

ITU-T Technical Report

(06/2026)

XSTR.kdc-qkdn

Key distribution centre (KDC)-based approaches in the service layer to manage keys supplied by quantum key distribution networks

Technical Report ITU-T XSTR.kdc-qkdn

Key distribution centre (KDC)-based approaches in the service layer to manage keys supplied by quantum key distribution networks

Summary

A QKDN exploits optical fibre channels or free-space optical channels to support quantum-channel networking and provide secure keys to devices connected to the QKDN. This Technical Report studies key distribution centre (KDC)-based approaches for managing keys supplied by a QKDN so that keys generated in the QKDN can be applied to various security services and to a large number of user devices, regardless of whether those devices are mobile or fixed.

Keywords

KDC (Key Distribution Centre), QKD (quantum key distribution), QKDN (QKD network).

Note

This is an informative ITU-T publication. Mandatory provisions, such as those found in ITU-T Recommendations, are outside the scope of this publication. This publication should only be referenced bibliographically in ITU-T Recommendations.

© ITU 2026

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope.....	1
2	References.....	1
3	Definitions	1
	3.1 Terms defined elsewhere	1
	3.2 Terms defined in this Technical Report	2
4	Abbreviations and acronyms	2
5	Introduction.....	2
6	Overview.....	3
	6.1 Countermeasures to resist quantum computing attacks.....	3
	6.2 QKDN.....	3
	6.3 KDC.....	4
7	Architecture	4
	7.1 Coupling model of user network and QKDN.....	4
	7.2 Conceptual architecture	5
8	Functionalities.....	7
	8.1 Rationale.....	7
	8.2 Management of the authentication and encryption keys	7
	8.3 Authentication and distribution of SKEYs.....	8
9	Trust model.....	12
	9.1 Trust model in the conventional system.....	12
	9.2 Trust model in the QKDN	12
	9.3 Trust model for integration of QKDN and user network supporting mobile devices	13
	Appendix I – Protocol for authentication and SKEY distribution (scheme 1)	15
	Appendix II – Protocol for authentication and SKEY distribution (scheme 2).....	18
	Appendix III – Protocol for authentication and SKEY distribution (Scheme 3).....	21
	Appendix IV – Brief introduction of Kerberos	24
	Bibliography.....	25

Key distribution centre (KDC)-based approaches in the service layer to manage keys supplied by quantum key distribution networks

1 Scope

This Technical Report studies key distribution centre (KDC)-based approaches in the service layer to manage keys supplied by a quantum key distribution network (QKDN). In particular, the scope of this Technical Report includes:

- An overview of the use of a KDC in the service layer to manage keys supplied by a QKDN;
- Architecture of a KDC in the service layer;
- Functionalities of a KDC in the service layer;
- Trust models for KDC-based integration of a QKDN and a user network.

2 References

None.

3 Definitions

3.1 Terms defined elsewhere

This Technical Report uses the following terms defined elsewhere:

3.1.1 key manager (KM) [b-ITU-T Y.3800]: A functional module located in a quantum key distribution (QKD) node to perform key management in the key management layer.

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

3.1.3 quantum key distribution network (QKDN) [b-ITU-T Y.3800]: A network comprised of two or more quantum key distribution (QKD) nodes connected through QKD links.

NOTE – A QKDN allows sharing keys between the QKD nodes by key relay when they are not directly connected by a QKD link.

3.1.4 quantum key distribution node (QKD node) [b-ITU-T Y.3800]: A node that contains one or more quantum key distribution (QKD) modules protected against intrusion and attacks by unauthorized parties.

NOTE – A QKD node can contain a key manager (KM).

3.1.5 secure channel [b-NIST-SP800-133Rev2]: A path for transferring data between two entities or components that ensures confidentiality, integrity, and replay protection as well as mutual authentication between the entities or components. The secure channel may be provided using cryptographic, physical, or procedural methods or a combination thereof.

3.1.6 trust [b-ITU-T-X.1252]: The confidence of one party or entity that another party or entity will behave in a well-defined way that does not violate agreed-upon rules, policies or legal clauses of the identity management system.

3.2 Terms defined in this Technical Report

This Technical Report defines the following terms:

3.2.1 key distribution centre (KDC): A centre that acquires a key from a QKDN and distributes that key to a pair of entities that each share distinct keys with the centre.

NOTE – Adapted from [b-ISO/IEC 11770-2:2018].

3.2.2 trust model: A collection of assumptions that characterize the trustworthiness of each component in a system.

NOTE – Adopted from [b-NIST SP 800-226].

4 Abbreviations and acronyms

This Technical Report uses the following abbreviations and acronyms:

AEAD	Authenticated Encryption with Associated Data
AS	Authentication Server
BER	Bit Error Rate
CA	Certificate Authority
HMAC	Hash-based Message Authentication Code
IT-secure	Information-theoretically secure
KDC	Key Distribution Centre
MAC	Message Authentication Code
OTP	One-Time Pad
QKD	Quantum Key Distribution
QKDN	QKD Network
SKEY	Session KEY
TGS	Ticket Granting Server

5 Introduction

A key is the most important security asset in a cryptosystem, as its security strength to a considerable extent determines whether the cryptosystem is secure or not. QKD (quantum key distribution) nodes can provide secure keys to cryptographic applications in the service layer. Keys generated by using QKD mechanism are anticipated to be widely used in many fields, such as government affairs, finance, energy and transportation for a higher security level. Security services which make use of keys from QKDN (quantum key distribution network) are expected to have a larger user scale since more and more services pose stricter security requirements. There are various types of security services, such as virtual private network, secure office, message encryption, and secure data transmission. User devices that support these security services can be classified into two categories: mobile devices and immovable devices. The typical examples of the former include a mobile phone, a pad and a notebook, while the typical examples of the latter include a desktop, a workstation and a server.

Key management is of utmost importance to a cryptosystem as any fault in the key management may lead to key disclosure. A KDC (key distribution centre) is a commonly used facility to manage keys for a larger user scale in the conventional system, whose functionalities at least include key generation, entity authentication, key distribution, and key life-cycle management. Kerberos [b-IETF RFC 4120] is a typical example of a KDC in a real-world system, which consists of an

authentication server (AS) and a ticket granting server (TGS). As a trusted third party, Kerberos enables the secure authentication of users to application servers over an unprotected network. It can also provide for the establishment of cryptographic keys between a client and an application server. In most cases the KDC shares a key with each of all the other parties. A KDC is viewed as a trustworthy infrastructure by users as it knows all keys of users. KDCs usually apply symmetric cryptographic algorithms for their operation. For example, it is well known that symmetric cryptographic algorithms with a 256-bit length key, such as AES-256, are resistant to quantum computing attacks. NIST categorizes the security strength of the PQC algorithms based on a classification on the range of security strengths offered by the existing NIST standards in symmetric cryptography, which NIST expects to offer significant resistance to quantum cryptanalysis. AES 256 is the highest security level to resist quantum-computing attacks when it is used as the reference objective for PQC algorithms[b-NIST-Criteria].

In this Technical Report, KDC-based approaches in the service layer to manage keys supplied by QKDN are studied, in order that keys generated in QKDN can be applied to various security services and a large number of user devices no matter whether these devices are movable or immovable.

6 Overview

6.1 Countermeasures to resist quantum computing attacks

Quantum computers are devices that use quantum mechanical phenomena to perform calculations and process data. Due to the powerful parallel computing capabilities, quantum computers have the ability to quickly complete calculations that cannot be done by classical computers. This poses a serious and realistic security threat to modern cryptographic algorithms. The security strength of a symmetric cryptographic algorithm will be reduced to half due to Grover's quantum algorithm. Many commonly used asymmetric cryptographic algorithms, such as RSA, DSA, and elliptic curve-based algorithms, will provide no security due to Shor's quantum algorithm. A quantum key distribution (QKD) scheme is viewed as an approach to alleviate the impact of quantum computing on the cryptographic algorithms. It provides a means to share a secret key between two distant parties securely against an eavesdropper with unlimited computational ability. A QKD link consists of a quantum channel to transmit quantum bits and a classical public channel to exchange public information for key distillation. When an adversary eavesdrops on the quantum channel in any way allowed by the laws of physics, the attack can be detected by monitoring an increase of bit error rate (BER) [b-Sasaki]. QKD is a provably secure scheme resilient to quantum computing attacks because it is a physics-based solution rather than one based on computational assumptions [b-Mayers].

6.2 QKDN

Quantum key distribution network (QKDN) is a network comprised of two or more quantum key distribution (QKD) nodes connected through QKD links [b-ITU-T Y.3800]. If the distance between QKD nodes exceeds a certain range, it is necessary to deploy QKD trusted relay nodes between QKD nodes.

QKDN can enable any two QKD nodes to share secure keys, even if these two nodes are not directly connected via a QKD link. For this purpose, the keys are relayed from one relay node to another until they arrive at a destination node. There are several schemes that can accomplish key relay. Among them, trusted relaying is a well-known and mature scheme, where keys are stored in QKD nodes (trusted nodes) and relayed to other distant QKD nodes via highly secure encryption, with one-time pad (OTP) recommended. The QKD node (trusted node) is assumed to be secure against intrusion and attacks by any unauthorized parties [b-ITU-T Y.3800].

There are three variations for forming QKDN and user networks in Appendix I of [b-ITU-T Y.3800], which are illustrated in Figure 1.

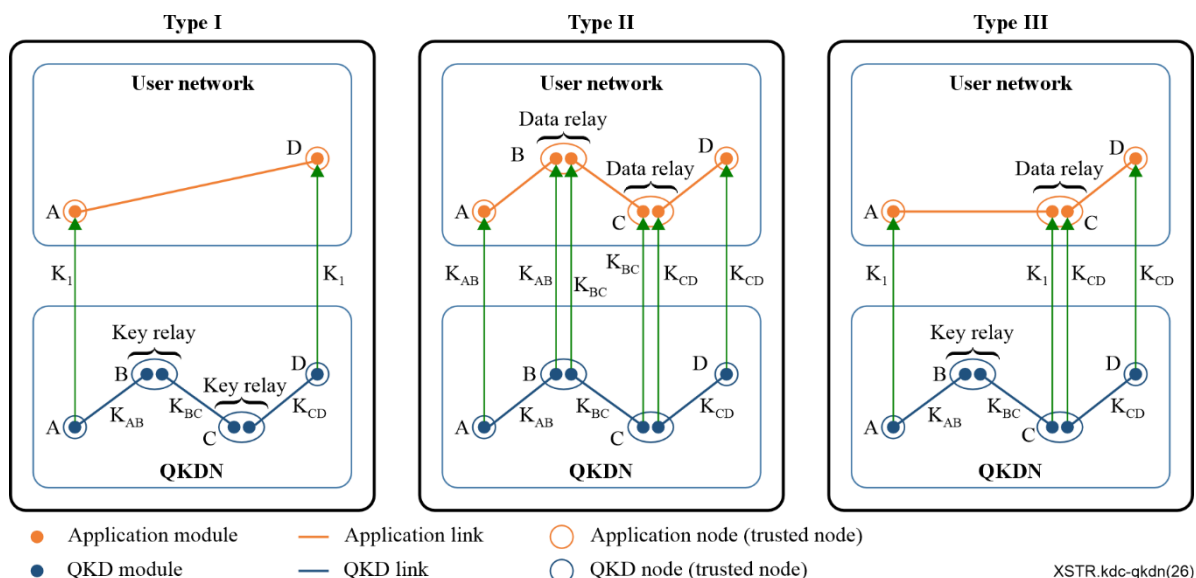


Figure 1 – Variations for forming QKDNs and user networks

In Type I, two application nodes which are connected to a QKDN obtain the same key (K_1) through the key relay performed by trusted relay nodes in the QKDN. In Type II, without using key relay in the QKDN, two application nodes conduct secure communications through data relay performed by trusted application nodes in the user network. The keys in QKD links are independently supplied directly, without any key relay, to the application modules in the user network. Type III is a combination of Type I and Type II.

The QKD node (trusted node) and the application node represent a logical node in the QKDN and the user network, respectively. In practice, these logical nodes are often located in the same physical node. The physical nodes in a QKDN are usually connected using optical fibres. In order to use keys generated by a QKDN on mobile devices, keys need to be distributed using a secure method.

NOTE – The mobile terminal obtains keys in a secure location, such as within a trusted node. Other methods (e.g., obtaining keys outside a trusted node) would make the keys less secure.

6.3 KDC

Key management is a challenge when only symmetric cryptographic algorithms are deployed in a system, as each participant has to manage n keys, and the total number of keys to be managed is $n \times (n-1)/2$, where n is the number of participants. If there are a large number of participants that need to establish secure connections, key management is not a trivial task. A KDC is introduced to address this issue.

In a KDC scheme, each participant shares a master key with the KDC. The KDC generates a SKEY and distributes it to two participants by encrypting it with the corresponding master keys. In this way, each participant can establish a secure connection with any other participant in a large group, while only one master key needs to be kept secret by each participant. Kerberos [b-IETF RFC 4120] is a typical example of a KDC in practical applications, which is briefly introduced in Appendix IV.

7 Architecture

7.1 Coupling model of user network and QKDN

As specified in [b-ITU-T Y.3800], a user network and a QKDN can be coupled in three types, where keys from the QKDN can be provided only to application nodes that are connected to the QKDN. In this Technical Report, a fourth type that binds a user network and a QKDN is presented to enable

keys from the QKDN to be distributed to mobile devices in a user network via a KDC, as illustrated in Figure 2.

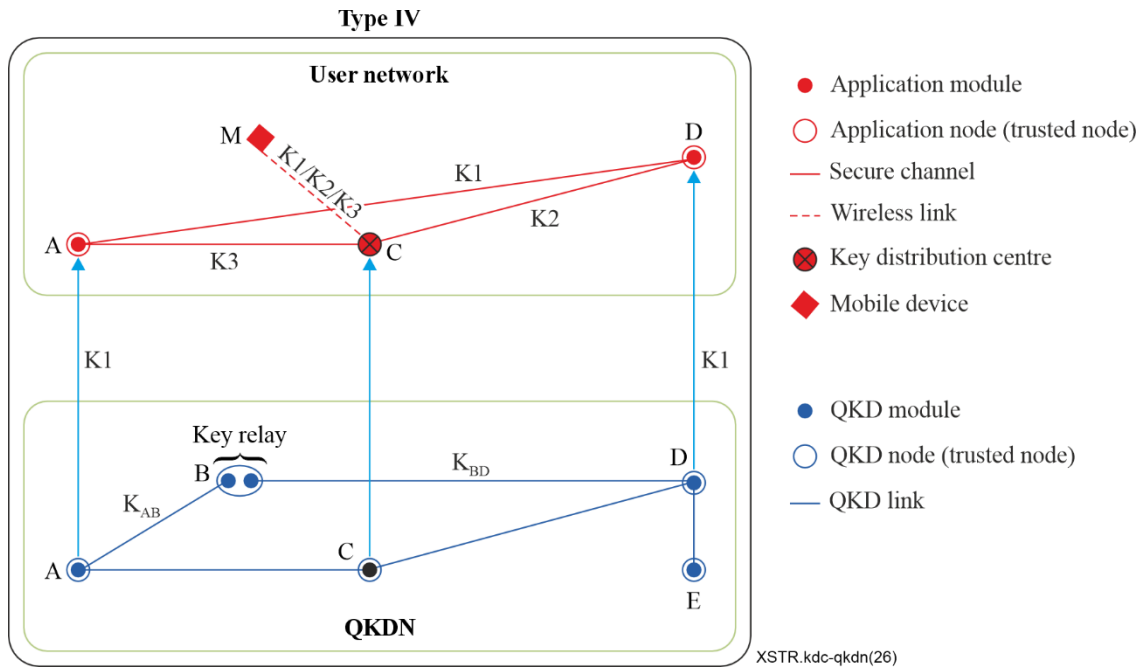


Figure 2 – Coupling model

As shown in Figure 2, each application node (i.e., endpoint) in the user network has a connection to the corresponding QKD node in the QKDN. Three kinds of keys (K1, K2 and K3) generated by the QKDN can be routed to a mobile device through the KDC in the user network and can be used as SKEYs for the mobile device to establish a secure connection with application node D in the user network.

Key K1 is supplied from the QKDN to two application nodes A and D, whose corresponding QKD nodes are not directly connected by a QKD link. Key relay is performed in the QKDN from node A to node D via node B. In the user network, the KDC (node C) is used to transfer key K1 from the QKDN to the mobile device (node M).

Key K2 is supplied from the QKDN to two application nodes C and D, whose corresponding QKD nodes are directly connected. In the user network, the KDC (node C) is used to transfer key K2 from the QKDN to the mobile device (node M).

Key K3 is supplied from the QKDN to two application nodes A and C, whose corresponding QKD nodes are directly connected. In the user network, the KDC (node C) is used to transfer the key K3 from the QKDN to the mobile device (node M).

This implies that keys from the underlying QKDN can be distributed in the user network with the help of a KDC.

The difference between Type III and Type IV is that key distribution can be performed in both the user network and the QKDN in Type IV, while for Type III key distribution is performed in the QKDN and data relay is performed in the user network.

7.2 Conceptual architecture

The conceptual architecture is composed of two networks: the underlying network, which is the QKDN, and the upper network, which is the user network, as shown in Figure 3. This architecture is aligned with the conceptual structure specified in [b-ITU-T Y.3800], except that a KDC is introduced into the user network in order to provide key services to mobile devices.

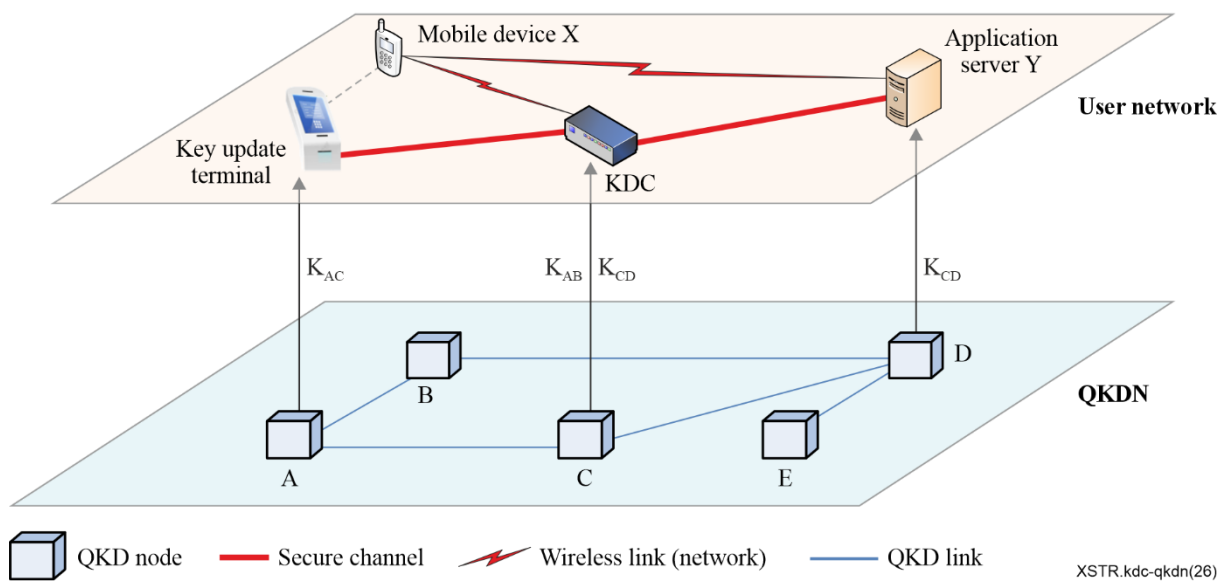


Figure 3 – Illustration of the conceptual architecture

The QKDN generates keys and supplies them to the user network. The user network consists of a KDC, key update terminals, servers, and mobile devices.

- **Mobile device:** It is a movable device which can be used anywhere. It can be a mobile phone, notebook, pad, etc. It has no direct connection to the QKDN. It wants to securely access an application server for a dedicated service by using keys from the QKDN. It communicates with the KDC and the application server via a wireless link (network).
- **Application server:** It provides services to mobile devices. It has a connection to the underlying QKD node to obtain keys from the QKDN. It communicates with the KDC via a secure channel, which is established using keys from two QKD nodes in the QKDN. These two QKD nodes are connected to the application server and the KDC, respectively.
- **Key update terminal:** It is usually located at a service hall of a mobile communication service operator in order to supply keys from the QKDN to mobile devices. It has a connection to the underlying QKD node to obtain keys from the QKDN. It communicates with the KDC via a secure channel, which is established using keys from two QKD nodes in the QKDN. These two QKD nodes are connected to the key update terminal and the KDC, respectively. The main objective of the key update terminal is to bind key metadata with the user identification of the mobile device, so that these keys delivered to the mobile devices can be used as authentication and encryption keys.
- **KDC:** It serves to authenticate mobile devices and distribute keys from the QKDN to them. It has a connection to the underlying QKD node to obtain keys from the QKDN. It applies authentication and encryption keys to perform mutual authentication with mobile devices. It fetches keys from the QKDN that can be used as SKEYs and securely relays them to mobile devices.

NOTE – The application server, key update terminal, and KDC are accommodated in trusted nodes, where keys are delivered from the QKDN securely.

8 Functionalities

8.1 Rationale

From an application point of view, mobile devices may access a variety of application servers to obtain different services. Without a KDC, in the case of a symmetric key system, mobile devices need to have pre-shared keys with these application servers separately in order to access them securely. This not only increases the storage burden of mobile devices but also imposes a heavy key management task. A KDC is introduced in this scheme to address this problem. A mobile device can obtain the SKEYs for communication with multiple application servers through the KDC, as long as it can be authenticated by the KDC. For this purpose, a mobile device needs to keep only one key shared with the KDC in advance, which is used for authentication as well as encryption of the SKEY.

8.2 Management of the authentication and encryption keys

Mobile devices inject keys from the QKDN in bulk, offline, through the key update terminal. These keys are generated between QKD node A (connected to the key update terminal) and QKD node C (connected to the KDC), i.e., keys K_{AC} in Figure 3. These keys are called authentication and encryption keys, as the KDC applies them to authenticate mobile devices and encrypt SKEYs. For high security, an authentication and encryption key is used only for one session. After that session, both the KDC and the mobile device delete the used authentication and encryption key. The process by which the KDC and a mobile device acquire authentication and encryption keys is depicted in Figure 4.

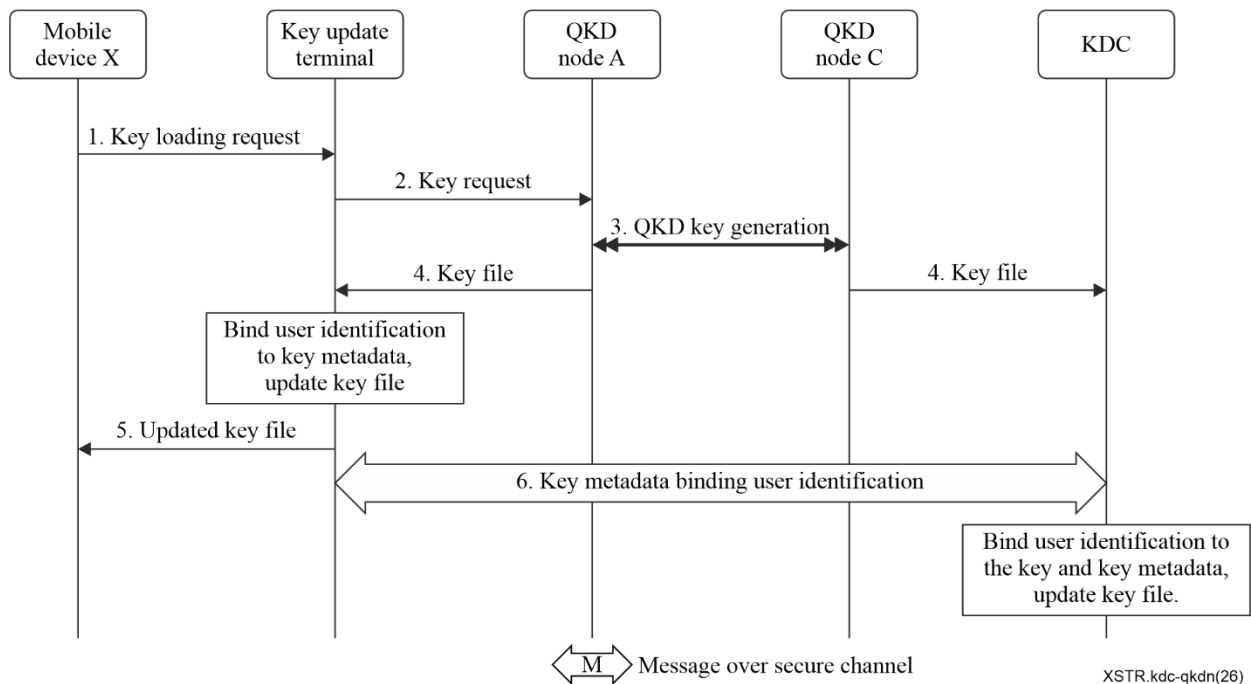


Figure 4 – Process to acquire keys for authentication and encryption

The detailed steps to acquire keys for authentication and encryption are as follows.

1. A mobile device user applies for loading a set of keys from the QKDN on the key update terminal. This implies that the mobile device invokes a key request to the key update terminal at the same location.
2. The key update terminal initiates a key request to the connected QKD node A.

3. QKD node A initiates the QKD key generation process with QKD node C connected to the KDC, generating a key file containing a set of keys and associated metadata for key management.
4. QKD node A supplies the key file to the key update terminal, and QKD node C supplies the key file to the KDC.
5. The key update terminal binds the user ID, which is the identification of the mobile device user, with the key metadata and updates the key file, and then delivers the updated key file to the mobile device.
6. The key update terminal transmits the key metadata bound to the user ID to the KDC through a secure channel between the key update terminal and the KDC, in order to assure the confidentiality and integrity of the user ID and key metadata. This secure channel is established using a key generated between QKD node A and QKD node C.
7. The KDC locates the corresponding keys using the key metadata, binds the user ID with these keys and the associated key metadata, and updates the key file.

Through the above steps, the KDC and the mobile device possess the same updated key file, which contains keys and associated key metadata. Each key has corresponding metadata. Examples of key metadata are specified in Table 1.

Table 1 – Examples of key metadata

Key identification (KeyID)	Each key of the two QKD nodes corresponds to a unique identifier
Key length	128 bits or 256 bits
Key provision time	The time point at which the QKD node provides the key to the cryptographic application
Application name	Name of cryptographic application
Source identification	(name of QKD node A, name of the key update terminal)
Destination identification	(name of QKD node C, name of the KDC)
User ID	Identification of the mobile device user

NOTE 1 – (name of QKD node A, name of the key update terminal) is a joint identification to uniquely determine at the source location which keys generated in which QKD node in QKDN are supplied to which key update terminal in the user network.

NOTE 2 – (name of QKD node C, name of the KDC) is a joint identification used to uniquely determine, at the destination location, which QKD node in the QKDN supplies keys to which KDC in the user network.

8.3 Authentication and distribution of SKEYs

8.3.1 Overview

As shown in Figure 3, the key update terminal, the KDC, and application server Y in the service layer are involved in key management. They are connected to the corresponding QKD nodes of the QKDN, which are node A (connected to the key update terminal), node C (connected to the KDC), and node D (connected to application server Y), respectively. The key K_{AD} generated between node A and node D, the key K_{CD} generated between node C and node D, as well as the key K_{AC} generated between node A and node C, can be delivered to the service layer. These keys can be used as SKEYs and distributed to a mobile device through the KDC. Accordingly, there exist the following three kinds of schemes.

8.3.2 Scheme 1: Distribution of the key K_{AD}

The KDC and a mobile device authenticate each other based on the pre-shared authentication and encryption keys between them. The key generated between QKD node A, which is connected to the

key update terminal, and QKD node D, which is connected to the application server, is used as a SKEY and is securely delivered to the mobile device. The procedure to perform authentication and the distribution of the SKEY is depicted in Figure 5.

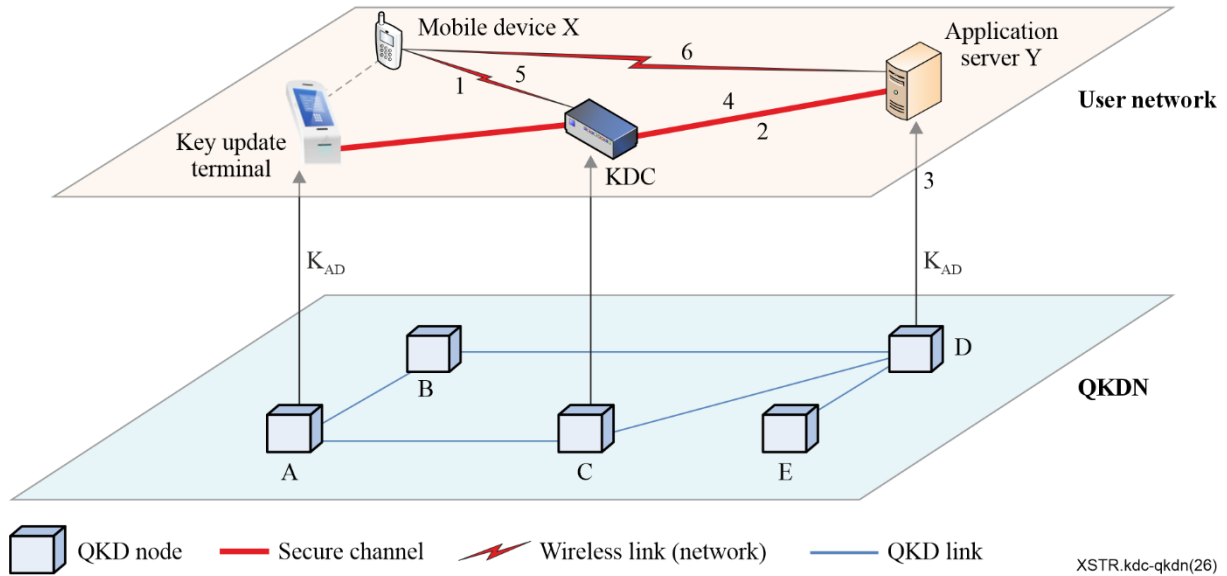


Figure 5 – Procedure to perform the authentication and SKEY distribution (scheme 1)

As shown in Figure 5, the detailed workflow is as follows.

1. Mobile device X initiates a SKEY request message to the KDC, whose integrity is protected using the pre-shared authentication and encryption key.
2. The KDC authenticates mobile device X by verifying the integrity of the SKEY request message. It delivers a session notification message to application server Y via a secure channel between the KDC and application server Y. This secure channel is established using the key generated between QKD node C and QKD node D.
3. Application server Y determines that a SKEY is required between itself and the key update terminal based on the session notification message. Accordingly, it sends a key request message to the underlying QKD node D to acquire the key generated between QKD node A and QKD node D in the QKDN.
4. Application server Y returns the obtained SKEY to the KDC through the secure channel.
5. The KDC securely delivers the SKEY using an authenticated encryption with associated data (AEAD) scheme [b-Rogaway] by using the authentication and encryption key. The output of the AEAD scheme consists of a ciphertext and a MAC tag. Mobile device X obtains the SKEY by decrypting the received ciphertext and authenticates the KDC by verifying the MAC tag.
6. Mobile device X initiates a secure session with application server Y using the SKEY, where they authenticate each other to establish a secure connection.

The protocol that details the contents of messages and procedures for authentication and SKEY distribution is specified in Appendix I.

8.3.3 Scheme 2: Distribution of the key K_{CD}

The KDC and a mobile device authenticate each other based on the pre-shared authentication and encryption keys between them. The key K_{CD} , which is generated between QKD node C connected to

the KDC and QKD node D connected to the application server, is used as a SKEY and is securely delivered to the mobile device. The procedure to perform authentication and the distribution of the SKEY is depicted in Figure 6.

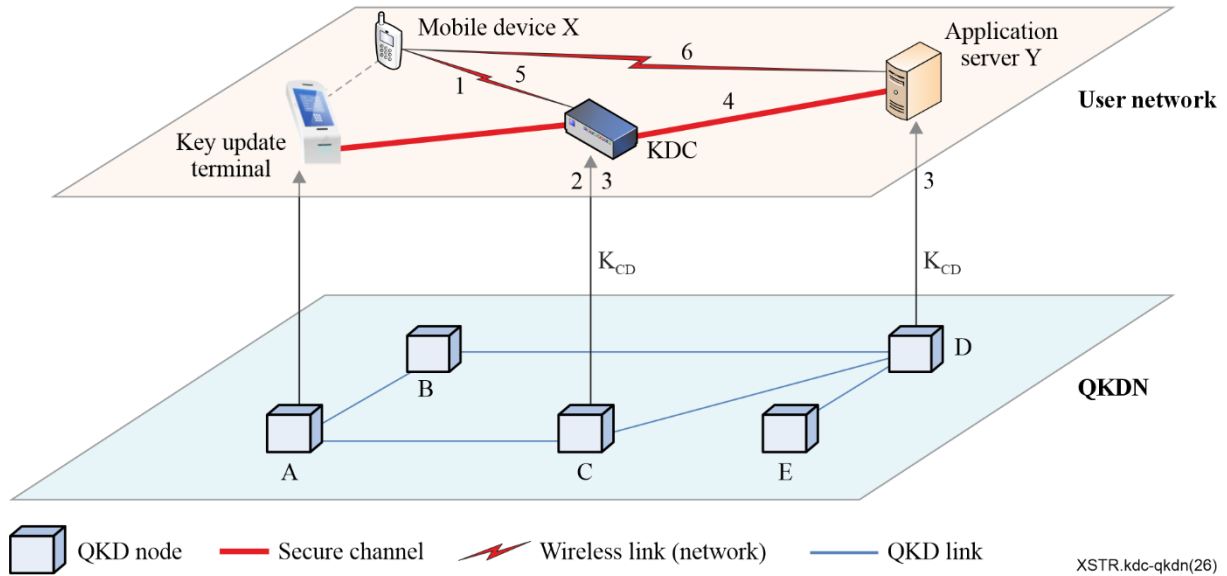


Figure 6 – Procedure to perform the authentication and SKEY distribution (scheme 2)

As shown in Figure 6, the detailed workflow is as follows.

1. Mobile device X initiates a SKEY request message to the KDC, whose integrity is protected using the pre-shared authentication and encryption key.
2. The KDC authenticates mobile device X by verifying the integrity of the SKEY request message. Accordingly, it delivers a key request message to the underlying QKD node C to acquire the key generated between QKD node C and QKD node D in the QKDN.
3. QKD node C and QKD node D deliver the key K_{CD} to the KDC and application server Y, respectively.
4. The KDC delivers a session notification message to application server Y via a secure channel between the KDC and application server Y. This secure channel is established using the key generated between QKD node C and QKD node D.
5. The KDC securely delivers the SKEY using an authenticated encryption with associated data (AEAD) scheme [b-Rogaway] by using the authentication and encryption key. The output of the AEAD scheme consists of a ciphertext and a MAC tag. Mobile device X obtains the SKEY by decrypting the received ciphertext and authenticates the KDC by verifying the MAC tag.
6. Mobile device X initiates a secure session with application server Y using the SKEY, and application server Y acquires the corresponding SKEY (K_{CD}) according to the session identifier obtained in step 4. They authenticate each other using K_{CD} to establish a secure connection.

The protocol that details the contents of messages and procedures for authentication and SKEY distribution is specified in Appendix II.

8.3.4 Scheme 3: Distribution of the key K_{AC}

The KDC and a mobile device authenticate each other based on the pre-shared authentication and encryption keys between them. The key K_{AC} , which is generated between QKD node C connected to the KDC and QKD node A connected to the key update terminal, is used as a SKEY and is securely delivered to the mobile device. The procedure to perform authentication and the distribution of the SKEY is depicted in Figure 7.

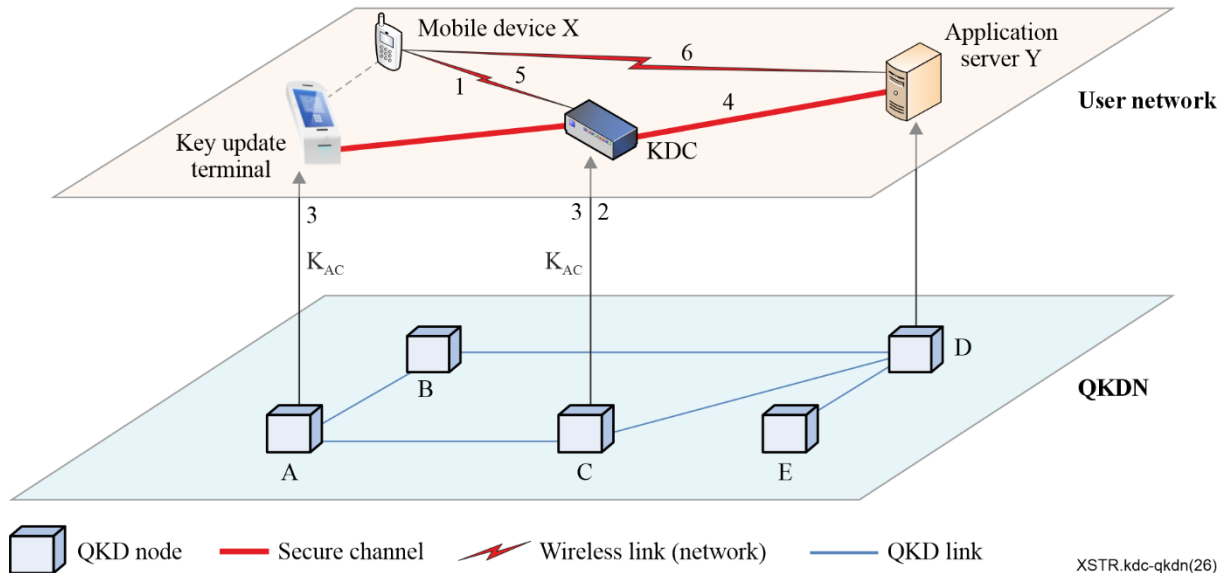


Figure 7 – Procedure to perform the authentication and SKEY distribution (scheme 3)

As shown in Figure 7, the detailed workflow is as follows.

1. Mobile device X initiates a SKEY request message to the KDC, whose integrity is protected using the pre-shared authentication and encryption key.
2. The KDC authenticates mobile device X by verifying the integrity of the SKEY request message. It delivers a key request message to the underlying QKD node C to acquire the key generated between QKD node C and QKD node A in the QKDN.
3. QKD node C and QKD node A deliver the key K_{AC} to the KDC and the key update terminal, respectively.
4. The KDC delivers a session notification message to application server Y via a secure channel between the KDC and application server Y. This secure channel is established using a key generated between QKD node C and QKD node D.
5. The KDC securely delivers the SKEY using an authenticated encryption with associated data (AEAD) scheme [b-Rogaway] by using the authentication and encryption key. The output of the AEAD scheme consists of a ciphertext and a MAC tag. Mobile device X obtains the SKEY by decrypting the received ciphertext and authenticates the KDC by verifying the MAC tag.
6. Mobile device X initiates a secure session with application server Y using the SKEY, and application server Y acquires the corresponding SKEY (K_{AC}) according to the session identifier obtained in step 4. They authenticate each other using K_{AC} to establish a secure connection.

The protocol that details the contents of messages and procedures for authentication and SKEY distribution is specified in Appendix III.

9 Trust model

9.1 Trust model in the conventional system

A trusted third party is usually involved to facilitate key establishment between two parties that do not know each other in advance. There are two approaches to introducing a trusted third party into a system: a KDC-based approach and a CA-based approach (see Figure 8).

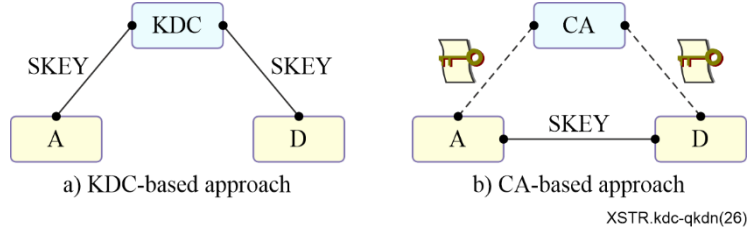


Figure 8 – Approaches for a trusted third party

KDC-based approach: As shown in Figure 8(a), the KDC acts as a trusted third party that is involved in the key establishment procedure for two participants (e.g., entity A and entity D in Figure 8(a)). Each participant trusts the KDC and shares a pre-shared key with the KDC. To establish the SKEY, the KDC first authenticates each participant using the corresponding pre-shared key. Then, the KDC encapsulates the SKEY with the respective pre-shared key and delivers it to each participant. This dramatically decreases the key management complexity, in that the total number of keys that need to be managed is reduced from $n \times (n-1)$ to n , where n is the total number of participants. The security of such a scheme relies on the trustworthiness of the KDC, as it knows all pre-shared keys and SKEYs (SKEYs). Kerberos is a typical example that uses this principle.

CA-based approach: As shown in Figure 8(b), the CA acts as a trusted third party to issue a certificate to each participant. Two participants (e.g., entity A and entity D in Figure 8(b)) perform mutual authentication based on certificates in order to ensure that the communicating partner is authentic. The two authenticated participants negotiate a SKEY. Although the CA does not know the SKEY, the trustworthiness of the CA is crucial to the security of the scheme. If a CA is compromised, the trust in the certificates it has issued collapses, rendering encrypted communications and authenticated services vulnerable. The CA-based approach has been used in IKEv2 and TLS for key establishment and entity authentication.

9.2 Trust model in the QKDN

A QKDN and user network are illustrated in Figure 9.

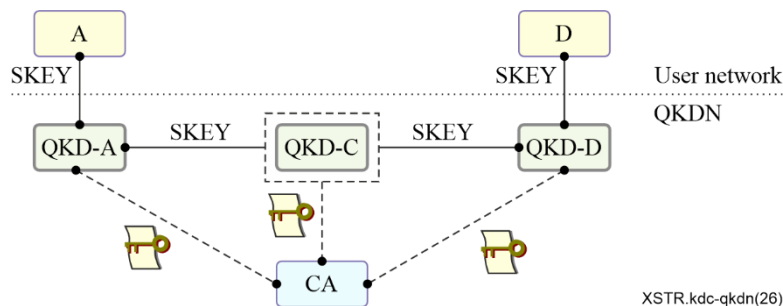


Figure 9 – QKDN and user network

A CA is introduced into the QKDN to serve authentication purposes between QKD nodes. Authenticated QKD nodes negotiate a shared SKEY using a QKD mechanism. The agreed SKEY in the QKDN is delivered to the corresponding application nodes in the user network. If the distance

between two QKD nodes (e.g., node QKD-A and node QKD-D in Figure 9) exceeds the allowed range, at least one trusted relay node (e.g., node QKD-C in Figure 9) must be deployed to relay the SKEY. Thus, a QKDN may contain one trusted third party (i.e., a CA) in the best case, while it may contain $n+1$ trusted third parties in the worst case, where n is the number of trusted relay nodes located in a QKD link. The more trusted third parties are involved, the lower the trustworthiness of a SKEY. The ideal case for key establishment is that the SKEY is known only to the two participants that wish to communicate with each other.

9.3 Trust model for integration of QKDN and user network supporting mobile devices

This model has a similar architecture to the QKDN system, except that a KDC is introduced into the system, as shown in Figure 10. Here, node A, node D, and node X correspond to the key update terminal, application server, and mobile device in the user network, respectively. It is evident that the proposed system has at least two trusted third parties: the KDC in the user network and the CA in the QKDN.

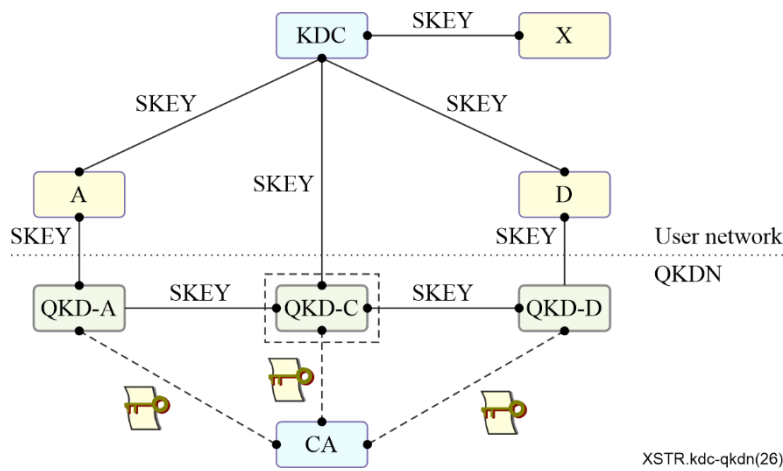


Figure 10 – Trust model in integration of QKDN and user network supporting mobile devices

This system has three different schemes to realize the delivery of the SKEY to the mobile device.

1) Scheme 1: Distribution of the key K_{AD}

The key K_{AD} , which is generated between QKD node A connected to the key update terminal (i.e., node A in the user network) and QKD node D connected to the application server (i.e., node D in the user network), is used as a SKEY and is securely delivered to the mobile device (i.e., node X in the user network). In this scheme, the SKEY (i.e., key K_{AD}) is known to the key update terminal and the KDC, in addition to the two participants that want to communicate with each other (the mobile device and the application server).

2) Scheme 2: Distribution of the key K_{CD}

The key K_{CD} , which is generated between QKD node C connected to the KDC and QKD node D connected to the application server (i.e., node D in the user network), is used as a SKEY and is securely delivered to the mobile device. In this scheme, the SKEY (i.e., key K_{CD}) is known only to the KDC, in addition to the two participants that want to communicate with each other (the mobile device and the application server).

3) Scheme 3: Distribution of the key K_{AC}

The key K_{AC} , which is generated between QKD node C connected to the KDC and QKD node A connected to the key update terminal (i.e., node A in the user network), is used as a SKEY and is securely delivered to the mobile device. In this scheme, the SKEY (i.e., key K_{AC}) is known to the key update terminal and the KDC, in addition to the two participants that want to communicate with each other (the mobile device and the application server).

As shown above, scheme 2 is the most appropriate approach for delivering a SKEY to a mobile device from a security perspective, since, except for the two participants that want to communicate with each other (the mobile device and the application server), the KDC is the only node that knows the SKEY in the user network. In contrast, in scheme 1 and scheme 3, the SKEY is also known to the key update terminal, which is not involved in the secure communication between the mobile device and the application server. Therefore, scheme 2 is suggested for future standardization.

Appendix I

Protocol for authentication and SKEY distribution (scheme 1)

The authentication and SKEY distribution protocol performs mutual authentication between a mobile device and the KDC and securely delivers the SKEY to the mobile device through the KDC. In addition, the mobile device and application server performs mutual authentication with the SKEY for a secure connection. The protocol flow is illustrated in Figure I.1.

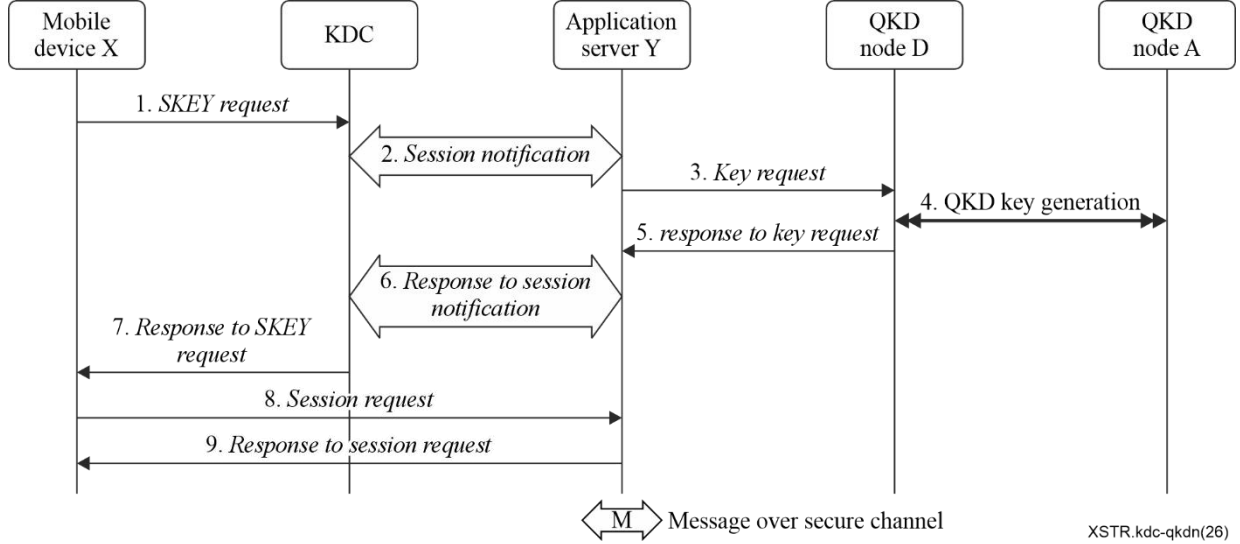


Figure I.1 – Protocol flow for authentication and SKEY distribution

1. Mobile device X sends a *SKEY request message* to the KDC, which contains:

(mobile device X → KDC): $ID_x, ID_y, N1_x, KeyID, MAC1$

Here, ID_x is the identifier of mobile device X, ID_y is the identifier of application server Y, $N1_x$ is a nonce generated by mobile device X, $KeyID$ is the identifier of the authentication and encryption key used by the mobile device, $MAC1$ is the message authentication code (MAC) generated based on the authentication and encryption key $AE-key$ by using the message authentication code algorithm, such as HMAC. The calculation formula is as follows:

$$MAC1 = \text{HMAC} (AE\text{-key}, ID_x \| ID_y \| N1_x \| KeyID)$$

Here $AE-key$ is the authentication encryption key corresponding to $KeyID$, and $\|$ is a string concatenation.

2. After receiving the *SKEY request message*, the KDC searches for the corresponding authentication and encryption key $AE-key$ in the store according to the $KeyID$ in the message. If it is not found, the process is terminated. When the authentication and encryption key $AE-key$ is found, $MAC1$ is verified by using this key. If the verification is successful, it proves that the *SKEY request message* has not been tampered with by the attacker and that the identity of the message sender is mobile device X. Based on ID_x and $KeyID$, the KDC determines from which mobile device X obtains the authentication and encryption key from which key update terminal, as well as the QKD node identifier in QKDN of that key update terminal. Consequently, the QKD source node (QKD node A) of the required key in QKDN is determined. According to ID_y , the KDC confirms that the QKD destination node for the required key in QKDN is QKD node D (connected to application server Y). The KDC sends a *session notification message* to the application server Y through a secure channel, which is:

(KDC $\rightarrow$ application server Y): $ID_x, ID_y, ID_{QKD-A}, ID_{QKD-D}$

Here ID_{QKD-A}, ID_{QKD-D} are the identifiers of QKD node A and QKD node D, respectively.

3. After receiving the *session notification message*, application server Y confirms that the key K_{AD} between QKD node A (connected to the key update terminal) and QKD node D node (connected to application server Y) is required according to ID_{QKD-A}, ID_{QKD-D} . Application server Y sends a *key request message* to the QKD node D, which is:

(application server Y $\rightarrow$ QKD node D): ID_{QKD-A}, ID_{QKD-D}

4. After receiving the *key request message*, QKD node D executes the key generation process between QKD node A and QKD node D and obtains the key K_{AD} .

5. QKD node D sends a *response to the key request message* to application server Y, which is:

(QKD node D $\rightarrow$ application server Y): K_{AD}

6. Application server Y makes use of the key K_{AD} as the *SKEY*. It generates a nonce N_y , and binds N_y with the *SKEY*. Application server Y sends a *response to the session notification message* through the secure channel to the KDC, which is:

(Application server Y $\rightarrow$ KDC): $SKEY, N_y$

7. The KDC returns to mobile device X a *response to the SKEY request message*, which is as follows:

(KDC $\rightarrow$ mobile device X): $ID_x, ID_{KDC}, N1x, N_y, [SKEY]_{AE-key}, MAC2$

Here ID_{KDC} is the identifier of the KDC. The *response to the SKEY request message* is the output of an authenticated encryption with associated data (AEAD) scheme [b-Rogaway]. $ID_x, ID_{KDC}, N1x, N_y$ are the associated data for the AEAD scheme. $[SKEY]_{AE-key}$ is the ciphertext, which means that the *SKEY* is encrypted with the AEAD scheme by using the authentication and encryption key *AE-key*. $MAC2$ is the message authentication tag, which is computed over the associated data and ciphertext by using the AEAD scheme.

8. After receiving the *response to the SKEY request message*, mobile device X compares $N1x$ in the *response to the SKEY request message* with $N1x$ in the *SKEY request message*. If they are equal, it can be determined that the *response to the SKEY request message* is not a replay message. It applies the AEAD scheme to obtain the *SKEY* and to verify $MAC2$ by using the authentication and encryption key *AE-key*. If the verification is successful, it proves that the communication partner is the KDC. mobile device X generates a *session request message* and sends it to application server Y, with the following content:

(mobile device X $\rightarrow$ Application server Y): $ID_x, ID_y, N2x, N_y, MAC3$

Here $N2x$ is a nonce generated by mobile device X, and $MAC3$ is a message authentication code generated by using the message authentication code algorithm (such as HMAC) on $ID_x, ID_y, N2x$ and N_y by using the *SKEY*. The calculation formula is as follows:

$$MAC3 = \text{HMAC} (SKEY, ID_x \| ID_y \| N2x \| N_y)$$

9. After receiving the *session request message*, application server Y compares the received N_y with the N_y sent in the *response to the session notification message*. If they are identical, it confirms that the *session request message* is not a replay message. It finds the corresponding *SKEY* based on N_y . Application server Y applies this key to verify $MAC3$. If the verification is successful, it proves the authenticity of mobile device X and the possession of the *SKEY*. Application server Y generates a *response to the session request message* and sends it to mobile device X, which is:

(Application server Y $\rightarrow$ mobile device X): $ID_y, ID_x, N2x, MAC4$

MAC4 is a message authentication code which is computed over *ID_y*, *ID_x*, *N2_x* with the message authentication code algorithm by using the *SKEY*, such as HMAC, the calculation formula is as follows:

$$MAC4 = \text{HMAC} (SKEY, ID_y \| ID_x \| N2_x)$$

After receiving the *response to the session request message*, mobile device X compares the received *N2_x* with the *N2_x* sent in the *session request message*. If they are equal, it confirms that the *response to the session request message* is not a replay message. Mobile device X applies the *SKEY* to verify *MAC4*. If the verification succeeds, it proves that the communication partner is application server Y and processes the *SKEY*.

Appendix II

Protocol for authentication and SKEY distribution (scheme 2)

The authentication and SKEY distribution protocol performs mutual authentication between a mobile device and the KDC and securely delivers the SKEY to the mobile device through the KDC. In addition, mobile device and application server performs mutual authentication with the SKEY for a secure connection. The protocol flow is illustrated in the Figure II.1.

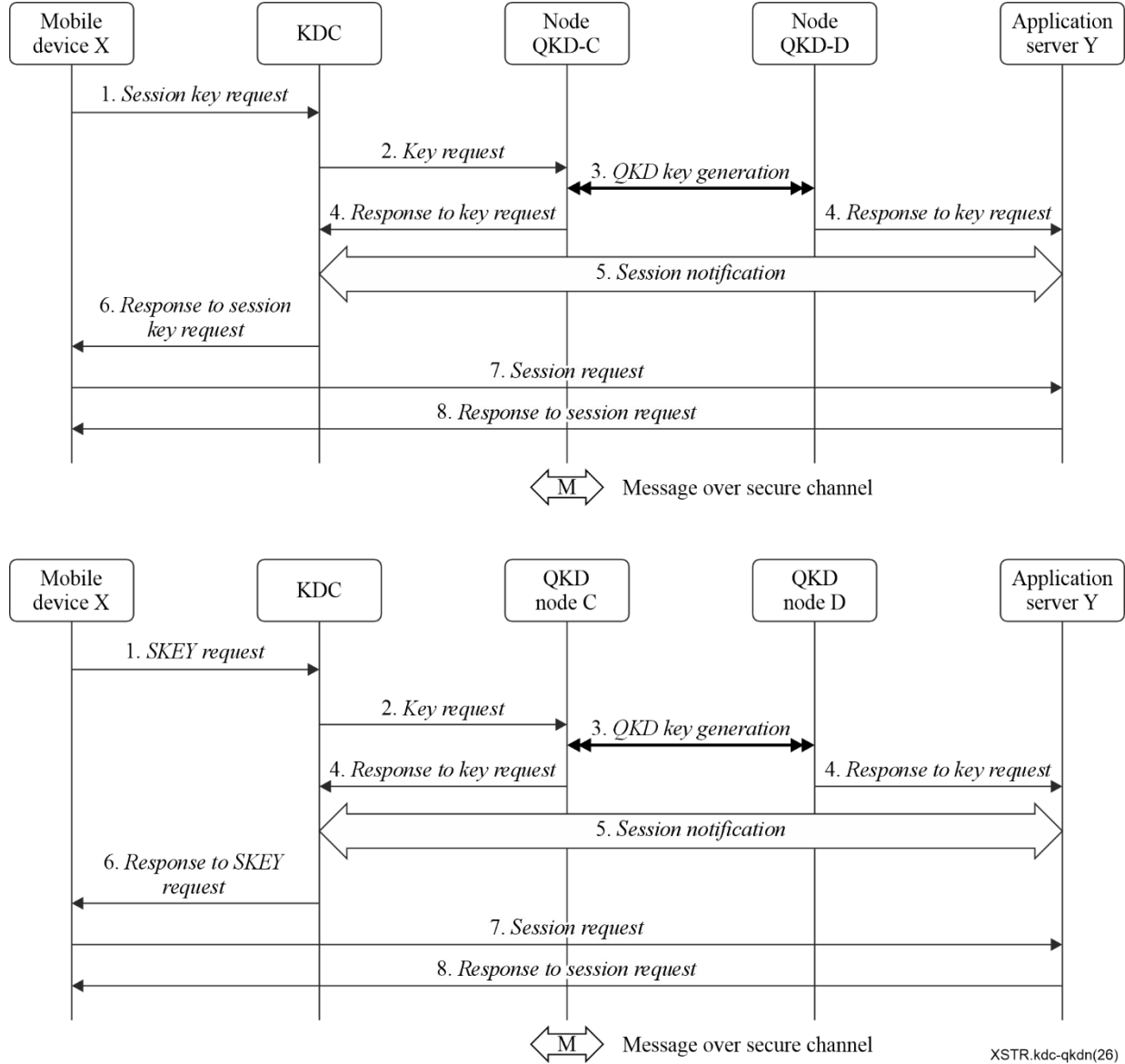


Figure II.1 – Protocol flow for authentication and SKEY distribution

1. Mobile device X sends a *SKEY request* message to the KDC, which contains:

(mobile device X $\rightarrow$ KDC): $ID_x, ID_y, N1_x, KeyID, MAC1$

Here, ID_x is the identifier of mobile device X, ID_y is the identifier of application server Y, $N1_x$ is a nonce generated by mobile device X, $KeyID$ is the identifier of the authentication encryption key used by the mobile device, $MAC1$ is the message authentication code generated based on the authentication and encryption key $AE-key$ by using the message authentication code algorithm, such as HMAC. The calculation formula is as follows:

$$MAC1 = \text{HMAC}(AE\text{-key}, IDx \| IDy \| N1x \| KeyID)$$

Here $AE\text{-key}$ is the authentication encryption key corresponding to $KeyID$, and $\|$ denotes the string concatenation.

2. After receiving the *SKEY request message*, the KDC searches for the corresponding authentication encryption key $AE\text{-key}$ in the store according to the $KeyID$ in the message. If it is not found, the process is terminated. When the authentication encryption key $AE\text{-key}$ is found, $MAC1$ is verified by using this key. If the verification is successful, it proves that the *SKEY request message* has not been tampered with by the attacker and that the identity of the message sender is mobile device X. According to IDx and IDy , KDC confirms that mobile terminal X requires a SKEY with the application server Y. According to IDy , KDC confirms that the QKD destination node for the required key in QKDN is node D (connected to application server Y). Therefore, KDC needs to request the key K_{CD} between node C (connected to KDC) and node D (connected to application server Y) from QKDN. KDC sends a *key request message* to the node C connected to it, which is:

$$(\text{KDC} \rightarrow \text{QKD node C}): ID_{QKD-C}, ID_{QKD-D}$$

Here, ID_{QKD-C} and ID_{QKD-D} are the identifiers of node C and node D, respectively.

3. After receiving the *key request message*, QKD node C executes the key generation process between QKD node C and QKD node D and obtains the key K_{CD} and key metadata containing the key identifier $KeyID_{CD}$. QKD node D also obtains both the key K_{CD} and the key metadata containing the key identifier $KeyID_{CD}$.

4. QKD node C sends a *response to key request message* to the KDC, which is:

$$(\text{QKD node C} \rightarrow \text{KDC}): K_{CD}, \text{key metadata:}$$

and QKD node D sends a *response to key request message* to application server Y, which is:

$$(\text{QKD node D} \rightarrow \text{application server Y}): K_{CD}, \text{key metadata}$$

5. KDC binds session information (IDx , IDy , $N1x$) with $KeyID_{CD}$ and sends the session information and $KeyID_{CD}$ to application server Y through a secure channel in the following *session notification message*.

$$(\text{KDC} \rightarrow \text{application server Y}): IDx, IDy, N1x, KeyID_{CD}$$

6. KDC returns to mobile device X a *response to SKEY request message*, which is as follows:

$$(\text{KDC} \rightarrow \text{mobile device X}): IDx, ID_{KDC}, N1x, [SKEY]_{AE\text{-key}}, MAC2$$

Here ID_{KDC} is the identifier of the KDC. The *response to SKEY request message* is the output of an authenticated encryption with associated data (AEAD) scheme [b-Rogaway]. IDx , ID_{KDC} , $N1x$ are the associated data for the AEAD scheme. $[SKEY]_{AE\text{-key}}$ is the ciphertext, which means that the SKEY is encrypted with the AEAD scheme by using the authentication and encryption key $AE\text{-key}$. $MAC2$ is the message authentication tag, which is output computed over the associated data and ciphertext by using the AEAD scheme.

7. After receiving the *response to SKEY request message*, mobile device X compares $N1x$ in the *response to SKEY request message* with $N1x$ in the *SKEY request message*. If they are equal, it can be determined that the *response to SKEY request message* is not a replay message. It applies the AEAD scheme to obtain the SKEY and to verify $MAC2$ by using the authentication encryption key $AE\text{-key}$. If the verification is successful, it proves that the communication partner is the KDC. mobile device X generates a *session request message* and sends it to application server Y, with the following content:

$$(\text{mobile device X} \rightarrow \text{application server Y}): IDx, IDy, N1x, N2x, MAC3$$

Here $N2x$ is a nonce generated by mobile device X, and $MAC3$ is a message authentication code generated by using the message authentication code algorithm (such as HMAC) on IDx , IDy , $N1x$, $N2x$ by using the SKEY. The calculation formula is as follows:

$$MAC3 = \text{HMAC} (SKEY, IDx \| IDy \| N1x \| N2x)$$

8. After receiving the *session request message*, application server Y locates the corresponding key identifier $KeyID_{CD}$ based on the session information (IDx , IDy , $N1x$), and subsequently retrieve the key K_{CD} , which serves as the SKEY. Application server Y employs this key to authenticate $MAC3$. A successful authentication confirms that mobile terminal X is legitimate and possesses the SKEY. Upon successful verification, application server Y generates a *session request response message* and transmits it to mobile terminal X with the following content:

(Application server Y $\rightarrow$ Mobile terminal X): IDy , IDx , $N2x$, $MAC4$

$MAC4$ is generated using a message authentication code algorithm, such as HMAC, based on the SKEY and the concatenation of IDy , IDx , and $N2x$. The formula for generating $MAC4$ is as follows:

$$MAC4 = \text{HMAC} (SKEY, IDy \| IDx \| N2x)$$

Upon receiving the session request response message, mobile terminal X compares the received $N2x$ with the previously transmitted $N2x$ to ensure it has not been replayed. Mobile terminal X then uses the SKEY to authenticate $MAC4$. A successful authentication confirms that the peer is indeed application server Y and that it possesses the SKEY.

Appendix III

Protocol for authentication and SKEY distribution (Scheme 3)

The authentication and SKEY distribution protocol performs mutual authentication between a mobile device and the KDC and securely delivers the SKEY K_{AC} to the mobile device through the KDC. In addition, the mobile device and application server perform mutual authentication with the SKEY for a secure connection. The protocol flow is illustrated in Figure III.1.

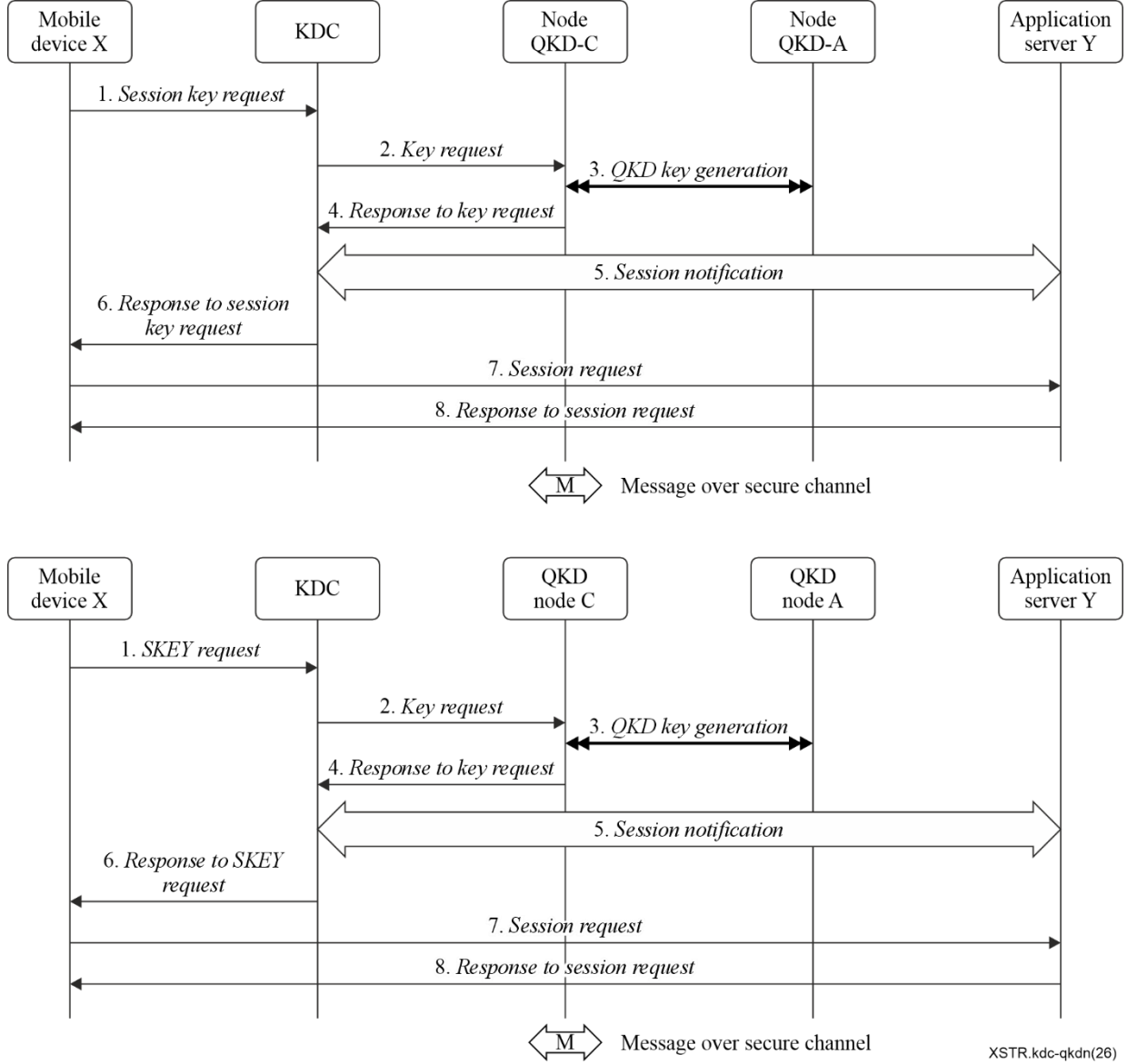


Figure III.1 – Protocol flow for authentication and SKEY distribution

1. Mobile device X sends a *SKEY* request message to the KDC, which contains:

(mobile device X $\rightarrow$ KDC): ID_x , ID_y , $N1_x$, $KeyID$, $MAC1$

Here, ID_x is the identifier of mobile device X, ID_y is the identifier of application server Y, $N1_x$ is a nonce generated by mobile device X, $KeyID$ is the identifier of the authentication and encryption key used by the mobile device, $MAC1$ is the message authentication code generated based on the authentication and encryption key AE -key by using the message authentication code algorithm, such as HMAC. The calculation formula is as follows:

$$MAC1 = \text{HMAC}(AE\text{-key}, IDx \| IDy \| N1x \| KeyID)$$

Here $AE\text{-key}$ is the authentication encryption key corresponding to $KeyID$, and $\|$ denotes the string concatenation.

2. After receiving the *SKEY request message*, the KDC searches for the corresponding authentication and encryption key $AE\text{-key}$ in the store according to the $KeyID$ in the message. If it is not found, the process is terminated. When the authentication and encryption key $AE\text{-key}$ is found, $MAC1$ is verified by using this key. If the verification is successful, it proves that the *SKEY request message* has not been tampered with by the attacker and that the identity of the message sender is mobile device X. According to IDx and IDy , KDC confirms that mobile terminal X requires a SKEY with the application server Y. According to IDy , KDC confirms that the QKD destination node for the required key in QKDN is node D (connected to application server Y). For this purpose, KDC can request the key K_{AC} between node QKD-C (connected to KDC) and node QKD-A (connected to the key update terminal) from QKDN. KDC sends a *key request message* to the node C connected to it, which is:

$$(\text{KDC} \rightarrow \text{QKD node C}): ID_{QKD-C}, ID_{QKD-A}$$

Here, ID_{QKD-C} and ID_{QKD-A} are the identifiers of QKD node C and QKD node A, respectively.

3. After receiving the *key request message*, QKD node C executes the key generation process between QKD node C and QKD node A and obtains the key K_{AC} and key metadata containing the key identifier $KeyID_{AC}$.

4. QKD node C sends a *response to the key request message* to the KDC, which is:

$$(\text{QKD node C} \rightarrow \text{KDC}): K_{AC}, \text{key metadata}$$

5. KDC binds session information ($IDx, IDy, N1x$) with K_{AC} and $KeyID_{AC}$ and sends the session information and $KeyID_{AC}$ to application server Y through a secure channel in the following *session notification message*.

$$(\text{KDC} \rightarrow \text{application server Y}): IDx, IDy, N1x, K_{AC}, KeyID_{AC}$$

6. KDC returns to mobile device X a *response to the SKEY request message*, which is as follows:

$$(\text{KDC} \rightarrow \text{mobile device X}): IDx, ID_{KDC}, N1x, [SKEY]_{AE\text{-key}}, MAC2$$

Here ID_{KDC} is the identifier of the KDC. The *response to the SKEY request message* is the output of an authenticated encryption with associated data (AEAD) scheme [b-Rogaway]. $IDx, ID_{KDC}, N1x$ are the associated data for the AEAD scheme. $[SKEY]_{AE\text{-key}}$ is the ciphertext, which means that the SKEY is encrypted with the AEAD scheme by using the authentication and encryption key $AE\text{-key}$. $MAC2$ is the message authentication tag, which is computed over the associated data and ciphertext by using the AEAD scheme.

7. After receiving the *response to the SKEY request message*, mobile device X compares $N1x$ in the *response to SKEY request message* with $N1x$ in the *SKEY request message*. If they are equal, it can be determined that the *response to SKEY request message* is not a replay message. It applies the AEAD scheme to obtain the SKEY and to verify $MAC2$ by using the authentication encryption key $AE\text{-key}$. If the verification is successful, it proves that the communication partner is the KDC. mobile device X generates a *session request message* and sends it to application server Y, with the following content:

$$(\text{Mobile device X} \rightarrow \text{application server Y}): IDx, IDy, N1x, N2x, MAC3$$

Here $N2x$ is a nonce generated by mobile device X, and $MAC3$ is a message authentication code generated by using the message authentication code algorithm (such as HMAC) on $IDx, IDy, N1x, N2x$ by using the SKEY. The calculation formula is as follows:

$$MAC3 = \text{HMAC} (SKEY, IDx \| IDy \| N1x \| N2x)$$

8. After receiving the *session request message*, application server Y locates the corresponding key identifier $KeyID_{AC}$ based on the session information (IDx , IDy , $N1x$), and subsequently retrieve the key K_{AC} , which serves as the SKEY. Application server Y employs this key to authenticate $MAC3$. A successful authentication confirms that mobile terminal X is legitimate and possesses the SKEY. Upon successful verification, Application server Y generates a *session request response message* and transmits it to mobile terminal X with the following content:

(Application server Y $\rightarrow$ mobile terminal X): IDy , IDx , $N2x$, $MAC4$

$MAC4$ is generated using a message authentication code algorithm, such as HMAC, based on the SKEY and the concatenation of IDy , IDx , and $N2x$. The formula for generating $MAC4$ is as follows:

$$MAC4 = \text{HMAC} (SKEY, IDy \| IDx \| N2x)$$

Upon receiving the session request response message, mobile terminal X compares the received $N2x$ with the previously transmitted $N2x$ to ensure it has not been replayed. Mobile terminal X then uses the SKEY to authenticate $MAC4$. A successful authentication confirms that the peer is indeed application server Y and that it possesses the SKEY.

Appendix IV

Brief introduction of Kerberos

Kerberos is a typical example of a KDC in practical applications. It utilizes symmetric cryptographic algorithms rather than asymmetric cryptographic algorithms to provide an authentication service and distribute SKEYs. An authentication server (AS) and a ticket granting server (TGS) are the two components that make up the KDC. The former is used to authenticate a client and create a ticket-granting ticket. The latter is used to issue a service-granting ticket to the client for a dedicated application server. The overview of Kerberos is illustrated in Figure IV.1.

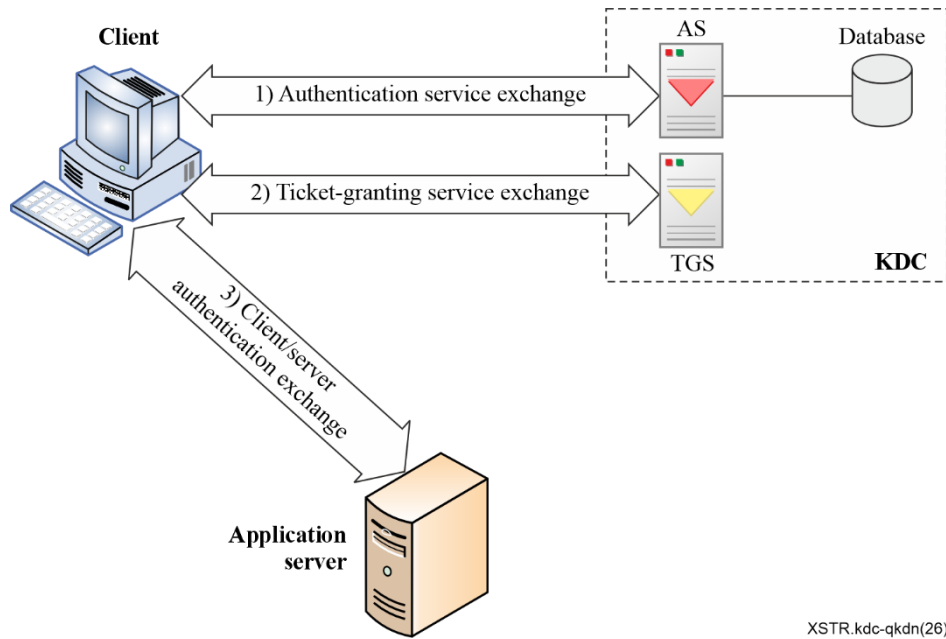


Figure IV.1 – Overview of Kerberos

The working flow of Kerberos consists of three phases: (1) the authentication service exchange, (2) the ticket-granting service exchange, and (3) the client/server authentication exchange and service request phase. In the authentication service exchange phase, the client initiates the need for a ticket-granting ticket request. The AS verifies the identity of the client by using the client's password stored in the database. For a successful authentication, AS creates a ticket-granting ticket and generates a SKEY. These results are encrypted using the key derived from the client's password and returned to the client. In the ticket-granting service exchange phase, the client decrypts the incoming message with its password, then sends the ticket-granting ticket and the authenticator containing the client's name, network address, and timestamp to TGS. TGS decrypts the ticket-granting ticket and authenticator, verifies the request, and creates a service-granting ticket which contains the SKEY in addition to the service-related information. The TGS then delivers the service-granting ticket to the client. In the client/server authentication phase, the client invokes a service request to an application server by using the service-granting ticket and authenticator. The application server verifies the ticket and authenticator, then grants access to the service. If mutual authentication is required, the application service responds to the client with an authenticator. Finally, the client and application server can protect their communications with the SKEY. The benefits of Kerberos are that the client does not need to maintain a credential for each application server for service access, instead the client needs merely one credential shared with AS to access a number of application servers.

Bibliography

- [[b-ITU-T X.1252](#)] Recommendation ITU-T X.1252 (2021), *Baseline identity management terms and definitions*.
- [[b-ITU-T Y.3800](#)] Recommendation ITU-T Y.3800 (2019), *Overview on networks supporting quantum key distribution*.
- [b-ETSI GR QKD 007] ETSI GR QKD 007 V1.2.1 (2026-01), *Quantum Key Distribution (QKD); Vocabulary*
- [b-IETF RFC 4120] IETF RFC 4120 (2005), *The Kerberos Network Authentication Service (V5)*.
- [b-ISO/IEC 11770-2:2018] ISO/IEC 11770-2:2018, *IT Security techniques – Key management – Part 2: Mechanisms using symmetric techniques*.
- [b-Mayers] D. Mayers (2001), *Unconditional security in quantum cryptography*, J. ACM 48,351-406.
- [b-NIST-Criteria] NIST: *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*,
<<https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>>
- [b-NIST-SP800-133Rev2] NIST Special publication 800-133 Revision2 (2020), *Recommendation for Cryptographic Key Generation*, June.
- [b-NIST SP 800-226] NIST Special publication 800-226 (2025), *Guidelines for Evaluating Differential Privacy Guarantees*.
- [b-Rogaway] P. Rogaway (2022), *Authenticated-encryption with associated-data*. *Ninth ACM Conference on Computer and Communications Security (CCS-9)*. ACM Press, 18 November.
- [b-Sasaki] Masahide Sasaki (2018), *Quantum Key Distribution and Its Applications*, IEEE Security & Privacy, Vol: 16 Issue: 5, September/October.
-