


Article

Secure Encapsulation Schemes Using Key Recovery System in IoMT Environments

Tae-hoon Kim ¹, Wonbin Kim ¹, Daehee Seo ² and Imyeong Lee ^{1,*}¹ Department of Software Convergence, Soonchunhyang University, Asan 31538, Korea; 20134101@sch.ac.kr (T.K.); wbkim29@sch.ac.kr (W.K.)² Faculty of Artificial Intelligence and Data Engineering, Sangmyung University, Seoul 03016, Korea; daehseo@smu.ac.kr

* Correspondence: imylee@sch.ac.kr

Abstract: Recently, as Internet of Things systems have been introduced to facilitate diagnosis and treatment in healthcare and medical environments, there are many issues concerning threats to these systems' security. For instance, if a key used for encryption is lost or corrupted, then ciphertexts produced with this key cannot be decrypted any more. Hence, this paper presents two schemes for key recovery systems that can recover the lost or the corrupted keys of an Internet of Medical Things. In our proposal, when the key used for the ciphertext is needed, this key is obtained from a Key Recovery Field present in the ciphertext. Thus, the recovered key will allow decrypting the ciphertext. However, there are threats to this proposal, including the case of the Key Recovery Field being forged or altered by a malicious user and the possibility of collusion among participating entities (Medical Institution, Key Recovery Auditor, and Key Recovery Center) which can interpret the Key Recovery Field and abuse their authority to gain access to the data. To prevent these threats, two schemes are proposed. The first one enhances the security of a multi-agent key recovery system by providing the Key Recovery Field with efficient integrity and non-repudiation functions, and the second one provides a proxy re-encryption function resistant to collusion attacks against the key recovery system.



Citation: Kim, T.; Kim, W.; Seo, D.; Lee, I. Secure Encapsulation Schemes Using Key Recovery System in IoMT Environments. *Sensors* **2021**, *21*, 3474. <https://doi.org/10.3390/s21103474>

Academic Editor: Damianos Gavalas

Received: 15 March 2021

Accepted: 13 May 2021

Published: 17 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: CL-PKC; key encapsulation; key recovery system; proxy re-encryption; signcryption

1. Introduction

In the era of the Fourth Industrial Revolution, as various countries and companies around the world have heavily invested in Information Technology (IT), the emergence of Internet of Things (IoT) environments has increasingly enabled a convenient and broad diversity of services to be distributed to consumers via various types of smart devices. There are various systems such as the Internet of Medical Things (IoMT), Intelligent Transportation Systems (ITS), smart home appliances, and connected cars that have been implemented on those smart devices and deploy a vast number of services to consumers [1,2]. Therefore, many current types of research have applied those IoT technologies to various environments.

Although the development of IoT has increased device convenience, it has also been accompanied by increasing threats to national, corporate, and personal information security [3]. According to the security threats, such as personal information leakage cases, encryption has rapidly become important to secure personal information [4]. Therefore, the importance of security issues in IoT environments has also increased. Furthermore, there is discussion regarding security issues related to key management, in which problems may arise where ciphertexts cannot be decrypted if the keys are lost or corrupted.

In general, key recovery is a system that provides the ability to reveal the key to an authorized user under specific conditions specified in advance [5]. This paper presents schemes to recover lost or corrupted keys using an encapsulation-based key recovery

system. When a user needs a key that was used to create a ciphertext, a newly defined field known as Key Recovery Field (KRF) can be used to recover the key. If key is lost or corrupted, the recovered key can be used to decrypt the ciphertext. Because security is necessary for key management and recovery in various environments using IoT, there is much research on key recovery systems for use with IoT. Guo et al. [6] proposed a secure group key distribution scheme for untrusted wireless networks. Guo et al. used the Self-healing Group Key Distribution (SGKD) protocol to ensure group communication security and improve communication efficiency. Instead of requiring the group manager to resend the missing key to update the message, Guo et al. proposed a scheme for group members to recover the lost session key from the current broadcast message.

Lee et al. [7] proposed an efficient and secure key distribution and key recovery mechanism suitable for the characteristics of the IoT environment. The proposed system added the key recovery function required to prevent the reverse function of the encryption and key recovery, providing security to both due to the communication device could not unilaterally recover the key. In addition, it is efficient because there is little information sent during key recovery, which is suitable for IoT environments.

Sung [8] proposed a scheme to support secure sensor data for cloud computing to activate services at the IoT application level. Sung proposed key management that enables continuous key authentication for the privacy of sensing information in such a cloud computing environment and enables secure recovery if the key is lost or corrupted.

Losing a key in an IoMT environment will prevent access to information such as previous medical treatment data and information on medications being taken, and impede accurate medical examination and treatment. Therefore, key management is important in IoMT environments [9]. There are four agents in a key recovery system in IoMT. The key Generation Center (KGC) can generate some parameters of a network participant's public key pair. The Medical Institutions (Med) have all the medical treatment data, The key Recovery Center (KRC) can recover the complete key. The key Recovery Agent or Key Recovery Auditor (KRA) can share the KRC's key recovery operations or monitor other agencies. If the patient loses the key used for the ciphertext, the key can be recovered with the help of remains the KRF, Med, KRC, and KRA.

However, the problem remains that the KRF may be forged or altered by a malicious user. To solve this problem, we propose our first scheme, which efficiently provides integrity and non-repudiation functions for the KRF and enhances the security of a multi-agents key recovery system.

The main contributions of our proposed scheme-I as follows:

- It provided a key recovery system based on secure encapsulation against various types of attacks and provides the ability to securely recover a lost or corrupted key.
- It uses signcryption to ensure KRF integrity and non-repudiation. In addition, it provides both digital signing and encryption at the same time to increase computational efficiency.
- It uses values that only authorized KRAs hold to prevent unauthorized KRAs and group-based authentication attacks. If some KRAs do not perform the key recovery properly, key recovery may be performed by other authenticated KRAs to prevent a single point of failure.

Furthermore, the Med, KRA, and KRC may collude and behave maliciously. To solve this problem, we propose scheme 2, which provides a proxy re-encryption function and enhances the security of a key recovery system against various types of attacks such as collusion attacks and the key escrow problem.

The main contributions of our proposed scheme-II as follows:

- It prevents the Med, KRC, and KRA from behaving maliciously to recover keys without authorization and prevents unauthorized entities from obtaining keys.
- It uses a partial private key generation scheme to prevent the KGC from generating private keys for all participants.

The remaining parts of the paper are organized as follows. Section 2 describes related work, and Section 3 describes system model for the proposed schemes. Section 4 describes scenarios and detailed protocols for the proposed scheme-I, and Section 5 describes scenarios and detailed protocols for the proposed scheme-II. Section 6 analyzes whether the proposed schemes satisfy the security requirements. Finally, Section 7 discusses our conclusions.

2. Related Work

This section reviews and discusses existing works related to key recovery systems and encryption schemes.

2.1. Encapsulation Key Recovery Systems

A key recovery system is an important part of an encryption system. If a private key or session key used for a ciphertext is lost or corrupted, or a Law Enforcement Agency (LEA) wishes to intercept suspicious ciphertexts lawfully, it must be possible to recover the key. There have been several proposals related to such key recovery systems. Kanyamee et al. [10] proposed a highly available distributed session key recovery system. It provides high availability and attack detection for secure session key management and group authentication while using Multi-Key Recovery Agents (M-KRA) to solve the single point of failure problem encountered in the traditional KRA approach. However, many problems remain, such as the risks of forgery, counterfeiting, and collusion attacks for user-generated KRFs, which can cause problems for the key recovery service.

Lim et al. [11] proposed an encapsulation-based M-KRA key recovery system. They attempted to solve the problem that the M-KRA must communicate directly with one or more KRAs in existing M-KRA scheme, and the user must directly perform a complex key recovery process. Their scheme provides secure session key management and recovery using a new type of M-KRA to solve this problem. However, problems may arise in the key recovery service the forgery or modification of KRFs and non-repudiation problems related to user-generated KRFs.

Kyusuk et al. [12] proposed an identity-based key escrow scheme to prevent malicious key use by LEAs. If an LEA maliciously obtains the key, it can read the encrypted data to the desired user. In other words, an LEA can intercept and obtain the users' keys to read all encrypted data. To solve this problem, the scheme prevents LEAs from obtaining a key by themselves after generating a user's key pair with the KGC generated master key and the user's ID. However, since it is a single KRA, it is vulnerable to problems such as a single point of failure weakness and group authentication attacks, causing problems with the key recovery service.

Huadpaknam [13] proposed the Security Key Recovery System with Channel Quality Awareness (SKRS-CQA) for smart grid applications. If a Smart Meter Unit (SMU) loses the keys used for correcting to the smart grid, it needs to be recovered. To solve this problem, key recovery proposed, providing improved reliability, system availability, and data confidentiality. In addition, system reliability was improved by using amplification and forwarding relay protocols and a cooperative communication network with optimal power allocation.

2.2. Multi-Agent Key Recovery

A single agent key recovery system is associated with service overload and security problems. Therefore, we use a multi-agent (at least two agents) key recovery system. The multi-agents receive a ciphertext that contains a key from the user or the KRC. Later, KRAs send pieces of the key to the KRC to allow the KRC to recover the complete key. However, various attacks and security breaches are possible, and efforts have been made to deal with these issues [14]. In our key recovery system using signcryption, we security by increasing availability and enhance security.

2.3. Signcryption

Encryption and digital signatures are two encryption tools that can ensure confidentiality, integrity, and non-repudiation. Until 1997, cryptographic systems used separate components to provide these security functions. In public key schemes, the traditional scheme is to digitally sign the message and then perform encryption (signature-then-encryption). However, there are two problems: the operation efficiency is low and the cost is high. To solve this signcryption was proposed. In 1997 Zheng [15] proposed the first signcryption scheme. Signcryption simultaneously performs digital signature and encryption. Signcryption compared to the traditional signature-then-encryption scheme, can effectively improve computational efficiency, by reducing computational cost and communication overhead. In addition, many other signcryption schemes have been proposed throughout the years, each of them having its problems and limitations while offering different levels of security and computational cost [16,17].

2.4. Secret Sharing

Secret sharing schemes are ideal for sensitive information. These pieces of information should be kept highly confidential, as their exposure could be disastrous. However, it is also critical that they should not be lost. Traditional encryption schemes are not suitable for achieving a high level of confidentiality and stability at the same time. When storing encryption keys, the user has to choose between keeping a single copy of the key in one location or multiple copies of the key in multiple locations for maximum security. The secret sharing scheme proposed by Shamir and Blakley [18,19] in 1979 is a scheme of dividing the secret value into several pieces so that the secret value can be recovered only when more than a certain number of pieces are collected. Such a scheme is called Shamir's (k, n) threshold scheme. This scheme divides the secret value into n pieces and entities may recover the secret value only when more than k pieces are collected. In another type of secret sharing scheme, there is one dealer and n players. The dealer gives a share of the secret to the players, but only when specific conditions are fulfilled will the players reconstruct the secret from their shares. The dealer accomplishes this by giving each player a share so that any group of t (for threshold) or more players can together reconstruct the secret but no group of fewer than t players can. In addition, many other secret sharing schemes have been proposed throughout the years with as in the case of signcryption, each of them having its problems and limitations while offering different levels of security and computational costs [20,21].

2.5. Proxy Re-Encryption

A Proxy Re-Encryption (PRE) scheme is a scheme that converts the ciphertext so that a proxy server can decrypt the ciphertext encrypted with user A's public key using user B's private key. In 1998, Blaze et al. [22] proposed the first two-way proxy re-encryption scheme. This scheme was designed using the ElGamal encryption scheme [23]. In 2007, Green et al. [24] proposed an ID-based proxy re-encryption scheme using ID-based encryption for the first time to solve the certificate management problem of the existing Public Key Infrastructure (PKI) based proxy re-encryption. ID-based encryption is a scheme of using the user's identity as a public key [25]. In this scheme, the user's identity itself is owned, so unlike in PKI-based environments there is no need to issue and manage certificates. In addition, since the KGC generates a private key corresponding to the identities and issues them to the users, it has the advantage of performing verification of the user through KGC in case of a dispute. However, the KGC issues all users' private keys, which causes a key escrow problem in which KGC knows the private keys. Therefore, to solve this problem, a Certificateless Public Key Cryptography (CL-PKC) system was developed. The CL-PKC scheme was proposed by Al-Riyami et al. [26], and it solves the key escrow problem by issuing partial private keys to the users by combining the user's identity and a random number. Building on these features, in 2010, Sur et al. [27] proposed Certificateless Proxy Re-Encryption (CL-PRE) using CL-PKC. CL-PRE is currently a representative form of

secure PRE because it can perform the purpose of proxy re-encryption without suffering the PKI certificate management problem or IBE key escrow problem [28–30].

3. System Model

This section describes the system models, system objects, and security requirements of the proposed schemes.

3.1. Common Proposed Key Recovery System Model

In this section, we present the two key recovery system models proposed in this study. Before describing each proposed model, we present the common elements of the proposed models.

3.1.1. Common Design Goals of Proposed Schemes

The two key recovery system models presented in this research were designed in different forms. However, the basic goal of both models is encapsulated key recovery. The first model proposed in this study is a key recovery system using signcryption. This process involves recovering the session key used for communication by using the encapsulated key recovery field. The second model proposed in this study is a key recovery system using proxy re-encryption. The basic goal is the same as the first model described above. However, the design and additional goals of the two models differ from each other. The similarities and differences between the two models can be seen in Figure 1, which will be described in detail below.

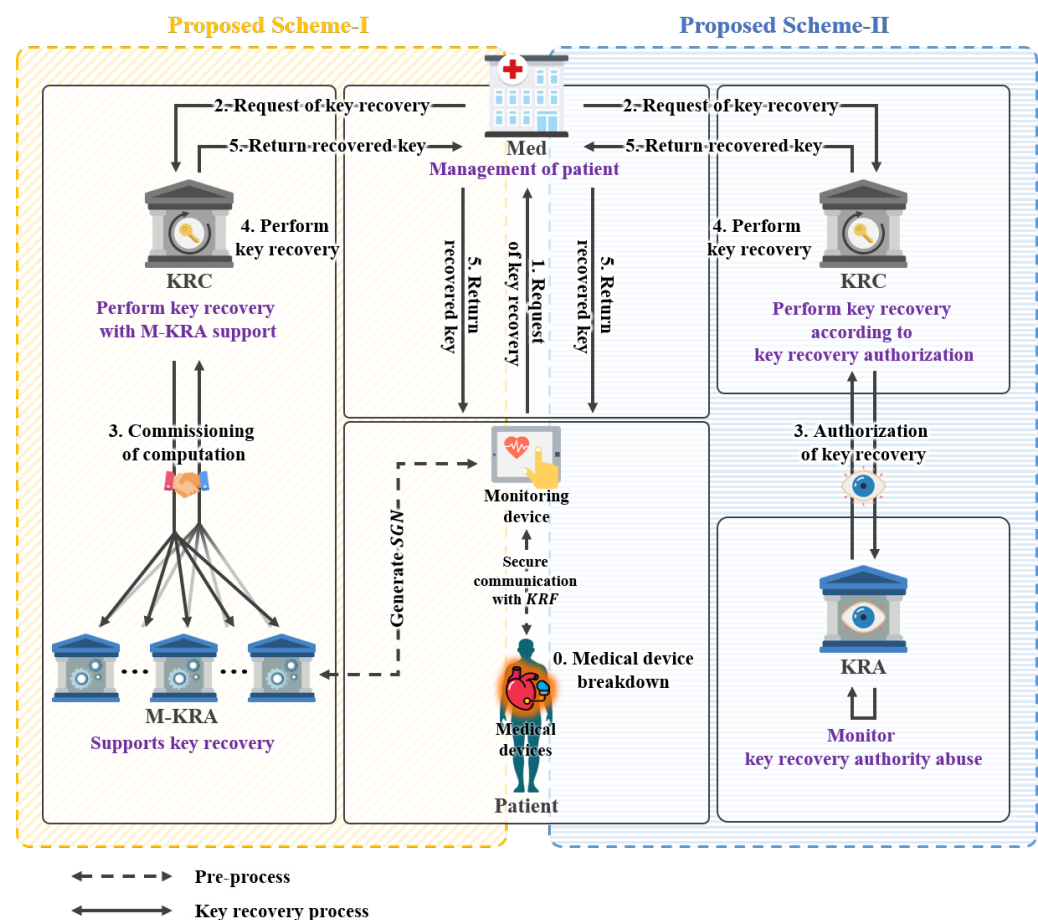


Figure 1. Summary and comparison of proposed key recovery system model.

3.1.2. Common Objects of Proposed Schemes

The composition of the two system models proposed in this study can be seen in Figure 1. In Figure 1, the difference between M-KRA and KRA methods is shown for the types of participants in the two models. The remaining differences are detailed in each model's respective section.

- **Key Generation Center (KGC):** Every participant *Part* must perform the KCG and key generation and communication steps to generate keys. All *Part* can generate a private key through the private key generation step with KGC, and a public key corresponding to the private key can be generated. The KGC publishes the public parameter *params* for performing encrypted communication with *Part*.
- **Devices (*Dev*):** *Dev* are medical devices and monitoring devices. Devices perform communication in the system managed by the *Med*. In this model, *Devs* must perform communication in the format designated by *Med*, and the basic format follows the form of $(C \parallel KRF)$, in which the ciphertext and KRF are concatenated. Devices participating in the communication need *Med*'s public parameters in order to make the session key used for message encryption into KRF. Furthermore, the generated KRF should be designed to only be controlled by KRC and KRA.
- **Medical Institution (*Med*):** *Med* is a medical institution that manages device authorization control and data on medical devices. When a device requests KRF key recovery, the *Med* verifies that it is the lawful owner of the KRF. In this paper, the step of confirming whether the KRF is a lawful owner is omitted. In addition, the *Med* sends the KRF to KRC to help recover the key.

3.2. Proposed Scheme-I (Key Recovery System Using Signcryption)

This section describes additional elements of the key recovery system model using signcryption, excluding the common elements of the two models proposed in Section 3.1.

3.2.1. Design Goals of Proposed Scheme-I

The model of the key recovery system using signcryption is a key recovery system that is used when a device key is lost or corrupted as shown in Figure 2. The device requests key recovery from Med and sends KRF. The Med receiving the KRF verifies that it is a lawful device of KRF. If it is a lawful device, it requests KRC to recover the key and sends KRF. After receiving KRF, KRA decrypts the KRF and sends the obtained KRF pieces to the M-KRA. Then, after receiving the pieces of KRF, M-KRA decrypts them and sends the session key pieces to KRC. It collects the session key pieces, generates a complete session key, and sends it to the device.

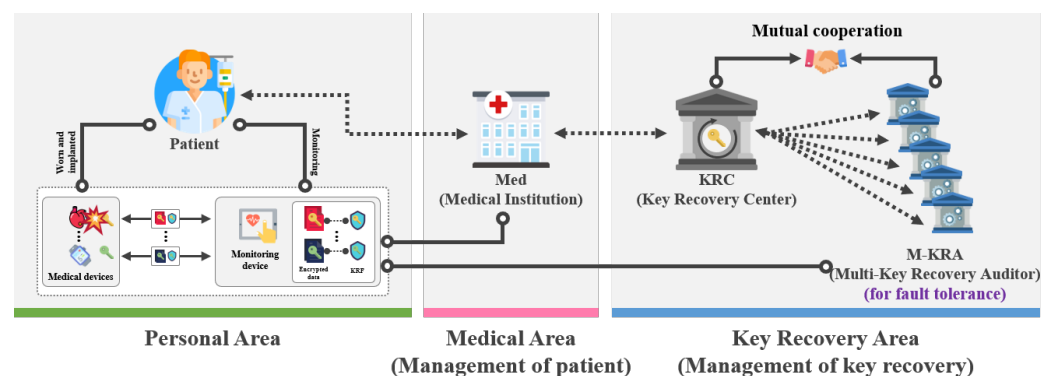


Figure 2. Proposed scheme-I.

3.2.2. Objects of Proposed Scheme-I

The system objects of the key recovery system using signcryption is shown in Figure 2. In addition, M-KRA additionally exists, and its roles are as follows:

- Participants: *Part* represents all participants (*Dev*, *Med*, *KRC*, *M-KRA*) who use the encrypted communication provided by *KGC*. *Part* can perform encrypted communication only by using *params* provided by *KGC*.
- Key Recovery Center (*KRC*): *KRC* is an organization in charge of key recovery and plays a central role in key recovery. The key recovery process is performed according to *Med*'s request for key recovery, and *KRF* is converted into a form that can be recovered using *KRC*'s private key. In this model, to reduce the burden of *KRC*'s key recovery operation, the help of *M-KRA* is needed.
- Multi-Key Recovery Agents (*M-KRA*): *M-KRA* is the agent that helps some operations of key recovery by reducing the burden on *KRC*. The *KRA* included in the *M-KRA* determines whether the *KRF* is suitable for recovery to prevent abuse of the *KRF*'s authority. When receiving a key recovery request from *KRC*, *M-KRA* perform the *KRF* recovery process using their private key. Furthermore, *M-KRA* send the obtained session key pieces to *KRC*.

3.2.3. Security Requirements of Proposed Scheme-I

The security requirements of the key recovery system using signcryption are as follows:

- *KRF* integrity: No participant in key recovery can maliciously transform *KRF* information from the device and *KRF* information required for key recovery cannot be changed.
- Data confidentiality: It should be possible for only authorized devices to decrypt encrypted data.
- Non-repudiation: The device should not be able to reject the fact that it generated the *KRF*. In addition, the fact that device-generated *KRF* should be clear after transmission, exchange, communication, and processing.
- Attack on group authentication detection: If a malicious third-party *KRA* pretends to be a lawful member of the key recovery group, *KRA* should be detected through group verification.
- Single point of failure protection: In *M-KRA*, some *KRAs* should be able to recover session keys even if another *KRA* fails to operate properly.

3.3. Proposed Scheme-II (Key Recovery System Using Proxy Re-Encryption)

This section describes additional elements of the key recovery system model using proxy re-encryption, excluding the common elements of the two models proposed in Section 3.1.

3.3.1. Design Goals of Proposed Scheme-II

The model of the key recovery system using proxy re-encryption is a key recovery system that is used when a device key is lost or corrupted as shown in Figure 3. The device requests key recovery from *Med* and sends *KRF*. The *Med* receiving the *KRF* verifies that it is a lawful device of *KRF*. If it is a lawful device, it generates a re-encryption key. Then, it requests key recovery from *KRC* and sends the obtained *KRF* and the re-encryption key. After receiving the *KRF* and re-encryption key, the *KRA* partially calculates *KRF* and sends the partially calculated *KRF* to *KRA*. After receiving the partial calculated *KRF*, *KRA* performs some calculations and sends partial calculated *KRF* to *KRC*. After receiving *KRF*, *KRC* sends it to the *Med*. The *Med* decrypts it, generates a session key, and sends it to the device.

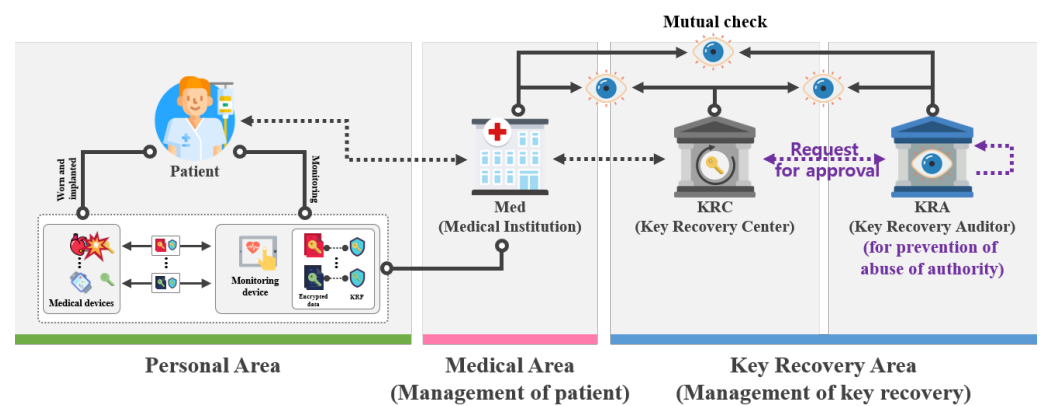


Figure 3. Proposed scheme-II.

3.3.2. Objects of Proposed Scheme-II

The system objects of the key recovery system using proxy re-encryption is shown in Figure 3.

- **Participants (*Part*):** *Part* represents all participants (*Dev*, *Med*, *KRC*, *KRA*) who use the encrypted communication provided by *KGC*. *Part* can perform encrypted communication only by using *params* provided by *KGC*.
- **Key Recovery Center (*KRC*):** *KRC* is an organization in charge of key recovery and plays a central role in key recovery. The key recovery process is performed according to *Med*'s request for key recovery, and *KRF* is converted into a form that can be recovered using *KRC*'s public key. However, in this model, key recovery can only be completed with the help of *KRA* to prevent abuse of privileges by *KRC*.
- **Key Recovery Auditor (*KRA*):** *KRA* is a monitoring agency that judges whether a key can be recovered by auditing the validity of key recovery. The *KRA* determines whether *KRF* is suitable for recovery to prevent abuse of authority through collusion between the *Med* and the *KRC*. If the key recovery request is deemed to be lawful, *KRA* will perform the *KRF* recovery process with its private key and sends it over to the *KRC*.

3.3.3. Security Requirements of Proposed Scheme-II

The security requirements of the key recovery system using proxy re-encryption are as follows:

- **KRF integrity:** No participant in key recovery can maliciously transform *KRF* information from the device and *KRF* information required for key recovery cannot be changed.
- **Data confidentiality:** It should be possible for only authorized devices to decrypt encrypted data.
- **Med applied for support:** The session key used for communication must be encrypted and stored in *KRF*. In the event of an emergency when it is necessary to view the device's data, the encrypted session key must be able to recover the encrypted message according to the procedure determined by *Med* as needed.
- **Collusion attack resistance:** Fewer than three participants among the *Med*, *KRC*, and *KRA* should not be allowed to obtain keys even if they are maliciously colluding.
- **Key escrow problem:** *KGC* can generate private keys for all participants, but the complete private key must not be known.

4. Proposed Scheme-I (Key Recovery System Using Signcryption)

In this section, we propose a key recovery scheme using signcryption. This scheme is a scheme for recovering the lost or corrupted device's key. This is mainly composed of a

setup phase, a key pair generation phase, a session key exchange and encryption phase, a KRF generation phase, and a session key recovery phase as shown in Figure 4.

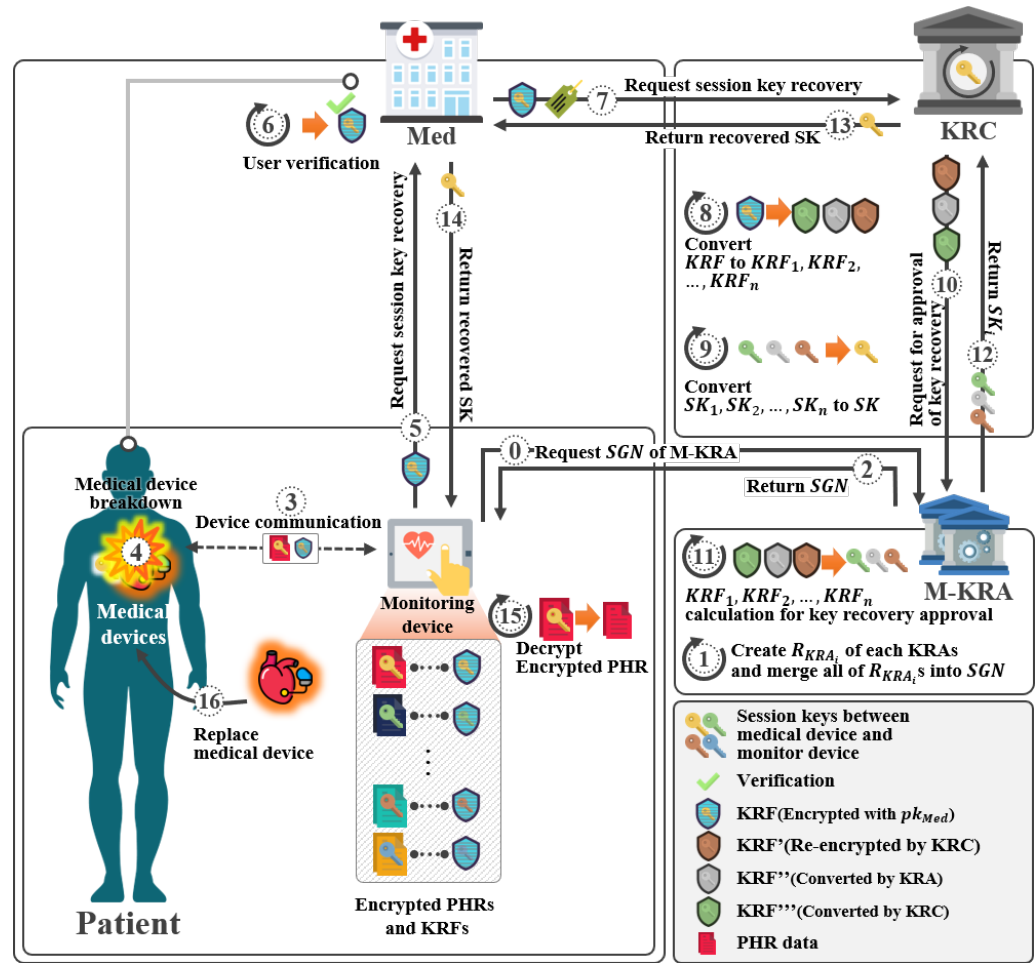


Figure 4. Scenario of proposed scheme-I.

4.1. System Parameters

The system parameters used in the proposed scheme-I are as follows.

- p : Prime number
- q : Prime factor of $p-1$
- \mathbb{G} : Cyclic group on prime p
- g : Random generator, $g \in \mathbb{G}$
- H : Hash function, $\{0, 1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_p^*$
- sk_M : Master private key, $sk_M \in \mathbb{Z}_p^*$
- pk_M : Master public key, $pk_M = g^{sk_M}$
- Dev_A : Monitoring devices
- Dev_B : Medical devices
- $Part_i$: Network Participant i , ($Dev_A, Dev_B, Med, KRC, KRA \in Part_i$)
- w_i, t_i, z_i, v_i : Random numbers, $w_i, t_i, z_i, v_i \in \mathbb{Z}_p^*$
- sk_i : $Part_i$'s private key, $sk_i = (d_i, z_i, v_i)$
- pk_i : $Part_i$'s public key, $pk_i = (X_i, Z_i, V_i)$
- a, b : Secret value of Dev_A and Dev_B , $a, b \in \mathbb{Z}_p^*$
- PSK_A, PSK_B : Partial session key of Dev_A and Dev_B
- SK : Session key between Dev_A and Dev_B
- x : Random number, $x \in \mathbb{Z}_p^*$ with $x \neq p-1$
- R_{KRA_i} : Random number of KRA_i , $R_{KRA_i} \in \mathbb{Z}_p^*$

- SGN : Group authentication values assigned to agents (Shared Group Number)
- c, r, s : Signcryption values
- c_i, r_i, s_i : i th signcryption pieces
- TT_i : Value containing the value to be recovered when some KRAs fail the key recovery operation
- Tc_i, Tr_i, Ts_i : i th TT_i pieces
- \mathcal{M} : Message space, $\mathcal{M} \in \{0, 1\}^n$
- M : Plaintext message between Dev_A and Dev_B ($M \in \mathcal{M}$)
- C : Ciphertext message (Encrypted M)
- KRF : Key recovery field, $E_{pk_{KRC}}(KRF_1 || KRF_2 || \dots || KRF_n || H(SGN))$
- KRF_i : i th key recovery field piece, $E_{pk_{KRA_i}}(c_i || r_i || s_i || SGN || TT_i)$

4.2. Setup Phase

In this phase, the KGC takes the security parameters as an input the security parameter 1^λ and generates public parameters.

- **Step 1:** The KGC selects λ -bit large prime p , where q is a large prime factor of $p - 1$ and group \mathbb{G} of prime order p . In addition, a random generator $g \in \mathbb{G}$ is selected.
- **Step 2:** A master private key $sk_M \in \mathbb{Z}_p^*$ is randomly selected and a master public key $pk_M \in \mathbb{G}$ is computed.
- **Step 3:** KGC selects Hash function H .
- **Step 4:** Then, public parameters $params = (G, n, p, q, g, S, H)$ are published.

4.3. Key Pair Generation Phase

In this phase, $Part_i$ receives a partial private key from KGC and uses it to generate full private key sk_i and public key pk_i .

- **Step 1:** KGC generates parameters $w_i, t_i \in \mathbb{Z}_p^*$ for participant $Part_i$ through the following operation and sends them to $Part_i$ through a secure channel.

$$X_i = g^{x_i} \quad (1)$$

$$d_i = x_i + sH(ID_i, w_i) \bmod q \quad (2)$$

- **Step 2:** Participant $Part_i$ who receives X_i, d_i from KGC, selects Random numbers $z_i, v_i \in \mathbb{Z}_p^*$ and sets $Part_i$'s private key sk_i .
- **Step 3:** Participant $Part_i$ generates Z_i, V_i and sets public key pk_i .

$$Z_i = g^{z_i} \quad (3)$$

$$V_i = g^{v_i} \quad (4)$$

4.4. Session Key Exchange and Encryption Phase

In this phase, the key recovery system uses signcryption to ensure integrity and non-repudiation and performs encryption of the session key simultaneously as shown in Figure 5.

- **Step 1:** Dev_A selects $a \in \mathbb{Z}_p^*$ and calculate partial session key $PSK_A = g^a$. Dev_B also selects $b \in \mathbb{Z}_p^*$ and calculates partial session key $PSK_B = g^b$. After that, Dev_A and Dev_B exchange PSK_A and PSK_B with each other.
- **Step 2:** Dev_A and Dev_B calculate the session key $SK = (PSK_B)^a = (PSK_A)^b$ using the exchanged values PSK_A and PSK_B .
- **Step 3:** Dev_A generates random number $x \in \mathbb{Z}_p^*$ and $k = pk_{KRC}^x \bmod p$, which is then divided in half into k_1 and k_2 .

$$k = (k_1 || k_2) \quad (5)$$

- **Step 4:** Dev_A generates c, r and s using k_1, k_2, sk_A, pk_A and SK .

$$c = E_{k_1}(SK) \quad (6)$$

$$r = H_{k_2}(g^x \bmod p, SK) \quad (7)$$

$$s = x / (r + sk_A) \bmod q \quad (8)$$

- **Step 5:** Dev_A divides c, r and s to c_i, r_i and s_i .

$$c = c_1 \oplus c_2 \oplus \dots \oplus c_n \quad (9)$$

$$r = r_1 \oplus r_2 \oplus \dots \oplus r_n \quad (10)$$

$$s = s_1 \oplus s_2 \oplus \dots \oplus s_n \quad (11)$$

where n is the number of KRA and $c_i = \{c_1, \dots, c_n\}, r_i = \{r_1, \dots, r_n\}, s_i = \{s_1, \dots, s_n\}$.

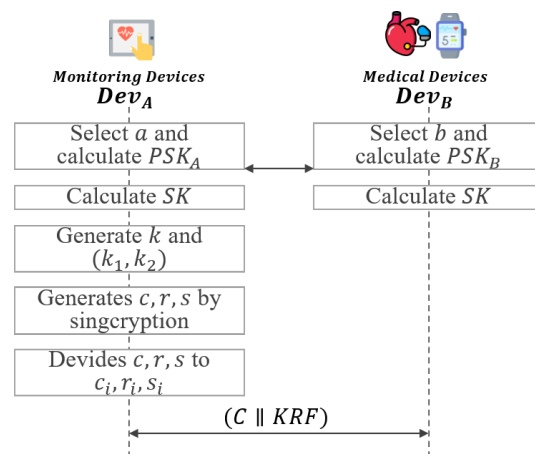


Figure 5. Session key exchange and encryption phase of proposed scheme-I.

4.5. KRF Generation Phase

In this phase, when the key is lost or corrupted, the necessary KRF is generated to recover the key as shown in Figure 6.

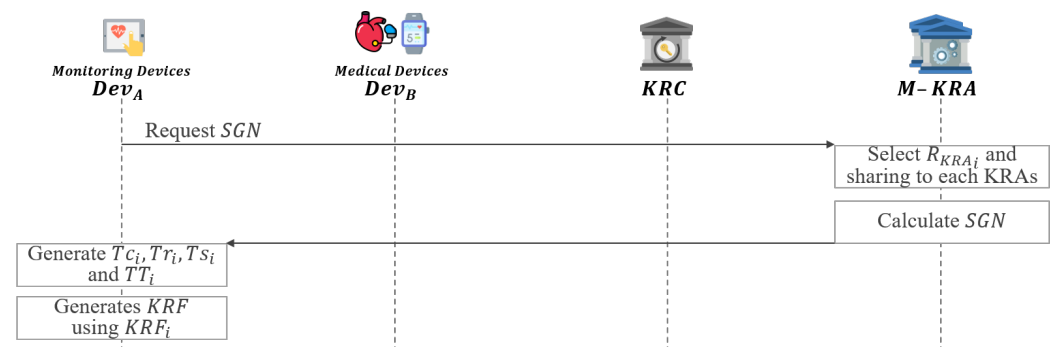


Figure 6. KRF generation phase of proposed scheme-I.

- **Step 1:** Dev_A requests SGN to M-KRA.
- **Step 2:** Each of the KRAs requested for SGN from Dev_A randomly selects $R_{KRA_i} \in \mathbb{Z}_p^*$. After that, each KRA generates an SGN by sharing R_{KRA_i} generated through a secure channel with each other.

$$SGN = R_{KRA_1} \oplus R_{KRA_2} \oplus \dots \oplus R_{KRA_n} \quad (12)$$

- **Step 3:** M-KRA send SGN to Dev_A .

- **Step 4:** Dev_A generates Tc_i, Tr_i, Ts_i using c_i, r_i, s_i and SGN . Then, TT_i is generated using Tc_i, Tr_i , and Ts_i .

$$Tc_i = c_i \oplus SGN \quad (13)$$

$$Tr_i = r_i \oplus SGN \quad (14)$$

$$Ts_i = s_i \oplus SGN \quad (15)$$

$$TT_i = (Tc_i \parallel Tr_i \parallel Ts_i) \quad (16)$$

- **Step 5:** Dev_A generates KRF using KRF_i .

$$KRF_i = E_{pk_{KRA_i}}(c_i \parallel r_i \parallel s_i \parallel SGN \parallel TT_i) \quad (17)$$

$$KRF = E_{pk_{KRC}}(KRF_1 \parallel KRF_2 \parallel \dots \parallel KRF_n \parallel H(SGN)) \quad (18)$$

- **Step 6:** Then, the generated KRF is attached to the ciphertext C .

$$(C \parallel KRF) \quad (19)$$

4.6. KRA Fault Recovery Phase

In this phase, if some KRAs fail to operate properly, the selected KRA or KRAs will instead perform key recovery as shown in Figure 7.

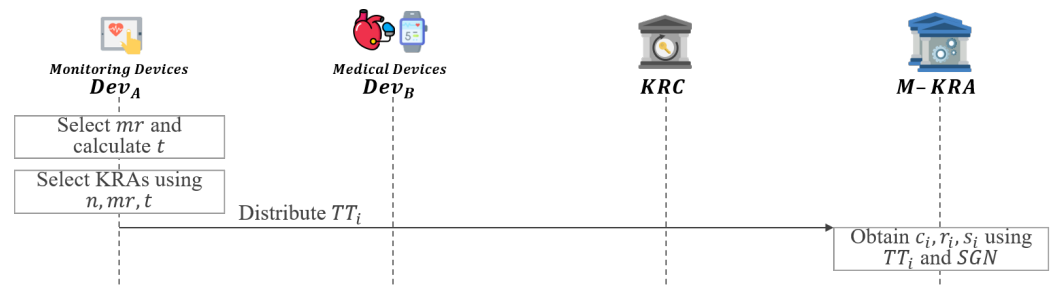


Figure 7. KRA fault recovery phase of proposed scheme-I.

- **Step 1:** Dev_A refers to the total number of KRAs n and the number of KRAs required for key recovery as mr .
- **Step 2:** Dev_A calculates the number of KRAs t required to distribute TT_i .

$$t = n - mr \quad (20)$$

- **Step 3:** Dev_A selects a KRA or KRAs to replace the failed KRA_i as follows:

$$j = i - mr \quad (21)$$

$$KRA_i \rightarrow \left\{ \begin{array}{l} KRA_{i+1}, KRA_{i+2}, \dots, KRA_{i+t} \ (i \leq mr) \\ KRA_{i+1}, \dots, KRA_n, KRA_1, \dots, KRA_j \ (i > mr \text{ and } i \neq n) \\ KRA_1, KRA_2, \dots, KRA_t \ (i = n) \end{array} \right\} \quad (22)$$

- **Step 4:** Dev_A distributes TT_i to selected KRA or KRAs.
- **Step 5:** If KRA_i fail to operate properly, the selected KRA or KRAs obtain c_i, r_i and s_i of failed KRA using the distributed TT_i and SGN .

$$\begin{aligned} TT_i \oplus SGN &= (Tc_i \parallel Tr_i \parallel Ts_i) \oplus SGN \\ &= (Tc_i \oplus SGN \parallel Tr_i \oplus SGN \parallel Ts_i \oplus SGN) \\ &= (c_i, r_i, s_i) \end{aligned} \quad (23)$$

4.7. Session Key Recovery Phase

This phase describes how to recover a key if the Dev_B requests key recovery as shown in Figure 8.

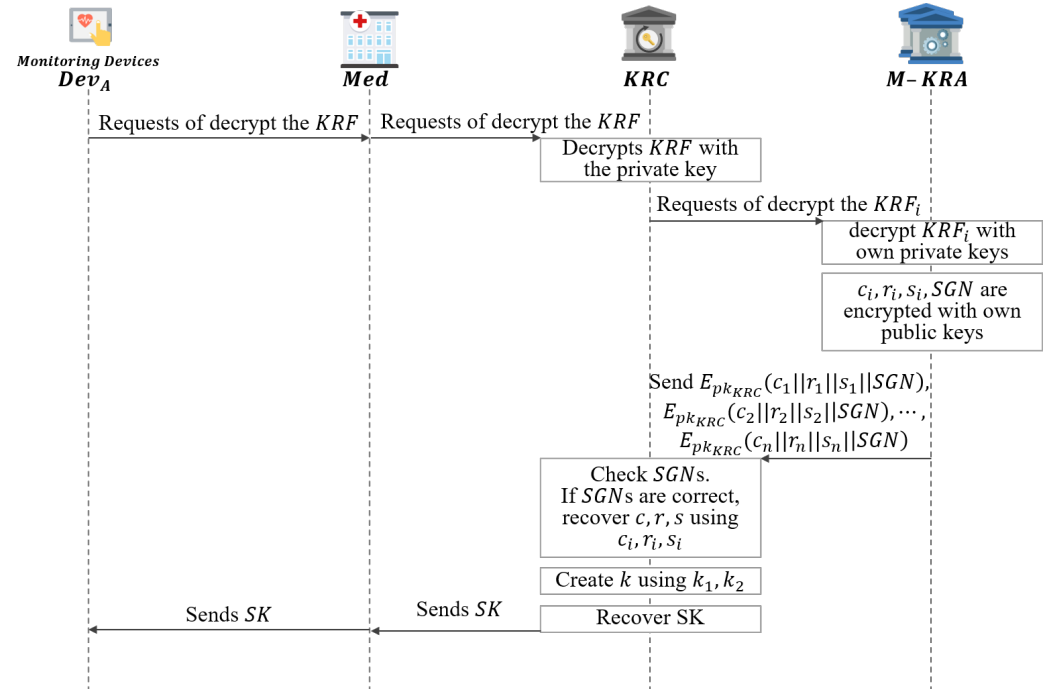


Figure 8. Session key recovery phase of proposed scheme-I.

- **Step 1:** When Dev_B requests KRF decryption from Med to recover SK , and sends KRF .
- **Step 2:** Then Med requests KRF decryption from KRC to recover SK , and sends KRF .
- **Step 3:** KRC upon receiving a request for KRF decryption, obtains KRF_i pieces after KRF decrypt with sk_{KRC} .

$$D_{sk_{KRC}}(E_{pk_{KRC}}(KRF_1 \parallel \dots \parallel KRF_n \parallel H(SGN))) \quad (24)$$

- **Step 4:** The obtained KRF_i pieces are sent to each $M-KRA$ to request decryption.
- **Step 5:** The requested $M-KRA$ obtain c_i, r_i, s_i, SGN, TT_i values with sk_{KRA_i} .

$$D_{sk_{KRA_i}}(E_{pk_{KRA_i}}(c_i \parallel r_i \parallel s_i \parallel SGN \parallel TT_i)) \quad (25)$$

- **Step 6:** Among the obtained values, c_i, r_i, s_i, SGN values are encrypted with pk_{KRC} and sends to the KRC .

$$E_{pk_{KRC}}(c_i \parallel r_i \parallel s_i \parallel SGN) \quad (26)$$

- **Step 7:** KRC compares SGN obtained by decrypting the received ciphertext with sk_{KRC} and $H(SGN)$. If they match, c_i, r_i, s_i pieces are collected and c, r, s are recovered.
- **Step 8:** KRC recovers the k value using the received ciphertext, public parameters, and recovered c, r, s .

$$k = H((pk_A \cdot g^r)^{s \cdot sk_{KRC}} \bmod p) \quad (27)$$

- **Step 9:** Then, KRC divides k by k_1, k_2 .
- **Step 10:** KRC recovers the SK using the obtained k_1 and c .

$$D_{k_1}(C) = D_{k_1}(E_{k_1}(SK)) = SK \quad (28)$$

- **Step 11:** KRC compares the calculated $H_{k_2}(SK)$ and r values using the obtained k_2 .
- **Step 12:** If it matches, KRC sends the recovered SK to Med.
- **Step 13:** Then, Med sends SK to Dev_B and the message is decrypted using the received SK .

$$D_{SK}(C) = D_{SK}(E_{k_1}(M)) = M \quad (29)$$

5. Proposed Scheme-II (Key Recovery System Using Proxy Re-Encryption)

In this section, we propose a proposed scheme-II. This scheme is a scheme recovering the lost and corrupted device's key. This system was designed based on the scheme of Yang et al. [31]. It consists of a setup phase, a key pair generation phase, a Med enforcement phase, and a session key recovery phase, as shown in Figure 9.

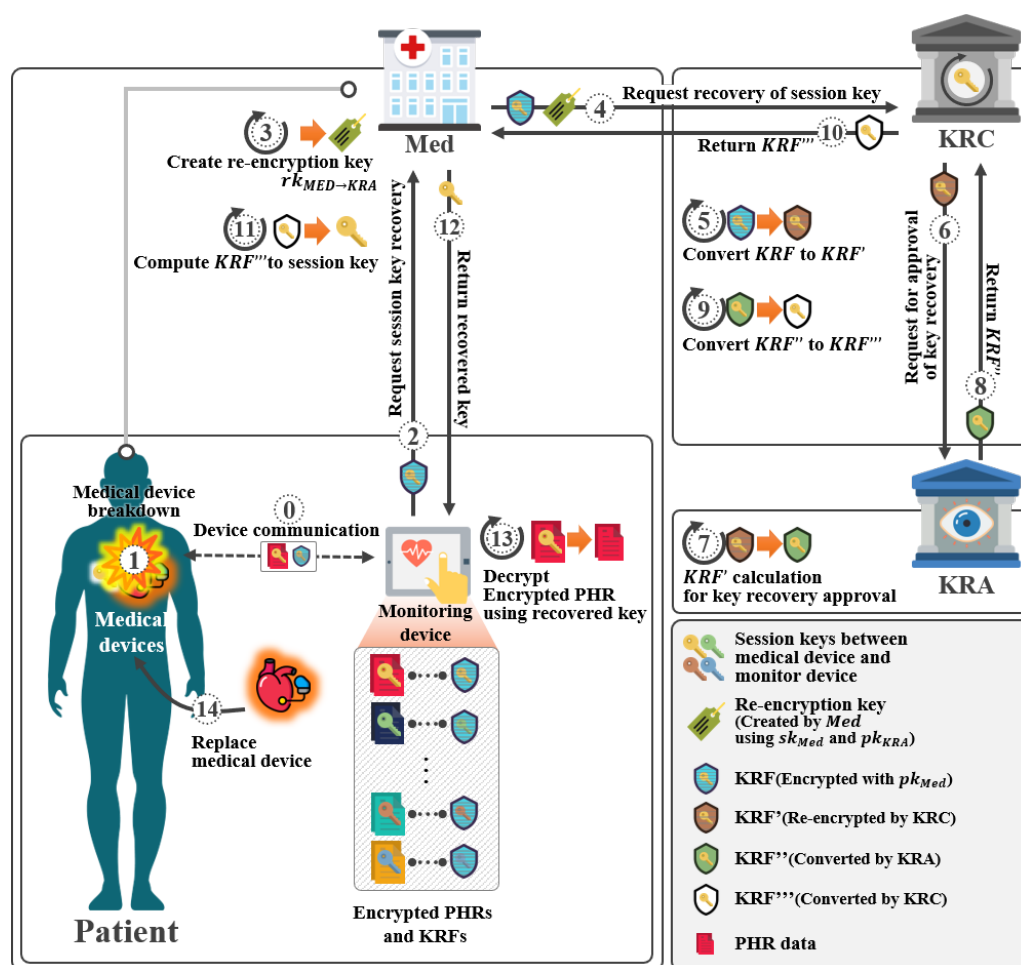


Figure 9. Scenario of proposed scheme-II.

5.1. System Parameters

The system parameters used in the proposed scheme-II are as follows:

- q : Prime number
- H_1 : Hash functions, $\{0, 1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$
- H_2 : Hash functions, $\mathbb{G} \rightarrow \{0, 1\}^{l_1+l_2}$ for some bit-length $l_1, l_2 \in \mathbb{N}$
- $H_3 - H_5$: Hash functions, $\{0, 1\}^* \rightarrow \mathbb{Z}_q^*$
- H_6 : Hash functions, $\mathbb{G} \rightarrow \mathbb{Z}_q^*$
- $Part_i$: System participant i , ($Dev_A, Dev_B, Med, KRC, KRA \in Part_i$)
- s : Master secret key of KGC, $s \in \mathbb{Z}_q^*$
- sk_i : $Part_i$'s private key, $sk_i = (d_i, y_i, z_i)$

- pk_i : $Part_i$'s public key, $pk_i = (X_i, Y_i, Z_i)$
- KRF : Key recovery field, $KRF = (KRF_1, KRF_2, KRF_3, KRF_4)$

5.2. Setup Phase

In this phase, the KGC takes the security parameter 1^λ as an input and generates public parameters.

- **Step 1:** KGC selects λ -bit large prime q and group \mathbb{G} of prime order q . In addition, a random generator $g \in \mathbb{G}$ is selected.
- **Step 2:** KGC randomly selects master secret key $s \in \mathbb{Z}_q^*$, and compute $S = g^s$.
- **Step 3:** KGC selects Hash function $H_1, H_2, H_3, H_4, H_5, H_6$.
- **Step 4:** The message space \mathcal{M} and public parameters $params = (\mathbb{G}, l_1, l_2, q, g, S, H_1, H_2, H_3, H_4, H_5, H_6)$ are published.

5.3. Key Pair Generation Phase

In this phase, $Part_i$ receives a partial private key from KGC and uses it to generate full private key sk_i and public key pk_i .

- **Step 1:** KGC generates parameters $x_i \in \mathbb{Z}_q^*$ for participant $Part_i$ through the following operation and sends them to $Part_i$ through a secure channel.

$$X_i = g^{x_i} \quad (30)$$

$$d_i = x_i + sH_1(ID_i, X_i) \bmod q \quad (31)$$

- **Step 2:** $Part_i$ who receives X_i, d_i from KGC, selects Random numbers $y_i, z_i \in \mathbb{Z}_q^*$ and sets $Part_i$'s private key sk_i .

$$sk_i = (d_i, y_i, z_i) \quad (32)$$

- **Step 3:** $Part_i$ generates Y_i, Z_i and sets public key pk_i .

$$Y_i = g^{y_i} \quad (33)$$

$$Z_i = g^{z_i} \quad (34)$$

$$pk_i = (X_i, Y_i, Z_i) \quad (35)$$

After that, $Part_i$ publishes public key pk_i .

5.4. Session Key Exchange and KRF Generation Phase

In this phase, a session key is exchanged between Dev_A and Dev_B , and a KRF is generated. Furthermore, in the KRF generation phase, after generating KRF, the ciphertext C is communicated with KRF as shown in Figure 10.

- **Step 1:** Dev_A selects $a \in \mathbb{Z}_q^*$ and calculate partial session key PSK_A .

$$PSK_A = g^a \quad (36)$$

Dev_B also selects $b \in \mathbb{Z}_q^*$ and calculates partial session key PSK_B .

$$PSK_B = g^b \quad (37)$$

After that, Dev_A and Dev_B exchange PSK_A and PSK_B with each other.

- **Step 2:** Dev_A and Dev_B calculate the session key $SK = (PSK_B)^a = (PSK_A)^b$ using the exchanged values PSK_A and PSK_B .
- **Step 3:** Dev_A generates the ciphertext message $C = E_{SK}(M)$ using the generated session key SK .

- **Step 4:** After that, Dev_A selects a random value $t, c \in \mathbb{Z}_q^*$ and $\sigma \in \{0,1\}^{l_2}$, and generates KRF using SK, pk_{Med}, pk_{KRC} and pk_{KRA} as follows:

$$\pi_i = X_i \cdot S^{H_1(ID_i, X_i)} \quad (38)$$

$$V_i = \pi_i^{H_6(Y_i)} \cdot Y_i \quad (39)$$

$$\tau = H_5(SK, \sigma, ID_{Med}, pk_{Med}) \quad (40)$$

$$\alpha = Y_{Med}^c \quad (41)$$

$$\beta = Z_{Med}^c \quad (42)$$

$$KRF_1 = g^t \quad (43)$$

$$KRF_2 = Y_{KRC}^\tau = g^{\tau \cdot y_{KRC}} \quad (44)$$

$$KRF_3 = (SK \parallel \sigma) \oplus H_2(g^c) \quad (45)$$

$$KRF_4 = (\alpha \cdot \beta)^{H_4(V_{KRA}^\tau) \cdot H_4(V_{Med}^\tau)} \quad (46)$$

$$KRF = (KRF_1, KRF_2, KRF_3, KRF_4) \quad (47)$$

After that, Dev_A and Dev_B communicate with each other using $(C \parallel KRF)$.

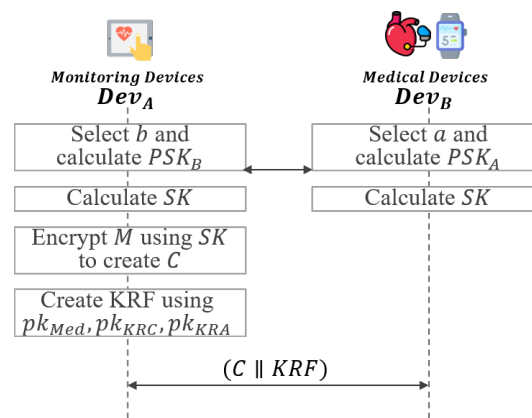


Figure 10. Session key exchange and KRF generation phase of proposed scheme-II.

5.5. Med Enforcement Phase

In this phase, Med will start recovering the encrypted session key between Dev_A and Dev_B at the request of Dev_A as shown in Figure 11.

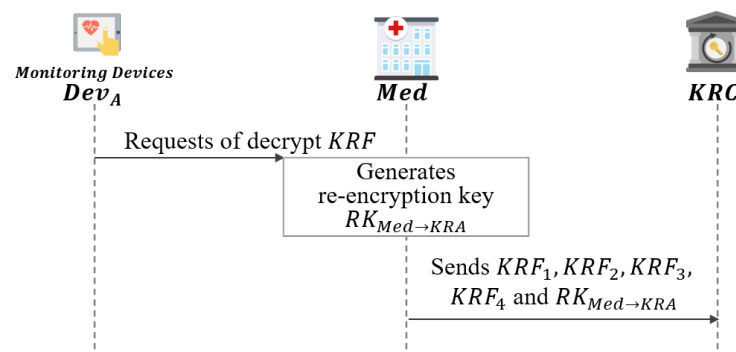


Figure 11. Med enforcement phase of proposed scheme-II.

- **Step 1:** Dev_A sends KRF to Med to recover the session key SK.

- **Step 2:** *Med* generates the re-encryption key $RK_{Med \rightarrow KRC}$.

$$\gamma_{KRC} = X_{KRC} \cdot S^{H_1(ID_{KRC}, X_{KRC})} \quad (48)$$

$$K_{Med-KRC} = H_3(\gamma_{KRC}^{z_{Med}}, Z_{KRC}^{z_{Med}}, ID_{Med}, pk_{Med}, ID_{KRC}, pk_{KRC}) \quad (49)$$

$$RK_{Med \rightarrow KRC} = (d_{Med} H_6(Y_{Med}) + y_{Med}) \cdot K_{Med-KRC} \quad (50)$$

- **Step 3:** *Med* requests key recovery by sending the $(KRF_1, KRF_2', KRF_4, KRF_6, RK_{Med \rightarrow KRC})$ to the *KRC*.

5.6. Session Key Recovery Phase

In this phase, *KRC* receives a key recovery request from *Med*. *KRF* calculates KRF_2' using its private key, and then requests key recovery from *KRA* as shown in Figure 12.

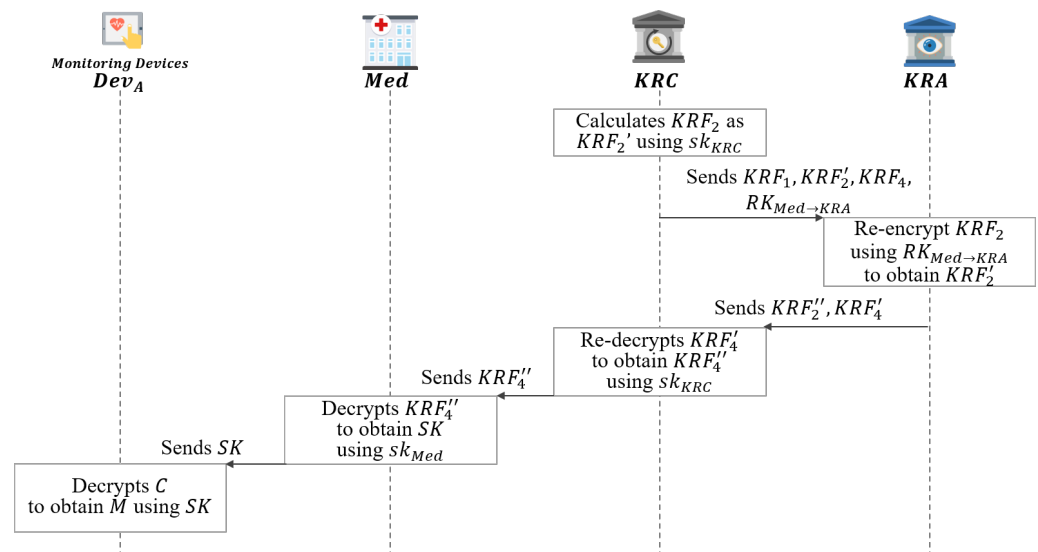


Figure 12. Session key recovery phase of proposed scheme-II.

- **Step 1:** After receiving $(KRF_1, KRF_2', KRF_4, RK_{Med \rightarrow KRC})$, *KRC* calculates KRF_2' as KRF_2' using its sk_{KRC} as follows:

$$\begin{aligned} KRF_2' &= KRF_2^{1/y_{KRC}} \\ &= \gamma_{KRC}^{\tau/y_{KRC}} \\ &= g^{\tau \cdot y_{KRC}/y_{KRC}} \\ &= g^{\tau} \end{aligned} \quad (51)$$

After that, *KRC* sends the generated $(KRF_1, KRF_2', KRF_4, RK_{Med \rightarrow KRC})$ to the *KRA*.

- **Step 2:** After receiving $(KRF_1, KRF_2', KRF_4, RK_{Med \rightarrow KRC})$, *KRA* re-encrypts KRF_2' as KRF_2'' using its $RK_{Med \rightarrow KRC}$ as follows:

$$\begin{aligned} KRF_4' &= KRF_4^{1/H_4(KRF_2'^{d_{KRA} \cdot H_4(Y_{KRA}) + y_{KRA}})} \\ &= (\alpha \cdot \beta)^{H_4(V_{KRA}^T) \cdot H_4(V_{Med}^T)/H_4(c_2^{d_{KRA} \cdot H_4(Y_{KRA}) + y_{KRA}})} \\ &= (\alpha \cdot \beta)^{H_4(V_{Med}^T)} \end{aligned} \quad (52)$$

$$KRF_2'' = KRF_2'^{RK_{Med \rightarrow KRC}} \quad (53)$$

After that, *KRA* sends (KRF_2'', KRF_4') to the *KRC*.

- **Step 3:** *KRC* re-decrypts KRF_4' to obtain KRF_4'' using sk_{KRC} as follows:

$$K_{Med-KRC} = H_3(Z_{Med}^{d_{KRC}}, Z_{Med}^{z_{KRC}}, ID_{Med}, pk_{Med}, ID_{KRC}, pk_{KRC}) \quad (54)$$

$$\begin{aligned} KRF_4'' &= KRF_4'^{1/H_4(KRF_2''^{1/K_{Med-KRC}})} \\ &= (\alpha \cdot \beta)^{H_4(V_{Med}^r)/H_4(KRF_2''^{1/K_{Med-KRC}})} \\ &= (\alpha \cdot \beta) \end{aligned} \quad (55)$$

After that, *KRC* sends KRF_4'' to *Med*.

- **Step 4:** *Med* decrypts KRF_4'' to obtain SK as follows:

$$\begin{aligned} g^c &= KRF_4''^{1/(y_{Med}+z_{Med})} \\ &= (\alpha \cdot \beta)^{1/(y_{Med}+z_{Med})} \\ &= (Y_{Med} \cdot Z_{Med})^{1/(y_{Med}+z_{Med})} \\ &= g^{c \cdot (y_{Med}+z_{Med})/(y_{Med}+z_{Med})} \end{aligned} \quad (56)$$

$$(SK \parallel \sigma) = KRF_3 \oplus H_2(g^c) \quad (57)$$

After that, *Med* sends SK to *Dev_A*.

- **Step 5:** *Dev_A* decrypts the message M using the obtained SK .

$$M = Dec_{SK}(C) \quad (58)$$

6. Analysis of the Proposed Schemes

This section explores whether the abovementioned security requirements are satisfied by the two proposed schemes, as shown in Table 1.

Table 1. Comparison of proposed schemes.

	[6]	[7]	[8]	[10]	[11]	[12]	[13]	Proposed Scheme-I	Proposed Scheme-II
KRF integrity	-	-	-	×	×	×	×	✓	✓
Non-repudiation	-	-	-	×	×	×	×	✓	×
Attack on group authentication detection	-	✓	-	✓	×	×	✓	✓	-
Single point of failure protection	×	×	×	✓	✓	×	×	✓	×
Data confidentiality	✓	✓	✓	✓	✓	✓	✓	✓	✓
Med applied for support	-	-	-	-	-	-	-	✓	✓
Collusion attacks resistance	✓	×	×	×	×	×	×	×	✓
Key escrow problem	×	×	×	✓	×	×	×	✓	✓

✓ : Provided / × : Not provided / - : Not considered

6.1. Proposed Scheme-I (Key Recovery System Using Signcryption)

- **KRF integrity:** The device, *Med*, *KRA*, and *KRC* participating in key recovery should not be able to transform a device key that generates a *KRF* maliciously. To solve this problem, this includes the session key hash in parameter r of the *KRF*. Therefore, *KRF* data cannot be forged. Only the device can access the *KRF* session key generated by the device.

$$r = ? r' \\ H_{k_2}(g^x \bmod p, SK) = ? H_{k_2}(g^x \bmod p, SK') \quad (59)$$

- Data confidentiality: In the proposed scheme-I, communication between devices is performed through a session key. Therefore, if the session key for the corresponding communication is unknown, the malicious user will not be able to obtain the message. In addition, as the *KRF* generated in the communication process contains the public keys of *KRC* and *M – KRA*, third-party besides *KRC* and *KRA* cannot know the contents of the corresponding *KRF*.
- Non-repudiation: If the device generates and uses the wrong *KRF*, *KRC* cannot recover the key. To solve this problem, the device should not be able to reject the fact that it generated *KRF*. Therefore, this includes the private key sk_A of the device in parameter *s* of the *KRF*. The device cannot deny that it generated the *KRF*.

$$\begin{aligned} k &= (pk_A \cdot g^r)^{s \cdot sk_{KRC}} \\ &= (g^{sk_A} \cdot g^r)^{s \cdot sk_{KRC}} \\ &= (g^{sk_A} \cdot g^r)^{x/(r+sk_A) \cdot sk_{KRC}} \\ &= g^{sk_{KRC}x} = pk_{KRC}^x \end{aligned} \quad (60)$$

- Attack on group authentication detection: Malicious key recovery by third-party *KRAs* should not be possible. Therefore, a lawful *KRA* group member applies an XOR operation on the values from R_{KRA_1} to R_{KRA_n} to generate a shared group value of *SGN* between groups. The device receives it from a lawful group member and hashes the *SGN* to include $H(SGN)$ in the *KRF*. When *KRC* recovers the complete key, it hashes and compares the *SGN* sent by the *M – KRA* with the *SGN* contained in the *KRF* to ensure it was received from a lawful *KRA*.

$$H(SGN) = ? H(SGN') \quad (61)$$

- Single point of failure protection: As both the *KRC* and all *KRAs* participate in session key recovery, it should be possible to recover the key even if some *KRAs* fail. Therefore, a special value *TT* is generated. If some *KRAs* fail to recover the session key pieces, other *KRAs* recover the session key pieces instead of the failed *KRA* and send them to the *KRC*. *TT* includes all c_i, r_i, s_i pieces and the *SGN* produced by the XOR operation. The other *KRA* (not the corresponding *KRA*) decrypts TT_i and sends it to the *KRC*, allowing the *KRC* to recover the complete session key.

$$TT_i = (Tc_i \parallel Tr_i \parallel Ts_i) \quad (62)$$

$$\begin{aligned} TT_i \oplus SGN &= ((c_i \oplus SGN) \parallel (r_i \oplus SGN) \parallel (s_i \oplus SGN)) \\ &= ((c_i \oplus SGN) \oplus SGN \parallel (r_i \oplus SGN) \oplus SGN \parallel (s_i \oplus SGN) \oplus SGN) \\ &= (c_i \parallel r_i \parallel s_i) \end{aligned} \quad (63)$$

- Med applied for support: *Med* should be able to view the encrypted data by acquiring the encrypted session key in the event of an emergency where it is necessary to view the device's data. Therefore, *Med* sends *KRF* to *KRC*, and *KRC* decrypts *KRF* to obtain *KRF* pieces. The acquired *KRF* pieces are sent to *M – KRA* and requested for recovery. Then, *M – KRA* obtains session key pieces by decrypting the acquired *KRF* pieces. The obtained session key pieces are sent to *KRC*, and *KRC* recovers the complete session key. After that, it sends the complete session key to the *Med*, allowing message decryption.
- Key escrow problem: The proposed scheme-I is based on a CL-PKC scheme. Therefore, as *KGC* can generate only a part of the private key during the private key genera-

tion process, the key escrow problem caused by KGC in ID-based encryption has been solved.

6.2. Proposed Scheme-II (Key Recovery System Using Proxy Re-Encryption)

- KRF integrity: In this proposed scheme-II, KRF is encrypted with the public keys of Med and KRA. Therefore, during the key recovery process, Med, KRC and KRA cannot be forged or modified KRF by alone.

$$KRF = (KRF_1, KRF_2, KRF_3, KRF_4) \quad (64)$$

$$\pi_{Med} = X_{Med} \cdot S^{H_1(ID_{Med}, X_{Med})} \quad (65)$$

$$V_{Med} = \pi_{Med}^{H_6(Y_{Med})} \cdot Y_{Med} \quad (66)$$

$$KRF_4 = (\alpha \cdot \beta)^{H_4(V_{KRA}^\tau) \cdot H_4(V_{Med}^\tau)} \quad (67)$$

- Data confidentiality: As the KRF generated in the communication process contains the public key of the Med and the secret values of KRC and KRA, third-party besides the Med, KRC, and KRA cannot know the contents of the corresponding KRF. In addition, even if all three of the Med, KRC, and KRA do not participate, each Med, KRC, and KRA cannot know the contents of the KRF.
- Med applied for support: Med can perform recovery of SK as needed. KRF is created using the public key of Med. Med can perform the key recovery process when it is determined that the key recovery is necessary for Dev that it manages. For this, Med can create $RK_{Med \rightarrow KRC}$ and request and execute the key recovery process through KRC and KRA.
- Collusion attack resistance: Fewer than three participants among the Med, KRC, and KRA must be prevented from maliciously acting together, thus preventing recovery of the key, and unauthorized entities must be prevented from obtaining the key. Therefore, the Med requires the cooperation of the KRC and KRA to decrypt KRF. Thus, even if the Med has colluded with a single participant among the KRC and KRA, the completed key recovery cannot be achieved without the assistance of the third participant as follows:

$$KRF = (KRF_1, KRF_2, KRF_3, KRF_4) \quad (68)$$

$$KRF_3 = (SK \parallel \sigma) \oplus H_2(g^c) \quad (69)$$

In order to obtain SK from the above $KRF = (KRF_1, KRF_2, KRF_3, KRF_4)$, KRF_3 must be decrypted. In order to decrypt KRF_3 , Med, KRC and KRA need to know c or g^c . However, c and g^c know only Dev. Therefore, it is necessary to obtain g^c by decrypting KRF_4 .

$$KRF_4 = (\alpha \cdot \beta)^{H_4(V_{KRA}^\tau) \cdot H_4(V_{Med}^\tau)} \quad (70)$$

Here, KRF_4 contains $\alpha \cdot \beta = Y_{Med}^c \cdot Z_{Med}^c$, so the attackers are $H_4(V_{KRA}^\tau)$ and $H_4(V_{Med}^\tau)$ should be computed.

$$\pi_i = X_i \cdot S^{H_1(ID_i, X_i)} \quad (71)$$

$$V_i = \pi_i^{H_6(Y_i)} \cdot Y_i \quad (72)$$

Since V_i can be created using a public key, anyone can create it. However, since τ only knows Dev, attackers must use KRF_2 to calculate $H_4(V_{KRA}^\tau)$ and $H_4(V_{Med}^\tau)$.

$$KRF_2 = Y_{KRC}^\tau = g^{\tau \cdot y_{KRC}} \quad (73)$$

Here, a KRC's private key y_{KRC} is required to obtain g^τ from KRF_2 . Therefore, KRC is required in the key recovery process.

$$KRF'_2 = g^\tau = KRF_2^{1/y_{KRC}} = g^{\tau \cdot y_{KRC} / y_{KRC}} \quad (74)$$

Next, since the attacker does not know τ , he has to perform the following operation to calculate $H_4(V_{KRA}^\tau)$. In the end, the KRA's private keys d_{KRA} and y_{KRA} are required, so KRA is also required.

$$\begin{aligned} KRF'_4 &= KRF_4^{1/H_4(KRF_2^{d_{KRA} \cdot H_4(y_{KRA}) + y_{KRA}})} \\ &= (\alpha \cdot \beta)^{H_4(V_{KRA}^\tau) \cdot H_4(V_{Med}^\tau) / H_4(c_2^{d_{KRA} H_4(y_{KRA}) + y_{KRA}})} \\ &= (\alpha \cdot \beta)^{H_4(V_{Med}^\tau)} \end{aligned} \quad (75)$$

Furthermore, an attacker who acquires $(\alpha \cdot \beta)$ must compute g^c to obtain $(SK \parallel \sigma)$ from $KRF_3 = (SK \parallel \sigma) \oplus H_2(g^c)$.

$$\begin{aligned} g^c &= (\alpha \cdot \beta)^{1/(y_{Med} + z_{Med})} \\ &= (Y_{Med} \cdot Z_{Med})^{1/(y_{Med} + z_{Med})} \\ &= (g^{c \cdot y_{Med}} \cdot g^{c \cdot z_{Med}})^{1/(y_{Med} + z_{Med})} \\ &= g^{c \cdot (y_{Med} + z_{Med}) / (y_{Med} + z_{Med})} \end{aligned} \quad (76)$$

In order to acquire g^c using KRF'_4 , Med's private keys y_{Med} and z_{Med} are required, so Med is also required. As a result, in order to obtain SK by decrypting KRF, all of Med, KRC, and KRA must participate.

- Key escrow problem: The proposed scheme-II is based on a CL-PKC scheme. Therefore, as KGC can generate only a part of the private key during the private key generation process, the key escrow problem caused by KGC in ID-based encryption has been solved.

7. Conclusions

This paper proposed key recovery systems based on key encapsulation secured from various attacks in IoMT environments in schemes II and II.

In the key recovery system, the session key used in the ciphertext is recovered via the KRF and used. However, the KRF can be forged and KRF owners can deny the fact that they generated the KRF. Furthermore, unauthorized KRAs can access the M-KRA and interfere with key recovery. To solve this problem, the key recovery system using signcryption includes the session key hash in the KRF. Therefore, the KRF data cannot be forged. In addition, this system includes the private key of a device in special value of the KRF. A device cannot deny that it generated the KRF. Furthermore, the system ensures the security requirements mentioned in Section 3, including KRF integrity and non-repudiation, are fulfilled.

Additionally, there is a problem that the key can be recovered by collusion attacks and key or message leakage among the Med, KRC, and KRA. To solve this problem, the Med must have the help of the KRC and KRA to recover the key by a proxy re-encryption function. In addition, the KRC or KRA would also need mutual help to recover a complete session key. That is, by limiting the information and processing capabilities of the three participants, the key recovery system can be expected to be secure against various attacks. Furthermore, because the KGC generates the private keys of all participants, there is the problem that the KGC's authority is strong. To solve this, a partial private key generation scheme is used. The KGC generates a partial private key and sends it to the participants. Participants who receive partial private keys use them to generate complete private keys and solve the KGC key escrow problem.

Future research is to check whether unexpected problems occur when the proposed schemes are implemented in actual systems. Furthermore, additional research is needed that can examine the amount of computations, time, and cost incurred when recovering

keys. In addition, further research is needed to determine whether the proposed schemes are secure against other types of security threats.

Author Contributions: Conceptualization, T.K., W.K. and I.L.; methodology, T.K., W.K., D.S. and I.L.; data investigation, T.K., W.K.; analysis and validation, T.K., W.K. and I.L.; writing—original draft, T.K.; writing—review and editing, T.K., W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2019R1A2C1085718) and the BK21 FOUR (Fostering Outstanding Universities for Research)(No. :5199990914048) and the Soonchunhyang University Research Fund.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qiu, T.; Chen, N.; Li, K.; Atiquzzaman, M.; Zhao, W. How can heterogeneous internet of things build our future: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2011–2027.
2. Shammam, E.A.; Zahary, A.T. The Internet of Things (IoT): A survey of techniques, operating systems, and trends. *Library Hi Tech* **2019**, *38*, 5–66.
3. Yu, J.Y.; Kim, Y.G. Analysis of IoT platform security: A survey. In Proceedings of the 2019 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, 28–30 January 2019; pp. 1–5.
4. Standard, D.E. *Federal Information Processing Standards Publication 46*; National Bureau of Standards; US Department of Commerce: Washington, DC, USA, 1977; Volume 23.
5. Denning, D.E.; Branstad, D.K. A taxonomy for key escrow encryption systems. *Commun. ACM* **1996**, *39*, 34–40.
6. Guo, H.; Zheng, Y.; Li, X.; Li, Z.; Xia, C. Self-healing group key distribution protocol in wireless sensor networks for secure IoT communications. *Future Gener. Comput. Syst.* **2018**, *89*, 713–721.
7. Lee, Y.; Park, Y.; Kim, C.S.; Lee, B. Threats Analysis and Mobile Key Recovery for Internet of Things. *J. Korea Multimed. Soc.* **2016**, *19*, 918–923.
8. Sung, S.H. Key Management for Secure Internet of Things (IoT) Data in Cloud Computing. *J. Korea Inst. Inf. Secur. Cryptol.* **2017**, *27*, 353–360.
9. Hatzivasilis, G.; Soultatos, O.; Ioannidis, S.; Verikoukis, C.; Demetriou, G.; Tsatsoulis, C. Review of security and privacy for the Internet of Medical Things (IoMT). In Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini, Greece, 29–31 May 2019; pp. 457–464.
10. Kanyamee, K.; Sathitwiriawong, C. High-availability decentralized cryptographic multi-agent key recovery. *Int. Arab J. Inf. Technol.* **2014**, *11*, 52–58.
11. Lim, S.; Kang, S.; Sohn, J. Modeling of multiple agent based cryptographic key recovery protocol. In Proceedings of the IEEE 19th Annual Computer Security Applications Conference, Las Vegas, NV, USA, 8–12 December 2003; pp. 119–128.
12. Han, K.; Yeun, C.Y.; Kim, K. New key escrow model for the lawful interception in 3GPP. In Proceedings of the IEEE 2009 Digest of Technical Papers International Conference on Consumer Electronics, Las Vegas, NV, USA, 10–14 January 2009; pp. 1–2.
13. Huadpaknam, P.; Pirak, C.; Mathar, R. A Security Key Recovery System with Channel Quality Awareness for Smart Grid Applications. *ECTI Trans. Comput. Inf. Technol. (ECTI-CIT)* **2016**, *10*, 1–14.
14. Gennaro, R.; Karger, P.; Matyas, S.; Peyravian, M.; Roginsky, A.; Safford, D.; Willett, M.; Zunic, N. Two-phase cryptographic key recovery system. *Comput. Secur.* **1997**, *16*, 481–506.
15. Zheng, Y. Digital signcryption or how to achieve cost (signature & encryption) \ll cost (signature) + cost (encryption). In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 1997; Springer: Berlin/Heidelberg, Germany, 1997; pp. 165–179.
16. Liu, Z.; Hu, Y.; Zhang, X.; Ma, H. Certificateless signcryption scheme in the standard model. *Inf. Sci.* **2010**, *180*, 452–464.
17. Li, F.G.; Zhong, D. A Survey of Digital Signcryption. *Netinfo Secur.* **2011**, *12*, 1–8.
18. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613.
19. Blakley, G.R. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*; IEEE Computer Society: New York, NY, USA, 1979; p. 313.
20. Beimel, A. Secret-sharing schemes: A survey. In *International Conference on Coding and Cryptology*; Springer: Qingdao, China, 2011; pp. 11–46.
21. Attasena, V.; Darmont, J.; Harbi, N. Secret sharing for cloud data security: A survey. *Vldb J.* **2017**, *26*, 657–681.
22. Blaze, M.; Bleumer, G.; Strauss, M. Divertible protocols and atomic proxy cryptography. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Espoo, Finland, 31 May–4 June 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 127–144.
23. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **1985**, *31*, 469–472.

24. Green, M.; Ateniese, G. Identity-based proxy re-encryption. In Proceedings of the International Conference on Applied Cryptography and Network Security, Zhuhai, China, 5–8 June 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 288–306.
25. Shamir, A. Identity-based cryptosystems and signature schemes. In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Santa Barbara, CA, USA, 19–22 August 1984; Springer: Berlin/Heidelberg, Germany, 1984; pp. 47–53.
26. Al-Riyami, S.S.; Paterson, K.G. Certificateless public key cryptography. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, 30 November–4 December 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 452–473.
27. Sur, C.; Jung, C.D.; Park, Y.; Rhee, K.H. Chosen-ciphertext secure certificateless proxy re-encryption. In Proceedings of the IFIP International Conference on Communications and Multimedia Security, Linz, Austria, 31 May–2 June 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 214–232.
28. Polyakov, Y.; Rohloff, K.; Sahu, G.; Vaikuntanathan, V. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur. (TOPS)* **2017**, *20*, 1–31.
29. Dodis, Y.; Goldwasser, S.; Kalai, Y.T.; Peikert, C.; Vaikuntanathan, V. Public-key encryption schemes with auxiliary inputs. In Proceedings of the Theory of Cryptography Conference, Zurich, Switzerland, 9–11 February 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 361–381.
30. Mambo, M.; Okamoto, E. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **1997**, *80*, 54–63.
31. Yang, K.; Xu, J.; Zhang, Z. Certificateless proxy re-encryption without pairings. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Korea, 27–29 November 2013; Springer: Cham, Switzerland, 2013; pp. 67–88.