ITUPublications

International Telecommunication Union

Standardization Sector

ITU-T Technical Report

(09/2023)

TR.qs-dlt

Guidelines for quantum-safe distributed ledger technology systems

Technical Report ITU-T TR.qs-dlt

Guidelines for quantum-safe distributed ledger technology systems

Summary

Technical Report ITU-T TR.qs-dlt assesses security threats to distributed ledger technology (DLT) systems when large-scale quantum computers are available. Construction methods for quantum-safe DLT systems are presented. Moreover, transition measures from the current to a quantum-safe DLT system are suggested. The accelerating development of quantum computing has a subversive impact on the field of traditional cryptography. DLT systems will be greatly affected by quantum computing as cryptography is one of their core technologies.

Keywords

Distributed ledger technology (DLT), migration, quantum computer, quantum key distribution (QKD), quantum-safe algorithm.

Note

Table of Contents

Technical Report ITU-T TR.qs-dlt

Guidelines for quantum-safe distributed ledger technology systems

1 Scope

This Technical Report provides guidelines for quantum-safe distributed ledger technology (DLT) systems, including:

– security assessments of cryptographic algorithms used in current DLT systems when large-scale quantum computers are available;

– construction requirements and guidelines for a quantum-safe DLT system; and

– measures for migration at the cryptographic algorithm level from the current to a quantum-safe DLT system.

2 References

None.

3 Definitions

3.1 Terms defined in elsewhere

This Technical Report uses the following terms defined elsewhere:

3.1.1 access control [b-ITU-T X.1252]: A procedure by which an administrator can restrict access to resources, facilities, services, or information based on pre-established rules and specific rights or authority associated with the requesting party.

3.1.2 account [b-ITU-T X.1400]: Representation of an entity whose data is recorded on a distributed ledger.

3.1.3 address [b-ITU-T X.1400]: Identifier for entity(s) performing transactions or other actions in a blockchain or distributed ledger network.

3.1.4 asset [b-ITU-T X.1400]: Representation of value.

3.1.5 asymmetric cryptographic algorithm [b-ITU-T X.810]: An algorithm for performing encipherment or the corresponding decipherment in which the keys used for encipherment and decipherment differ.

NOTE – With some asymmetric cryptographic algorithms, decipherment of ciphertext or the generation of a digital signature requires the use of more than one private key.

3.1.6 authentication [b-ITU-T Y.2014]: A property by which the correct identity of an entity or party is established with a required assurance. The party being authenticated could be a user, subscriber, home environment or serving network.

3.1.7 bitcoin [b-ITU-T X.1400]: An example of a blockchain using proof of work.

3.1.8 block [b-ITU-T X.1400]: Individual data unit of a blockchain, composed of a collection of transactions and a block header.

3.1.9 block header [b-ISO 22739]: Structured data that includes a hash link to the previous block, if present.

3.1.10 blockchain [b-ITU-T X.1400]: A type of distributed ledger which is composed of digitally recorded data arranged as a successively growing chain of blocks with each block cryptographically linked and hardened against tampering and revision.

3.1.11 Byzantine fault tolerance [b-ITU-T X.1400]: Property that enables a system to continue [to] operate properly even if some of its components fail or existence if there are intentional bad actors.

3.1.12 confidentiality [b-ITU-T X.800]: The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

3.1.13 consensus [b-ITU-T X.1400]: Agreement that a set of transactions is valid.

3.1.14 consensus mechanism [b-ITU-T X.1400]: Rules and procedures by which consensus is reached.

3.1.15 crash fault tolerance [b-ITU-T X.1400]: Property that enables a system to continue operating properly even if some of its components fail.

3.1.16 data integrity [b-ITU-T X.800]: The property that data has not been altered or destroyed in an unauthorized manner.

3.1.17 digital signature [b-ITU-T X.800]: Data appended to, or a cryptographic transformation of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery, e.g., by the recipient.

3.1.18 DLT; distributed ledger technology [b-ISO 22739]: Technology that enables the operation and use of distributed ledgers.

3.1.19 DLT system; distributed ledger system; distributed ledger technology system [b-ISO 22739]: System that implements a distributed ledger.

3.1.20 genesis block [b-ITU-T X.1400]: The first block in a blockchain that serves to initialize the blockchain.

3.1.21 immutability [b-ISO 22739]: Property of a distributed ledger wherein ledger records cannot be modified or removed once added to that distributed ledger.

3.1.22 key establishment [b-ISO/IEC 11770-4]: Process of making available a shared secret key to one or more entities.

NOTE – Key establishment includes key agreement, key transport and key retrieval.

3.1.23 Merkle tree [b-NIST IR 8202]: A data structure where the data is hashed and combined until there is a singular root hash that represents the entire data structure.

3.1.24 node [b-ITU-T X.1400]: Device or process that participates in a distributed ledger network.

3.1.25 permissioned [b-ISO 22739]: Requiring authorization to perform a particular activity or activities.

3.1.26 permissionless [b-ISO 22739]: Not requiring authorization to perform any particular activity.

3.1.27 proof of work [b-ITU-T X.1400]: Consensus process to solve a difficult (costly, time-consuming) problem that produces a result that is easy for others to verify.

3.1.28 proof of stake [b-ITU-T X.1400]: Consensus process, where an existing stake in the distributed ledger system (e.g., the amount of that currency that you hold) is used to reach consensus.

3.1.29 quantum computing [b-ISO/TS 80004-12]: Use of quantum phenomena for computational purposes.

3.1.30 quantum key distribution [b-ETSI GR QKD 007]: Procedure or method for generating and distributing symmetrical cryptographic keys with information theoretical security based on quantum information theory.

3.1.31 quantum safe [b-ETSI TR 103 619]: Not vulnerable to quantum computing attack.

3.1.32 symmetric cryptographic algorithm [b-ITU-T X.810]: An algorithm for performing encipherment or the corresponding algorithm for performing decipherment in which the same key is required for both encipherment and decipherment.

3.1.33 transaction [b-ITU-T X.1400]: Whole of the exchange of information between nodes. A transaction is uniquely identified by a transaction identifier.

3.1.34 wallet [b-ITU-T X.1400]: Software and/or hardware used to generate, manage and store both private and public keys and addresses, which enable distributed ledger technology (DLT) users to transact. Some wallets may interact with smart contracts and allow single and/or multi-signature.

3.2 Terms defined in this Technical Report

This Technical Report defines the following terms:

3.2.1 migration: Set of processes, procedures and technologies required to transition from a current to a quantum-safe distributed ledger technology system.

NOTE – Adapted from the migration entry in clause 3.1 of [b-ETSI TR 103 619].

3.2.2 nested hash function: A hash function formed by nesting at least two different types of hash function.

3.2.3 permissioned distributed ledger technology system: A distributed ledger technology system in which permissions are required to maintain and operate a node.

NOTE – Adapted from permissioned distributed ledger system in [b-ITU-T X.1400].

3.2.4 permissionless distributed ledger technology system: A distributed ledger technology system where permissions are not required to maintain and operate a node.

NOTE – Adapted from permissionless distributed ledger system [b-ITU-T X.1400].

3.2.5 quantum computer: Computer that uses quantum mechanical phenomena to operate.

NOTE – Adapted from [b-Wiki-QC].

3.2.6 quantum-safe cryptographic algorithm: A cryptographic algorithm that is resistant to attacks by both classical and quantum computers to keep information assets secure even after a large-scale quantum computer has been built.

NOTE – Adapted from [b-ETSI-QSC].

3.2.7 quantum-safe distributed ledger technology system: A distributed ledger technology system that uses quantum-safe cryptographic algorithms and is not vulnerable to quantum-computing attack.

3.2.8 quantum-secured distributed ledger technology system: A distributed ledger technology system that is protected by a quantum key distribution network working in the physical layer to defend against quantum-computing attacks.

4 Abbreviations and acronyms

This Technical Report uses the following abbreviations and acronyms:

AES	Advanced Encryption Standard
BIKE	Bit-Flipping Key Encapsulation

CA	Certification Authority
DES	Data Encryption Standard
DLT	Distributed Ledger Technology
ECDSA	Elliptic Curve Digital Signature Algorithm
FALCON	Fast Fourier Lattice-based Compact signatures Over Number theory research unit
HQC	Hamming Quasi-Cyclic
KEM	Key Encapsulation Mechanism
MSP	Membership Service Provider
NHF	Nested Hash Function
NP	Non-deterministic Polynomial
P2P	Peer to Peer
PBFT	Practical Byzantine Fault Tolerance
PKE	Public Key Encryption
PKI	Public Key Infrastructure
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
PQC	Post-Quantum Cryptograph
QKD	Quantum Key Distribution
RSA	Rivest–Shamir–Adleman
SIKE	Supersingular Isogeny Key Encapsulation
TLS	Transport Layer Security

5 Conventions

None.

6 Overview

DLT enables a number of participants whose trustworthiness is unknown to reach agreement by leveraging cryptographic algorithms and consensus protocols [b-ITU-T FG DLT D5.1] without relying on a centralized authority. Thanks to its accountability and transparency in transactions, DLT can be applied to a wide range of use scenarios, such as finance, manufacturing and healthcare.

Currently, most popular DLT systems deploy conventional cryptographic algorithms to enable secure transactions. For example, cryptographic hash algorithms are utilized to ensure the integrity of a transaction; digital signature algorithms are applied to ensure the authentication of a transaction. The security of these conventional cryptographic algorithms relies on some hard mathematical problems. Once these mathematical problems are solved, the security of the cryptographic algorithms is consequently compromised. This can have serious impacts on the security of DLT systems [b-ITU-T X.1401].

Recently a 54-qubit quantum computer has been announced, which has achieved quantum supremacy [b-Arute]. It has demonstrated that this quantum computer performs a calculation in 200 s that would have taken the world's most powerful supercomputer 10 000 years. Availability is anticipated of large-scale quantum computers in 10 years [b-Juskalian]. Large-scale quantum computers put most currently used cryptographic algorithms at risk [b-Rodenburg]. The security strength of a symmetric cryptographic algorithm is reduced to half due to Grover's quantum algorithm. Many commonly used asymmetric cryptographic algorithms, such as Rivest–Shamir–Adleman (RSA), digital signature algorithm and elliptic curve-based algorithms, provide no security due to Shor's quantum algorithm.

Most currently used DLT systems are vulnerable to quantum attacks due to the compromise of the underlying cryptographic algorithms in the quantum era. In general, there are two approaches to protect against quantum attacks in the context of DLT systems. One is called the quantum-secured DLT system where a quantum key distribution (QKD) network works in the physical layer to protect applications including the DLT system in the upper layer [b-Kiktenko], see Appendix IV. The other is called the quantum-safe DLT system that works in the application layer without relying on the security of the underlying layer. The main objective of this Technical Report is to study the quantum-safe DLT system.

A quantum-safe DLT system requires that countermeasures to protect against quantum-computing attacks are taken for their normal operation, which involves methods not only to replace conventional cryptographic algorithms with those that are quantum safe, but also to adopt quantum-safe algorithms that are appropriate to a DLT system. The lack of interoperation among the DLT systems has become a big barrier or bottleneck to information exchange across DLT systems. Each system has been developed according to its own requirements and methods, and without a standard giving guidance on the construction of a DLT system, even at a high level. Construction of quantum-safe DLT systems presents an opportunity to change this situation, as such systems are currently rarely available. Some guidance on the design of a quantum-safe DLT system, at least at a high level, needs to be proposed in order to avoid interoperational issues.

The rapid development and usage of DLT systems all over the world also place requirements on the guidance for transition to quantum-safe DLT systems in order to prevent an attacker from altering existing transactions and controlling user accounts. The various systems may have different requirements for the history data, migration time and organizational structure that will impact the update method. Methods (e.g., hard fork) to migrate and update from the current DLT system to one that is quantum safe are suggested.

This Technical Report describes the following.

– The security impact on DLT systems due to large-scale quantum computers, including threats to their cryptoalgorithms and interoperation at different quantum-safe levels.
– Guidelines and requirements for the construction of quantum-safe DLT systems. The guidelines include using quantum-safe cryptographic algorithms, identifying the factors that will impact implemention of quantum-safe algorithms for different usages or different types of DLT systems and identifying the guidance matrix model in the implementation process. Some components of DLT structure for a future smooth migration, such as the use of a quantum-safe ITU-T X.509 certificate when needed in DLT, is also included.

7 Security assessments on DLT systems

7.1 Impact of quantum computing on cryptographic algorithms

7.1.1 Impact on symmetric cryptographic algorithms

The encryption key is the same as the decryption key in symmetric cryptography, which is usually used for data encryption. The [b-Grover] quantum algorithm provides quadratic acceleration for unstructured search problems. When the Grover algorithm is applied to symmetric cryptography, the *n* bit key can be recovered by *O* (2*n*/2) quantum operations. The research shows that doubling the key length is sufficient to eliminate the threat of quantum computing to symmetric cryptographic algorithms. As an example, advanced encryption standard-256 (AES-256) will withstand quantum computer attacks until way after 2050 [b-ETSI ETSI GR QSC 006].

7.1.2 Impact on asymmetric cryptographic algorithms

There is a pair of public and private keys in an asymmetric cryptographic algorithm. To provide encryption, the private key for decrypting is securely kept by an individual while the public key for encryption is generally made public. Asymmetric cryptographic algorithms can be used in a variety of scenarios, such as digital signatures, identity authentication and data encryption. To provide a digital signature, the private key for digital signature generation is securely kept by an individual while the public key for digital signature verification is generally made public.

Most widely used asymmetric cryptographic algorithms are based on two well-known hard mathematical problems, which are those of integer factorization and discrete logarithms. The integer factorization problem involves finding the prime factors of a given integer. This is the basis of modern cryptographic algorithms like RSA, which relies on the difficulty of factorizing large integers to provide security. The discrete logarithm problem is another that is fundamental and involves finding the exponent (or logarithm) to which a given base must be raised to obtain a specific residue modulo a prime or a finite cyclic group. The widely used elliptic curve digital signature algorithm (ECDSA) depends on the discrete logarithm problem.

With quantum computers, the [b-Shor] algorithm can solve the integer factorization problem and also enables quantum computers to solve discrete logarithm problems on finite fields and elliptic curves. Both problems can be solved in polynomial time, which is reasonable to solve a problem efficiently. Therefore, quantum computers eventually weaken the security of the asymmetric cryptographic algorithms. As an example, it is optimistically estimated that as early as 2027 a quantum computer could exist that can break the elliptic curve signature scheme (e.g., ECDSA) in less than 10 min [b-Aggarwal].

7.1.3 Impact on hash algorithms

A hash algorithm can map a string of arbitrary length binary input values into a string of short, fixed length binary values, which is called hash value, also known as digest or fingerprint. Collision is a measure of the security of hashing algorithms. To find collisions, there is currently no publicly available quantum algorithm that is more effective than classical algorithms [b-Amy]. For example, SHA-2 and SHA-3 with an output length of at least 256 bits will withstand quantum-computing attacks until way after 2050 [b-ETSI ETSI GR QSC 006].

7.2 Impact of quantum computing on the DLT systems

Transaction procedures in DLT systems involve the usage of cryptographic algorithms (see Appendix I). Both permissionless and permissioned DLT systems have the common characteristics that the data on the ledger cannot be tampered with and the data on the ledger can be trusted. However, there are significant differences in the trust model, access control and account management, which lead to significant differences in the scope and intensity regarding the usage of asymmetric cryptographic algorithms.

7.2.1 The impact of quantum computing on permissionless DLT system

The permissionless DLT system has no access control. It opens to anyone so that any participant can perform operations on the ledger. Most permissionless DLT systems apply competing consensus mechanisms to determine the recording node and write transaction data into the block. The impact of quantum computing technology on various functions in a permissionless DLT system is as follows.

- **Account management**. The permissionless DLT system generally applies the user client (hereinafter referred to as the client) for account management. The client can contain multiple public-private key pairs for executing transactions, i.e., there is no one-to-one relationship between the client and the public-private key pairs. The account of the client is generally protected by using passphrases. As long as the effective length of the passphrase is not smaller than 256 bits, quantum computing technology cannot crack the user's private keys in reasonable time.
- **Access control**. It is not affected by quantum computing technology, because the permissionless DLT system does not have access control.
- **Transaction process**. When transacting on permissionless DLT system, the sender signs the transaction information with its own private key, and publishes the transaction over the peer-to-peer (P2P) network, including the transaction information, digital signature, receiver's address and public key corresponding to the private key. The recipient uses the received public key to verify the digital signature to determine the authenticity of the transaction information. Since transactions make use of asymmetric cryptographic algorithms, quantum-computing technology has a significant impact on transactions. The attacker can derive the private key from the public key on the transaction by using the Shor algorithm, thereby can tamper with the transaction information and forge the digital signature. Some permissionless DLT systems recommend that one public-private key pair be used for transaction only once so that this impact can be mitigated to a certain extent.
- **Consensus mechanism**. Competing consensus mechanisms (e.g., proof of work (PoW) and proof of stake (PoS)) are widely used in permissionless DLT systems, which involve the usage of hash functions more or less. At present, it is generally believed that the standardized hash function (such as SHA-256), which has not been cracked, can resist the attack of quantum computing, because the Grover quantum algorithm and its variants are not as fast as the traditional algorithm to find the collision solution of the hash function. Therefore, quantum-computing technology has no effect on most permissionless DLT systems. However, a few permissionless DLT systems, such as EOS, apply a cooperative mechanism (e.g., delegated PoS) to reach a common state among participants, where messages are usually signed to ensure their authenticity consensus mechanism. Delegated PoS is vulnerable to quantum-computing attacks as it involves the usage of asymmetric cryptographic algorithms.
- **Integrity protection of the ledger**. The integrity of the transaction data on the block is guaranteed by the hash value. Each block is linked with the previous block by using a hash function to calculate the hash value of the block header of the previous block, thereby forming a chain data structure. To tamper with the transaction data on the ledger, an attacker needs to be able to crack the hash function. It is known that standardized hash functions (such as SHA-256) that have not been cracked are resistant to quantum-computing attacks. Current research shows that the quantum-computing technology has basically no threat to the integrity of transaction data on the chain, as these data are chained by using hash function which is resistant to quantum-computing attacks.
- **Data confidentiality on the ledger**. If the data on the ledger is encrypted using a symmetric algorithm, the security strength is equivalent to the effect of halving the key length in classic calculations.
- **Message transmission**. Messages exchanged in the permissionless system are usually transmitted in clear text so that it is not affected by quantum computing.

7.2.2 Impact of quantum computing on permissioned DLT system

A permissioned DLT system, such as Hyperledger Fabric, relies on public key infrastructure (PKI) technology to control user access to the system. The participants in the DLT system do not fully trust each other. A permissioned DLT system is generally based on cooperative mechanisms to enable the distributed nodes to reach a consensus. The impact of quantum-computing technology on the various functions of permissioned DLT is as follows.

- **Account management**. Certificates are issued by the certification authority (CA) and the identity of the user is verified with these certificates. Generally, a user has only one certificate per account on a ledger in a permissioned DLT system. The access control of the private key corresponding to the public key in the certificate storage generally adopts the method of passphrase. As long as the effective length of the passphrase is not smaller than 256 bits, quantum-computing technology cannot obtain the access control to the private key within the effective time. However, because the certificate is public information, the attacker can derive the corresponding private key from the public key in the certificate by using the Shor algorithm. Thus, account management is vulnerable to quantum-computing attacks.
- **Access control**. Quantum-computing technology has a great influence on access control of a permissioned DLT system, which relies on PKI technology for access control. PKI is basically constructed by using an asymmetric cryptographic algorithm such as the digital signature algorithm. With the Shor algorithm, attackers can deduce the private key of a certificate from its public key, so that they can access the permissioned DLT system with any forged user identity.
- **Transaction process**. When transactions are performed on the DLT system, senders sign the transaction information with their private key, and publish the transaction over the network including the transaction information, digital signature and the certificate corresponding to the private key. The receiver uses the received public key to verify the signature and confirm the authenticity of the transaction information. Quantum-computing technology has a significant impact on the transaction because of the asymmetric algorithm used in the transaction. After cracking the private key of CA by running the Shor algorithm, attackers can forge any certificate and initiate a transaction. This makes transactions on the ledger lose authenticity.
- **Consensus mechanism**. Permissioned DLT systems mainly deploy cooperative consensus mechanisms, such as practical Byzantine fault tolerance (PBFT), delegated PoS and proof of authority (PoA). They usually use a signature to prove the origin and integrity of messages exchanged during the consensus procedure. They are therefore prone to quantum-computing attacks as digital signatures are generated by using asymmetric cryptographic algorithms, which can be broken by the Shor algorithm. As a result, consensus cannot be reached or its results are not authentic. Raft is an exception because its cooperative consensus mechanisms are not affected by quantum computing as its work does not involve the use of a cryptographic algorithm.
- **Integrity protection of the ledger**. Similar to the permissionless DLT system, the integrity of data on the block is guaranteed by a hash function. Blocks are linked based on the hash value of the block header. In order to tamper with the data on the chain, the attacker needs to be able to crack the hash function used. However, it is generally recognized that the standardized hash function can protect against quantum-computing attacks. Therefore, quantum-computing technology has no threat to the integrity of transaction data on the ledger.
- **Data confidentiality on the ledger**. If the symmetric algorithm is used to encrypt the data on the ledger, its security strength is equivalent to the effect of halving the key length in classical calculation.
- **Message transmission**. Messages exchanged in a permissioned DLT system may be protected by using transport layer security (TLS) channels in order to protect all information (including ledger) from leakage outside the group. However, this feature cannot be retained if large-scale quantum computers are available, as the establishment of TLS session keys usually relies on asymmetric cryptographic algorithms.

7.3 Summary

The impact of quantum-computing technology on permissionless DLT systems and permissioned DLT systems is summarized in Table 1.

Table 1 – Summary of the impact of quantum-computing technology on DLT systems

	Permissionless DLT systems	**Permissioned DLT systems**
Account management	Affected but can be mitigated	Broken
Access control	Not applied	Broken
Transaction process	Affected but can be mitigated	Broken
Consensus mechanism	Most not affected, a small part broken	Most broken, a small part not affected
Integrity protection of the ledger	Not affected	Not affected
Data confidentiality on the ledger	Affected but can be easily mitigated	Affected but can be easily mitigated
Message transmission	Not affected	broken

Functions such as integrity protection of the ledger and message transmission for permissionless DLT systems are not affected by quantum-computing attacks. Although quantum-computing technology has impacts on the security of the transaction process and account management in permissionless DLT systems, these can be mitigated by some measures mentioned in clause 7.2.1. Regarding consensus mechanisms, most of them used in permissionless DLT systems are not affected, and a small part are broken, in contrast, most of them used in permissioned DLT systems are broken, and a small part are not affected. For the permissioned DLT systems, most functions including the account management, access control, transaction process, and message transmission are at risk. This is because the operation of permissioned DLT systems relies on the heavy usage of asymmetric algorithms that will be broken in the quantum era.

8 Requirements and guidelines to build a quantum-safe DLT system

8.1 Requirements of a quantum-safe DLT system

8.1.1 Quantum-safe cryptographic algorithms

A quantum-safe DLT system should employ quantum-safe cryptographic (also called post-quantum cryptograph (PQC)) algorithms to protect against quantum-computing attacks. Quantum-safe cryptographic algorithms can be divided into those that are symmetric and asymmetric.

8.1.1.1 Quantum-safe symmetric cryptographic algorithms

It is easy to realize quantum-safe symmetric cryptographic algorithms by doubling the key size of currently used symmetric cryptographic algorithms. [b-CSA QSS] shows that the current recommended key size of 256 bits is considered to be safe, even against the Grover algorithm.

8.1.1.2 Quantum-safe asymmetric cryptographic algorithms

Quantum-safe asymmetric cryptographic algorithms need to be invented. The current consensus is that problems that are non-deterministic polynomial (NP) hard and NP complete problems cannot be solved in polynomial time even using quantum computers [b-Aaronson] [b-Buchman]. The quantum-safe asymmetric cryptographic algorithms are basically designed based on the NP-hard problem, which includes the following.

- **Lattice-based algorithms**. A lattice is the set of points in *n*-dimensional spaces with a periodic structure. The well-known NP-hard problems used for cryptography include: shortest vector problem. which is to find the shortest non-zero vector in a given lattice; the closet vector problem, which is find a lattice vector that minimizes the distance from another target lattice.
- **Code-based algorithms**. Code-based algorithms rely on some error-correcting codes, where encoding schemes are difficult to decode efficiently, even for a quantum computer. For example, the [b-McEliece] cryptosystem is based on the NP-hard problem of decoding a general linear code.
- **Multivariate algorithms**. The security of multivariate algorithms relies on the difficulty of solving systems of non-linear multivariate polynomial equations over finite fields, which has been demonstrated to be NP-hard [b-Garey].
- **Supersingular isogeny-based algorithm**s. Supersingular isogeny-based algorithms are constructed on the basis of the difficulty of recovering an unknown isogeny between a pair of supersingular elliptic curves that are known to be isogenous.

In addition, **hash-based asymmetric cryptographic algorithms**, whose security relies on the characteristics of the underlying cryptographic hash function, are also proven to resist quantum attacks.

In 2016, the National Institute of Standards and Technology (NIST) started to standardize PQC algorithms. In July 2022, NIST selected four algorithms for the fourth round as final candidates. For further details, see Appendix II.

8.1.2 Supporting heterogeneous nodes and clients

To support heterogeneous nodes and clients, there needs to be devised a quantum-safe DLT system, whose computing resource capability ranges from constrained (such as in Internet of things nodes and clients) to sufficient (such as in mining nodes), as heterogeneous nodes are possibly involved in running the DLT system. Efficient quantum-safe cryptographic algorithms require application in nodes and clients with constrained computing resources. In contrast, there is no strong need to utilize efficient quantum-safe cryptographic algorithms in nodes and clients with sufficient computing resources, although efficient cryptographic algorithms are preferred. Thus, a quantum-safe DLT system needs simultaneously to support various quantum-safe cryptographic algorithms so that a node and a client can choose the appropriate algorithms to meet their requirements.

8.1.3 Flexible deployment of cryptographic algorithms

In the conventional DLT system, such as Bitcoin, a unique set of cryptographic algorithms (including hash algorithms and signature algorithm) is deployed to serve the cryptographic functions required by the DLT system. There is no possibility that these algorithms can be replaced with new ones. Such design assumes that the confidence in the algorithms used is solid. However, this confidence may be broken for different reasons. For example, the SHA-1 algorithm has been considered to be insecure to compute the hash value as the computation cost to find a collision is far lower than expected [b-Wang]. In addition, the data encryption standard (DES) algorithm is not suggested for use as a modern computer can recover the key in a reasonable time due to the short key size used. In a nutshell, the conventional DLT system with a fixed set of cryptographic algorithms will be undermined if one of these algorithms is broken.

Confidence in quantum-safe algorithms is not well established as they are relatively new compared to those that are conventional. NIST has spent many efforts to standardize quantum-safe algorithms. Even if these algorithms become standards, they could be compromised in the future by currently unknown attacks. This has happened with conventional cryptographic algorithms, such as SHA-1. Therefore, a quantum-safe DLT system should be devised to support the flexible deployment of cryptographic algorithms. This means that various sets of quantum-safe algorithms may be provided in the system and that these algorithms may be replaced with new ones to ensure the security of the DLT system.

8.2 Guidelines to build a quantum-safe DLT system

8.2.1 Permissionless quantum-safe DLT system

A quantum-safe permissionless DLT system is composed of clients and nodes, as shown in Figure 1. The necessary functional modules in a client include: wallet; transaction; as well as cryptographic algorithm selection and determination. A node needs to have the following functional modules: transactions; consensus mechanism; block management; as well as cryptographic algorithm selection and determination.

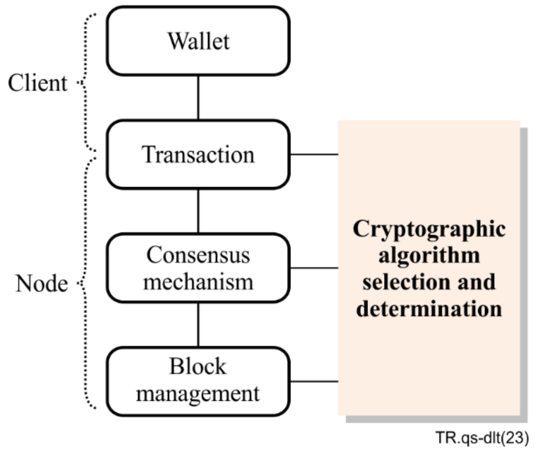

Figure 1 – Quantum-safe permissionless DLT system

A general rule to build a quantum-safe permissionless DLT system is that the quantum-safe rather than conventional cryptographic algorithms are deployed in the system. The substantial difference between a quantum-safe and a conventional DLT system is that a cryptographic algorithm selection and determination module is introduced into the latter to enable flexible deployment of cryptographic algorithms in the system, which is described in Appendix III. For each functional module of the system, the corresponding measures to protect against quantum-computing attacks are as follows. The effective length of the passphrase used to protect the wallet should not be smaller than 256 bits, so that attackers with large-scale quantum computers cannot acquire the private keys in the wallet. The client applies cryptographic algorithm selection and determination module to choose appropriate quantum-safe algorithms according to service and security requirements. The client signs the transaction with the chosen quantum-safe signature algorithm, so that the transaction cannot be forged by quantum attacks.

After receiving the transaction, the node verifies the signature and determines the validity of the chosen algorithms by using the cryptographic algorithm selection and determination module. The node then applies the determined algorithms to the consensus mechanism and block management modules. The functions of the block management module include forming the block and updating the ledger.

8.2.2 Permissioned quantum-safe DLT system

A quantum-safe permissioned DLT system is composed of clients and nodes, as shown in Figure 2. The necessary functional modules in a client include: account; access control; transaction; as well as cryptographic algorithm selection and determination. A node needs to have the following functional modules: access control; transactions; consensus mechanism; block management; as well as cryptographic algorithm selection and determination.

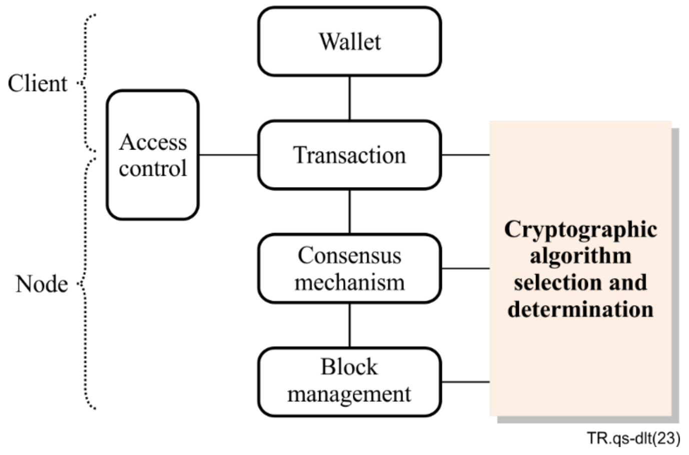

Figure 2 – Quantum-safe permissioned DLT system

The main difference between a permissionless and a permissioned DLT system is that access control is added in the latter. Only authorized clients and nodes can be granted to access the system. This is usually achieved by using a PKI to issue the certificates to clients and nodes. To prevent forgery of a certificate from a quantum attacker, a quantum-safe PKI that applies quantum-safe cryptographic algorithms to bind the identity and public key should be deployed. The remaining functional modules in a permissioned are similar to those in permissionless DLT system. Thus, countermeasures to prevent quantum attacks in a permissionless DLT system specified in clause 8.2.1 can be applied to one that is permissioned to make it quantum safe.

NOTE 1 – The wallet in a permissionless DLT system is equivalent to the account in one that is permissioned in the context of function.

NOTE 2 – Quantum-safe cryptographic algorithms are not necessarily considered in the mechanism module where cryptographic algorithms are not used during the consensus procedure. An appropriate quantum-safe cryptographic algorithm should be chosen in the mechanism module where a cryptographic algorithm is applied to the consensus process.

9 Measures for migration from the current to a quantum-safe DLT system

9.1 Overview

The migration from the current to a quantum-safe DLT system is essentially equivalent to the cryptographic algorithm transition, i.e., conventional cryptographic algorithms are replaced by those that are quantum safe in the DLT system. Usually it is a long ongoing process to perform the cryptographic algorithm transition. For example, the transition from DES to triple-DES to AES took place over decades [b-IETF RFC 7696]. It is anticipated that a switch to quantum-safe cryptographic algorithms will take at least a decade [b-Nyczepir]. Besides the long transition process, the switch to quantum-safe cryptographic algorithms in a DLT system faces a number of challenges. At least the following two issues have to be addressed in a DLT system when performing the cryptographic algorithm transition as they are critical to the security of transactions and operation stability.

1) Integrity verification of data on the current DLT system

A DLT system relies on underlying cryptographic algorithms to ensure the security of its digital assets. There is a strong need to take measures to detect whether data on the current DLT system have been altered while a cryptographic algorithm transition is carried out over quite a long time. There are two reasons for this. First, during the long transition period, a breakthrough on cracking a hash algorithm cannot be excluded. As a result, the data on the DLT system may lose integrity protection. Second, even in the quantum era, transactions on a quantum-safe DLT system may also need to quote data on the current one, because there are many digital assets that are recorded on the latter.

2) Asynchronous software update

A DLT system is inherently a decentralized system. Current DLT systems lack a mechanism to perform synchronous software updates. During the transition from the current to a quantum-safe DLT system, the clients and nodes of the DLT network may have inconsistent software versions, i.e., some clients and nodes have been upgraded to support the quantum-safe signature algorithm, while others still support only that which is conventional. A problem that needs solution is a method of ensuring that the DLT system can still conduct transactions without interruption when the software versions of the client and nodes are inconsistent.

9.2 Verification DLT system

To verify the integrity of data on the current DLT system, even if a hash algorithm used in the current DLT system is broken, a verification DLT system has been developed, which is described in detail in Appendix V. A verification DLT system is generated based on the current DLT system and a nested hash function (NHF) in order to detect whether the data on the current DLT system has been altered. Blocks in the current and those in the verification DLT system have a one-to-one correlation relationship. The body of each block in the verification DLT system except that of genesis is generated by using an NHF, which is formed by nesting at least two different types of hash function.

During the transition from a current to a quantum-safe DLT system, the data of a block on the current DLT system and the corresponding block on the verification DLT system need to be verified simultaneously. According to the data to be verified, the corresponding block to be checked is found in the current DLT system. When the block is verified in the current DLT system, the verification DLT system is used to detect whether the corresponding block has been tampered with. Any change to the data on the current DLT system can be detected on the corresponding block in the verification DLT system even if the hash function used in the current DLT system is compromised, since the the NHF used in the verification DLT system contains at least two different hash functions. For detailed steps to check the integrity of data on the current DLT system by using a verification DLT system, see Appendix V.

9.3 Hybrid signature method

The following methods used to sign transactions are applied to clients so that transactions can be verified in the nodes of a DLT system, even if the software versions of the client and nodes are inconsistent.

- The client that does not support quantum-safe signature algorithms signs all information in a transaction by using the conventional signature algorithm.
- The client that supports quantum-safe signature algorithms signs all information in a transaction by using the hybrid signature method.

The basic idea of the hybrid signature method is that the client first utilizes the conventional signature algorithm specified on the current DLT system to sign all information in a transaction to produce the first signature, and then applies the quantum-safe signature algorithm to sign all information in the transaction and the first signature to generate the second signature. When a node that does not support the quantum-safe algorithm receives a transaction request generated by using the hybrid signature method, it only needs to verify the first signature and ignore the second signature. In this way, the transaction verification in a node is not interrupted, even if the software versions of clients and nodes are different.

The detailed procedure to implement the hybrid signature method is described in Appendix VI.

9.4 Migration plan

The migration plan comprises the following three stages [b-ETSI TR 103 619]:

- inventory compilation: identification of the set of cryptographic assets and processes in the system;
- migration preparation;
- migration execution.

Appendix I

Overview of transaction procedure in conventional DLT systems

I.1 Classification of DLT systems

A DLT system can create and maintain an immutable ledger to record transactions among un- or semi-trusted participants by using cryptographic algorithms and consensus mechanisms. DLT systems can be classified into two categories according to the grant method to a participant of access the system: permissionless and permissioned [b-NIST IR 8202]. In a permissionless DLT system, anyone can access to the system without authorization, where participants are un-trusted. [b-Bitcoin] and [b-Ethereum] are typical permissionless DLT systems. In a permissioned DLT system, participants can access only after they have been proved to have the right to enter into the system, where participants do not fully trust each other (semi-trusted). [b-Hyperledger Fabric] is a typical permissioned DLT system. The transaction procedure in DLT systems involves the usage of cryptographic algorithms, which are introduced in clauses I.2 and I.3 for permissionless and permissioned DLT systems, respectively.

I.2 Transaction procedure in permissionless DLT system

The transaction procedure in permissionless DLT system is depicted in Figure I.1.

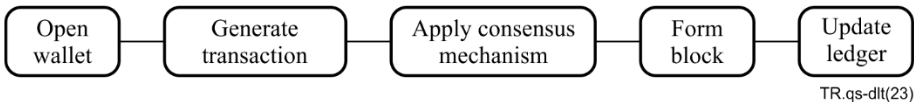

Figure I.1 – Transaction procedure in permissionless DLT system

The detailed steps in the transaction process are as follows.

1) Open wallet

Anyone can download the software for the permissionless DLT system, which contains the wallet. Accounts can be created without any identification or authorization process. Participants open the wallet with the information they know, such as the passphrase, in order to retrieve the private key stored in the wallet.

2) Generate transaction

Participants sign the transaction information with their private key in order to authorize the transaction. The transaction usually contains at least the transaction information, digital signature, public key of the sender and address of the receiver. The address in a permissionless DLT system is usually derived from the public key of a participant by hashing it.

3) Apply consensus mechanism

After generating the transaction, the participant propagates the transaction over the P2P network. A node in the network applies the consensus mechanism in order to acquire the right to form the block. Currently permissionless DLT systems mainly apply competing consensus mechanisms to select the node to create the block, where the resource of nodes (computing power, amount of asset) is the deciding factor for selection. PoW and PoS consensus mechanisms are typical examples, which are introduced as follows.

– **PoW**. The principle of PoW is that a node that first solves a computationally intensive puzzle problem wins the right to form the block. The so-called puzzle problem is to find an eligible *nonce* value that satisfies the requirement that a hashed block header must be less than (or equal to) the *target hash value* by repeatedly performing hash computations in a node.

– **PoS**. The principle of PoS is that a node is randomly selected to produce the next block based on its stake. The selected node is called a validator. Its security relies on the assumption that most stakes are held by honest participants who are intended to receive transaction fees as a reward in the system. PoS consumes comparatively less resources (electricity and processing power) as there is no need to perform the resource intensive computations found in PoW.

A small proportion of permissionless DLT systems apply a cooperative consensus mechanism to select the node to create the block without needing intensive computation, which is mainly deployed in a permissioned DLT system. Delegated PoS is a typical example, which is introduced in clause I.3.

4) Form the block

The node that wins in the consensus process verifies the digital signature in the transaction to ensure the authenticity of the transaction. Moreover, it checks that the transaction is correctly formatted and the providers of digital assets in each transaction. After that, the node forms the block by collecting a set of transactions and writing them into the block. A block is composed of a block header and block data. The block header contains metadata for this block. The block data contains a list of validated and authentic transactions, which are usually managed in a Merkle tree. The Merkle tree is used efficiently to summarize and verify the integrity of large sets of data. The leaves of the tree represent the transactions, and its root the digital fingerprint of the entire set of transactions that is computed from the leaves to the root by using hash function. The root of the tree is written into the block header.

5) Update the ledger

After forming the block, the accounting node broadcasts the block over a P2P network. The nodes in the network first check the validity of the block, and update the ledger independently by appending it to the last block of the ledger. The ledger is a blockchain that is constructed in such a way that each block header contains the hash digest of the previous block header.

Messages transmitted in the P2P network for a permissionless DLT system are usually in cleartext so that they can be seen by anyone. In a particular case, the transaction may contain encrypted information for secure communication between two participants.

I.3 Transaction procedure in a permissioned DLT system

The transaction procedure in a permissioned DLT system is depicted in Figure I.2.

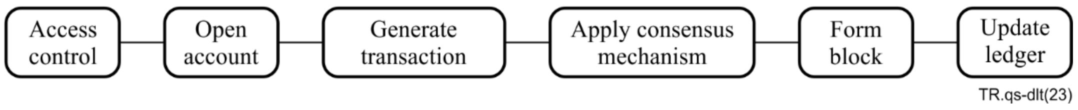

Figure I.2 – Transaction procedure in a permissioned DLT system

The detailed steps in the transaction process are as follows.

1) Access control

A user can gain access to a permissioned DLT system only after it has been granted the access right. This is usually achieved by using membership service provider (MSP) defined in Hyperledger Fabric [b-Androulaki]. An MSP maintains the identities of all nodes in the system and is responsible for issuing node credentials that are used for authentication and authorization. MSP relies on the PKI to manage node certificates for binding public keys and identities. For this, either a standard CA or a stand-alone CA can be exploited. Each node has its unique identity in a ledger, i.e., it has one certificate.

2) Open account

A user can retrieve its private key directly without inputting a passphrase if the account is not protected by one (Fabric manages the account in this way). Users can retrieve their private keys by inputting the passphrase they know if the account is protected by one.

3) Generate transaction

Users sign transaction proposals with their private key. A transaction proposal contains the identity of the sender, the transaction payload and transaction identifier. A transaction at least contains the certificate of the sender, transaction proposal and digital signature.

4) Apply consensus mechanism

After its generation, the participant delivers the transaction to the permissioned DLT system for implementation of the consensus mechanism. A permissioned DLT system mainly applies a cooperative consensus mechanism to reach the same state among participants, where one leader may be cooperatively selected by them. A typical example follows.

– **Raft**. This is a consensus algorithm designed to achieve the same state among a cluster of computers by assuming that all nodes are trusted but not excluding that some nodes in the network may crash (fault-tolerance) [b-Raft]. Only the elected leader is responsible for accepting new transactions and replicating these transactions for other followers. After the leader receives feedback from a certain number of followers who have written the transactions, the transactions will be committed. The messages exchanged in the Raft algorithm are not signed and verified as it is assumed that all participants are trusted.

– **PBFT**. This is a consensus algorithm that tolerates Byzantine faults [b-Castro], which assumes that the number of fault-tolerant nodes is less than one-third of the total number of nodes. After passing three phases, i.e., pre-prepared, prepared and commit, nodes in the network can reach valid consensus. PBFT applies a digital signature to prevent spoofing and replays, and to detect corrupted messages.

– **PoA**. This is an efficient consensus algorithm by assuming that partial nodes in the network are trusted. A subset of nodes on the network is designated as containing fully trusted validators, i.e., authorities, and cooperates to maintain steady operation of the blockchain. Digital signatures are applied to messages to ensure their authenticity.

– **Delegated PoS**. A small number of participants is treated as trustworthy (delegates) to write the data on the ledger by the system. Participants vote to elect and revoke the rights of delegates by using cryptographically signed messages.

It is worth noting that most consensus mechanisms used in a permissioned DLT system exploit the digital signature to ensure the authenticity of the messages. Raft, which does not use a digital signature, is the exception.

5) Form the block

After receiving transactions, the selected leader (i.e., the orderer in Fabric) establishes a total order on all submitted transactions, and assembles multiple transactions into blocks by applying the techniques used in the permissionless DLT system. The leader then broadcasts the blocks to nodes in the network.

6) Update the ledger

On its reception, nodes in the network independently validate the block and append it to the locally stored ledger.

Messages transmitted in the network for a permissioned DLT system could be set to be visible only to members by establishing a TLS channel among nodes. It is possible that these messages are visible to anyone by disabling the TLS channel among nodes.

Appendix II

NIST post-quantum cryptograph standardization

Since 2016, NIST has made many efforts to promote the standardization of PQC algorithms, i.e., quantum-safe asymmetric cryptographic algorithms. In the first round, NIST accepted 69 candidate schemes, including 20 digital signature schemes and 49 public key encryption (PKE) or key encapsulation mechanisms (KEMs). On 30 Jan 2019, NIST selected 26 algorithms as the second round of candidate algorithms, including nine digital signature schemes, and 17 PKE and key establishment schemes [b-NIST IR 8240]. In July 2020, NIST selected seven algorithms as the third round of final algorithms [b-NIST IR 8309], including four PKE and key establishment schemes, which are CRYSTALS-Kyber, Classic McEliece, Number Theory Research Unit and SABER, and three digital signature schemes, which are CRYSTALS-Dilithium, fast Fourier lattice-based compact signatures over Number Theory Research Unit (FALCON) and Rainbow. At the same time, NIST selected eight algorithms as alternative candidate algorithms for the third round, including five PKE and key establishment schemes, and three digital signature schemes. In the context of asymmetric cryptographic algorithms, just signature algorithms are usually involved in the DLT systems. Thus the performance of quantum-safe signature algorithms, including public key size, signature size and security level, is an important factor to choose an algorithm to be deployed in a DLT system.

In July 2022, NIST selected four algorithms as the fourth round of final algorithms [b-NIST IR 8413]. NIST identified four candidate algorithms for standardization: one for key establishment and three for digital signature. NIST will recommend two primary cryptographic algorithms to be implemented for most use cases: CRYSTALS-Kyber (key-establishment) and CRYSTALS-Dilithium (digital signatures). In addition, the additional signature schemes FALCON and SPHINCS+ will also be standardized.

CRYSTALS-Kyber (key-establishment) and CRYSTALS-Dilithium (digital signatures) are standardized for their strong security and excellent performance. FALCON will also be selected since there may be use cases for which CRYSTALS-Dilithium signatures are too large. SPHINCS+ will also be selected to avoid relying only on the security of lattices for signatures.

The following candidate KEM algorithms will advance to the fourth round: bit-flipping key encapsulation (BIKE), Classic McEliece, Hamming quasi-cyclic (HQC), supersingular isogeny key encapsulation (SIKE). Both BIKE and HQC are based on structured codes, and either would be suitable as a general-purpose KEM that is not based on lattices. NIST expects to select at most one of these two candidates for standardization at the conclusion of the fourth round. SIKE will not be considered further for standardization, although it holds small key and ciphertext sizes. This is because it has been broken [b-Castryck] just after publication of the report NIST IR 8413. Classic McEliece was a finalist but is not being standardized by NIST at this time. Although Classic McEliece is widely regarded as secure, NIST does not anticipate it being widely used due to its large public key size. NIST may choose to standardize Classic McEliece at the end of the fourth round.

In Table II.1, a comparison between conventional signature algorithms and NIST PQC signature candidates is given.

Table II.1 – Comparison between traditional signature algorithms and NIST PQC signature algorithms to be standardized

Classification	Problem base	Algorithm	Public key size (bytes)	Private key size (bytes)	Signature size (bytes)	Security strength of quantum algorithm
Conventional signature algorithm	Integer Factorization	RSA 3072	384	384	384	–
	EC Discrete logarithm	ECDSA 384	48	48	48	–
Signature algorithms to be standardized	Lattice-based	CRYSTALS-Dilithium II	1 312	2 528	2 420	Security category 2
		CRYSTALS-Dilithium III	1 952	4 000	3 293	Security category 3
		CRYSTALS-Dilithium V	2 592	4 864	4 595	Security category 5
		FALCON 512	897	7 553	666	Security category 1
		FALCON 1024	1 793	13 953	1 280	Security category 5
	Hash-based	SPHINCS+-128s	32	64	7 856	Security category 1
		SPHINCS+-192s	48	96	16 224	Security category 3
		SPHINCS+-256s	64	128	29 792	Security category 5

NOTE – NIST specifies five security strength categories to categorize the computational complexity of attacks [b-NIST SREC].

A transaction message in DLT system usually contains the public key and signature of the sender besides the transaction information. Thus, the quantum-safe signature algorithm that has the smallest sum of public key size and signature size is preferred for use in the DLT system. As shown in Table II.1, FALCON has the smallest sum of public key size and signature size among all NIST PQC signature candidates. Thus, it may be the best choice for use as a quantum-safe DLT system.

Appendix III

Cryptographic algorithm selection and determination mechanism in DLT systems

This appendix provides the cryptographic algorithm selection and determination mechanism in DLT systems. The basic idea of this solution is to make the software version numbers of all clients and network nodes in the DLT system consistent through the synchronization mechanism for software version number. When the software version number is consistent, the user selects the cryptographic algorithms and determines the algorithms.

III.1 Permitted cryptographic algorithm bundle

The DLT system is equipped with the permitted cryptographic algorithm bundle, including a permitted signature algorithm list, a permitted hash algorithm list and a permitted encryption algorithm list, which are described as follows.

Signature algorithm list = {*signature algorithm 1, signature algorithm 2, ... Signature algorithm n*}

Example: signature algorithm list = {*ECDSA, FALCON 512, CRYSTALS-Dilithium II*}

Hash algorithm list = {*hash algorithm 1, hash algorithm 2, ... Hash algorithm m*}

Example: hash algorithm list = {*SHA-256, SHA-384, SHA-3*}

Encryption algorithm list = {*encryption algorithm 1, encryption algorithm 2, ... Encryption algorithm k*}

Example: encryption algorithm list = {*AES 256, ZUC-256*[b-ZUC-256], *SNOW-V* [b-Ekdahl]}

A user can select a set of cryptographic algorithms from the permitted cryptographic algorithm bundle. The selected set of cryptographic algorithms at least contains one signature algorithm and one hash algorithm.

III.2 Synchronization mechanism for software version number

All clients and network nodes in the DLT system set a unified timer T to start software version update. Each client has its own time point of joining the DLT network as PTJ (point of time to join). For example, the time point of joining the DLT network of client 1 is PTJ1, and its corresponding startup time points for software version update are PTJ1 + T, PTJ1 + 2T, PTJ1 + 3T. Similarly, each network node has its own activation time point as PTA (point of time for activation). For example, the activation time point of network node 1 is PTA1, and its corresponding startup time points for software version update are PTA1 + T, PTA1 + 2T, PTA1 + 3T. The software version number synchronization mechanism of DLT system is shown in Figure III.1.

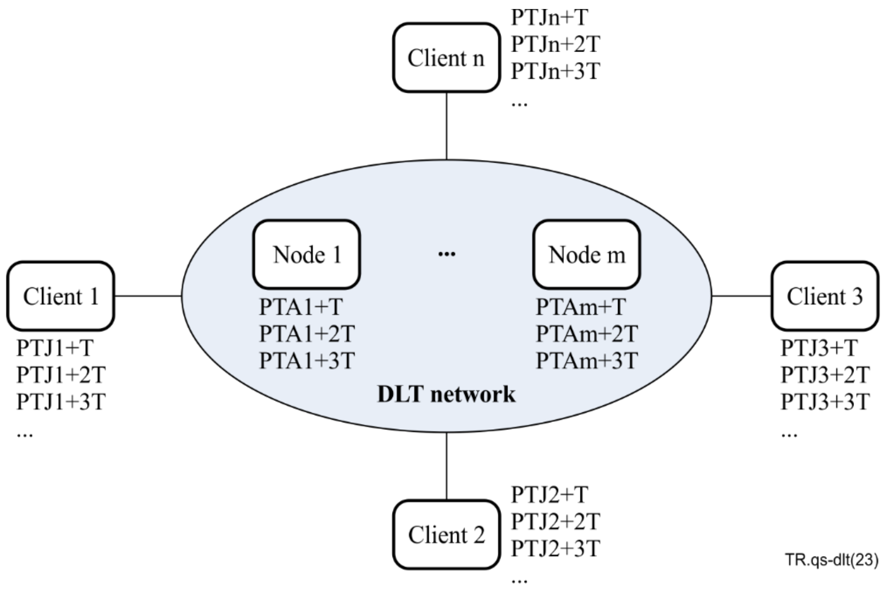

Figure III.1 – Software version number synchronization mechanism

When a certain client or network node reaches the time point of starting the software version update, it will query the software version server for the latest software version. The initiator sends the information to other clients and network nodes according to the latest software version information received. All clients and network nodes are upgraded according to the latest software version information (all download the latest version from the software version server), so that the software version numbers of all clients and network nodes are consistent. No upgrade is required to clients and network nodes that are the latest software version. At the same time, each client and network node update the time point of starting the software version update, that is, add one T to the current update time point.

When the client accesses the network or the network node is activated, the previously mentioned software version update process will also be started.

III.3 Procedure to select and determine cryptographic algorithms

The procedure to select and determine cryptographic algorithms is shown in Figure III.2.

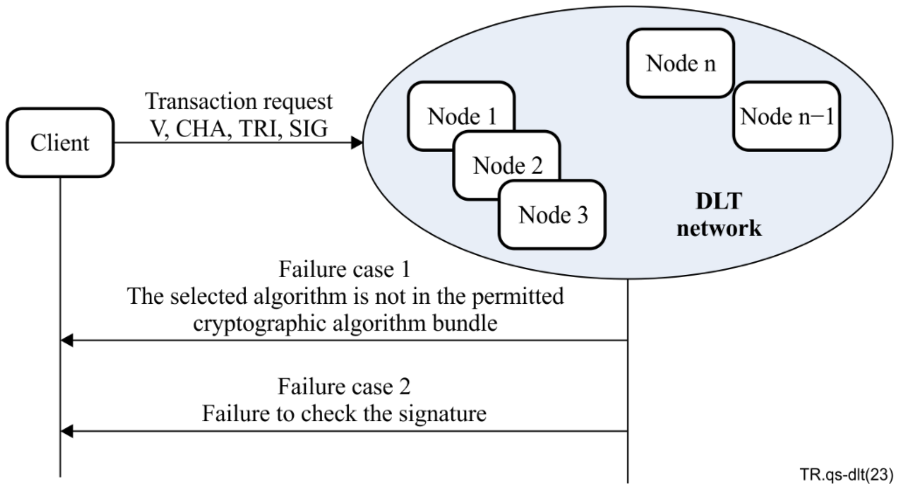

Figure III.2 – Procedure to select and determine cryptographic algorithms

The detailed process to select and determine cryptographic algorithms is as follows.

1. The user applies the client to select a set of cryptographic algorithms (or alternatively uses the default system settings related to the cryptographic algorithms) and generates a transaction request message, whose content includes the client software version number (V), the selected cryptographic algorithms, transaction-related information and signature. The client broadcasts a transaction request message to the DLT network.

   For a permissionless DLT system, the transaction-related information is the transaction content, the public key of the sender and the address of the recipient. For a permissioned DLT system, the transaction-related information is the sender's certificate, the transaction proposal.

2. After each node of the DLT network receives the transaction request message, it compares the selected cryptographic algorithms with the permitted cryptographic algorithm bundle. If found, the user applies the selected signature algorithm to verify the transaction message. After signature verification is successful, the nodes in the network use the selected cryptographic algorithms in the subsequent transaction process, including applying the consensus mechanism, forming blocks and updating the ledger.

3. After each node of the DLT network receives the transaction request message, it may fail to process it, including the following situations.

   1) The selected algorithm is not in the permitted cryptographic algorithm bundle: the node returns a message to the client indicating that it is not there, and informs the client of its software version number. After receiving the message returned by the node, the client checks whether its software version number is lower than that of the node in the network, and if so, the user updates the client software. The node in the network compares its software version number with that of the client, and if it is lower, the node in the network updates its software.

   NOTE – Under normal circumstances, the client software version number in the DLT system is consistent with the software version number of network node on the software server, i.e., the client software number is changed, and the software version number of network node is also changed accordingly.

   2) Signature verification failure: Although the selected algorithms are in the permitted cryptographic algorithm bundle, the network node fails to verify the signature in the message of the transaction request. The node returns a message to the client that the verification of the signature has failed. After receiving this message, the client may re-initiate the transaction request.

Appendix IV

Quantum-secured DLT system

A quantum-secured DLT system relies on a QKD network to protect against quantum-computing attacks. There are two kinds of approaches to building a quantum-secured DLT system: loose couple and tight couple. The former is a straightforward method, where the underlying QKD network and upper-layer DLT system run independently. The QKD network provides the secure links between DLT system nodes, which are quantum resistant so that transactions among DLT system nodes cannot be attacked by adversaries with quantum computers. The advantage of this approach is that conventional DLT systems can still be used without change. [b-Kiktenko] is a typical example of the tight couple approach, where each pair of nodes in a QKD network establishes an information-theoretically secure key, and these established keys in a QKD network are applied to DLT system nodes to generate authentication tags to verify the validity of transactions, as shown in Figure IV.1. This approach is resilient to quantum-computing attacks as it applies an information-theoretically secure hash function with keys established in a QKD network to check the integrity of transactions instead of applying a conventional digital signature scheme. The tight couple approach is more complicated than the loose one, as the conventional DLT system has to be modified by replacing the digital signature scheme with the information-theoretically secure hash function. The deployment of a quantum-secured DLT system is inflexible compared to that of a quantum-safe DLT system, as it requires that the underlying QKD network be available.

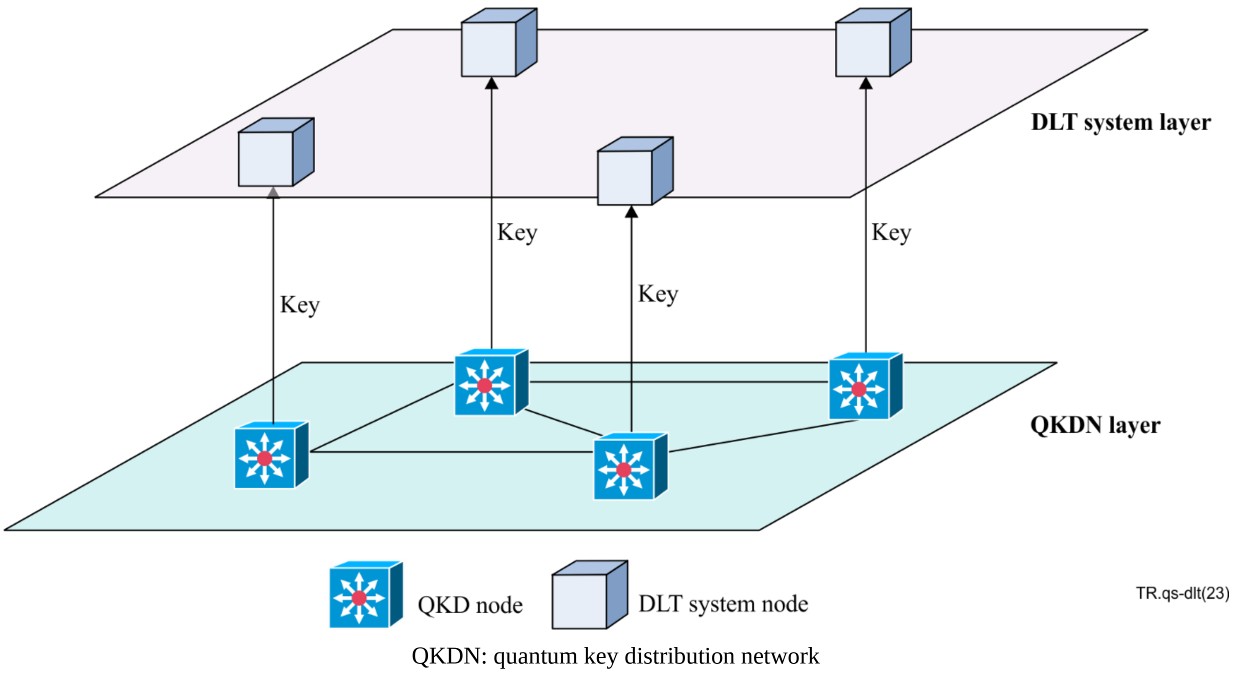

Figure IV.1 – Tight couple approach for quantum-secured DLT system

Appendix V

Verification DLT system

V.1 Principle

A verification DLT system is generated on the basis of the current DLT system by using the NHF. It serves to detect whether the data on the current DLT system has been tampered with. A block of the verification DLT system is composed of a header and a body. The block header contains the block association indication corresponding to the block of the current DLT system, which can be either a timestamp (such as that in the block header of Bitcoin) or a block number (such as that in the block header of Hyperledger). The block body of the genesis block contains a description of NHF to indicate which hash functions are used to form the NHF. With the exception of the genesis block, the body of each block is generated by using the NHF to hash the block header, the previous block, and the block associated with this block on the current DLT system. The structure of the verification DLT system is shown in Figure V.1.

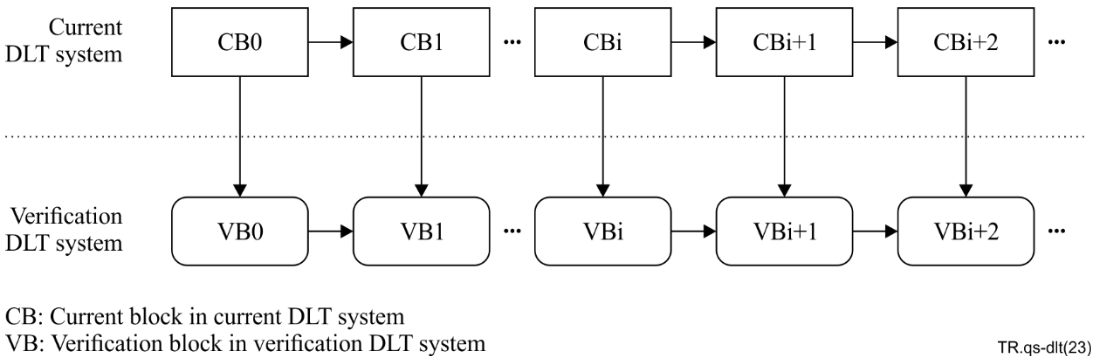

Figure V.1 – Structure of the verification DLT system

V.2 Generation

1) Genesis block

The block header of the genesis block of the verification DLT system is composed of the timestamp or block number in the block header of the genesis block of the current DLT system, i.e.,

Block header: *VB0. header = CB0. header. (TS/BN)*

Here *VB0* represents the genesis block of the verification DLT system, the *header* represents the block header, *CB0* represents the genesis block of the current DLT system, *TS* (Time Stamp) represents the timestamp, and *BN* (Block Number) represents the block number.

The block body of the genesis block of the verification DLT system contains a description of the NHF, indicating which hash functions are used. An NHF is specified as follows:

$$\text{NHF} = H1\ (H2\ (\ldots(Hn(M)))$$

In the formula, NHF nests hash functions $H1$, $H2$... Hn. Here $H1$, $H2$... Hn are different types of hash function, so even if attackers break one of the hash algorithms, they cannot crack the whole NHF. The hash operation of message M starts from the innermost hash function, namely Hn (M), and the operation result is input into the upper hash function, until the outermost hash function $H1$. In the formula, $n \geq 2$, i.e., NHF nests at least two hash functions, e.g., SHA-256 (SHA-3 (M)).

2) Ordinary block

The block header of the ordinary block *VBi* ($i \geq 1$) of a verification DLT system is composed of the timestamp or block number in the block header of the ordinary block *CBi* of the current DLT system, that is:

Block header: *VBi. header = CBi. header. (TS/BN)*

The block body of the ordinary block *VBi* of the verification DLT system is the operation result of the NHF. The input of the NHF is *VBi. header* (block header), *VBi*−1 (the previous block), and *CBi* (the corresponding block of the current DLT system). The block body of *VBi* can be expressed as:

Block body: *VBi. body* = NHF (*VBi. header, VBi−1, CBi*)

NHF hash operation contains the previous block, so as to realize its link with this block. This chained data structure increases the difficulty for attackers to forge data.

V.3 Verification

During the transition from the current to the quantum-safe DLT system, in order to confirm that the data in the chain have not been tampered with, it is necessary to verify both the data on the current and verification DLT systems. According to the data to be verified, the corresponding block *CBi* in the current DLT system is found. The block verification mechanism of the current DLT system is applied to verify the block. If the verification is successful, the corresponding block *VBi* and the previous block *VBi*−1 are found on the verification DLT system through the timestamp or block number in the block header of the current DLT system. The NHF is used to hash the block header *VBi.header* of the corresponding block, the previous block *VBi*−1 of the corresponding block, and the block *CBi* of the current DLT system to check whether the result is equal to the block body *VBi.body* of the corresponding block, i.e.,

$$VBi.body \triangleq \text{NHF}\,(VBi.header,\ VBi-1,\ CBi)$$

Here ≙ represents comparison operation. If the operation result of the NHF is equal to *VBi.body*, it can be affirmed that the block *CBi* on the current DLT system has not been tampered with, as long as one of the hash functions used in the NHF has not been cracked.

Appendix VI

Hybrid signature method

VI.1 Principle

During the transition from the current to the quantum-safe DLT system, the software versions of the clients and nodes in the DLT network may be inconsistent. There are four cases when a client initiates a transaction to a node: (1) neither the client nor the node verifying the transaction supports the quantum-safe signature algorithm; (2) the client does not support the quantum-safe signature algorithm, but the node verifying the transaction already supports the quantum-safe signature algorithm; (3) the client supports the quantum-safe signature algorithm, but the node verifying the transaction does not support the quantum-safe signature algorithm; (4) Both the client and the node verifying the transaction support the quantum-safe signature algorithm. The four scenarios are shown in Figure VI.1.

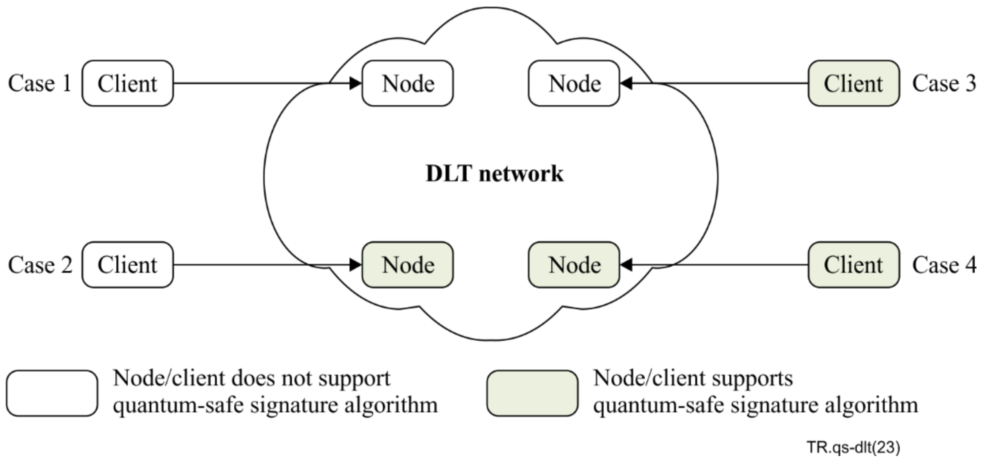
Case 1
Client
Node
Node
Client
Case 3
DLT network
Case 2
Client
Node
Node
Client
Case 4
Node/client does not support quantum-safe signature algorithm
Node/client supports quantum-safe signature algorithm
TR.qs-dlt(23)

Figure VI.1 – Four cases of transactions

In order to enable transactions to be verified by nodes in the DLT network that do not support quantum-safe algorithms, clients that do support quantum-safe algorithms need to use a hybrid signature method for signing transactions. The basic idea is that the client first uses the conventional signature algorithm specified on the current DLT system to sign all information in the transaction to generate the first signature, and then signs all information in the transaction and the first signature to generate the second signature. When a node that does not support the quantum-safe algorithm receives a transaction request generated by using the hybrid signature method, it only needs to verify the first signature, and ignore the second signature without performing signature verification. As a result, the transaction verification in the DLT network will not be interrupted, even if the software versions of nodes in the network are different.

There is a significant difference between permissionless and permissioned DLT systems in the management of public keys. The permissionless DLT system does not use certificates to manage public keys, while the permissioned DLT system does. As a result, current permissionless and permissioned DLT systems apply the hybrid signature method to process transaction information differently when transitioning to quantum-safe DLT systems. The following describes transition schemes from current permissionless DLT systems and permissioned DLT systems to quantum-safe DLT systems.

VI.2 Permissionless DLT system

The transaction request initiated by the client in the current permissionless DLT network includes the transaction content (*TC*), the sender's conventional public key (*CPK*), the receiver's address (*AD*), and the signature based on the conventional algorithm (*CSign*).

During the transition to quantum-safe DLT system, in order to prevent degradation attacks, that is, to prevent attackers from degrading the hybrid signature method to the conventional signature method, the transaction needs to carry the signature type identifier (*STI*), which is used to indicate whether the signature adopts the hybrid signature method or the conventional signature method. The *STI* and other information in the transaction are first signed by using the conventional signature algorithm. This ensures that the *STI* is not tampered with by attackers. The network node can determine whether it is under degradation attack according to the *STI* and the number of signatures (one or two).

NOTE – Network nodes that do not support quantum-safe algorithms need to be slightly upgraded to be able to identify *STI*.

(1) The first transaction case

When a client initiates a transaction to a node, neither the client nor the node verifying the transaction supports the quantum-safe signature algorithm.

The transaction request initiated by the client includes the *TC*, the sender's *CPK*, the *AD*, the *STI*, and the *CSign*. The calculation of *CSign* is to sign all the information in the transaction, which can be expressed as follows:

$$CSign = Signc\ (TC, CPK, AD, STI)$$

Here *Signc* is the conventional signature algorithm specified on the current DLT system.

After receiving the transaction request, the node verifying the transaction first detects the *STI* field to determine whether it is a conventional signature method. Based on the conventional signature algorithm, the signature *CSign* is verified by using the sender's *CPK*.

(2) The second transaction case

When a client initiates a transaction to a node, the client does not support the quantum-safe signature algorithm, but the node verifying the transaction already does so.

In this case, the generation of the transaction request message at the client and the validation at the network node are the same as in the first transaction case.

(3) The third transaction case

When a client sends a transaction to a node, the client supports the quantum-safe signature algorithm, but the node verifying the transaction does not support the quantum-safe signature algorithm.

The transaction request initiated by the client includes the *TC*, the sender's *CPK*, the *AD, STI, CSign*, i.e., first signature, quantum-safe signature algorithm identifier (*QSSAI*), The sender's quantum-safe public key (*QPK*) and signature based on quantum-safe algorithm (*QSign*), i.e., second signature. The calculation of *CSign* is to sign all the information in the transaction, while the calculation of *QSign* is to sign all the information in the transaction and *CSign,* which are expressed as follows:

$$CSign = Signc(TC, CPK, AD, STI)$$

$$QSign = Signq(TC, CPK, AD, STI, QSSAI,\ QPK, CSign)$$

Here, *Signc* is the conventional signature algorithm specified on the current DLT system, and *Signq* is the quantum-safe signature algorithm specified by *QSSAI.*

After the transaction request is received by the node that verifies the transaction, the *STI* field is detected as the hybrid signature method. However, because the node that verifies the transaction does not support the quantum-safe signature algorithm, the node only applies the *CPK* to verify the *CSign.*

(4) The fourth transaction case

When a client sends a transaction to a node, the client and node verifying the transaction both support quantum-safe signature algorithms.

In this case, the generation of transaction request message at the client is the same as the third transaction case.

After the transaction request is received by the node that verifies the transaction, the *STI* field is detected as the hybrid signature method. The node first applies the *CPK* to verify the *CSign*, and then uses the *QPK* to verify the *QSign* by using the algorithm specified by *QSSAI*.

VI.3 Permissioned DLT system

The transaction request initiated by the client in the current permissioned DLT network includes the sender's certificate (*Cert*), and the transaction proposal (*TP*) is based on *CSign*.

When the client supports the hybrid signature method, the ITU-T X.509 certificate that supports multiple cryptographic algorithms can be used [b-ITU-T X.509], which is the certificate with an extension (*CertE*). In addition to the information of the conventional cryptographic algorithm, it also contains the relevant information of the alternate cryptographic algorithm in the extension of the certificate, including the alternate public key information in the `AltPublicKeyInfo` extension, the alternate signature algorithm value in the `altSignatureAlgorithm` extension, and the alternate signature value in the `altSignatureValue` extension [b-ITU-T X.509]. Here, the alternative cryptographic algorithm can be a quantum-safe signature algorithm.

The conventional certificate *Cert* or *CertE* can be carried in the transaction. The former indicates that the transaction employs the conventional signature method, while the latter indicates that the transaction uses the hybrid signature method. The certificate in the transaction is protected by the signature *Csign* and any alteration on the certificate by an attacker can be detected by checking the signature *Csign*. Therefore, it does not need to carry an *STI* in the transaction request to prevent downgrade attack.

NOTE – Nodes that do not support quantum security algorithms need to be slightly upgraded to be able to identify the *CertE*.

(1) The first transaction case

When a client initiates a transaction to a node, neither the client nor the node verifying the transaction supports the quantum-safe signature algorithm.

The transaction request initiated by the client includes the sender's *Cert*, *TP*, and *CSign*. The calculation of *CSign* is to sign all the information in the transaction, which can be expressed as follows:

$$CSign = Signc(Cert, TP)$$

Here *Signc* is the conventional signature algorithm indicated in *Cert*.

After receiving the transaction request, the node verifying the transaction first detects that the certificate is a conventional certificate and thus determines that a conventional signature method has been applied to the transaction. The *CPK* is used to verify the signature based on the conventional signature algorithm, which is obtained from the certificate *Cert*.

(2) The second transaction case

When a client initiates a transaction to a node, the client does not support the quantum-safe signature algorithm, but the node verifying the transaction already supports the quantum-safe signature algorithm.

In this case, the generation of the transaction request message at the client and the validation at the network node are the same as in the first transaction case.

(3) The third transaction case

When a client initiates a transaction to a node, the client supports the quantum-safe signature algorithm, but the node verifying the transaction does not.

The transaction request initiated by the client includes the sender's *CertE*, *TP*, and signature based on conventional algorithm (*CSign*), i.e., first signature, and *QSign*, i.e., second signature. The calculation of *CSign* is to sign all the information in the transaction, while the calculation of *QSign* is to sign all the information in the transaction and *CSign*, which are represented as follows:

$$CSign = Signc(CertE, TP)$$

$$QSign = Signq(CertE, TP, CSign)$$

Here *Signc* is the conventional signature algorithm indicated by *CertE*, *Signq* is the quantum-safe signature algorithm indicated by *CertE*. *CertE* contains the *CPK* and quantum-safe public key [b-ITU-T X.509].

After receiving the transaction request, the node first detects the *CertE* to determine that it is the hybrid signature method. However, as the node does not support the quantum-safe signature algorithm, it only acquires the *CPK* from the certificate *CertE*, and uses this public key to verify the signature *CSign* based on the conventional signature algorithm.

(4) The fourth transaction situation

When a client initiates a transaction to a node, the client and the node verifying the transaction both support the quantum-safe signature algorithm.

In this case, the generation of transaction request message at the client is the same as the third transaction case.

After receiving the transaction request, the node verifying the transaction first detects the certificate with an extension (*CertE*) to determine that the hybrid signature method has been applied. The node obtains the *CPK* and the quantum-safe public key from the certificate *CertE*. First, it utilizes the *CPK* to verify the signature *CSign* based on the conventional signature algorithm indicated in the certificate *CertE*, and then applies the quantum-safe public key to verify the signature *QSign* based on the quantum-safe signature algorithm indicated in the certificate *CertE*.

Bibliography

[b-ITU-T X.509]	Recommendation ITU-T X.509 （2019) | ISO/IEC 9594-8:2020, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
[b-ITU-T X.800]	Recommendation ITU-T X.800 (1991) | ISO 7498-2:1991, *Security architecture for Open Systems Interconnection for CCITT applications*.
[b-ITU-T X.810]	Recommendation ITU-T X.810 (1995) | ISO/IEC 10181-1:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview*.
[b-ITU-T X.1252]	Recommendation ITU-T X.1252 (2021), *Baseline identity management terms and definitions*.
[b-ITU-T X.1400]	Recommendation ITU-T X.1400 (2020), *Terms and definitions for distributed ledger technology*.
[b-ITU-T X.1401]	Recommendation ITU-T X.1401 (2019), *Security threats of distributed ledger technology*.
[b-ITU-T Y.2014]	Recommendation ITU-T Y.2014 (2010), *Network attachment control functions in next generation networks*.
[b-ITU-T FG DLT D5.1]	Technical Report ITU-T FG DLT D5.1 (2019). *Outlook on distributed ledger technologies*.
[b-IETF RFC 7696]	IETF RFC 7696 (2015), *Guidelines for cryptographic algorithm agility and selecting mandatory-to-implement algorithms*. Available [viewed 2024-04-21] at: https://datatracker.ietf.org/doc/html/rfc7696
[b-ISO/IEC 11770-4]	ISO/IEC 11770-4: 2017, *Information technology – Security techniques – Key management – Part 4: Mechanisms based on weak secrets*.
[b-ISO 22739]	ISO 22739:2024, *Blockchain and distributed ledger technologies – Vocabulary*.
[b-ISO/TS 80004-12]	ISO/TS 80004-12:2016, *Nanotechnologies – Vocabulary – Part 12: Quantum phenomena in nanotechnology*.
[b-ETSI GR QKD 007]	Group Report ETSI GR QKD 007 V1.1.1 (2018), *Quantum key distribution (QKD) – Vocabulary*.
[b-ETSI GR QSC 006]	Group Report ETSI GR QSC 006 V1.1.1 (2017), *Quantum-safe cryptography (QSC); Limits to quantum computing applied to symmetric key sizes*.
[b-ETSI-QSC]	ETSI (2024). *Quantum-safe cryptography (QSC).* Sophia Antipolis: ETSI. Available [viewed 2024-04-20] at: https://www.etsi.org/technologies/quantum-safe-cryptography
[b-ETSI TR 103 619]	Technical Report ETSI TR 103 619 V1.1.1 (2020), *CYBER; Migration strategies and recommendations to quantum safe schemes.*
[b-NIST IR 8202]	Internal Report NIST IR 8202-2018, *Blockchain technology overview*.

[b-NIST IR 8240]	Internal Report NIST IR 8240-2019, *Status report on the first round of the NIST post-quantum cryptography standardization process*.
[b-NIST IR 8309]	Internal Report NIST IR 8309-2020, *Status report on the second round of the NIST post-quantum cryptography standardization process*.
[b-NIST IR 8413]	Internal Report NIST IR 8413-2022, *Status report on the third round of the NIST post-quantum cryptography standardization process*.
[b-NIST SREC]	National Institute of Standards and Technology (2016). *Submission requirements and evaluation criteria for the post-quantum cryptography standardization process*. Gaithersburg, MA: National Institute of Standards and Technology. 25 pp. Available [viewed 2024-04-21] at: https://csrc.nist.gov/CSRC/media/Projects/Post- Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
[b-Aaronson]	Aaronson, S. (2008). The limits of quantum. *Sci, Am*. **298**, pp. 62-69.
[b-Aggarwal]	Aggarwal, D., Brennen, G., Lee, T., et al. (2018). Quantum attacks on bitcoin, and how to protect against them. *Ledger* **3**, pp. 68-90. https://doi.org/10.5195/ledger.2018.127
[b-Amy]	Amy, M., Di Matteo, O., Gheorghiu, V. et al. (2017), Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In: Avanzi, R., Heys, H. (eds). *Selected areas in cryptography – SAC 2016. Lecture notes in computer science*, vol 10532. Cham: Springer. https://doi.org/10.1007/978-3-319-69453-5_18
[b-Androulaki]	Androulaki, E., Barger, A., Bortnikov, V. et al. (2018). Hyperledger Fabric: A distributed operating system for permissioned blockchains. In: *EuroSys '18: Proc. Thirteenth EuroSys Conference*, pp. 1-15. New York, NY: Association for Computing Machinery. Available [viewed 2024-04-21] at: https://dl.acm.org/doi/epdf/10.1145/3190508.3190538
[b-Arute]	Arute, F., Arya, K., Babbush, R. *et al*. (2019). Quantum supremacy using a programmable superconducting processor. *Nature* **574**, pp. 505-510.
[b-Bitcoin]	Bitcoin (2009-2024). *Bitcoin is an innovative payment network and a new kind of money*. Las Vegas, NV: Bitcoin Foundation. Available [viewed 2024-04-21] at: https://bitcoin.org
[b-Buchman]	Buchmann, J.A., Butin, D., Goefert, F., Petzoldt, A. (2016). Applied quantum-safe security: Quantum-resistant algorithms and quantum key distribution. In: Ryan, P.Y.A., Naccache, D., Quisquater, J.-J., eds. *The new codebreakers*, pp.88-108. Berlin: Springer.
[b-Castro]	Castro, M., Liskov, B. (1999). Practical Byzantine fault tolerance. In: *Proc. Third Symposium on Operating Systems Design and Implementation*, pp. 173-186. Berkeley, CA: USENIX Association.
[b-Castryck]	Castryck, W., Decru, T. (2023). An efficient key recovery attack on SIDH. In: *Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part V*, pp. 423–447. Berlin: Springer. 10.1007/978-3-031-30589-4_15

[b-CSA QSS]	Quantum-safe Security Working Group (2017). *Applied quantum safe security*. Seattle, WA: Cloud Security Alliance. Available [viewed 2024-04-20] from: https://cloudsecurityalliance.org/download/applied-quantum-safe-security
[b-Ekdahl]	Ekdahl, P., Johansson, T., Maximov, A., J. Yang, J. (2019). A new SNOW stream cipher called SNOW-V. *IACR Trans. Symm. Cryptol*. **2019**(3), pp. 1–42. Available [viewed 2024-04-22] from: https://tosc.iacr.org/index.php/ToSC/article/view/8356
[b-Ethereum]	Ethereum (2024). *Welcome to Ethereum*. Zug: Ethereum. Available [viewed 2024-04-21] at: https://ethereum.org
[b-Garey]	Garey, M.R., Johnson, D.S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York, NY: W.H. Freeman. 338 pp.
[b-Grover]	Grover, L K. (1996). A fast quantum mechanical algorithm for database search. In: *Proc. 28th annual ACM Symposium on the Theory of Computing*, pp. 212–219. New York, NY: Association for Computing Machinery. doi.org/10.1145/237814.237866
[b-Hyperledger Fabric]	Hyperledger Foundation (2024). *Hyperledger Fabric*. San Francisco, CA: Linux Foundation. Available [viewed 2024-04-21] at: https://www.hyperledger.org/use/fabric
[b-Juskalian]	Juskalian, R. (2017). Practical quantum computers. *MIT Technol. Rev*. Available [viewed 2024-04-20] at: https://www.technologyreview.com/technology/practical-quantum-computers/
[b-Kiktenko]	Kiktenko, E.O., Pozhar, N.O., Anufriev, M.N. et al. (2018). *Quantum-secured blockchain*. arXiv:1705.09258v3. 7 pp. Available [viewed 2024-04-20] at: https://arxiv.org/pdf/1705.09258.pdf
[b-McEliece]	McEliece, R.J. (1978). A public-key cryptosystem based on algebraic coding theory. In*: Deep space network progress report*, DSN PR 42-44, pp. 114–116. Bibcode:1978DSNPR. Available [viewed 2024-04-20] at: https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PD
[b-Nyczepir]	Nyczepir, D. (2022). Post-quantum cryptography experts brace for long transition despite White House deadlines. *Fedscoop*. Available [viewed 2024-04-21] at: https://www.fedscoop.com/quantum-crytography-experts-long-transition/
[b-Raft]	Raft (Internet). *The Raft consensus algorithm*. San Francisco, CA: GitHub Pages. Available [viewed 2024-04-21] at: https://raft.github.io/
[b-Rodenburg]	Rodenburg, B., Pappas, S.P. (2017). *Blockchain and quantum computing*, Mitre Technical Report. Princeton, NJ: Mitre. 16 pp. Available [viewed 2024-04-20] at: https://www.mitre.org/sites/default/files/publications/17-4039-blockchain-and-quantum-computing.pdf
[b-Shor]	Shor, P.W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**(5), pp. 1484–1509. https://doi.org/10.1137/S0097539795293172

[b-Wang]	Wang, X., Yin, Y.L., Yu, H. (2005). Finding collisions in the full SHA-1, In: Shoup, V. (ed.). *Advances in Cryptology – CRYPTO.2005. Lecture Notes in Computer Science*, vol. 3621, pp. 17–36. Berlin: Springer. https://doi.org/10.1007/11535218_2
[b-Wiki-QC]	Wikipedia (2024). *Quantum computing*. San Francisco, CA: Wikimedia Foundation. Available [viewed 2024-04-20] at: https://en.wikipedia.org/wiki/Quantum_computing
[b-ZUC-256]	ISCAS (2018). *The ZUC-256 stream cipher*, version 1.1. Beijing: Institute of Software of the Chinese Academy of Sciences. 9 pp. Available [viewed 2024-04-22] at: http://www.is.cas.cn/ztzl2016/zouchongzhi/201801/W020180416526664982687.pdf