A Secure Healthcare Personal Data Sharing Scheme Based on Updatable Encryption and Proxy Re-Encryption

Ahmad Afiouni, Bian Yang, and Slobodan Petrović

Norwegian University of Science and Technology (NTNU)
P.O. box 191, N-2802 Gjøvik, Norway
ahmad.afiouni@ntnu.no
bian.yang@ntnu.no
slobodan.petrovic@ntnu.no

Abstract. We propose a secure hybrid Updatable Encryption - Proxy Re-Encryption (UE-PRE) data sharing scheme intended for use in a healthcare information system. It is assumed that data in this system are stored in a cloud, which may be untrusted. A user of the system can delegate access rights to other users using Proxy Re-Encryption and revoke them using Updatable Encryption. In addition, encryption keys can be rotated regularly and/or occasionally without the need for re-encryption. The concrete parameters of the components of the proposed hybrid UE-PRE scheme are given. Security and efficiency of the proposed solutions are analyzed.

Keywords: updatable encryption · proxy re-encryption · healthcare information system.

1 Introduction

Healthcare information systems are very complex and the requirements for data privacy, security, and integrity are very strict for them. At the same time, data traffic in these systems is intensive, which requires high efficiency of the applied security solutions. This holds for key management too. Two tasks related to key management are typical in these systems: access rights delegation and key rotation. The task of efficient access rights delegation assumes that these rights must be possible to switch efficiently between different users without the need for re-encryption by the data owner. The task of key rotation assumes that it must be possible to change keys efficiently, either upon a request (for example, due to key compromise) or regularly, again without the need for re-encryption. The requirement of avoiding re-encryption in performing these tasks is dictated by the fact that the ciphertexts are usually stored in a cloud, and it is highly unlikely that a simple download of a ciphertext, decryption, and subsequent re-encryption with a new key could be efficient enough for a smooth operation of such an information system.

Efficient access rights switching can be ensured by using Proxy Re-Encryption (PRE) schemes. In these schemes, instead of downloading the old ciphertext, re-encrypting with another user's key and uploading the new ciphertext to the cloud, the data owner sends so-called *re-encryption key* to a special proxy server. That server uses the re-encryption key to transform the ciphertext in such a way that a new user can decrypt it. When it comes to efficient key rotation, it can be ensured by using Updatable Encryption (UE) schemes. In such schemes, instead of downloading the old ciphertext by the data owner, decrypting, re-encrypting with the new key, and uploading again to the cloud, the data owner sends just a small piece of information, so-called *token*, to the cloud. The token is used to update the ciphertext to the new version. In such a way, the efficiency of the key rotation is significantly improved, see for example [3].

In order to be adequate for application in a healthcare information system for data protection, any PRE and/or UE scheme is required to use a cloud that need not be trusted. Thus, the cloud's own security solutions (that are always available and offered by cloud service providers) are not required to be used. The data owner (in this case, a user of the healthcare information system) has an opportunity to use his/her own security algorithms, but this makes them responsible for their security and efficiency.

This paper extends the work done in [1] regarding the proposal of a new hybrid UE-PRE scheme applicable in a healthcare information system. We give a detailed description of multiplicative and additive variants of the PRE and UE algorithms. We also propose the concrete parameters of the PRE component of our scheme (the Umbral scheme [7]) and the UE component of the scheme (the SHINE scheme [5]). These parameters have been selected in such a way that the PRE and UE algorithms ensure data security in a healthcare system. At the same time, our proposed hybrid UE-PRE scheme is efficient enough to guarantee smooth operation of the overall system. We analyze the security of the proposed UE-PRE scheme with the concrete parameters as well as its efficiency.

The paper is organized as follows: In Section 2, we expose mathematical fundamentals of updatable encryption and proxy re-encryption. We also describe the SHINE UE scheme, as given in [5], and the simplified (thresholdless) version of the Umbral PRE scheme. In Section 3, we describe and give the concrete parameters of our hybrid UE-PRE scheme suitable for efficient application in a healthcare environment. We also give some implementation details and discuss the efficiency of the implementation, illustrated by some experimental results. In Section 4, we analyze the security of the proposed hybrid UE-PRE scheme with the concrete parameters. Section 5 concludes the paper.

2 Preliminaries and problem definition

In this Section, we expose the fundamentals of UE and PRE. We first give Boneh's definition of key homomorphism. Then we explain the application of this concept in UE and PRE. We also define the problem of storage and data

re-encryption in a partially trusted cloud and possible ways to solve it using UE, PRE, and their combination.

2.1 Key homomorphism, UE, and PRE

The functions that realize UE and PRE must possess the key homomorphism property. Boneh [3] uses the following definition of key homomorphism: Let $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ be a pseudorandom function. If there exists an efficient algorithm to compute $F(k_1 \oplus k_2, x)$ from $F(k_1, x)$ and $F(k_2, x)$, then the function F is said to be *key homomorphic* i.e., F is homomorphic with respect to an element from the set $\mathcal{K}$ - the key. The operation $\oplus$ is a group operation on the keys k_1 and k_2. The security proofs of such functions have been given in the random oracle model. In [3], the security proof was given for the standard model and the first construction of such function was proposed based on Learning With Errors (LWE).

As we have stated in Section 1, in the case of practical healthcare data protection mechanisms, two applications of key homomorphism are needed - UE and PRE. The corresponding practical problem is the following: data (for example, patient records) have to be shared between some parties in the healthcare system, but they must be protected from disclosure to unauthorized parties i.e., *confidentiality* has to be ensured. We assume that the amount of data is so large that storing them in a cloud is the only reasonable possibility. The cloud is *partially* trusted, which means that we cannot trust completely the cloud provider's own security mechanisms. Instead, the healthcare system uses its own (private) protection algorithms for access control, encryption, integrity protection, and so on.

A typical data protection mechanism ensuring confidentiality is *encryption*. It is well known that, to ensure confidentiality even if some keys are compromised, *key rotation* must be implemented i.e., the keys are regularly changed and the data have to be re-encrypted after such a change. The same procedure must be used as an acute workaround if key compromise is detected or a party that has had right to access some pieces of data loses this right (for example, an ex-employee). Re-encryption implies downloading of enormous quantities of encrypted data to a local host, decryption using the old key, encryption using the new key, and uploading the new ciphertext to the cloud again. In most cases, this is not efficient enough, which may create bottlenecks in the system that is critical for saving lives. Possible solution to this problem is using UE. In a UE scheme, a key homomorphic pseudorandom function is used to create a new ciphertext from the old one. Instead of the whole procedure of decryption and re-encryption, we only upload a small piece of data, so-called *token* to the cloud, which serves as an update to the old key. Thus, we obtain the new ciphertext without the need for decryption and re-encryption. Of course, it is asumed that such a pseudorandom function is secure in a standard model, as discussed by Boneh in [3]: an adversary cannot reconstruct the plaintext given the old ciphertext, the new ciphertext, and the token. The time period of validity of a key in an UE system is called an *epoch*.

In data protection systems (including the healthcare system), it is sometimes necessary to transform the ciphertext encrypted with one key (of one user) to a ciphertext encrypted with another key (of another user). In such a way, the owner of this second key gets access to the data. Again, the trivial procedure of decryption with one key and encryption with the other key is not efficient enough. Instead, the transformation of one ciphertext to another (of the same plaintext) is outsourced to a partially trusted proxy and a PRE scheme is used to perform this task. In a PRE scheme, apart from the ciphertext encrypted with the key of the first user, the proxy is provided with the piece of information (re-encryption key) Δ that serves it to compute the ciphertext encrypted with another user's key. The PRE scheme also ensures that the proxy cannot reconstruct the plaintext from the original ciphertext and the re-encryption key Δ.

2.2 UE schemes

Several UE schemes have been proposed, for example the RISE scheme [6], based on ElGamal-based proxy re-encryption and re-randomization of ciphertexts. In this paper, we focus on the UE scheme proposed in [4] – SHINE (Secure Homomorphic Ideal-cipher Nonce-based Encryption). In this scheme, the encryption algorithm uses a permutation to obfuscate the input to the exponentiation function. Updating the ciphertext requires an exponentiation function, which takes the update token as input. The token is the quotient of the division of the new epoch key by the previous epoch key. The length of the plaintext determines the variant of the SHINE scheme that is to be used - the simplest variant, SHINE0, is used for processing of short plaintexts and uses only one permutation. The second variant, MirrorSHINE, uses two permutations for the same input. Finally, the variant OCBSHINE can be used for any length of the input plaintext. It uses a family of permutations. For all the three variants of SHINE, security and integrity proofs are given in [4], where the criterion of UE security stronger than the previously defined ones is given. Regarding the implementation, the paper [4] gives some general recommendations, for example AES256/512 or Threefish block cipher for the permutation and special NIST elliptic curves (256 or 512) for the exponentiation. However, no concrete parameters have been given. The goal of this paper is to provide the concrete parameters to the recommended cipher systems involved in the SHINE scheme, which ensure both high level of security and acceptable efficiency of the overall UE algorithm. The details are exposed in the following sections.

2.3 PRE schemes

Most PRE schemes use public key encryption algorithms, whose security is based on difficulty of solving the discrete logarithm problem. Examples are the first proposed PRE scheme by Blaze et al. [2] and the threshold PRE scheme Umbral proposed by Nuñez [7], on which we base our proposal. We give an example of a PRE scheme based on a simplified (thresholdless) version of the Umbral scheme. It is assumed that the owner of the data (Alice) delegates the access rights to

other participants. We describe the scheme in the multiplicative form first and then in the additive form (elliptic curve-based).

2.3.1 The multiplicative form of the PRE scheme

In the multiplicative form, the base cryptosystem is the following: Let p be a prime and let g be a generator in the group $Z_p^\times$. Let H be a cryptographic hash function. p, g, and H are used as global parameters. Alice's secret key is a, $0 < a < p$, and her public key is $g^a \bmod p$. Bob's secret key is b, $0 < b < p$, and his public key is $g^b \bmod p$. Alice wants to encrypt a message m and send it to Bob. She takes Bob's public key g^b, computes the ciphertext $C_{AB} = (C_{AB1}, C_{AB2})$, where $C_{AB1} = g^b \bmod p$ and $C_{AB2} = m(g^b)^a \bmod p = mg^{ab} \bmod p$, and sends it to Bob. Bob knows that the ciphertext that he received was sent by Alice (Alice's authentication was successful), so to decrypt he takes Alice's public key g^a and computes $C_{AB2}(g^a)^{-b} \bmod p = mg^{ab}g^{-ab} \bmod p = m$. Note that C_{AB1} is kept in the ciphertext for the purpose of proxy re-encryption defined below and without proxy re-encryption it is not needed in the cryptosystem.

To create a PRE system based on this cryptosystem, we define the re-encryption key and the re-encryption transformation. The re-encryption key is generated by Alice that wants to delegate the access rights for the message m to a new user Charlie, whose secret key is c and the public key is g^c. The re-encryption key produced by Alice is:

$$\Delta_{AC} = a(H((g^c)^a))^{-1} \bmod p = a(H(g^{ac}))^{-1}. \tag{1}$$

The re-encryption key given by the equation (1) is sent to the proxy, which transforms the ciphertext C_{AB} to the ciphertext C_{AC} intended for Charlie using the re-encryption transformation:

$$C_{AC1} = (C_{AB1})^{\Delta_{AC}} \bmod p = (g^b)^{\Delta_{AC}} \bmod p = g^{ab(H(g^{ac}))^{-1}}. \tag{2}$$

Note that C_{AB2} is not changed i.e., $C_{AC2} = C_{AB2}$. Charlie knows that the ciphertext that he received was produced by Alice (and the proxy) so to decrypt he takes Alice's public key g^a and computes

$$\begin{aligned} &C_{AC2}\left(C_{AC1}^{H((g^a)^c)}\right)^{-1} \bmod p = \\ &= mg^{ab}\left(\left(g^{ab(H(g^{ac}))^{-1}}\right)^{H(g^{ac})}\right)^{-1} \bmod p = mg^{ab}g^{-ab} \bmod p = m. \end{aligned} \tag{3}$$

2.3.2 The additive form of the PRE scheme

In the additive form, the base cryptosystem is the following: Let p be a prime and let g be a point on an elliptic curve E modulo p. Let H be a cryptographic hash function, whose argument x is obtained by concatenating the coordinates of a point on E i.e., for the point $P = (x_1, x_2)$ it holds $x = x_1||x_2$. Note that

the output of the hash function H is a *scalar*. p, g, and H are used as global parameters. Alice's secret key is a, $0 < a < p$, and her public key is $a \cdot g$. Bob's secret key is b, $0 < b < p$, and his public key is $b \cdot g$. Alice wants to encrypt a message m (represented by the point P_m on the curve E using for example the Koblitz method, see for example [9]) and send it to Bob. She takes Bob's public key $b \cdot g$, computes the ciphertext $C_{AB} = (C_{AB1}, C_{AB2})$, where $C_{AB1} = b \cdot g$ and $C_{AB2} = P_m + ab \cdot g$, and sends it to Bob. Bob knows that the ciphertext that he received was sent by Alice (Alice's authentication was successful), so to decrypt he takes Alice's public key $a \cdot g$ and computes $C_{AB2} - ab \cdot g = P_m + ab \cdot g - ab \cdot g = P_m + \mathcal{O} = P_m$, where $\mathcal{O}$ is the point at infinity. Note that, as it is the case with the multiplicative form of the cryptosystem, C_{AB1} is kept in the ciphertext for the purpose of proxy re-encryption defined below and without proxy re-encryption it is not needed in the cryptosystem.

In a PRE system based on this cryptosystem, the re-encryption key is generated by Alice that wants to delegate the access rights for the message m to a new user Charlie, whose secret key is c and the public key is $c \cdot g$. The re-encryption key produced by Alice is:

$$\Delta_{AC} = a(H(ac \cdot g))^{-1} \bmod p. \tag{4}$$

Note that Δ_{AC} is a scalar. The re-encryption key given by the equation (4) is sent to the proxy, which transforms the ciphertext C_{AB} to the ciphertext C_{AC} intended for Charlie using the re-encryption transformation:

$$C_{AC1} = \Delta_{AC} \cdot C_{AB1} = ab(H(ac \cdot g))^{-1} \cdot g. \tag{5}$$

Note that C_{AB2} is not changed i.e., $C_{AC2} = C_{AB2}$. Charlie knows that the ciphertext that he received was produced by Alice (and the proxy) so to decrypt he takes Alice's public key $a \cdot g$ and computes

$$\begin{aligned} C_{AC2} - H(ac \cdot g) \cdot C_{AC1} &= P_m + ab \cdot g - H(ac \cdot g) \cdot \left(ab(H(ac \cdot g))^{-1} \cdot g\right) = \\ &= P_m + ab \cdot g - ab \cdot g = P_m + \mathcal{O} = P_m. \end{aligned} \tag{6}$$

Using either the additive or the multiplicative form of the PRE scheme described above, the ciphertext from Alice addressed to Bob is converted to the ciphertext from Alice addressed to Charlie and the piece of data m is now shared between Alice and Charlie.

3 The proposed hybrid UE-PRE scheme

In this Section, we propose a hybrid UE-PRE encryption scheme adequate for data protection in a system such as a healthcare information system. We first explain the difference between public key encrypted data storage and public key encrypted communication. Then we give the global description of the proposed hybrid UE-PRE system. Finally, we provide the concrete parameters of the proposed system adequate for efficient implementation.

3.1 Storage vs. communication with public key systems

To explain our proposed scheme, we first have to explain the difference between public key encrypted remote storage and public key encrypted communication between two parties. The use case is the following: Alice wants to encrypt data using a public key system and, since her local storage resources are limited, she wants to store the encrypted data in a cloud for later use. At the time of retrieval, she downloads the encrypted data from the cloud and decrypts it. Since the cipher system uses a public key and a secret key, to encrypt data and store in a cloud without compromising the confidentiality, Alice has to encrypt data with her own public key. Then, Alice is the only one that can decipher the data at the time of retrieval since she is the only one that possesses both the secret key and the public key. In the case of secret communication between Alice and Bob, where Alice sends data to Bob, she has to pick Bob's public key and encrypt data using that key. Then Bob is the only one that can decrypt since he is the only one that possesses the secret key.

3.2 Global description of the proposed hybrid UE-PRE system

We now describe our proposed hybrid UE-PRE data sharing scheme. It consists of a symmetric cipher component and the asymmetric cipher component. These components are used by both the PRE part of the scheme and the UE part.

3.2.1 The PRE part of the hybrid UE-PRE scheme

We first explain the operation of the PRE part, which is used to share data between parties in the information system. Suppose Alice wants to store encrypted data m in the cloud and she wants to delegate access to the data to other participants at some later moment upon request. To achieve this, she generates the encryption key k_{eA} for the symmetric cipher component of the scheme and encrypts the data (the plaintext m) with it. The resulting ciphertext C_{eA} is uploaded to the cloud. Alice also generates the public encryption key pk_A and the secret encryption key sk_A for the asymmetric cipher component of the scheme and encrypts the key k_{eA} using the pair (pk_A, sk_A). The resulting ciphertext C_{kA} is sent to the PRE server (in the cloud). Alice is the only one that can decrypt C_{kA} since only Alice possesses the secret part of the key sk_A. Both the public key system ciphertext and the secret key system ciphertext can be uploaded to a semi-trusted cloud without compromising the confidentiality since the cloud cannot decrypt the symmetric key k_{eA} encrypted by Alice by means of the public key cipher without knowing her secret key, and consequently the cloud cannot decrypt the ciphertext C_{eA}.

When Alice wants to share her data m with the approved recipient Bob, she first has to permit Bob decipher the symmetric cipher key k_{eA}. To do this, she retrieves Bob's public key, pk_B and generates the re-encryption key Rk_{AB}. The PRE server retrieves the re-encryption key Rk_{AB} and converts the ciphertext C_{kA} from that created under Alice's public key pk_A to the ciphertext created using Bob's public key pk_B. The resulting ciphertext C_{kAB} is made available to

Bob. He decrypts C_{kAB} with the asymmetric cipher using his secret key sk_B and thus gets access to the key k_{eA}. The cloud makes available the ciphertext C_{eA} to Bob, which he decrypts with the symmetric cipher component of the scheme using the symmetric cipher key k_{eA}. The data sharing process between Alice and Bob using the PRE component of our proposed scheme is schematically presented in Fig. 1.

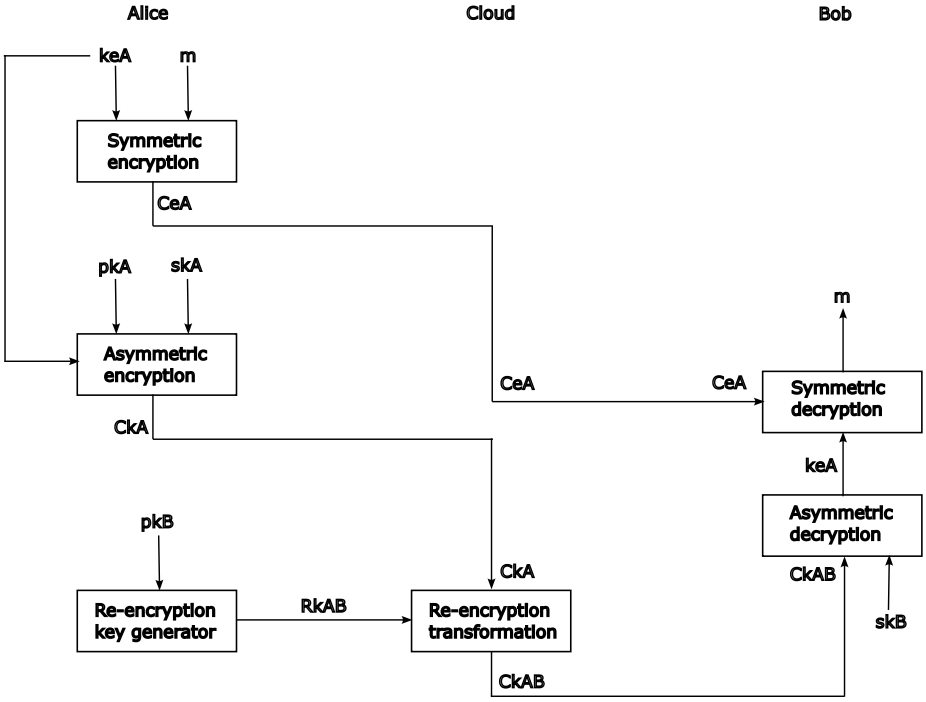

Fig. 1. Alice uses PRE to share data m with Bob.

We illustrate the operation of the PRE part of the proposed scheme on an example, using the public key cipher system described in Section 2.3. In the case of using the multiplicative variant of the cipher, Alice's secret key is sk_A, the computation is performed modulo the prime p and g is a generator of the multiplicative group $Z_p^{\times}$. The hash function used in the PRE process is H. Alice's public key is $pk_A = g^{sk_A} \bmod p$.

To store the piece of data k_{eA} in the encrypted form in the cloud, Alice encrypts it using her own public (and secret) key. She computes the ciphertext $C_{kA} = (C_{kA1}, C_{kA2})$, where $C_{kA1} = pk_A = g^{sk_A} \bmod p$ and $C_{kA2} = k_{eA}(pk_A)^{sk_A} = k_{eA}(g^{sk_A})^{sk_A} \bmod p = k_{eA}g^{(sk_A^2)} \bmod p$.

Suppose that Bob is to get access to Alice's piece of data (the encryption key for the symmetric cipher) k_{eA} stored in the cloud. The proxy converts the

ciphertext $C_{kA} = (C_{kA1}, C_{kA2})$ that only Alice can decrypt to the ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$ that Bob can decrypt. We know from Section 2.3 that $C_{kAB2} = C_{kA2}$. It is C_{kA1} that must be transformed. The re-encryption key that Alice computes and sends to the proxy re-encryption server is

$$Rk_{AB} = sk_A \left(H\left(pk_B^{sk_A} \right) \right)^{-1} \bmod p = sk_A \left(H\left(\left(g^{sk_B} \right)^{sk_A} \right) \right)^{-1} \bmod p = sk_A \left(H\left(g^{sk_A sk_B} \right) \right)^{-1} \bmod p. \quad (7)$$

The proxy applies the re-encryption transformation

$$C_{kAB1} = pk_A^{RkAB} \bmod p = \left(g^{sk_A} \right)^{RkAB} \bmod p = \left(g^{sk_A} \right)^{sk_A \left(H\left(g^{sk_A sk_B} \right) \right)^{-1}} \bmod p = \left(g^{\left(sk_A^2 \right)} \right)^{\left(H\left(g^{sk_A sk_B} \right) \right)^{-1}}. \quad (8)$$

Thus, the proxy converted the ciphertext $C_{kA} = (C_{kA1}, C_{kA2})$ to the ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$ that Bob can retrieve from the cloud and decipher.

To decipher the ciphertext C_{eA} produced by Alice using the symmetric cipher, Bob first has to recover the key k_{eA}. To this end, he retrieves the ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$ produced by the proxy and deciphers it by computing

$$C_{kAB2} \left(C_{kAB1}^{H\left(pk_A^{sk_B} \right)} \right)^{-1} \bmod p = C_{kAB2} \left(C_{kAB1}^{H\left(\left(g^{sk_A} \right)^{sk_B} \right)} \right)^{-1} \bmod p = k_{eA} g^{\left(sk_A^2 \right)} \left(\left(g^{sk_A^2 \left(H\left(g^{sk_A sk_B} \right) \right)^{-1}} \right)^{H\left(g^{sk_A sk_B} \right)} \right)^{-1} \bmod p = k_{eA} g^{\left(sk_A^2 \right)} \left(g^{\left(sk_A^2 \right)} \right)^{-1} \bmod p = k_{eA}. \quad (9)$$

Then Bob can decipher C_{eA} using the symmetric cipher and the key k_{eA} and recover the message m.

In the case of using the additive variant of the cipher, Alice's secret key is also sk_A, the computation is performed modulo the prime p and g is a point on the chosen elliptic curve E. The hash function used in the PRE process is H. Alice's public key is $pk_A = sk_A \cdot g$.

To store the piece of data k_{eA} in the encrypted form in the cloud, Alice encrypts it using her own public (and secret) key. She computes the ciphertext $C_{kA} = (C_{kA1}, C_{kA2})$, where $C_{kA1} = pk_A = sk_A \cdot g$ and $C_{kA2} = P_{k_{eA}} + sk_A pk_A = P_{k_{eA}} + sk_A^2 \cdot g$, where $P_{k_{eA}}$ is the point on the elliptic curve E that represents the plaintext k_{eA}, see Section 2.3.

If Alice wants to authorize Bob to get access to Alice's piece of data (the encryption key for the symmetric cipher) k_{eA} stored in the cloud, she gives the instruction to the proxy to convert the ciphertext $C_{kA} = (C_{kA1}, C_{kA2})$ that

only Alice can decrypt to the ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$ that Bob can decrypt. For the proxy to be able to perform this conversion, Alice sends the re-encryption key Rk_{AB} to it. We know from Section 2.3 that $C_{kAB2} = C_{kA2}$. It is C_{kA1} that must be transformed. The re-encryption key that Alice computes and sends to the proxy re-encryption server is

$$Rk_{AB} = sk_A(H(sk_A pk_B))^{-1} = sk_A(H(sk_A sk_B \cdot g))^{-1}. \tag{10}$$

The proxy applies the re-encryption transformation

$$C_{kAB1} = pk_A Rk_{AB} = sk_A^2(H(sk_A sk_B \cdot g))^{-1} \cdot g. \tag{11}$$

Thus, the proxy converted the ciphertext $C_{kA} = (C_{kA1}, C_{kA2})$ to the ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$ that Bob can retrieve from the cloud and decipher.

To decipher the ciphertext C_{eA} produced by Alice using the symmetric cipher, Bob has to recover the key k_{eA} first. To this end, he retrieves the ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$ produced by the proxy and deciphers it by computing

$$\begin{aligned} C_{kAB2} - H(sk_B pk_A)C_{kAB1} = \\ P_{k_{eA}} + sk_A^2 \cdot g - H(sk_A sk_B \cdot g)sk_A^2(H(sk_A sk_B \cdot g))^{-1} \cdot g \\ = P_{k_{eA}}. \end{aligned} \tag{12}$$

Bob is now capable of deciphering C_{eA} using the symmetric cipher and the key k_{eA} and recover the message m.

3.2.2 The UE part of the hybrid UE-PRE scheme

We now describe the UE part of our proposed hybrid UE-PRE scheme. It is activated every time epoch key rotation is needed. A typical use case is a situation when the current epoch key is compromised. Another use case is regular epoch key rotation - it is a preventive measure against epoch key compromise. In our explanation of the UE segment of the hybrid UE-PRE scheme, we continue considering Alice the owner of the data. The UE transformation is performed in the cloud. The owner of the data, Alice, generates the new epoch key $k_{(e+1)A}$ and this new epoch key and the old epoch key k_{eA} serve as inputs to the token generator that produces the token Δ_{e+1}, which is then sent to the cloud. The cloud server updates the asymmetric cipher ciphertext C_{keA} to the new version $C_{k(e+1)A}$ by means of the token Δ_{e+1} received from Alice. What remains is to grant access to the new epoch key $k_{(e+1)A}$ to the intended recipient. Here we may have two cases: key compromise and regular key rotation.

In the case of old epoch key compromise, the user Bob, at whose site this compromise had happened is to lose access rights to Alice's data. This is achieved automatically after the ciphertext C_{keA} has been updated to $C_{k(e+1)A}$ if PRE server is not run to re-encrypt the old ciphertext $C_{kAB} = (C_{kAB1}, C_{kAB2})$, see Fig. 2.

If Alice wants to continue granting permission to the symmetric cipher encryption key to the same recipient Bob, the PRE server has to be activated and

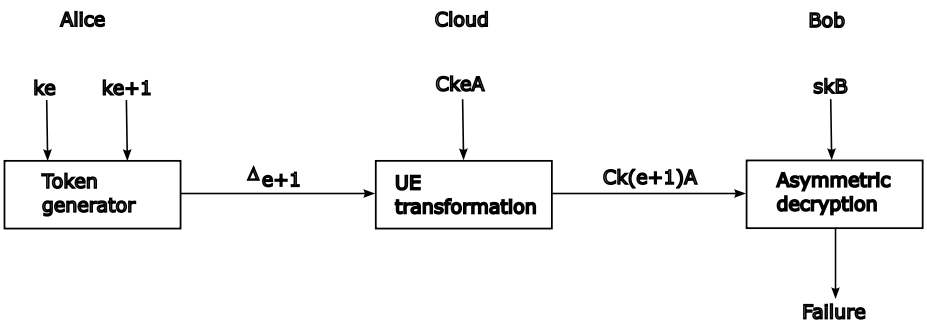

Fig. 2. Alice uses UE to revoke access grant previously given to Bob.

the new asymmetric cipher ciphertext $C'_{kAB} = (C'_{kAB1}, C'_{kAB2})$ must be produced. If Alice does not want to grant access to Bob after the old epoch key had been compromised, she may grant access to another user, Charlie, by activating the PRE server after the UE process had finished. Then Charlie can decrypt the asymmetric cipher ciphertext $C_{kAC} = (C_{kAC1}, C_{kAC2})$ and reconstruct the new epoch key $k_{(e+1)A}$. Consequently, Charlie gets access to the data m by decrypting $C_{k(e+1)A}$.

In the case of regular key rotation, nobody is to lose access to Alice's data. To this end, after UE process completes, the PRE process must be activated to produce the new version of the ciphertext $C'_{kAB} = (C'_{kAB1}, C'_{kAB2})$. Then the old user, Bob, can decrypt the symmetric cipher ciphertext $C_{(e+1)A}$ and get access to data again.

We illustrate the operation of the UE procedure described above by using the SHINE UE scheme introduced in Section 2.2. For simplicity, we assume that the piece of data m owned by Alice is short enough and the SHINE0 version is capable of processing it. Since any implementation of SHINE in general and SHINE0 in particular consists of two ciphers - the symmetric one (the instantiation of an ideal (random) permutation, see [4]) and the asymmetric one, two keys are needed for its practical operation - the epoch key k_{eA} and the symmetric cipher key k_s. The epoch key k_{eA} in SHINE is not used for symmetric cipher encryption. It is used for updating the ciphertext in the UE procedure. This does not challenge the general UE concept described above. The difference of the SHINE implementation compared to it is only technical, as explained below. The UE token in SHINE0 is $\Delta_{e+1} = k_{(e+1)A} \cdot k_{eA}^{-1} \bmod p$. Note that the token is a scalar. The elliptic curve E and the modulus p are public parameters. Both $C_{(e+1)A}$ and C_{eA} are produced using the AES256 block cipher in our practical implementation. Given the old epoch key k_{eA}, a point c on the elliptic curve E needed for the UE process is computed in the following way: the ciphertext $C_{eA} = AES256(\mathrm{CBC}, k_s, m, iv)$ is produced first (AES256 is run in the CBC mode using the initialization vector iv and padding is performed, if necessary). The obtained ciphertext C_{eA} is then embedded in the coordinate x of a point on the elliptic curve E. Thus, a point (x, y) is obtained on E. Finally, the point c

is obtained as $c = k_{eA} \cdot (x, y)$ and stored in the cloud. The UE procedure generates the updated point c' on the curve E that corresponds to the new epoch key $k_{(e+1)A}$. It is obtained as $c' = \Delta_{e+1} \cdot c = k_{(e+1)A} \cdot k_{eA}^{-1} \cdot k_{eA} \cdot (x, y) = k_{(e+1)A} \cdot (x, y)$.

To decrypt the symmetric cipher, any recipient must possess the new epoch key $k_{(e+1)A}$. It is provided through the PRE scheme described above. Thus, the combination of the PRE scheme and the UE scheme enables efficient key rotation, regular or occasional.

3.3 Implementation and experimental results

The (pedagogical) implementation of the UE component of the hybrid PRE-UE scheme proposed in this paper was done in Python. The additive variant of the algorithms was chosen, which required implementation of elliptic curve-related routines. Our own implementation of these routines was done, although there are 3rd party software packages that could be more efficient. The elliptic curve chosen as a global parameter was sec256k1: $y^2 = x^3 + 7 \bmod p$, where the prime modulus $p = 2^{256} - 2^{32} - 977$. Embedding of the AES256 ciphertext in the UE part of the scheme was implemented in a trial-and-error manner, which implied a random selection of a point given one fixed coordinate and then check whether such a point belonged to the curve. This did not reduce the efficiency of the implementation significantly. For the PRE part of the hybrid scheme, a ready-made implementation of the full Umbral system available in [7] was used.

The UE part of the implemented hybrid PRE-UE scheme consumes most of resources (time, memory) and this consumption depends on the length of the plaintext. The time consumption of the PRE part of the scheme is not affected by the length of the plaintext and represents a small fraction of the total time consumption. For example, in our experiment on an ordinary office machine with 16 MB of RAM, when OCBSHINE variant of the SHINE scheme was used, the time consumption varied between 12.19 seconds for a 10 kB of plaintext and 132 seconds for a 100 kB of plaintext, which indicates linear dependence of the UE component time consumption on the length of the plaintext. Unlike the UE part of the scheme, the time consumption of the PRE part of the scheme is much smaller than that of the UE part and independent of the length of the plaintext. More details on the experiments with the hybrid scheme are available in [1].

4 Security of the proposed UE-PRE sscheme

Any hybrid UE-PRE scheme must satisfy a set of security criteria defined in advance. Each individual component of the scheme (UE, PRE) must satisfy these criteria as well. The set of security criteria to satisfy by these schemes has evolved over time. For example, the authors of the SHINE family of UE schemes also updated the set of security criteria used previously for UE schemes (see [4]).

The following security criteria have been used for PRE schemes: unidirectionality, collusion resistance, non-interactivity, correctness, confidentiality, key privacy, security against Chosen-Plaintext Attacks (CPA) and Chosen-Ciphertext

Attacks (CCA). For UE schemes, the set of security criteria that has been developed includes correctness, confidentiality, integrity of ciphertexts (INT-CTXT), indistinguishability of output(s) from random data (IND), key privacy, as well as security against CPA and CCA attacks. More details on security criteria can be found in [8] for PRE schemes and [4] for UE schemes.

Different use cases and threat models determine the security criteria that UE-PRE schemes must satisfy. In this paper, we provide the security analysis of the instantiation of the UE-PRE scheme proposed in previous sections. It represents the concretization of the security analysis presented in [1]. The instantiation of the general UE-PRE scheme is the following: as the PRE component of the scheme we use the threshold scheme Umbral [7] whereas as the UE component we use the SHINE scheme [4][5].

Regarding the PRE component of our proposed scheme, [8] provides security analysis of similar schemes, whose conclusions apply directly to security of Umbral. In addition, in [7] where the Umbral scheme is described, a short security analysis is given as well. Umbral satisfies the criteria for PRE schemes listed above. It is unidirectional, non-interactive, and CCA-secure. As for the UE scheme, the SHINE scheme used in our proposal, in particular the OCBSHINE variant, satisfies the IND-CPA, INT-CCA, and INT-CTXT criteria [4].

The greatest advantage of our proposed UE-PRE scheme over other schemes regarding security is the fact that all the keys, including the epoch keys, re-encryption keys, and update tokens are generated at the user's premises, without intervention by cloud service (and other) providers. This is particularly convenient in healthcare information systems, where data confidentiality and integrity must be provided by the users and any third party intervention is prohibited by the law. This in turn increases the users' responsibility for proper generation and key management since the flaws in these procedures jeopardize the main security requirements satisfaction.

5 Conclusion

In this paper, we have proposed a hybrid Updatable Encryption - Proxy Re-Encryption scheme (UE-PRE), whose application field is general, but it is particularly sutable for use in a healthcare information system data protection due to possibility of the user(s) to generate and distribute all the relevant security elements (in particular, the keys) independent from cloud service providers. The concrete analyzed instantiation of the proposed scheme combines the Umbral threshold PRE scheme [7] and the SHINE UE scheme [4][5]. A detailed description of the hybrid UE-PRE scheme with possible instantiations in the additive and the multiplicative forms is given, as well as the security and efficiency analysis of the Umbral-OCBSHINE scheme instantiation. It is shown that the PRE component of the scheme satisfies the unidirectionality, non-interactivity, and CCA-security criteria whereas the UE component of the scheme satisfies the IND-CPA, INT-CCA, and INT-CTXT criteria. Regarding efficiency, the time consumption of the scheme is primarily determined by the time consumption of

the UE component. It linearly depends on the size of the plaintext. Overall, the hybrid UE-PRE scheme proposed in this paper is suitable for data protection and access delegation in a healthcare information system. A quantum-safe variant of the proposed UE-PRE scheme may be needed, which is the goal of the future work.

References

1. Afiouni A.: Privacy preserving data sharing with partially-trusted cloud services. Master thesis, Norwegian University of Science and Technology (NTNU), Gjøvik (2023).
2. Blaze M., Bleumer G., and Strauss M.: Divertible protocols and atomic proxy cryptography. In: Proceedings of EUROCRYPT 1998, LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998).
3. Boneh D., Lewi K., Montgomery H., and Raghunathan A.: Key homomorphic PRFs and their applications. In: Proceedings of Crypto 2013, LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013).
4. Boyd C., Evans G., Gjøsteen K., and Jiang Y., Fast and secure updatable encryption. In: Proceedings of Crypto 2020, LNCS, vol. 12170, pp. 464–493. Springer, Heidelberg (2020).
5. Jiang Y.: Cryptographic tools for cloud security. PhD thesis, Norwegian University of Science and Technology (NTNU), Trondheim (2021)
6. Lehmann A., and Tackmann B., Updatable encryption with post-compromise security. In: Proceedings of Eurocrypt 2018, part III, LNCS, vol. 10822, pp. 685–716. Springer, Heidelberg (2018).
7. Nuñez D.: Umbral: A threshold proxy re-encryption scheme. `https://github.com/nucypher/pyUmbral`, last accessed 2025/11/03.
8. Nuñez D., Agudo I., and Lopez J., Proxy Re-Encryption: Analysis of constructions and its application to secure access delegation. Journal of Network and Computer Applications, **87**(2017), 193–209 (2017).
9. Trappe W., and Washington L.C., Introduction to Cryptography with Coding Theory, 2nd edn., Pearson Prentice Hall, Upper Saddle River, NJ, USA (2006).