cryptographic material (using DH), and authenticate the server (and optionally the client) using certificates tied to public/private key pairs. The outcome of this handshake is the derivation of shared symmetric keys used to secure the communication channel efficiently.

Figure 3.2 illustrates the full handshake mechanisms of TLS versions 1.2 and 1.3, which are recommended today. In both versions, the negotiation begins with a **ClientHello** message (from the Client), followed by a **ServerHello**

message (from the Server); however, TLS 1.3 introduces a significant architectural shift to optimise both performance and security. By including key share extensions directly in the initial **ClientHello** message, TLS 1.3 allows the server to derive session keys immediately, reducing the handshake to a trip, compared to the two trips required by TLS 1.2. Furthermore, TLS 1.3 provides enhanced security by encrypting the handshake metadata from the **EncryptedExtensions** message, which includes the server's identity. Finally, TLS 1.3 strengthens the protocol by removing legacy, insecure ciphers and mandating key exchanges based on modern elliptic curves as a fundamental requirement.

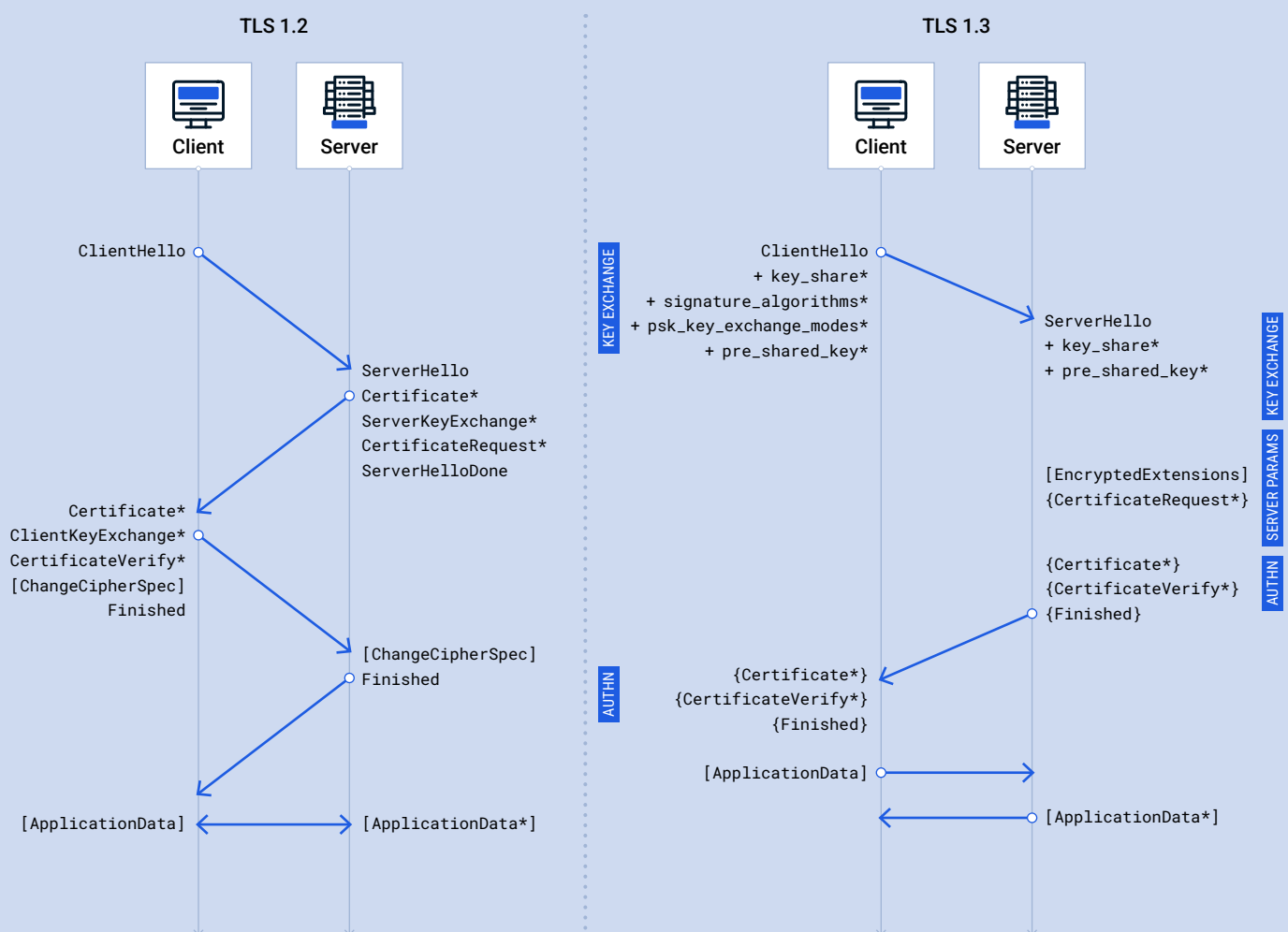


Figure 3.2. Message flow for a full handshake (TLS 1.2 (left)¹⁰ and TLS 1.3 (right)¹¹)

During this message flow, several intermediary secret keys are derived before generating the final session keys.

¹⁰ T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, IETF, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>

¹¹ E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, IETF, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>

Similarly, as shown in Figure 3.3, SSH begins its connection establishment with a key exchange phase in which client and server agree on algorithms (**KEXINIT** message), perform an exchange based on DH (**KEXDH** and **NEWKEYS** messages), and authenticate the server's identity through its host key. Optionally, user authentication often relies on asymmetric key pairs (**USERAUTH** messages), eliminating password-based weaknesses. Once user authentication is successful, information is encrypted and exchanged (Application Data).



Figure 3.3. SSH Message flow using DH and public key cryptography for user authentication¹²

¹² T. Ylonen and C. Lonvick, The Secure Shell (SSH) Transport Layer Protocol, RFC 4253, IETF, January 2006, and The Secure Shell (SSH) Authentication Protocol, RFC 4252, IETF, January 2006, subsequently updated by later RFCs.

3.2. STORAGE WITH OPENPGP

HNDL is not limited to network communications' 'data in transit'; it can also apply to encrypted files' 'data at rest'. An adversary might collect encrypted files, archives or database dumps today, hoping to decrypt them later when keys are compromised. One common example is encrypted email or documents protected with OpenPGP.

OpenPGP is the open standard for PGP (Pretty Good Privacy), which utilises asymmetric cryptography to protect data at rest and in transit, ensuring confidentiality, integrity and

authenticity. PGP generates a random symmetric session key (using algorithms such as AES) to encrypt the actual data (e.g. a message or a file). This session key is then encrypted using the public key of the intended recipient and bundled with the ciphertext (see Figure 3.4). Upon receiving the file, the recipient uses their private key to decrypt the session key, which in turn decrypts the data. This hybrid scheme allows secure large files and storage volumes without the significant performance penalties associated with pure public key cryptosystems.

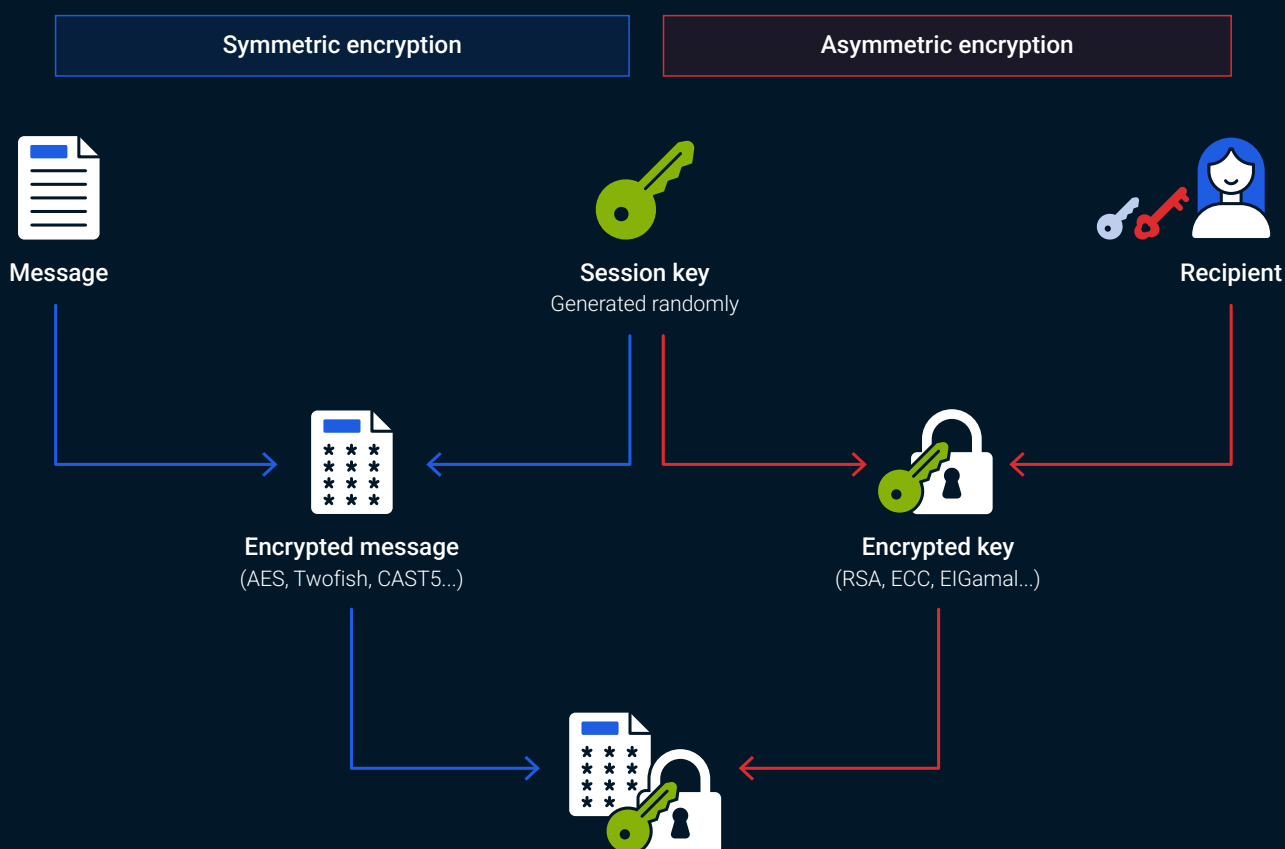


Figure 3.4. PGP Hybrid Encryption

This report focuses on PGP-based file encryption as a representative storage encryption scenario, and it analyses what an attacker would need to decrypt such data later.

3.3. HNDL VULNERABILITY ANALYSIS FOR TLS

To perform a HNDL attack in TLS, a passive adversary records the entire handshake and encrypted traffic of a session, then later uses quantum computing (or other cryptanalytic breakthroughs) to get private keys and break the handshake's cryptography and recover the session keys.

The data required to break the handshake's cryptography will depend on the TLS version and the type of key exchange used. In general, the key exchange methods used in TLS1.2 and TLS1.3 can be grouped into two types:

► **Non-forward secrecy key exchange:**

In these types of key exchange, the private key of the server is reused, meaning that if we manage to break it, we will be able to decrypt past and future handshakes without breaking any other key.

► **Perfect forward secrecy (PFS) key exchange:**

In these types of key exchanges, the private keys of the server are not reused, or even used at all during the handshake. A new pair of public/private keys is used for each exchange, usually referred as ephemeral keys. In practice, it means that the attacker will have to break a private key for each handshake they want to recover.

TLS 1.2 supports the two types of key exchange, whereas TLS 1.3 only supports PFS key exchange methods, and it is therefore more secure. Meaning that, in TLS 1.2, if the TLS session used a non-forward secrecy key exchange (RSA or other static modes), then obtaining the server's private key can be enough to decrypt past recorded TLS 1.2 sessions.

If the session used forward secrecy, as in TLS 1.2 with ephemeral (DHE), or based on elliptic curves (ECDHE), then the long-term private key alone is not sufficient, and each recorded session would need to be broken individually.

TLS version	WITH forward secrecy	WITHOUT forward secrecy
TLS1.3	TLS_AES_128_GCM_SHA256 TLS_AES_256_GCM_SHA384 TLS_AES_128_CCM_SHA256 TLS_AES_128_CCM_8_SHA256 <i>TLS_CHACHA20_POLY1305_SHA256</i> <i>(Not listed by NIST, but recommended by IETF RFC8446)</i>	None
TLS1.2	TLS_ECDHE_ECDSA_WITH_* TLS_ECDHE_RSA_WITH_* TLS_DHE_RSA_WITH_* TLS_DHE_DSS_WITH_* (*) AES 128/256 in GCM (recommended) or CBC (allowed) mode, using SHA256/384 <i>(IETF RFC9325 recommends only AES in GCM mode to provide AEAD)</i>	TLS_DH_DSS_WITH_* TLS_DH_RSA_WITH_* TLS_ECDH_ECDSA_WITH_* TLS_ECDH_RSA_WITH_* (*) AES 128/256 in CBC or GCM mode, using SHA256/384
TLS1.1 TLS1.0 <i>Deprecated</i>	N/A	N/A

Table 1 TLS versions and cipher suites recommended by NIST SP 800-52 Rev2

In TLS 1.3, the server's private key, associated with the server's public key certificate, is used only for authentication purposes. Therefore, compromising it later does not reveal prior session plaintexts; prior sessions would only be at risk if the attacker later recovers per-session handshake secrets or session resumption secrets (for example, Pre-Shared Keys (PSKs) or resumption tickets stored at an endpoint).

Forward secrecy protects against future compromise of long-term keys, but under the quantum threat model even ephemeral key exchanges become vulnerable, although an attacker would need to break each individual key exchange, significantly increasing the computational overhead. While this makes a full-scale attack more difficult, it is by no means impossible for a sufficiently powerful quantum computer.

Table 1 summarises cipher suites that provide forward secrecy and those that do not. In TLS 1.3 all ciphers provide PFS, because it is not negotiable. In TLS 1.2, cipher suites starting with TLS_DHE or TLS_ECDHE provides it. A detailed analysis can be found in the paper 'On the Practical Feasibility of Harvest Now, Decrypt Later Attacks'¹³. Other minor TLS versions are deprecated because these are vulnerable and considered insecure.

To summarise, both TLS 1.2 and TLS 1.3 are vulnerable to HNDL attacks, but the severity differs. In TLS 1.2 with RSA key transport (non-forward secrecy), compromising a single server key would allow an adversary to decrypt all past and future sessions protected by that key. When ephemeral Diffie–Hellman (DHE or ECDHE) is used instead, TLS 1.2 gains forward secrecy: the attacker would need to break each recorded session's key exchange individually. TLS 1.3

¹³ Blanco-Romero, J., Mendoza, F.A., Rubio, C.G., Campo, C. and Sánchez, D.D. (Cornell University), 'On the Practical Feasibility of Harvest-Now, Decrypt-Later Attacks', 1 March 2026, <http://arxiv.org/abs/2603.01091>.

enforces this stronger model by design, removing RSA key transport and mandating ephemeral exchanges for every full handshake, and therefore compromising a server's long-term key alone cannot decrypt past traffic. However, since the ephemeral public values are already present in the recorded handshake, a sufficiently powerful quantum computer could solve the Diffie–Hellman problem to obtain each session's shared secret and decrypt the corresponding traffic; additionally, resumed sessions, relying on pre-shared keys derived from a broken handshake, would also be exposed if the key is not updated (conversely, if the attacker failed to capture the original full handshake, the resumed session remains protected, since the pre-shared key cannot be reconstructed). In short, TLS 1.2 with RSA is the most exposed configuration, whereas TLS 1.3 limits the damage to per-session attacks,

although both remain ultimately vulnerable under the quantum threat model.

In both TLS versions, an HNDL attacker would record the entire connection, including the handshake negotiation, all public key exchange values and the encrypted data, even though none of it can be read at that point. The attacker stores this recording and waits. When the necessary cryptographic capability becomes available (whether through key compromise, theft, or a future quantum computer), the attacker can work backwards from the recorded public values to recover the session secrets and decrypt everything. Figure 3.5 illustrates this process for a TLS 1.3 session: once the shared secret from the key exchange is recovered, all session keys follow automatically.

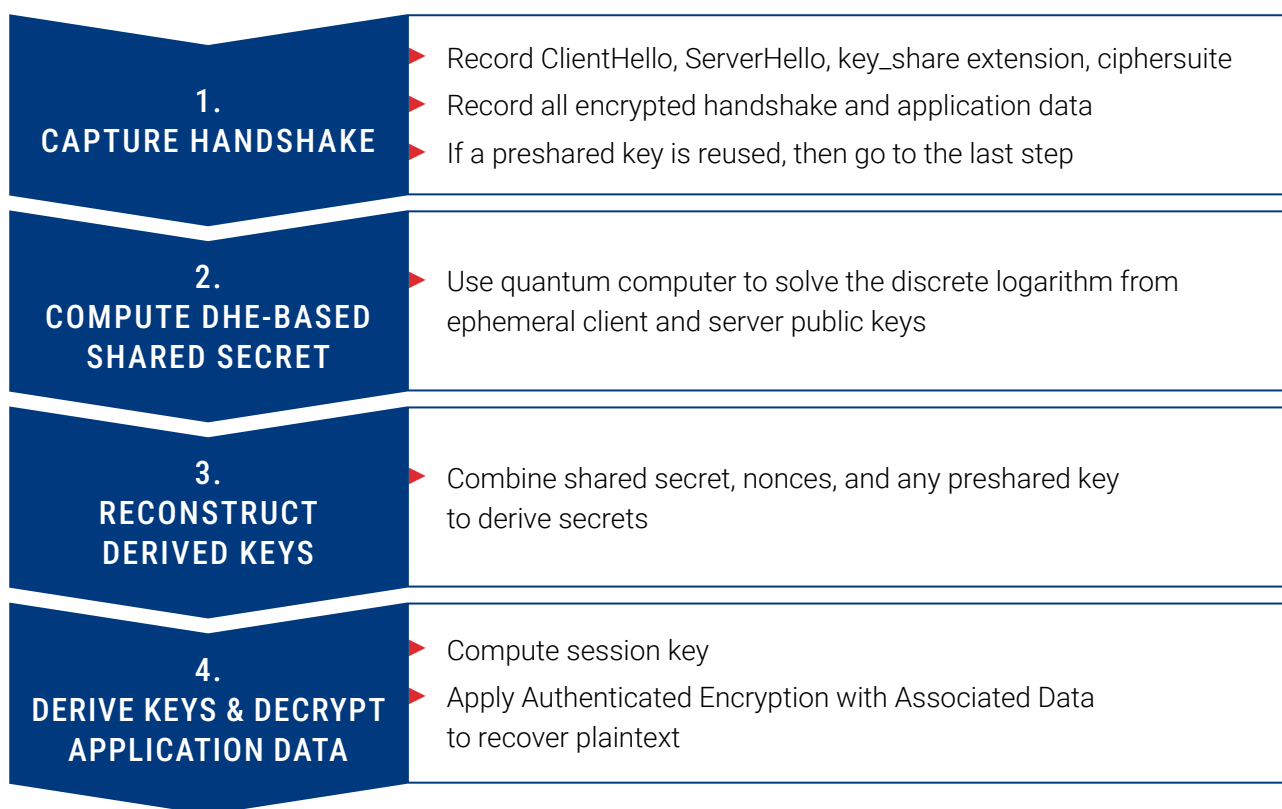


Figure 3.5. Decryption Steps for a Recorded TLS 1.3 Session

Two additional points are worth noting. First, TLS 1.3 offers a 'fast reconnection' mode (called 0-RTT) that reuses secrets from a previous session; if the attacker recorded the original full handshake and later breaks it, the reconnected session is exposed as well. However, if the original handshake was not captured, the reconnected

session remains safe. Second, although TLS 1.3 allows endpoints to refresh their encryption keys during a session – something called re-keying – these refreshed keys are derived from the original handshake secret and would therefore also be recoverable once the underlying key exchange is broken.

3.4. HNDL VULNERABILITY ANALYSIS FOR SSH

In the case of SSH communications, under the default and recommended settings, SSH's key exchange is ephemeral and forward-secret. The Diffie–Hellman shared secret key established during the handshake is not stored in the long term; once the session is over and keys are discarded, an attacker cannot recover that secret unless they can solve the DH problem. Importantly, the server host private key (e.g. an RSA, DSA, or key based on elliptic curves (Ed25519)) is used only to authenticate the exchange (by signing certain handshake data) and is not used to encrypt the session key itself. Thus, if an adversary records an SSH session and later steals the server host key, they still cannot derive the session key from the recording. Past SSH sessions remain confidential, illustrating forward secrecy: a compromise of long-term keys does not compromise past session keys. This property protects from HNDL in the classical scenario.

However, as with TLS, the caveat comes with quantum computing. If an attacker can one day break the DH exchange (e.g. using Shor's algorithm to compute the discrete log of the recorded public values), they could then obtain the shared secret key and decrypt it. All widely used SSH key exchange methods (diffie-hellman-group14-sha256, ecdh-sha2-nistp256, curve25519-sha256) rely on the difficulty of the discrete logarithm problem; thus, they would all be vulnerable to a quantum HNDL adversary just as TLS would be. It is worth noting that an older key exchange method for SSH based on RSA (RSA Key Exchange for SSH) involves the server encrypting a secret key with its RSA key, such as TLS 1.2 RSA mode. That method does not provide forward secrecy. However, this RSA-based key exchange has been deprecated and is not commonly used today. All modern SSH connections use the DH family, which provide forward secrecy, including based on elliptic curves; however, as discussed for TLS, a future quantum adversary could still decrypt archived sessions by breaking the recorded Diffie–Hellman exchange.

SSH protocol version	WITH forward secrecy	WITHOUT forward secrecy
SSH-v1 (Obsolete, insecure!)	N/A	N/A
SSH-v2 (Current standard)	RFC 9142 curve25519-sha256 diffie-hell- man-group16-sha512 ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-sha2-nistp521 gss-curve25519-sha256-* gss-group14-sha256-* gss-nistp256-sha256-* gss-nistp384-sha384-* gss-nistp521-sha512-*	

Table 2 SSH version and Key Exchange mechanisms recommended by IETF RFC 9142

Figure 3.6 illustrates the steps of a complete HNDL attack against SSH. A passive adversary captures the SSH handshake, including the SSH_MSG_KEXINIT algorithm negotiation, the SSH_MSG_KEXDH_INIT/REPLY messages containing ephemeral public keys, and all subsequent encrypted traffic. Using a future quantum computer, the attacker solves the elliptic curve discrete logarithm to recover one party's

ephemeral private key, then computes the shared secret. With the shared secret and the captured exchange, the attacker applies a hash function to derive all six session keys and decrypt the entire archived session. The single-stage key derivation from the shared secret and hash means all session keys are immediately recoverable once the ECDH shared secret is obtained.

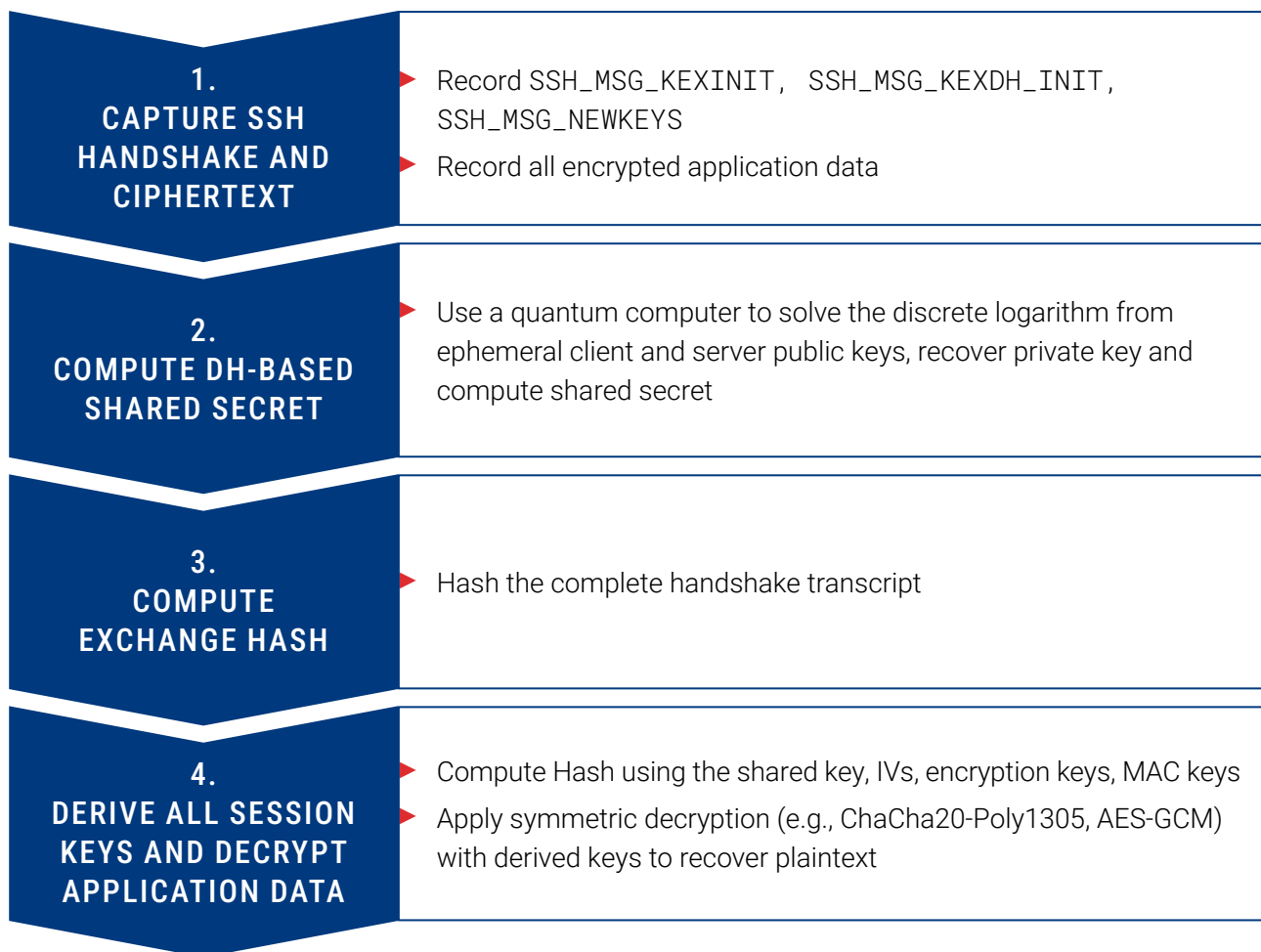


Figure 3.6. Decryption Steps for a recorded SSH session under the HNDL model

Finally, it is worth mentioning that SSH can perform a 're-key' after a connection has started, to refresh its security. It does this by starting a new key exchange (SSH_MSG_KEXINIT) right inside the existing secure connection. This creates a brand-new secret key for that specific part of the session, though the session's 'original ID' stays the same. An HNDL adversary who records the entire session (including re-key exchanges) can break each key exchange independently and derive the corresponding session keys for each epoch.

Unlike a TLS 1.3 key update, which derives new keys deterministically from the previous secret, SSH re-keying performs a full new DH exchange, theoretically requiring the attacker to break each exchange separately. However, this provides no additional quantum resistance, since each re-key has the same underlying ECDH problem.

3.5. HNDL VULNERABILITY ANALYSIS FOR STORAGE

As mentioned before, OpenPGP implements a hybrid encryption scheme: a random symmetric key (also called a session key) is used to encrypt content (with a cipher like AES). Then, that random session key is encrypted with the intended recipient's public key (for example, using RSA or El Gamal encryption) to ensure only the intended recipient (i.e. public key owner) can access it. The result is an encrypted session key, which is stored in the message alongside the ciphertext of the content. Upon receiving it, the recipient uses their private key to decrypt the session key and then uses that session key to decrypt the content.

If an attacker performed an HNDL attack on PGP-encrypted data, two main things would be needed:

1. The **encrypted data itself**, including the encrypted session key packet(s). In practice, this means the attacker must get a copy of the PGP-encrypted file or message. For example, the attacker might intercept an encrypted email in transit or steal a backup that contains PGP-encrypted files.
2. The **corresponding private key of one of the recipients** (along with the ability to use it, i.e. the passphrase or password if it is passphrase-protected) at some point in the future.

Data encrypted with PGP for recipients Alice and Bob will have the following format:

```
-----BEGIN PGP MESSAGE-----  
... (some header info)...  
<Encrypted Session Key for Alice>  
<Encrypted Session Key for Bob>  
<Encrypted content data>  
-----END PGP MESSAGE-----
```

The encrypted session key for Alice is a blob that only Alice's private key can decrypt (it typically appears as a PKESK packet in the binary encoding), and similarly for Bob. An eavesdropper can see these blobs but cannot decode them without the private key.

Now, suppose the attacker archives this message. Later, an attacker with a quantum computer, having collected the public key of Alice, could derive Alice's private PGP key from her public key. Once the private key is compromised, the attacker can use it to decrypt any session key blob intended for her. That yields the session (symmetric) key that was used to encrypt the content. Finally, the attacker takes the encrypted content and decrypts it with this session key, obtaining the plaintext. This entire process will succeed as long as the cryptographic algorithms remain unbroken and the private key is obtained. There is no additional ephemeral secret key needed, because the 'ephemeral' session key was stored and wrapped by the public key. In essence, the PGP message itself contains all the information needed to decrypt (the content cipher text and the session key encrypted for the private key), and thus possession of the private key at any time will unlock it.

It is also worth noting that PGP private keys are often protected by a passphrase. Consequently, an attacker might obtain an encrypted private key (e.g. by hacking someone's computer or a key server), and then later also guess or crack the passphrase. Once they have the actual private key material, decryption of past messages is straightforward. If the algorithms used for public key encryption (RSA, ElGamal, ECC) are broken through advances (such as quantum computing factoring RSA), then an adversary could derive it from the public key and then do the same even if the private key was not directly stolen. In the context of quantum threats, RSA and ElGamal (which relies on discrete log) would be broken by quantum algorithms, meaning an attacker could potentially compute the recipient's private key from the public key in the future.

4.

MITIGATION STRATEGIES FOR THE HNDL THREAT

Because HNDL targets data that are already being transmitted and stored today, mitigation must be proactive; waiting until quantum computers arrive would be too late for traffic that has already been recorded. The measures available fall into two broad categories: at technical level, organisations can enforce perfect forward secrecy, adopt post-quantum cryptography (PQC), and deploy hybrid key exchanges that combine classical and quantum-resistant algorithms; while at management level they can apply data minimisation, shorter key and data life cycles, and cryptographic agility programmes to prepare for the longer-term transition.

4.1. MITIGATION STRATEGIES AT TECHNICAL LEVEL

4.1.1. SECURE COMMUNICATION PROTOCOLS (DATA IN TRANSIT)

To mitigate HNDL risks, in the short term, the fastest reduction in exposure often comes from using modern protocol versions and secure defaults. Organisations must use the latest and recommended TLS 1.3 and SSH2 versions. Implementations must be updated, and legacy configurations and deprecated protocol versions

avoided; that is, organisations must disable TLS 1.0, TLS 1.1, and SSH1 where they are still present. These versions are deprecated due to having known vulnerabilities, being insecure nowadays, and increasing the likelihood of near-term compromise (HNDL).

In addition, some features also require conscious extra caution for TLS:

-
- ▶ **TLS 1.3 0-RTT early data:** this mode does not provide forward secrecy for early data. Must be restricted to low-risk use cases, while avoiding sending sensitive data and using key update.
 - ▶ **Session resumption and pre-shared keys (PSKs):** session resumption is widely used for performance, but it needs careful configuration. Where resumption occurs without a fresh key exchange, a future compromise of the resumption secret can expose more than one session's data. (A PSK is a secret key reused to resume or speed up a connection.)
-

Where TLS 1.2 must be used for compatibility, it should be restricted to forward-secret Diffie–Hellman key exchange modes (DHE or ECDHE) and RSA key transport avoided. This also applies to SSH key exchange modes. Algorithms that use static public/private keys, such as RSA, must be disabled entirely. Although PFS does not defeat a future quantum adversary (who could still solve the ephemeral key exchange), it substantially raises the cost of an HNDL attack: instead of breaking one long-term key to decrypt years of traffic, the attacker must break every recorded session individually.

If possible, the transition to configurations that support PQC should be prioritised (see Section 4.3). Even if full PQC migration is not yet feasible, hybrid key exchanges (see Section 4.4) that combine classical ephemeral DH with quantum-resistant algorithms should be implemented. Nevertheless, for many organisations, migrating to PQC is the long-term goal, but it is not yet consistently deployable across all systems, products and partners.

4.1.2. STORAGE ENCRYPTION (DATA AT REST)

For encrypted data at rest, such as files or archives protected with OpenPGP, the main HNDL risk is that the long-term private key could be compromised or broken by a future quantum computer, exposing every file ever encrypted to that key. In the short term, organisations should ensure that encryption keys are rotated regularly and that older ciphertexts are re-encrypted with fresh keys when practical. Where possible,

migrating to PQC-capable encryption tools (for example, GnuPG builds that support post-quantum public key algorithms) will provide longer-term protection. Equally important is limiting the volume of sensitive data stored in encrypted form: data that are no longer needed should be securely deleted rather than archived indefinitely, since every retained ciphertext is a potential future target.

4.1.3. POST-QUANTUM CRYPTOGRAPHY (PQC) ADOPTION

Post-Quantum Cryptography, also known as quantum-resistant cryptography, is a new set of cryptographic algorithms that are secure against both classical and future quantum computers. This can be applied to both data in transit and data at rest. These algorithms use public key schemes either for a key encapsulation mechanism (KEM) or for digital signatures. Unlike key agreement protocols such as Diffie–Hellman (where both parties contribute to the shared secret key), a KEM allows one party to use a public key to generate and wrap a shared secret (encapsulation), and the other party to unwrap it with the corresponding private key (decapsulation). A digital signature defines both the generation and verification of signatures.

KEM algorithms include three main operations: key pair generation, encapsulation and decapsulation.

Encapsulation uses a public key to generate a ciphertext and a shared secret key, while decapsulation uses the corresponding private key and the ciphertext to generate the same shared secret key. Signature algorithms also include three operations: key pair generation, signing and verification. Signing uses a private key to sign a message, and verification checks the validity of the signature using the message and the public key.

The algorithms proposed so far are mainly based on five core schemes: lattices, multivariate polynomials, error-correcting codes, hash-based signatures and isogeny of elliptic curves. A summary of the main characteristics of each scheme is provided in the table below. Each algorithm has different versions, with variations in their parameters or encryption and hash functions, which offer different security levels.

Scheme family	Main characteristics
Lattice-based	Flexible construction and good balance between performance, key size, and ciphertext/signature size. Based on computationally hard problems over high-dimensional lattices, such as Learning with Errors (LWE) and Short Integer Solution (SIS) and their variants. Representative examples include CRYSTALS-Kyber for KEM, and CRYSTALS-Dilithium and FALCON for digital signature. CRYSTALS-Kyber and CRYSTALS-Dilithium have already been standardized: FIPS 203: ML-KEM and FIPS 204: ML-DSA, respectively. FALCON-derived FN-DSA standard (FIPS 206) is still under development.

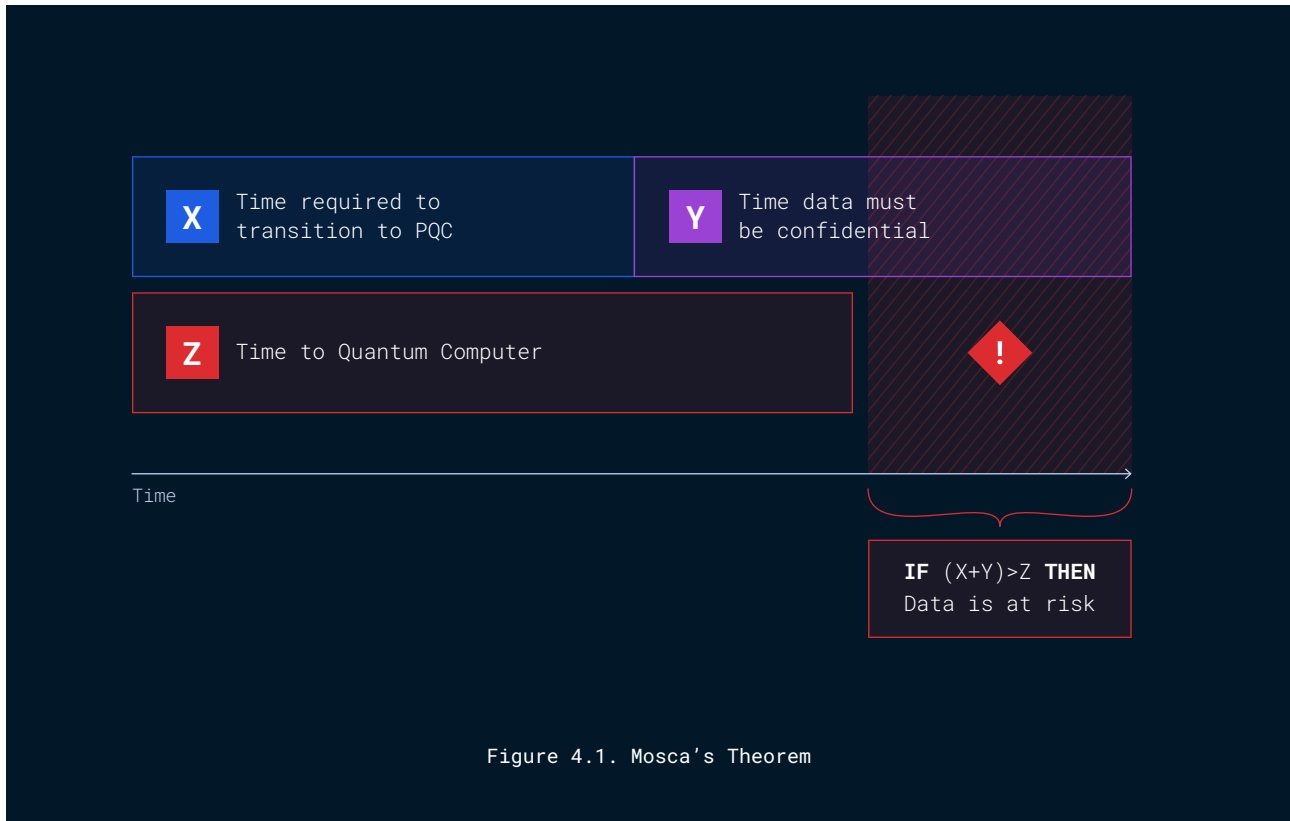
Scheme family	Main characteristics
Multivariate	Specifically investigated for digital signatures. Often trade relatively large public keys for compact signatures and relatively efficient signing and verification. Based on the difficulty of solving systems of multivariate polynomial equations. MAYO, QR-UOV, SNOVA, and UOV (Unbalanced Oil and Vinegar) are currently third-round candidates in NIST's Additional Digital Signature Schemes process. MQOM is also in such round; although NIST classifies it as an MPC-in-the-Head signature, its underlying hard problem is the multivariate quadratic (MQ) problem.
Code-based	Long-standing security foundations, although their key (generally large public key) and ciphertext (generally small ciphertext) vary across constructions. Based on hard decoding problems in error-correcting codes. Representative examples include Classic McEliece, BIKE, and HQC. HQC was selected by NIST for standardization in 2025 following the 4th round of the original PQC process; its final FIPS standard is still pending. SDitH, a third-round additional-signature candidate, uses an MPC-in-the-Head construction based on the syndrome-decoding problem.
Hash-based	Designed exclusively for digital signatures. Compact public and private keys, but comparatively large signatures, involving significant signing and verification costs depending on the construction. Relying primarily on the security properties of cryptographic hash functions. Includes both stateful and stateless variants. SPHINCS+ is an example of a stateless hash-based algorithm, which was standardized by NIST as SLH-DSA in FIPS 205.
Isogeny-based	Despite the collapse of SIDH-based protocols like SIKE (Supersingular Isogeny Key Encapsulation), isogeny-based cryptography remains an active field of interest through candidates like SQISign, which is characterized by very compact public keys and signatures. Based on hard problems involving isogenies between elliptic curves, SQISign is currently a third-round signature candidate, although its compactness comes at the cost of comparatively high computational overhead.

Table 3 PQC algorithms

In addition to the families shown in the table, NIST's third-round additional signature candidates include FAEST, which NIST classifies as a symmetric-based signature scheme. FAEST relies on symmetric-key primitives combined with zero-knowledge/MPC-style techniques and does not naturally fall into the five traditional PQC families considered above.

The implementation and integration of these algorithms should be guided by Mosca's Theorem. Cryptographers use this logic to determine the urgency of the transition: if X (the years your data must remain secret) plus Y (the years needed to

upgrade the infrastructure) is greater than Z (the years until a cryptographically relevant quantum computer exists), then the security of the data is already at risk, as shown Figure 4.1.



4.1.3.1. PRACTICAL APPLICATION OF PQ ALGORITHMS IN SECURE COMMUNICATION PROTOCOLS

PQC algorithms are available through open-source implementations such as liboqs¹⁴, PQCclean¹⁵ and Bouncy Castle¹⁶, among others. These are increasingly integrated into industry-standard secure protocol libraries like OpenSSL¹⁷, wolfSSL¹⁸ and OpenSSH¹⁹, facilitating their adoption within TLS and SSH and other secure communication protocols.

To maintain backward compatibility, most implementations utilise hybrid schemes, combining post-quantum algorithms with classical elliptic curves, by default. Furthermore, other secure communication protocols, for instance VPN protocols such as WireGuard and IPsec, are being updated to support PQC, enabling organisations to establish ‘quantum-safe tunnels’ or secure connections that protect internal company secrets against both current and future

cryptographic threats. Likewise, DTLS is also supporting PQC to be used for IoT protocols (mainly) and other protocols over UDP (User Datagram Protocol).

However, the efficacy of these protocols relies heavily on a globally recognised PQC-based public key infrastructure (PKI) to distribute the digital certificates used to authenticate servers. Consequently, organisations should prioritise inventorying their existing certificates and testing hybrid X.509 certificates in isolated environments; this ensures that legacy network hardware can manage the significantly larger data sizes typical of post-quantum signatures. Parallel to these infrastructure updates, emerging research is exploring alternative architectures that aim to eliminate dependency on traditional digital certificates.

4.1.3.2. PRACTICAL APPLICATION OF PQ ALGORITHMS IN SECURE STORAGE PROTOCOLS

Securing data at rest presents a unique challenge: the encryption must remain robust for the entire lifecycle of the data.

The IETF (Internet Engineering Task Force) is currently finalising the crypto-refresh for OpenPGP to include support for ML-KEM (Kyber) and Composite Signatures. Thus, users can encrypt files using a hybrid scheme where the file is locked with both a traditional RSA/ECC key and a PQ ML-KEM key. This ensures that even if an attacker recovers the RSA private key, for instance in 2035, the ML-KEM layer still protects the file content.

In an equivalent manner, cloud providers that offer ‘Zero-Knowledge’ encryption (where only the user has the key) are transitioning to PQC to protect user privacy against future state-level decryption. Providers are beginning to use ML-KEM to wrap the symmetric keys (AES-256) used to encrypt large data volumes and ML-DSA to sign file metadata.

¹⁴ Github’s liboqs page: <https://github.com/open-quantum-safe/liboqs>.

¹⁵ Github’s PQCclean page <https://github.com/PQClean/PQClean>.

¹⁶ Bouncy Castle’s website: <https://www.bouncycastle.org/>.

¹⁷ OpenSSL’s website: <https://www.openssl.org/>.

¹⁸ WolfSSL’s website: <https://www.wolfssl.com/>.

¹⁹ OpenSSH’s website: <https://www.openssh.org/>.

4.2. MITIGATION STRATEGIES AT MANAGERIAL LEVEL

4.2.1. CRYPTOGRAPHIC AGILITY AND LONG-TERM MIGRATION TO PQC

The measures described above address the HNDL threat at protocol level. However, cryptography is not a one-time deployment: algorithms age, new vulnerabilities emerge, and standards evolve. The overarching lesson of the quantum threat is also

that organisations need **cryptographic agility**: the ability to update the algorithms and parameters used across their systems with predictable effort and minimal disruption.

In practice, building cryptographic agility involves several complementary steps:

-
- ▶ First, organisations should inventory the cryptography they currently use. This includes TLS termination points, VPN gateways, SSH servers, backup and archival encryption, email encryption, databases, and any internal services that rely on certificates or key material. Without a clear map of where cryptography is deployed, planning an upgrade is guesswork.
 - ▶ Second, they should identify which data carry long-term confidentiality requirements. Not all data need to remain secret for decades, but categories such as personal health records, classified government communications, intellectual property and law enforcement case files may well exceed a ten-year sensitivity window, placing them squarely in the HNDL risk zone.
 - ▶ Third, organisations can reduce the volume of data available to harvest by applying data minimisation principles: enforcing retention limits, securely deleting data that are no longer needed, and segmenting high-value datasets so that a single point of interception does not expose everything.
 - ▶ Fourth, key management practices should be toughened. Long-term private keys should be stored in hardware security modules (HSMs) where possible, access should be monitored, and incident response plans should cover key compromise scenarios.
 - ▶ Finally, the transition to PQC should be treated as a programme rather than a single project. Organisations should track vendor and standards body roadmaps (including NIST's post-quantum standards and the European Commission's PQC migration recommendations), test interoperability in non-production environments, adopt hybrid schemes where they are available today, and plan phased rollouts that prioritise the highest-risk systems first.
-

None of these steps requires waiting for a quantum computer to exist. The goal is to be ready before one arrives, so that the window during which harvested data remain vulnerable is as narrow as possible.

4.3. HYBRID KEY EXCHANGE

Because PQC deployment takes time and must work across diverse systems, many organisations treat **hybrid key exchanges** as a prudent interim approach. This reflects the fact that PQC algorithms are relatively new compared to long-established schemes such as RSA or ECC. In simple terms, a hybrid approach runs two key exchanges in parallel. One is classical (for example, ECDH), and one is post-quantum (for example, ML-KEM). The final session key is derived from both. The intention is that the resulting protection remains strong even if one of the two approaches later proves weaker.

Using hybrid schemes, an attacker would typically need to defeat both the classical and the post-quantum component to recover the session key. The aim is to reduce exposure to HNDL threats without discarding the maturity and operational familiarity of existing classical approaches. However, hybrid deployments add complexity and can introduce interoperability and operational challenges, so they are best treated as a transitional step rather than an end state.

Hybrid key exchange is also moving from research into deployment. In TLS, the IETF has been developing a standard construction for hybrid key exchange in TLS 1.3²⁰. IPsec and its key exchange mechanism, IKEv2, can be extended to perform multiple key exchanges and combine secret keys, providing a standards track route to hybrid post-quantum key establishment²¹.

Some widely used systems have also started adopting hybrid designs in practice, including OpenSSL and OpenSSH (starting with version 9.0) shipping a hybrid post-quantum and classical key exchange as default²².

Finally, it is worth noting that another hybrid key exchange, combining a PQC-based key with one generated by quantum nodes, is currently being deployed and tested in highly sensitive environments and research projects. Furthermore, quantum random number generators (QRNGs) are being integrated into key generation processes to begin incorporating quantum computing devices into current scenarios.

²⁰ IETF, 'Hybrid key exchange in TLS 1.3', 20 May 2026, <https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/>.

²¹ IETF, 'Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)', on the RFC 9370 website, May 2023, <https://www.rfc-editor.org/rfc/rfc9370>.

²² OpenSSH, 'OpenSSH 9.0 Release Notes', April 2022, <https://www.openssh.org/txt/release-9.0>.

5.

CONCLUSIONS

This report demonstrates that the HNDL phenomenon presents a credible long-term threat to widely used communication protocols such as TLS and SSH. Under specific conditions, encrypted data intercepted today could be decrypted in the future once sufficiently powerful quantum computing capabilities become available.

However, executing an HNDL attack requires significant resources: large-scale data harvesting and long-term storage, eventual access to a cryptographically relevant quantum computer, and the capability to process and break substantial volumes of key material. Due to these high costs and technical barriers, HNDL is currently mostly viable against high-value targets where long-term confidentiality is critical, such as governmental communications or confidential business information.

Although there is currently no clear evidence that HNDL is being systematically exploited at scale, past cyber incidents involving the exfiltration of encrypted data from high-value organisations suggest that the prerequisites for future decryption may already be in place. Additionally, the risk cannot be indefinitely confined to high-value targets. With quantum capabilities

maturing and costs decreasing, the threat may expand to broader sectors.

While HNDL poses a genuine strategic risk, it is not as immediately pervasive as threats such as ransomware. It primarily concerns sensitive data that must remain confidential for extended periods. Notably, the continued use of deprecated protocols such as TLS 1.0 and TLS 1.1 creates an immediate **'Harvest Now, Decrypt Now'** (HNDN) risk, as these versions can be compromised using classical computing techniques without the need for quantum advances.

Overall, organisations should view HNDL not as a speculative concern but as a forward-looking risk that requires proactive cryptographic resilience and migration planning today, to safeguard long-term confidentiality.

MITIGATION AND STRATEGIC RESPONSE

The long-term defence against HNDL is **migration to Post-Quantum Cryptography (PQC)**, designed to resist both quantum and classical attacks. However, mitigation should begin now by strengthening cryptographic agility. Organisations should first inventory where cryptography is deployed – across TLS, VPNs, SSH, certificates, backups, databases and internal services – to understand upgrade dependencies. They should then identify data requiring long-term confidentiality, such as health records, classified communications, trade secrets or law enforcement files, and which fall within the HNDL risk window.

Data minimisation can reduce exposure by limiting retention, securely deleting unnecessary data, and segmenting high-value datasets. At the same time, key management should be toughened through secure storage (e.g. HSMs), strict access controls, monitoring and prepared key rotation procedures.

Finally, the PQC transition should be managed as a phased programme: tracking standards developments, testing hybrid schemes, and prioritising high-risk systems. These measures can and should be implemented now, to minimise the future impact of harvested encrypted data.

While this report provides general guidance on mitigating the HNDL threat, each organisation must conduct its own Quantum Threat Risk Assessment to identify specific vulnerabilities and prioritise mitigation efforts based on its unique data assets, systems and risk tolerance. Developing a phased implementation roadmap aligned with these findings is crucial for effective long-term security.

The quantum threat may be uncertain in its timing, but the imperative is clear: **action must begin now to adopt post-quantum cryptography and secure the future of digital communications.**

GLOSSARY

ASYMMETRIC ENCRYPTION or public key cryptography uses two related keys: a public key and a private key. The public key can be shared. The private key must remain secret. It is widely used for key exchange, digital signatures and identity systems.

AUTHENTICATION is the process of checking or verifying identity. For example, a website may authenticate itself with a certificate, and a user may authenticate with a password, a token or a key.

AUTHENTICITY means you can trust that a person, device, server or message is really what it claims to be. In simple terms, authenticity answers the question: 'Is this really from the right source?'. It helps prevent impersonation, fake websites and forged messages.

CERTIFICATE is a digital document that links a public key to an identity, such as a website name. It is usually issued or signed by a trusted authority. Certificates help browsers and other software decide whether they are talking to the real server.

CONFIDENTIALITY means that information is only visible to the people or systems that are allowed to see it. A common way to protect confidentiality is encryption, which scrambles data so outsiders cannot read it.

DECRYPTION is the reverse of encryption. It uses the correct key to turn ciphertext back into normal readable information.

DIFFIE-HELLMAN (DH) is a way for two parties to agree on a shared secret key over an insecure network without sending the secret key itself through a public channel. That shared secret can then help create a session key. It is a foundational idea behind many modern secure protocols.

DIGITAL SIGNATURE is a cryptographic way to sign a message or document using a private key so that others can verify it with the matching public key. Digital signatures support integrity, authenticity, and often non-repudiation.

EPHEMERAL KEY is a temporary cryptographic key that is created for a short period of time, usually for a single session or connection. It is used during a secure exchange (such as setting up an encrypted internet connection) and then discarded once it is no longer needed.

ENCRYPTION is the process of turning readable information into unreadable data called ciphertext. Only someone with the right key can turn it back into readable form.

FORWARD SECRECY means that if a long-term key is exposed in the future, past recorded sessions should remain protected. This is often

achieved with temporary key exchange methods such as ephemeral Diffie–Hellman.

HNDL describes an attacker strategy in which encrypted traffic or files are collected today and stored for the future. The attacker hopes to decrypt that material later when better tools or more powerful computers become available. This matters particularly when information stays sensitive for many years, such as in the case of government, health, legal or long-term research data.

HASH FUNCTION turns data of any size into a short fixed-size value. Even a tiny change in the original data should produce a quite different hash. Hashes are widely used to check integrity and support signatures and authentication mechanisms.

HOST KEY is the long-term key that an SSH server uses to identify itself to clients. If the host key changes unexpectedly, it may signal a server rebuild, a key rotation or an attack.

HYBRID KEY EXCHANGE combines a traditional method and a post-quantum method in the same connection. The goal is to get the benefits of both during the transition period.

INTEGRITY means that information has not been changed in an unauthorised way. The message you receive should be the same as the message that was sent.

KEM (KEY ENCAPSULATION MECHANISM) is a cryptographic method used to establish or transport a shared secret key securely. Many post-quantum designs use KEMs instead of classic Diffie–Hellman style key exchange.

NON-REPUDIATION means that someone cannot credibly deny an action they performed, such as signing a document or sending a signed message. It usually depends on digital signatures and reliable records.

OPENPGP is a standard and an ecosystem used to encrypt and sign emails, files and messages. It often creates a new session key for each message and then protects that key with public key cryptography. It is widely known from tools such as GnuPG.

PKI is the overall system of certificates, trusted authorities, rules and processes used to manage public keys and digital identity. It is the standardised trust framework behind much of the secure web and many enterprise security systems.

PQC refers to cryptographic algorithms designed to resist both today's normal computers and future quantum attacks that could break some current public key systems.

PRIVATE KEY is the secret partner to a public key. It may be used to decrypt, to sign, or to prove identity, depending on the protocol. If a private key is stolen, the identity or protection linked to that key may be compromised.

PUBLIC KEY is the key that can be shared openly. Depending on the system, it can be used to encrypt data, verify a signature or help establish a secure connection. The key point is not secrecy but trust: people must believe the public key really belongs to the right party.

SESSION KEY is a temporary symmetric key used for one connection, one conversation, or one message. It is usually created at the start of a secure session and then discarded later. Temporary keys reduce the amount of data exposed if a single key is ever compromised.

SSH (SECURE SHELL) is a protocol used for secure remote login, server administration, secure file transfer and encrypted tunnels. It protects command-line access and other remote management tasks.

SYMMETRIC ENCRYPTION uses the same secret key to encrypt and decrypt data. It is fast and efficient, so it is commonly used to protect the actual content of messages and connections. This is why many protocols first agree on a temporary symmetric key and then use it for the rest of the session.

TLS is the protocol that protects many online connections, including HTTPS websites. It helps verify the server, establish session keys and encrypt traffic. When you see a secure website connection, TLS is usually working in the background.

HARVEST NOW, DECRYPT LATER

PDF | ISBN: 978-92-9414-153-8 | DOI: 10.2813/1882350 | QL-01-26-020-EN-N

Neither the European Union Agency for Law Enforcement Cooperation nor any person acting on behalf of the agency is responsible for the use that might be made of the following information.

Luxembourg: Publications Office of the European Union, 2026

© **European Union Agency for Law Enforcement Cooperation, 2026**

Reproduction is authorised provided the source is acknowledged.

For any use or reproduction of photos or other material that is not under the copyright of the European Union Agency for Law Enforcement Cooperation, permission must be sought directly from the copyright holders.

While best efforts have been made to trace and acknowledge all copyright holders, Europol would like to apologise should there have been any errors or omissions. Please do contact us if you possess any further information relating to the images published or their rights holder.

Cite this publication: Europol, *Harvest now, decrypt later*, Publications Office of the European Union, Luxembourg, 2026.

This publication and more information on Europol are available on the Internet.

www.europol.europa.eu