IOWN GLOBAL FORUM ™

Functional Architecture for Protection of Data in Motion: Multi-Factor Security Key Exchange and Management

Classification: APPROVED REFERENCE DOCUMENT

Confidentiality: PUBLIC

Version 1.0

October 24, 2024

[MFS Functional Architecture]

Legal

Contents

List of Figures

List of Tables

1. Introduction

- In recent years, there has been a substantial amount of research on quantum computers – machines that exploit quantum mechanical phenomena to solve mathematical problems that are difficult or intractable for conventional computers. If large-scale quantum computers (e.g. fault-torrent quantum computer capable of handling 1 million qubits) are ever built, they will be able to break many of the public-key cryptosystems currently in use [NIST Post-Quantum Cryptography Standardization]. Even though a quantum computer capable of breaking cryptography does not yet exist, an attack in which an attacker collects encrypted data over a long period of time now and attempts to decrypt it in the future when the performance of quantum computers improves is called a "store now, decrypt later attack"[Store Now and decrypt later] and should be viewed as a threat.
- In the Technology Outlook of Information Security [IOWN GF SEC OUTLOOK], the Multi-Factor Security (MFS) concept is described as a means of achieving End-to-End post-quantum cryptographic communications with high crypto-agility on IOWN infrastructure in the quantum computers era where existing public key cryptography-based key exchange methods are no longer secure.
- This document describes a functional architecture for protection of data in motion, especially, key exchange and management for realizing quantum-safe cryptographic communication based on the MFS concept on IOWN infrastructure.
- When a key exchange method is compromised, the compromised security method should be quickly replaced with a secure security method. In addition, a mechanism is to quickly understand the situation that how and where these compromised security methods are being used. The objective of this document is to define an open architecture for achieving high crypto-agility on a key exchange in a multi-vendor implementation.
- The scope of this document is as follows;
  - Clarification of functional requirements for achieving post-quantum cryptographic communication based on the IOWN infrastructure.
  - Definition of an open architecture that fulfills the functional requirements clarified above (hereafter MFS key exchange and management functional architecture).
  - Descriptions of use case examples of MFS key exchange and management functional architecture.

2. Overview of MFS Key Exchange and Management Functional Architecture

2.1. Positioning of MFS Key Exchange and Management Functional Architecture

This section describes the overall cryptographic communication architecture using the MFS concept and the positioning of the MFS key exchange and management functional architecture focused that is the subject of this document.

Figure 2.1-1 shows an overview of a cryptographic communication architecture based on the MFS concept. It is a hierarchical structure consisting of three layers, i.e., MFS key exchange, MFS key management and cryptographic communication layers.

- The MFS key exchange layer: these layers provide key exchange between two nodes. The MFS key exchange layer may use multiple networks to reduce the risk of eavesdropping.
- The MFS key management layer: this layer provides key synchronization between two nodes, as well as key combining and key update management on the nodes. This layer is independent of the cryptographic protocol (e.g., IPsec, MACsec, OTNsec) required for key management.
- The cryptographic communication layer: this layer provides encryption/decryption functions for secure data communication between two nodes. This layer depends on the cryptographic protocol.

This document focuses on the MFS key exchange layer and the MFS key management layer in the cryptographic communication architecture while excluding the cryptographic communication layer.

We state the requirements for the MFS key exchange layer and the MFS key management layer in section 3 and define the basic functional architecture including object model and semantic level I/F in section 4. From the point of view of utilizing MFS key exchange and MFS key management functions, we also briefly describe the cryptographic communication layer. Since the functional definition of such a layer may vary depending on the implementation, only specific examples on the configuration of encryption/decryption in the cryptographic communication layer are discussed.

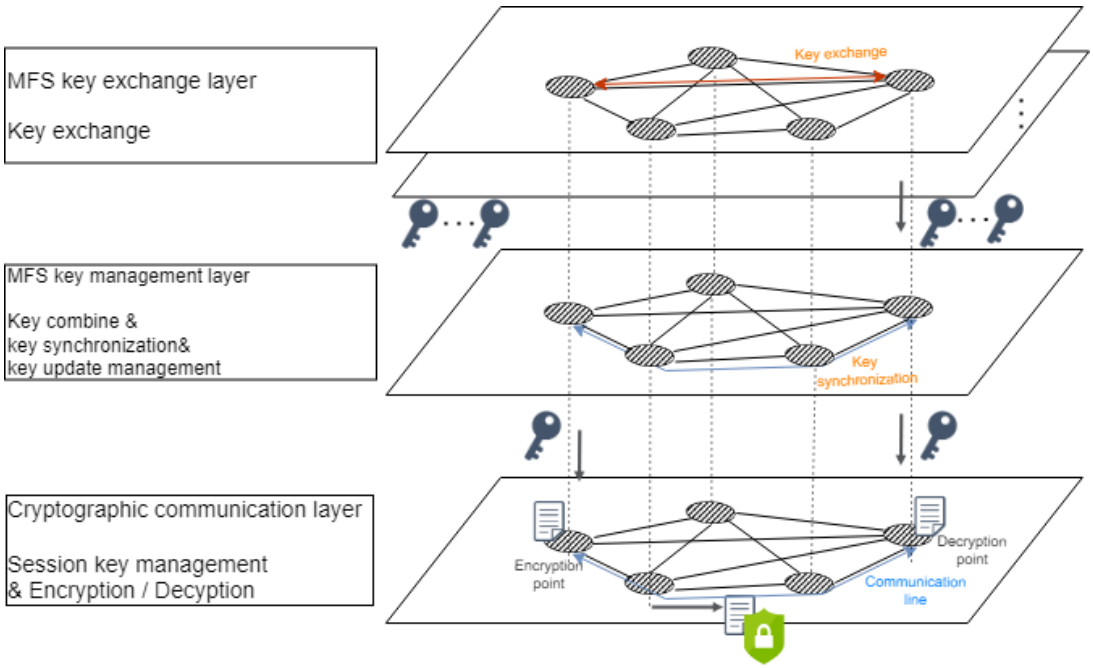

Figure 2.1-1: Overview of cryptographic communication architecture based on the MFS concept

2.2. Communication Endpoint of MFS Key Exchange and Management Functional Architecture

Communication Endpoint

For the cryptographic communication layer, the communication starts at data communication endpoints, e.g., application processes. Typically, encryption/decryption points as shown in Figure 2.1-1 matches the data communication endpoints. The exact definition of “endpoint” in IOWN infrastructure is the point where protected data are generated, processed, or consumed. Generally, the “endpoint” will be application processes within an end node.

It is also possible that encryption/decryption points and data communication endpoints do not match (it is referred to as “channel encryption” and is described in section 5.1).
However, when considering end-to-end data protection, even in the context of channel encryption, the “endpoint” should continue to follow the exact definition above, i.e., that it is the application process.

Near-endpoint

Near-endpoint is *a location* of the processes or devices independent of the endpoint, where an encryption/decryption process can exist independent of the endpoint application process. It should be noted the definition does not directly mean an entity that provides encrypted communications as a service and is not necessarily consistent with hardware acceleration (offload*). If processes or devices independent of the endpoint are secured at the same level as the application processes that are the endpoints, some implementers may choose to implement encryption functions in such an embedded process and device. IOWNsec should allow such deployments.

However, the security level of such deployments depends on the security level of the execution environment containing the application process and the encryption function. Measures for the protection of this environment are outside the scope of this version of IOWNsec. It is important to note that the security level of systems that rely on near-endpoint encryption functions depends on the nature of the specific implementation. Further, it should be noted that in many cases the near-endpoint is owned by a third party.

Specific examples of implementing an encryption function using subdevice and processes which are embedded on the devices as near-endpoint are as follows. CPS/AIC use case workloads include an image analysis process and TLS encryption/decryption service to protect data transfer, and these two workloads work together in a single node. The former is considered an endpoint, and the latter is considered a near-endpoint. Another example is a mobile device which has a single application. In this case, the device's security level is almost the same as the application process's one.

The near-endpoint encryption/decryption can be a case where encryption and decryption are performed in a separate process in object encryption, which applies to entire data objects, or a case where the service is implemented in channel encryption, which provides an encrypted channel (The conceptual distinction of object and channel encryption will be explained in section 5.1).

In the case of object encryption, there are cases where an encryption function is implemented in processes and devices other than endpoint application processes within the end node. This case corresponds to near-end implementation. Similarly, in the case of channel encryption, the encryption function is implemented in the device or process in the point of edge, e.g., a network interface card and APN-T, instead of application processes. Therefore, it also corresponds to the near-end point implementation.

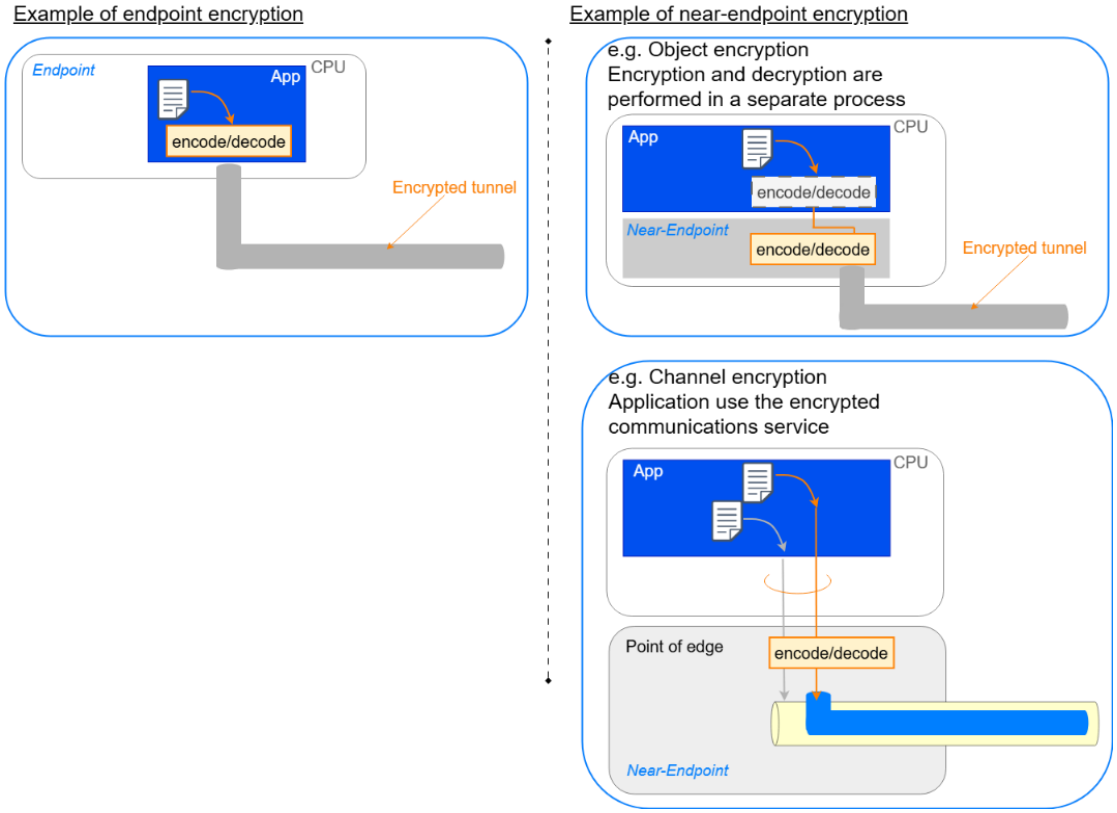

Figure 2.2-1: Example of Near-endpoint encryption

*The definition of the word "offload" is strictly different from that of "near-endpoint." The word means that an entity that performs encrypted communication can outsource the process to an external device or shared component equipped with hardware acceleration functions to share the load. In IOWN infrastructure, possible offload destinations include APN-T, flexible bridges, network devices separate from the Open APN, smart NICs, VNICs, kernels, and so on. Offload is an implementation option that is common in both object encryption and channel encryption. In the basic concept of offload, the OS and hardware such as smart-NICs only provide encode-decode, not the encrypted communication itself. It is a computational aid like a coprocessor. However, in actual implementations, there are cases where encrypted communication is terminated at the offload destination.

3. Requirements and Directions for Realizing MFS Key Exchange and Management Functional Architecture

3.1. Achieving the Post-quantum Security with High Crypto-agility

To maintain the security of the IOWN GF architecture over the long term, the IOWN GF architecture must be prepared for new threats to cryptographic algorithms. This is because one never knows when a vulnerability will be discovered, even if it is a state-of-the-art algorithm. With the advent of quantum computers, these threats are becoming more and more prevalent. In other words, it is desirable to increase “crypto-agility”, which is the ability of crypto systems to respond quickly and flexibly to new threats. Conventional cryptographic protocols consist of vertically integrated key exchange functions, key management functions, encryption functions, etc. In such a configuration, if a new attack method is discovered and some of these functions are compromised, it is difficult to update them without affecting other parts of the system. Therefore, in order to achieve crypto-agility, it is necessary to have a configuration that can flexibly update these functions without having to integrate them in advance (i.e. disaggregation).

To summarize the above requirements, MFS key exchange and management functional architecture must have the following features.

- A feature that generates a cryptographic key by combining keys obtained from multiple quantum-resistant key exchange methods.
- A feature that switches the key exchange methods to maintain security at any time depending on the situation.

Candidates for post-quantum key exchange methods that systems in the IOWN era should be equipped with include: post-quantum cryptography key encapsulation mechanism (PQC KEM), quantum key distribution, and pre-shared key.

3.2. Achieving a Balanced Cost-security Level

Since various security levels are required for the services realized by the IOWN GF architecture, it is difficult to uniformly specify security measures and strengths. For some systems, expensive, complex, and resource-intensive security solutions are not always optimal. Therefore, it is necessary to be able to flexibly select the means and strength of the data protection solution to maintain a balance between system security and cost.

An open and disaggregated architecture is required that will allow users to flexibly select the security solution that best meets their individual needs. By having an architecture that allows users to choose only the key exchange means, for example, rather than the conventional vertically integrated architecture, users can select the optimal balance between security level and cost according to their use cases.

An open and disaggregated architecture that is independent of terminals and cryptographic protocols can also reduce the time and cost of cryptographic migration. In general, replacing a compromised scheme with an alternative secure scheme takes time and leaves the user exposed to threats for the duration of the migration period. An open and disaggregated architecture is expected to shorten this migration period by making it easier to switch between partial security measures.

3.3. Achieving Crypto-Agility without Compromising the Benefits of IOWN GF Technologies

It is necessary to ensure that the security features we develop do not interfere with the performance requirements that the IOWN infrastructure is trying to achieve (e.g., high capacity, low latency, and high energy efficiency).

To fulfil these requirements, the following directions could be considered.

- High capacity and low latency communication requires fast encryption and decryption. To achieve it, offloading of the cryptographic processing to an accelerator or other device should be considered.
- Repeated use of an encryption key with the same source key increases the risk that the key will be decrypted. It is necessary to have a mechanism to continue supplying keys even if the communication between the key exchange layer, key management layer, and the cryptographic communication layer fails.

3.4. Achieving Zero Trust Security Based on the IOWN GF Security Model

This section describes several requirements to ensure zero trust security for data in motion. From a business continuity perspective, it is necessary not only to provide advanced security services using cutting-edge technologies, but also to provide the freedom to allow applications to design security from the perspective of their own business continuity requirements. Therefore, the three common security requirements for the IOWN GF are as follows.

(1) Freely assign application data protection classes to isolated, secure areas:
IOWN-GF service, e.g., network services and computing services, should be provided so that applications can operate in terms of business continuity (minimize damage and minimize recovery), assuming that they will one day be penetrated. The applications perform resource definition considering the impact in the event of data leakage or tampering. Resources mapped into an isolated, secure area provided by the IOWN GF should be securely transferred by encrypted data communication. Simply using a single encrypted communication to prevent leakage or tampering is not sufficient for zero trust security. When considering zero trust, an application must be able to define application data protection classes (resources by magnitude of attack) and map these classes to isolated secure areas (multiple encrypted communications) to manage based on a "need-to-know" policy. To protect data communication based on zero trust security, granular control over encryption and decryption is required so that security for each dataset is separately managed depending on the impact in the event of data leakage/tampering.

(2) Provide cutting-edge security services:
It is accepted that malicious attacks from criminals and the security measures that thwart such attacks will continue to evolve in reaction to one another. As a result, an application service provided by IOWN GF needs to be most secure at that time, so that cutting-edge security solutions capable of thwarting advanced attacks are necessary to continue to ensure such requirements. For example, applying the concept of MFS, which combines various methods of key-exchange will be described in section 4. In addition, cutting-edge key-exchange technology and its use in the MFS key exchange and management system will be described in section 5.

(3) Provide monitoring functions within the platform to detect unknown attacks and report them to the application:
Even if the security of IOWN GF infrastructure is highly advanced, there is a possibility that a new, hitherto unknown method of attack could penetrate secure data communication and/or key exchange systems at some point. Therefore, it is necessary to monitor for potential intrusions by advanced attackers using new techniques by monitoring application behavior for unusual activity, preferably before the attack can have an impact on the system. When an unknown attack (i.e., detection of unusual behavior) is detected, it should be reported to the application so that appropriate action can be taken according to the data protection class defined by the application.

As an example, consider a case where QKD-based key exchange method is applied. In this case, it is assumed that the QKD network is deployed on the MFS key exchange layer. The QKD network enables the application to detect security incidents such as a malicious and unknown attack by monitoring physical performance information defined by ITU-T Y.3811. In addition to pre-defined detection mechanisms, detection mechanisms based on inconsistencies, and unusual frequency of accesses, or other anomalous behavior. etc., by monitoring internal data related to the implementation of the QKD system, need to be considered. These functions can monitor the possibility of unknown attacks by detecting unusual behaviors that can be monitored within the implementation system. These monitoring functions are out of scope of this document.

4. MFS Key Exchange and Management Functional Architecture

4.1. MFS Key Exchange and Management Functional Architecture

The increase in computing power and the emergence of future quantum computers are expected to jeopardize current cryptographic protocols. AES256, which is currently used in cryptography, and PQCs, which are currently being standardized are unlikely to be compromised anytime soon, but the discovery of new attacks could well change the specification of new hardware and software implementations. In order to quickly take measures against these threats without making major changes to the system infrastructure, an architecture that supports crypto-agility is required, which allows the cryptographic algorithm to be replaced with another one if it becomes compromised.

Conventional cryptographic protocols consist of vertically integrated key exchange functions, key management functions, encryption functions, etc. In such a configuration, if a new attack method is discovered and some of these functions are compromised, it is difficult to replace them without affecting other parts of the architecture. Therefore, to achieve crypto-agility, it is necessary to have a configuration that can flexibly update these functions without having to integrate them in advance (i.e. modularity).

The figure below shows MFS key exchange and management functional architecture.

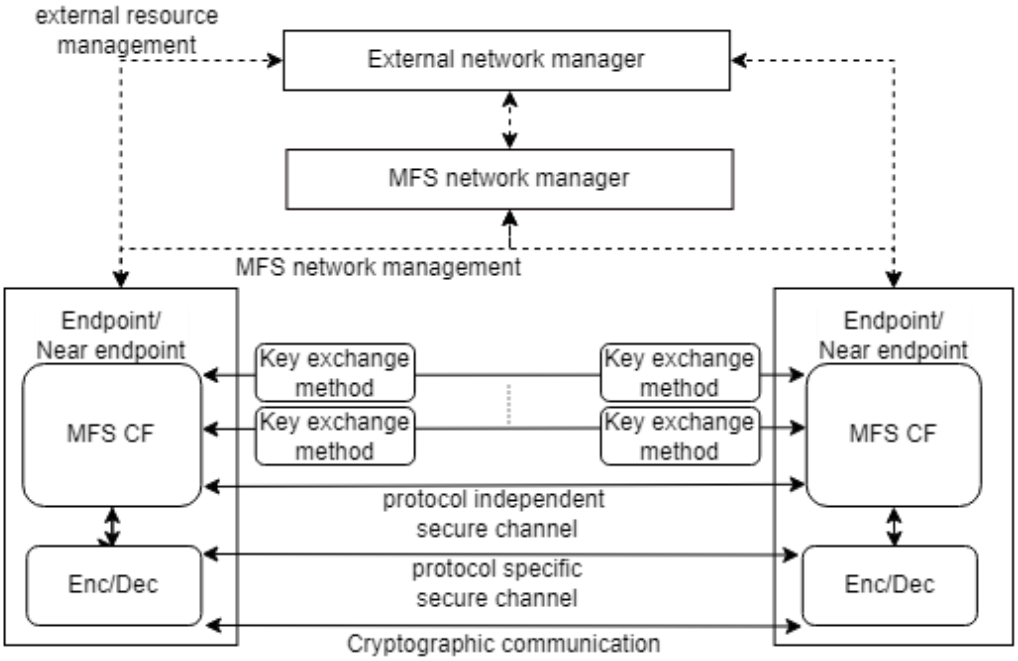

Figure 4.1-1: MFS key exchange and management functional architecture

The definitions of MFS key exchange and management functional architecture are given below.

- Enc/Dec (Encryption/Decryption): An entity that performs the encryption and decryption processes, which may be embedded in the application that handles data to be protected or may be external to the application. It is necessary to synchronize keys between the encryption point and decryption point so they can share the common knowledge of which data is encrypted by which key. Key management functions that depend on cryptographic protocols are included in this object.

- MFS Control Function (MFS CF): An entity that operates one or more key exchange method modules to derive encryption/decryption keys and provides the keys to Enc/Dec. This entity includes functions such as:
  - Key request function to transfer exchanged keys with the communication counterpart from key exchange methods.
  - Key derivation function (KDF) to generate keys that will be supplied to Enc/Dec. A key combiner may be included in this function.
  - Cryptographic communication protocol-independent-key management functions such as negotiating key exchange methods, sharing key ID with the communication counterpart and updating keys.
  - Function obtaining network information for key exchange from the network manager.
- MFS Network Manager(optional): An entity that manages the key exchange and encryption protocols used by MFS CF, the routing information used for key exchange and encrypted communication, and key information (e.g. identifier, intended use). MFS network refers to a network consisting of the MFS CFs and key exchanges. It may communicate with an external network manager, which manage networks outside of the MFS network, to request a network such as out-of-band control channels and key exchanges, and to get network connectivity status information.
- Key exchange method: A key exchange module that includes the key exchange service provided by trusted third party. There are possible modules that implement a single key exchange method and hybrid key exchange modules that are equipped with multiple key exchange methods. KDF (key derivation function) including key combining may be included in this function. Each key exchange algorithm used must have post-quantum security. Furthermore, it is desirable that the algorithms to be combined have different reliability grounds from the viewpoint of the combination effect.
- External network manager: External controllers that control networks outside of MFS network (e.g., container orchestrater, path computation element, QKDN-C, and APN-C).

4.2. Object Model of MFS Key Exchange and Management Functional Architecture

The figure below shows an object model for MFS key exchange and management functional architecture.

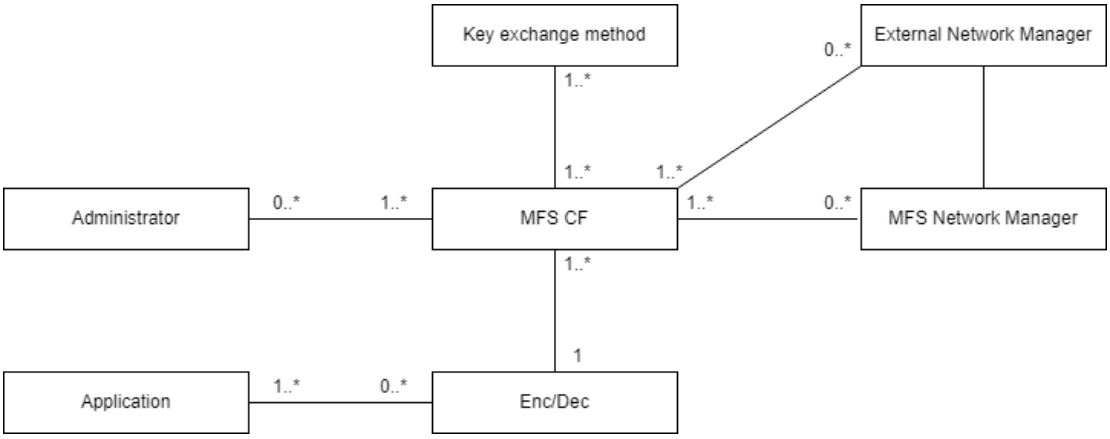

Figure 4.2-1: Object model for MFS key exchange and management functional architecture

The definitions of MFS key exchange and management functional architecture objects are given below.

- Application: Application here refers to software that handles data to be protected.
- Administrator: An administrative user who performs settings, changes, maintenance, and operations related to the MFS key exchange and management system.

4.3. MFS Key Exchange and Management Functional Architecture I/F

4.3.1. I/F from application deployer and administrator view

The figure below shows MFS key exchange and management functional architecture I/F from the application deployer and administrator view.

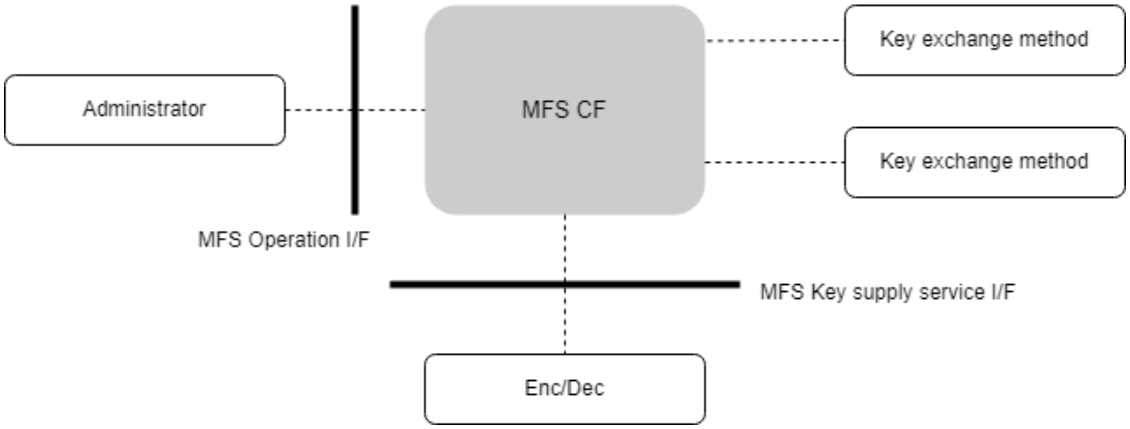

Figure 4.3-1: MFS key exchange and management functional architecture I/F from application deployer and administrator view

- MFS Key Supply Service I/F

  An I/F that supplies keys to Enc/Dec. This I/F also allows the MFS CF to obtain information from the Enc/Dec in order for the MFS Network Management to manage the key information (e.g. identifier, intended use). This may provide keys on a session and resource basis.

  Two major patterns are assumed: a pull-type key supply that accepts key get requests from Enc/Dec and periodic push-type key supply.

- MFS Operation I/F

  An I/F for administrators to set up key exchange methods and KDF and to perform system maintenance and operation.

4.3.2. I/F from MFS key exchange and management system deployer view

The figure below shows MFS key exchange and management functional architecture I/F from MFS key exchange and management system deployer view.

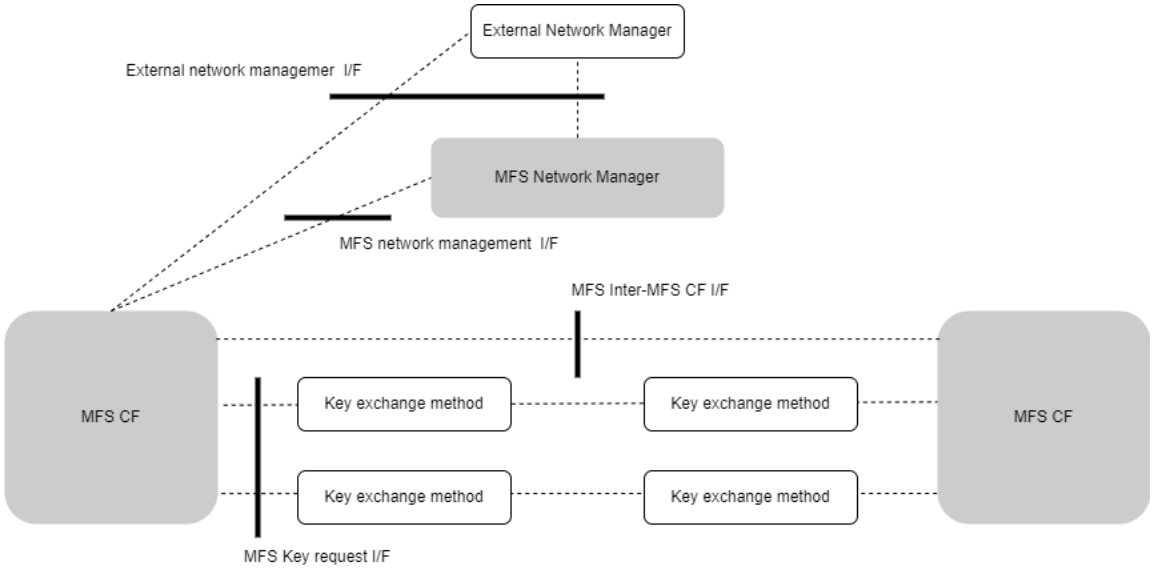

Figure 4.3-2: MFS key exchange and management functional architecture I/F from MFS key exchange and management system deployer view

- MFS Key request I/F

  An I/F to request key exchange method modules connected to the MFS CF to exchange keys with its counterpart.

- MFS Network Management I/F(optional)

  I/F for managing status of key exchange and encryption protocols.

- MFS Inter-MFS CF I/F

  A key management I/F that shares key IDs of exchanged keys with the opposing MFS CF. Sharing or negotiating key exchange methods and key derivation function (KDF) may be performed through this I/F. Key management through this I/F is independent of cryptographic communication protocols and devices. Cryptographic communication protocol-dependent key management is performed on the Enc/Dec side.

- External network manager I/F(optional)

  An I/F to obtain network information necessary for key exchange with the communication partner from External Network Manager. This I/F can also be used to request the network for key exchange and encrypted communication to the external network manager.

4.4. MFS Key Exchange and Management Functional Architecture for Modularity

- Figure 4.4-1 shows the MFS key exchange and management functional architecture for modularity.

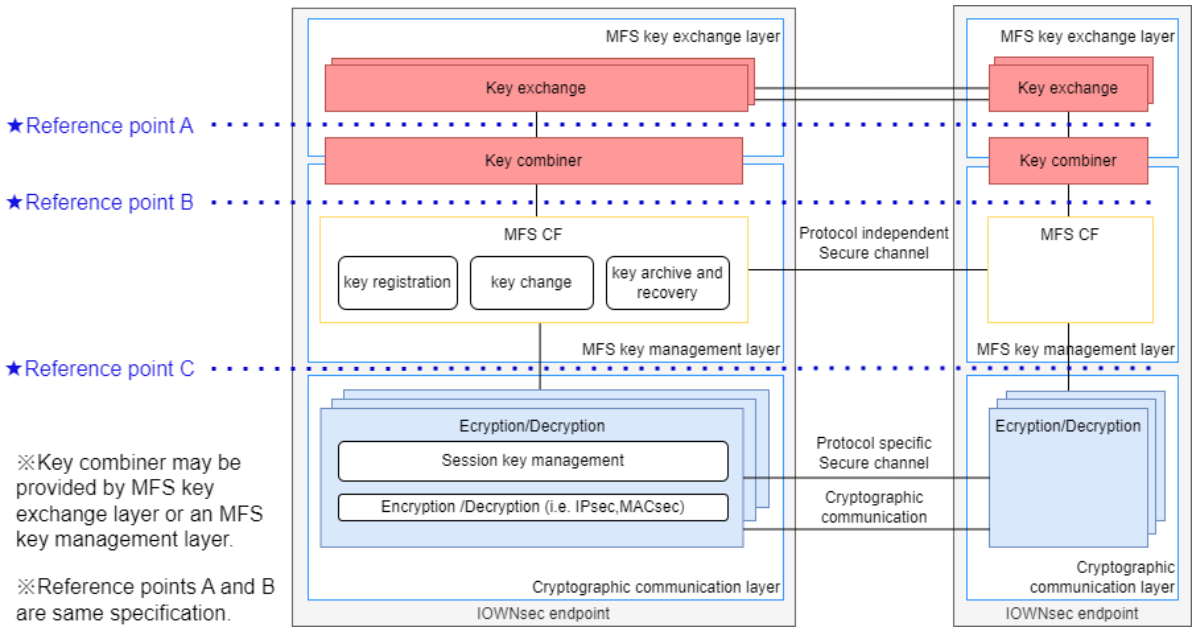

Figure 4.4-1: MFS key exchange and management functional architecture for modularity

The term is defined below.

- Key combiner: Entity to combine secure methods of key exchanges. The specific key combining methods are described in [NIST SP 800-133 Rev.2].
- Session key management: A Management function involving the handling of session keys and other related security parameters during the entire lifecycle of the keys, including their generation, storage, establishment, entry and output, use, and destruction.
- Key registration: A function in the lifecycle of a cryptographic key; the process of officially recording the keying material by a registration authority. [NIST SP 800-57 Part 1 Rev. 5]
- Key archive and recovery (optional): A function in the lifecycle of keying material; mechanisms and processes that allow authorized entities to retrieve or reconstruct keying material from key backup or archive. [NIST SP 800-57 Part 1 Rev. 5]
- In addition, the reference points that provide the secure key are defined as follows.
  - Reference point A: This is the reference point for obtaining the key in the MFS key request I/F. The encryption key is generated by the key derivation function. Since MFS CF and encryption protocols are integrated, the risk of keys being intercepted is low without special measures, but it is difficult to change encryption protocols flexibly.
  - Reference point B: This is the reference point for obtaining the combined key at the MFS key request I/F. Provides combined keys for MFS CF. The encryption key is generated by the key derivation function. Since MFS CF and encryption protocols are integrated, the risk of keys being intercepted is low without special measures, but it is difficult to change encryption protocols flexibly.
  - Reference point C: This is the reference point for obtaining the encryption key at the MFS key supply service I/F. Encryption keys can be entered directly. Since the key is not generated using a key derivation

function, it is possible to provide a new random key directly as an encryption key. Since key management and cryptographic protocols are separated, it is possible to manage keys without relying on the physical environment. However, since the encryption key is entered directly, a mechanism is required to prevent the key from being intercepted. And since the encryption key can be intercepted, it is possible to decrypt and examine the key during communication. This reference point that separates key management from specific cryptographic protocols enables crypto-agility to use the latest security algorithms and provides session-based encrypted tunnels over the network.

*Reference points A and B has the same specifications.

5. Specific Examples of MFS Key Exchange and Management Usage

5.1. Object and Channel Encryptions

It is useful to make the conceptual distinction between object encryption and channel encryption to consider the implementation of MFS key exchange and management functional architecture. Note that the difference between object and channel encryption is a matter of perspective. The object encryption refers to encryption which apply to entire data objects. Users themselves encrypt and decrypt their own data objects. On the other hand, channel encryption provides an encrypted channel over which objects may be carried transparently, but the channel has no special knowledge about object boundaries. Users of encrypted communication pass their user data objects to the network connection service to use the encrypted communications service.

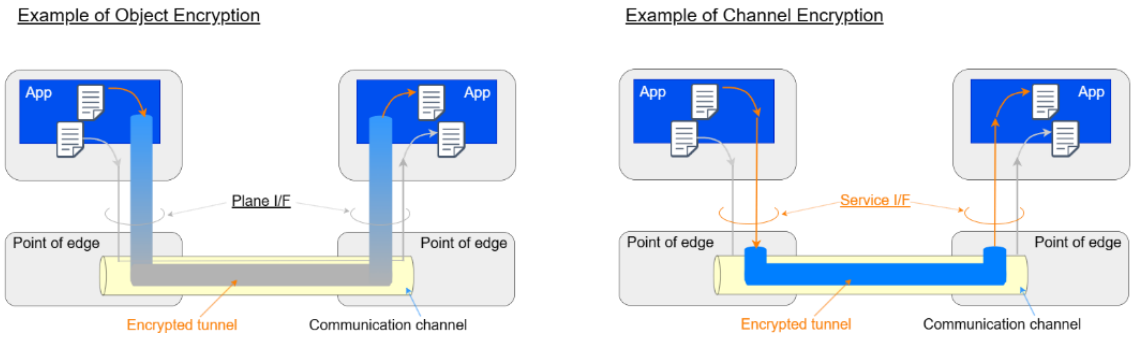

Figure 5.1-1: Object and channel encryption

Such encryption types need to be selected considering their features. Object encryption provides fine control functions such as encryption per dataset that is managed separately depending on the impact in the event of data leakage/tampering. On the other hand, channel encryption enables the simultaneous encryption function of multiple datasets. This function makes encryption processing simple if there is no need to differentiate control of each dataset depending on the impact of data leakage for each data. In addition, the increase in the number of encryption sessions managed by the security management function can be mitigated even when the number of endpoints increases.

5.2. Object and Channel Encryption for Zero Trust

Advanced security services using state-of-the-art technology make isolated and secure areas available for applications. Then, resources mapped into such isolated secure areas provided by the IOWN infrastructure should be securely transferred by encrypted data communication. Simply using a single encrypted communication to prevent leakage/tampering is not sufficient for zero trust security. When considering zero trust, comprehensively subdivide resources by the magnitude of attack and map these resources to multiple encrypted communications to manage based on a "need-to-know" policy.

5.3. Variation of MFS Key Exchange and Management Configuration

This section contains references related to MFS implementation patterns in the Open APN. The Open APN is a network that connects endpoints directly with optical paths. It provides high-speed, ultra-reliable, and low-latency connections. In today’s network, optical paths are disjointed and operated on a segment-by-segment basis. By contrast, the Open APN will enable one optical path to span across multiple segments. This will enable end-to-end communication with deterministic performance [Open APN Functional Architecture].

Section 5.1 describes the object and channel encryptions to which MFS applies. This section presents variations on implementing MFS when applied to an Open APN. MFS implementation patterns are classified into the following five patterns.

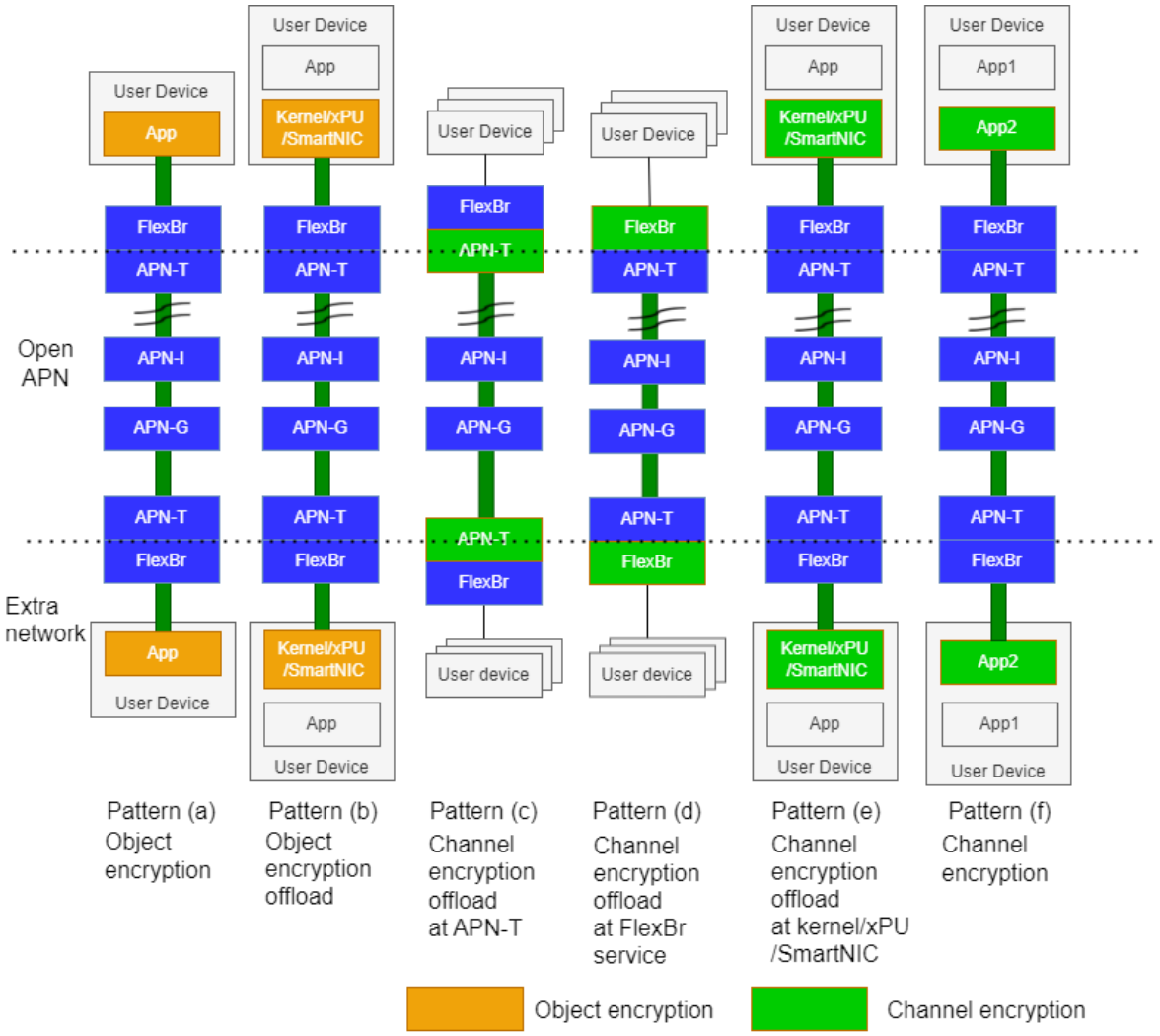

Figure 5.3-1: Implementation patterns of MFS

- Pattern (a) Object encryption
  In this case, the application and the encryption endpoint is same. The encryption endpoint recognizes the data to be encrypted and decrypted, and the encryption endpoint encrypts and decrypts the data. The application encrypts the communication to achieve end-to-end encryption. Because it uses end-to-end encryption, there is no need to consider the risk of third-party intervention other than the risk of eavesdropping on the hardware itself. Each application must have an encryption function.
- Pattern (b) Object encryption offload
  Applications offload cryptographic processing to the user device's kernel or hardware (xPU/SmartNIC). In this case, the encryption endpoint knows what to encrypt. The encryption process is offloaded and the application can continue processing without being affected by encryption. On the other hand, implementations that can offload cryptographic processing are required If the devices that perform offload processing are the same hardware, there is no need to consider the risk of a third party intervening, except for the risk that the hardware itself will be compromised. However, there may be risks from the platform operator providing the hardware.

- Pattern (c) Channel encryption offload at APN-T
  In channel encryption, the encryption point is transparent and does not know the object boundary. Application data is encrypted at the optical layer of an APN-T. Because it is encrypted at the physical layer, the risk of eavesdropping can be reduced. Applications do not need to consider cryptographic processing since it is handled transparently. Since an APN-T is a transceiver, it cannot modify the communication, so it can only encrypt the communication on a fiber or wavelength basis. Also, since the part from the application to an APN-T is not encrypted, additional security measures are required, such as placing it in a reliable location.
- Pattern (d) Channel encryption offload at FlexBr (Flexible Bridge) service
  Application data is encrypted at the aggregation layer of FlexBr. This is useful if the user device cannot implement encryption or if the user device does not have the capacity or capability for encryption. Applications do not need to consider cryptographic processing since it is performed between FlexBr. However, since the part from the application to FlexBr is not encrypted, additional security measures are required, such as placing it in a reliable location.
- Pattern (e) Channel encryption offload at kernel/xPU/SmartNIC
  Application data encryption processing is offloaded to the user device's kernel or hardware (kernel/xPU/SmartNIC). Since the application handles it transparently, there is no need to consider cryptographic processing. Encryption is processed at a lower layer than the application. It can be implemented on a variety of devices, and encryption can be performed at a fine granularity. If the devices that perform offload processing are the same hardware, there is no need to consider the risk of a third-party intervening, except for the risk that the hardware itself will be eavesdropped.
- Pattern (f) Channel encryption
  App1 communicates between user devices App2 encrypts the communication to achieve end-to-end communication. Since the application handles it transparently, there is no need to consider cryptographic processing. Because it uses end-to-end encryption, there is no need to consider the risk of third-party intervention other than the risk of eavesdropping on the hardware itself. Each application must have an encryption function.
  *The difference from Pattern (b) is that the App1 is not involved in cryptographic processing.

The above is summarized in the table below:

Table 5.3-1: implementation pattern of MFS

		PATTERN (A)	PATTERN (B)	PATTERN (C)	PATTERN (D)	PATTERN (E)	PATTERN (F)
System deployer perspective	Encryption type	object	object	channel	channel	channel	channel
System deployer perspective	Encryption point	application	offload	APN-T	FlexBr	offload	application
System deployer perspective	Encryption unit	application	application	fiber or wavelength	aggregation unit	aggregation unit	application

System deployer perspective	APN	**not required**	**not required**	required	required	**not required**	**not required**
Application deployer perspective	Processing load on Application	high	**low**	**low**	**low**	**low**	high
Application deployer perspective	Modification of application	required	required	**not required**	**not required**	**not required**	**not required**

The security level and performance of these configurations depend on the implementation, but the following selection patterns can generally be considered:

- If handling cryptographic processing on the user side without relying on telecommunications carriers or platform providers: (a) or (f)
  *(f) is when cryptographic processing needs to be separated from the application.
- When cryptographic processing is performed on the user side as described above, if performance also needs to be considered (e.g. when high traffic volume is expected): (b) or (e) (utilizing hardware accelerator, etc.)
  *(e) is when offloading cryptographic processes needs to be separated from the application.
- If leaving cryptographic processing to telecom carriers or platform providers, or if for some reason the user devices cannot allocate sufficient computing resources to cryptographic processing: (c) or (d) (These patterns also apply when telecom carriers protect their own communication path.)

5.4. Specific Reference Examples for MFS Key Exchange and Management Functional Architecture

This subsection describes examples that map the MFS key exchange and management functional architecture in section 4 to specific hardware based on the configuration example described in section 5.3.

Pattern(a) Object encryption between smart phone and server

Communication between a smart phone and a server. The communication endpoints are the application processes of a customer premise device and a server. Cryptographic communication using MFS key exchange and management between an Application on the Smart phone and an Application on Main CPU of server are performed as following.

- MFS key exchange and management using a PSK based key exchange method and a PQC based key exchange method are performed,
- The obtained two keys are combined to generate an encryption/decryption key and
- Data is encrypted with proprietary cryptographic function of application using the key which is supplied from MFS CF.

On the smartphone, the subscriber identity module (SIM) is used to store the PSK securely. On the server, the hardware security module (HSM) is used to protect the PSK.

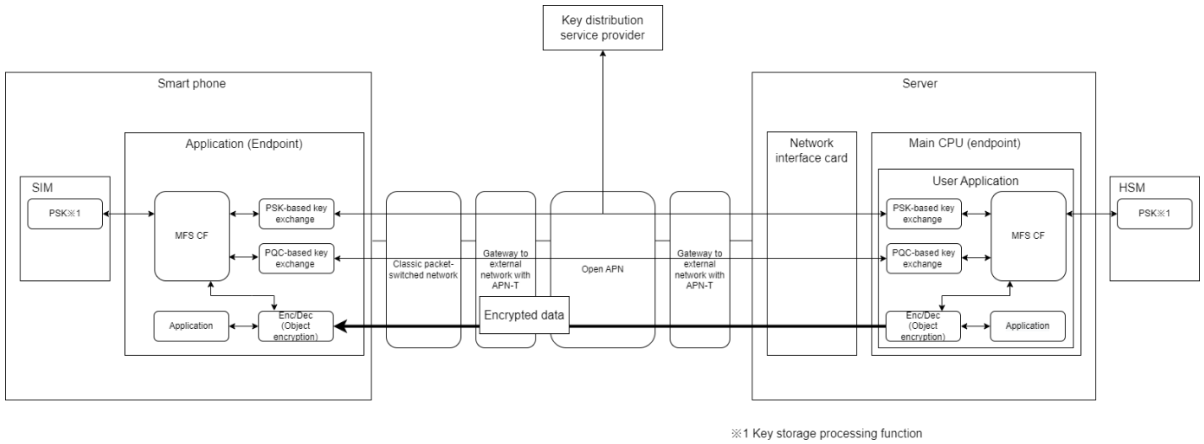

Figure 5.4-1: Specific example of object encryption with MFS key exchange and management using PSK and PQC between smart phone and server

Pattern(b) Offloaded object encryption between servers

The communication endpoints of this example are application processes on each server's Main CPU. Applications on each Main CPU offload object encryption processing including MFS key exchange and management to the Network interface card (e.g., DPU/IPU/Smart NIC) which is the near-endpoint. Cryptographic communication using MFS key exchange and management between the Network interface card are performed as following.

- MFS key exchange and management using two different PQC based key exchange methods are performed,
- The obtained two keys are combined to generate an encryption/decryption key and
- Data is encrypted with proprietary cryptographic application using the key which is supplied from MFS CF.

Although described as an example of using TEE to the protect execution environment of the application process on the Main CPU, measures for protecting against malicious acts on the Main CPU and the Network interface card (i.e., Protection of data in use) is outside of the scope of this document. In other words, the actual security level of this example system depends on the individual implementation. Data protection between the endpoint and the Network interface card should also be considered separately.

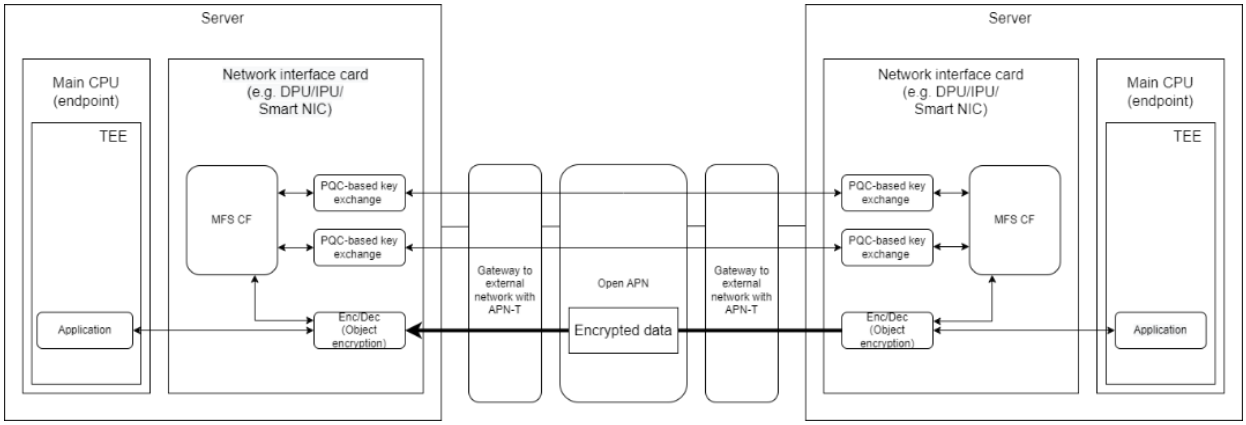

Figure 5.4-2: Specific example of offloaded object encryption with MFS key exchange and management using a hybrid of two PQCs between servers

Pattern(c) Channel encryption between servers

The communication endpoints of this example are application processes on each server's Main CPU. APN-T embedded into the Network interface card (e.g., DPU/IPU/Smart NIC) provides channel encryption processing. Cryptographic communication using MFS key exchange and management between the Network interface card are performed as following.

- MFS key exchange and management using a QKD based key exchange method and a PQC based key exchange method are performed,
- The obtained two keys are combined to generate an encryption/decryption key and
- Data is encrypted with channel encryption such as OTNsec at APN-T using the key which is supplied from MFS CF.

The QKD network allows sharing keys between two distinct servers by provisioning it from QKD nodes, in which the key relay can be realized through key management agent (KMA) and the key at the QKD end node is provided to the application server by key supply agent (KSA) in key management layer. [ITU-T Y.3802]

Figure 5.4-3 shows the case that obtained two keys are combined and the resulting key is used to encrypt data with channel encryption such as OTNsec at APN-T. As another case (not shown in the figure), for example, data is first encrypted with channel encryption such as IPsec using PQC-based key as outer encryption/decryption, and then the encrypted data is further encrypted with channel encryption such as OTNsec using QKD-based key as inner encryption/decryption.

Although described as an example of using TEE to protect execution environment of application on the Main CPU, measures for protecting against malicious acts on the Main CPU and the Network interface card (i.e., Protection of data in use) are outside of the scope of this document. Data protection between the endpoint and the APN-T should also be considered separately. QKD nodes are treated as trusted nodes, and it is also possible to protect the link between the endpoint and the QKD node using PQC or PSK-based pairing.

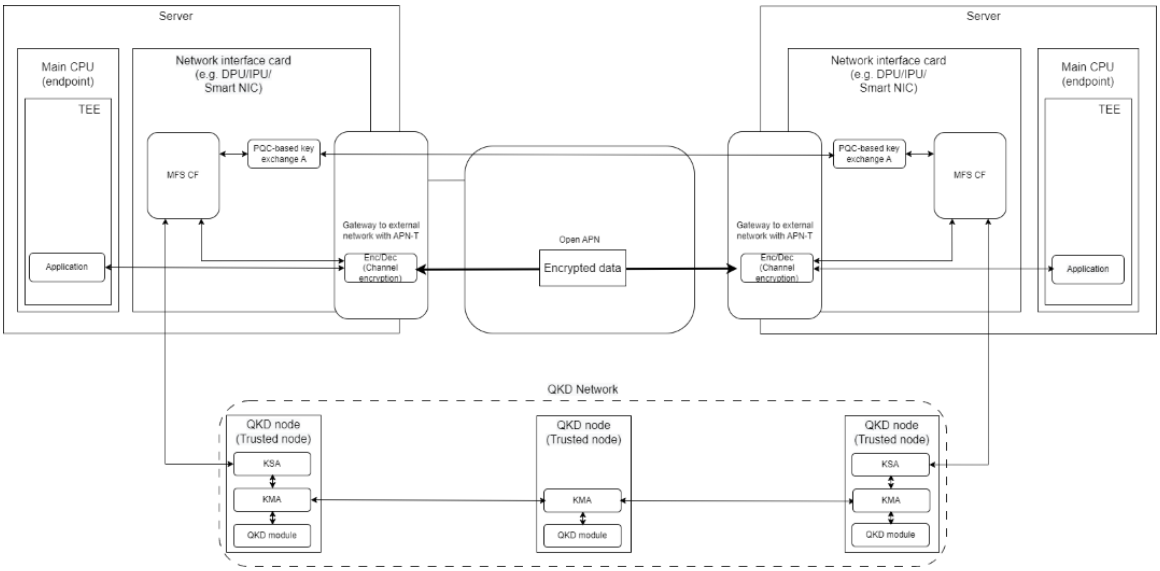

Figure 5.4-3: Specific example of channel encryption with MFS key exchange and management using a hybrid of QKD and PQC between servers

Pattern(d) Channel encryption offload at FlexBr between servers

The communication endpoints of this example are application processes on each server's Main CPU. Applications on each Main CPU offload object encryption processing including MFS key exchange and management to FlexBr which is the near-endpoint. Cryptographic communication using MFS key exchange and management between FlexBrs are performed as following.

- MFS key exchange and management using two different PQC based key exchange methods are performed,
- The obtained two keys are combined to generate an encryption/decryption key and
- Data is encrypted with channel encryption such as IPsec and MACsec using the key which is supplied from MFS CF.

Although described as an example of using TEE to protect execution environment of application process on the Main CPU, measures for protecting against malicious acts on the Main CPU and the Network interface card (i.e., Protection of data in use) is outside of the scope of this document. In other words, the actual security level of this example system depends on the individual implementation. Data protection between the endpoint and the Network interface card should also be considered separately.

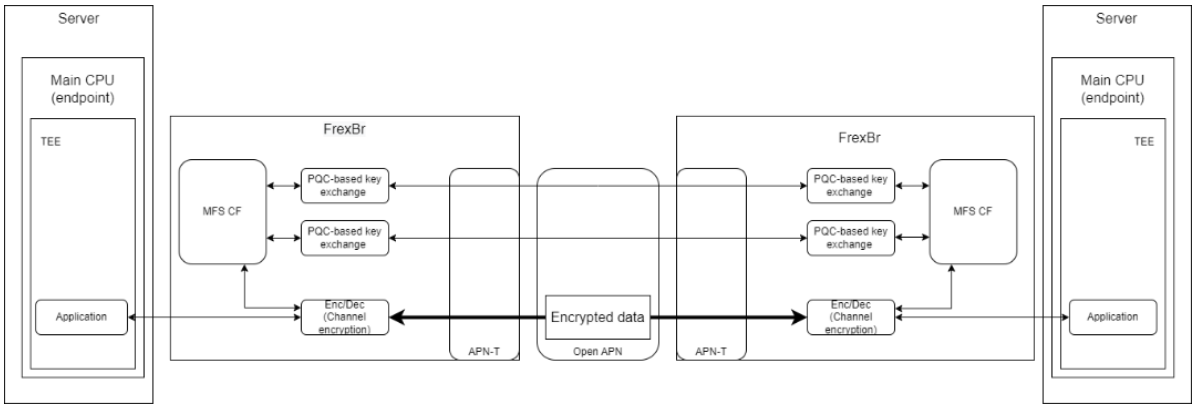

Figure 5.4-4: Specific example of offloaded object encryption with MFS key exchange and management using a hybrid of two PQCs between servers

Pattern(e) Channel encryption offload at SmartNIC between servers

The communication endpoints of this example are application processes on each server's Main CPU.Applications are processed on each Main CPU and encryptions are offloaded to Network interface card (e.g., DPU/IPU/Smart NIC) which is the near-endpoint. Cryptographic communication using MFS key exchange and management between Network interface cards are performed as following.

- MFS key exchange and management using two different PQC based key exchange methods are performed,
- The obtained two keys are combined to generate an encryption/decryption key and
- Data is encrypted with channel encryption such as IPsec and MACsec using the key which is supplied from MFS CF.

Although described as an example of using TEE to protect execution environment of application process on the Main CPU, measures for protecting against malicious acts on the Main CPU and the Network interface card (i.e., Protection of data in use) is outside of the scope of this document. In other words, the actual security level of this example system depends on implementation. Data protection between the endpoint and the Network interface card should also be considered separately.

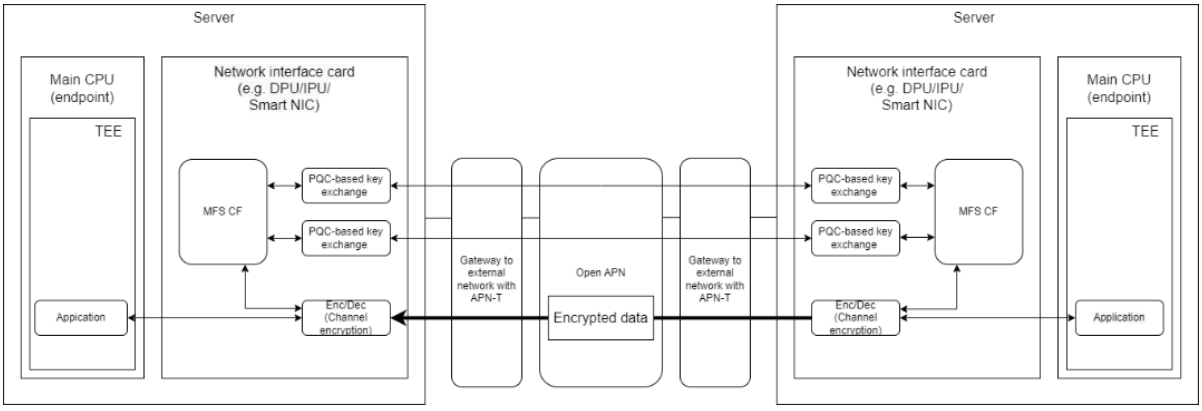

Figure 5.4-5: Specific example of channel encryption offload with MFS key exchange and management using a hybrid of two PQCs between servers

Pattern(f) Channel encryption between VMs on servers

The communication endpoints of this example are application processes on each server's Main CPU. Applications on each Main CPU use channel encryption provided by computing platform including MFS key exchange and management

which is the near-endpoint. Cryptographic communication using MFS key exchange and management between VMs are performed as following.

- MFS key exchange and management using two different PQC based key exchange methods are performed on VM 1,
- The obtained two keys are combined to generate an encryption/decryption key and
- Data is encrypted with channel encryption such as IPsec and MACsec using the key which is supplied from MFS CF on VM 2.

Measures for protecting against malicious acts on the Main CPU (i.e., Protection of data in use) are outside of the scope of this document. In other words, the actual security level of this example system depends on the specific implementation.

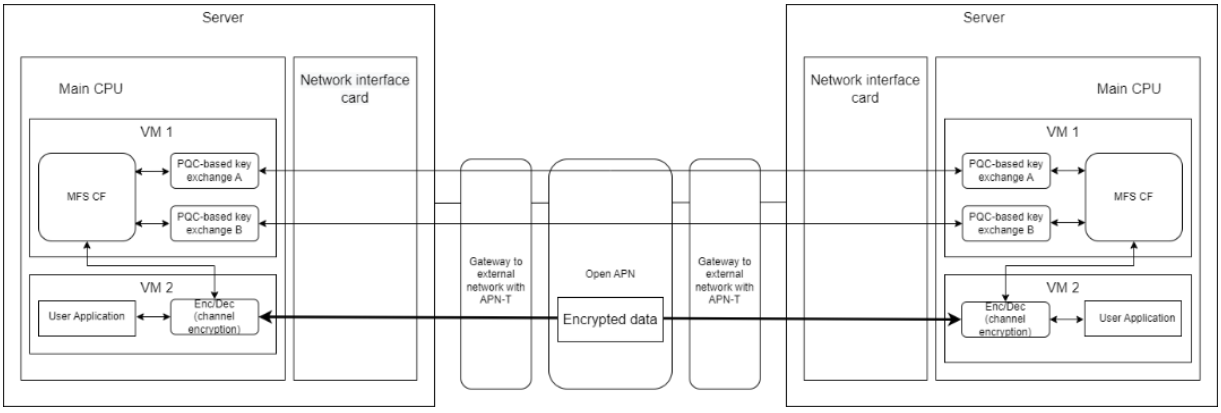

Figure 5.4-6: Specific example of channel encryption with MFS key exchange and management using a hybrid of two PQCs between servers

6. Conclusion

This document provided the MFS key exchange and management functional architecture required to achieve post-quantum cryptographic communication with high crypto-agility and specific examples of cryptographic communication using that architecture based on “Technology outlook of information security” [IOWN GF SEC OUTLOOK]. By following this architecture, secure and flexible cryptographic communication between IOWN endpoints in the era of IOWN infrastructure can be achieved.

References

[Store now decrypt later]: National Institute of Standards and Technology (NIST) Special publication 1800-38B Migration to Post-Quantum Cryptography Quantum Readiness: Cryptographic Discovery

[IOWN GF SEC OUTLOOK]: IOWN Global Forum, “Technology Outlook of Information Security”, February 2023, https://iowngf.org/wp-content/uploads/formidable/21/IOWN-GF-RD-SEC-Technology_outlook_of_Information_Security.pdf

[ITU-T Y.3811]: ITU-T Recommendation Y.3811, “Quantum key distribution networks – Functional architecture for quality of service assurance Amendment 1”, November 2023, https://www.itu.int/rec/T-REC-Y.3811

[NIST Post-Quantum Cryptography Standardization]: National Institute of Standards and Technology (NIST), Post-Quantum Cryptography Standardization, May 2024, https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization

[NIST SP 800-133 Rev.2]: National Institute of Standards and Technology (NIST), NIST Special Publication 800-133 Revision 2, Recommendation for Cryptographic Key Generation, June 2020, https://csrc.nist.gov/publications/detail/sp/800-133/rev-2/final

[Open APN Functional Architecture]: IOWN Global Forum, “Open All-Photonic Network Functional Architecture”, October 2023, https://iowngf.org/wp-content/uploads/formidable/21/IOWN-GF-RD-Open_APN_Functional_Architecture-2.0.pdf

Abbreviations

AES Advanced Encryption Standard

APN All-Photonic Network

APN-C Open APN Controller

APN-I Open APN Interchange

APN-G Open APN Gateway

APN-T Open APN Transceiver

CPU Central Processing Unit

Dec Decryption

DPU Data Processing Unit

Enc Encryption

FlexBr Flexible Bridge

HSM Hardware Security Module

IPU Infrastructure Processing Unit

KDF Key derivation Function

MFS Multi-Factor Security

MFS CF Multi-Factor Security Control Function

RIM Reference Implementation Model

OS Operating system

PoC Proof of Concept

PQC Post-Quantum Cryptography

PQC KEM Post-quantum cryptography key encapsulation mechanism

PSK Pre-Shared Key

QKD Quantum Key Distribution

QKDN Quantum Key Distribution Network

SIM Subscriber identity Module

Smart NIC Smart Network Interface Card

TEE Trusted Execution Environment

vNIC virtual NIC (Network Interface Card)

VM Virtual Machine

History

Revision	Release Date	Summary of Changes
1.0	October 24, 2024	Initial Release