Multi-recipient Encryption in Heterogeneous Setting

Puwen Wei[1,⋆], Yuliang Zheng[2,⋆], and Wei Wang[1,⋆]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
{pwei,weiwangsdu}@sdu.edu.cn
[2] Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
yzheng@uncc.edu

Abstract. This paper presents an efficient method for securely broadcasting a message to multiple recipients in a heterogeneous environment where each recipient is allowed to choose his or her preferred secure encryption scheme independently of other recipients' choices. Previous work pertinent to this direction of research, namely multi-recipient encryption scheme (MRES), generally requires all recipients adhere to the same public key encapsulation mechanism (KEM) for the sake of delivering promised savings in computation and bandwidth via randomness reuse. Our work eliminates the requirement of using the same KEM by all recipients, whereby removing a practical barrier to the adoption of MRES in real world applications. A second advantage is the method's capability to cope with a dynamically changing group of recipients where old recipients may be deleted and new recipients may be added, while ensuring the security of messages shared in future. Additional features of our method include decryption by a sender, anonymity of recipients and stateful key encapsulation which significantly reduces computational costs for securely transmitting or sharing new messages. All these attributes would be useful in building applications for secure data sharing in a cloud computing environment.

Keywords: multi-recipient, public key encryption, sender recovery, KEM, DEM.

1 Introduction

The primary goal of multi-recipient encryption is for a sender to transmit encrypted messages to multiple recipients in an efficient manner in terms of computation and bandwidth. Although in its broadest form multi-recipient encryption could be defined to allow multiple messages intended for different recipients to be included in a single ciphertext, we focus our attention on the case of most relevance to practical applications, namely single message multi-recipient encryption whereby the same message is transmitted to all intended recipients.

⋆ Corresponding Authors.

X. Huang and J. Zhou (Eds.): ISPEC 2014, LNCS 8434, pp. 462–480, 2014.
© Springer International Publishing Switzerland 2014

A simple construction for a multi-recipient encryption scheme (MRES) could follow an approach specified in S/MIME [1]: generate a symmetric key k for a data encapsulation mechanism (DEM), encrypt k for intended recipients followed by encrypting the message using the DEM under k. It is however unclear whether the ad hoc construction admits provable security under reasonable assumptions. A further issue is with efficiency in terms of computational and bandwidth requirements when a full-fledged public key encryption scheme is employed as a key encapsulation mechanism (KEM) for each recipient.

In pursuit of a more efficient MRES that admits a rigorous security analysis, Kurosawa [2] proposed the first MRES with a shortened ciphertext. Smart [3] introduced the notion of mKEM, which can be viewed as an efficient key encapsulation technique for multiple recipients. Bellare et al. [4][5] systematically studied the technique of randomness reuse and provided several generic and efficient constructions for MRES. Specifically, they introduced a so-called reproducibility theorem [5] to determine whether a standard encryption scheme permits secure randomness reuse. This was followed by Barbosa and Farshim [6] who proposed the notion of weak reproducibility which enabled them to construct a wider class of efficient (single message) MRESs. Hiwatari et al. [7] considered yet another approach by way of examining the behavior of a simulator in a security proof. We also note an earlier effort to design a multi-recipient signcryption scheme in which a sender and all recipients share the same group parameters [8]. While (public key) broadcast encryption [9] and multi-recipient encryption share a similar goal, researchers differentiate between these two types of security techniques by noting how public/secret key pairs for recipients are generated. Generally speaking, with broadcast encryption all recipients' public/secret key pairs are generated centrally by a sender or by a trusted key generation center, whereas with multi-recipient encryption each recipient is allowed to generate his or her own public/seciret key pair. In theory, multi-recipient encryption offers more flexibility for recipients than does broadcast encryption. In practice though, the flexibility is curtailed by previously proposed MRESs which generally require that all recipients use the same KEM or at the least share some easy-to-handle mathematical structures such as the same cyclic group. The main reason for the necessity of imposing limitations on recipients' choices of public/secret key pairs is due to the fact that these early MRESs employ randomness reuse as a tool to achieve savings in computation and bandwidth. We believe that the limitation as explained above would be a barrier to the adoption of MRES in some real world applications where each recipient may prefer to choose his or her own public/secret key pairs due to either organizational policy requirements or purely personal preferences. As a concrete example, one can imagine that in a cloud computing environment, recipient 1 might have to use RSA in order to comply with his company's security policy, recipient 2 might prefer to use elliptic curve cryptography (ECC), recipient 3 might be quite happy to use the latest standard technique for lattice based cryptography, and recipient 4 might want to stick to her choice of group parameters for ECC that are different from those for recipient 2.

1.1 Our Contributions

The goal of this paper is to address the above problem by designing an efficient MRES with provable security for a heterogeneous environment where a recipient is free to choose a secure KEM of his or her liking. Further, our scheme can be optimized in a stateful manner. That is, state information that includes randomness and other data used to generate a ciphertext can be reused to improve the efficiency of the multi-recipient encryption scheme. More interestingly, the scheme can be securely extended to a more general, heterogeneous setting where some of the recipients may not possess public/secret key pairs but instead share with the sender some symmetric keys which may be established in an out-of-band manner or as an outcome of a previous secure communication session. An added feature of our scheme is decryption of a ciphertext by the sender which was first advocated in [10,11]. Such a property would be useful in practice when the sender wishes to recover a message he or she sent earlier from the ciphertext only without access to the original plaintext or any of the recipients' secret decryption keys.

In some applications it may be important to protect the identity of a recipient. We show how to adapt our scheme in such a way that other than the sender and an intended recipient herself, no-one else can determine whether the recipient is an intended one or not. For the security model of anonymity, we mostly follow the idea of [12] on the definition of anonymity for broadcast encryption. However, modifications must be made to reflect a major difference: unlike broadcast encryption, the public/secret key pairs of our MRES are generated by different recipients instead of a key generation center. So it provides a potential opportunity for an adversary to choose public keys of his liking and compromise anonymity. We define a security model for anonymity in a heterogeneous environment and prove that our adapted scheme satisfies the stringent requirement of anonymity. All these attributes make our scheme a useful tool for secure data sharing in a heterogeneous environment, e.g., encrypted file system in a cloud computing environment.

1.2 Overview of Main Techniques

Map t Strings to One String. As mentioned above, researchers constructed efficient MRES by employing a key encapsulation mechanism (KEM) in lieu of full-fledged public key encryption. However, previous efficient schemes, e.g., [2,3,4,5,6,7] are not applicable in a heterogeneous public key setting. The main difficulty of employing KEMs is how to map ephemeral keys that are independently generated with different recipients' KEMs, to the same string that is used as a key in a symmetric data encapsulation mechanism (DEM). We overcome this difficulty with the help of a universal hash function family with collision accessibility. Cryptographic applications of a universal hash function family with collision accessibility, denoted by UH_{CA}, were discussed extensively in early work including [13] and explored more recently in [11]. Simply speaking, a

t-universal hash function family with collision accessibility is one that, given t initial strings, the hash values of them can be made to collide with one another. By using UH_{CA}, [11] provides an efficient method for public key encryption with sender recovery. Extending the work of [11], we find that ephemeral keys for different recipients' KEMs can be mapped into the same symmetric key for a DEM thanks to the collision accessibility of UH_{CA}. However, it turns out that this straightforward extension has its drawback in that it is not quite scalable when the number of recipients increases. To address this drawback we consider alternative structures to a "flat" t-universal hash function family. An example of such alternatives is derived from the Merkle-Damgard hash chain. Another example takes the form of a hash tree. These alternative structures enable us to use a 2-universal hash function family only to realize in an efficient and scalable manner the "collision" of t ephemeral keys and more importantly, to work out a security proof using an idea similar to length-extension attack.

Secure State Reuse. Stateful public key cryptosystem introduced in [14] permits the reuse of state information across different encryptions, resulting in a more efficient technique. The authors of [14] demonstrated stateful versions of the DHIES and Kurosawa-Desmedt schemes which requires one exponentiation only to encrypt. A potential problem with the randomness reuse, when applied to a stateful (single message) MRES as advocated in [14,5], lies in the fact that the symmetric key for a subsequent DEM is fixed. This issue of fixed DEM keys can be seen clearly when one applies the above randomness reuse technique to the construction of stateful MRESs from concrete MRESs in [3,6,7]. With a fixed DEM key, the resultant stateful MRESs are exposed to potential attacks when a recipient is removed from the group of intended recipients, as the removed recipient would still be able to decrypt future ciphertexts as long as the sender does not reset the state. While resetting the state could prevent such attacks, it would severely decrease the efficiency of a stateful MRES. To better understand it we note that all previous MRESs work more or less like this: a symmetric key (or the seed of the symmetric key) k is chosen at random and then "mapped"[1] to t ciphertexts for KEMs, which correspond exactly to t recipients. Since the state depends on k, it has to be changed once k is changed.

The MRES we propose in this paper can be viewed as the exact opposite of the above: t random ephemeral keys and their corresponding KEM ciphertexts, say $(k_1, c_1), (k_2, c_2), ..., (k_t, c_t)$, are generated for all t recipients independently; they are then "mapped" to the same DEM key k by UH_{CA}. One can see that if $(k_1, c_1), (k_2, c_2), ..., (k_t, c_t)$ are cached the first time when they are computed, they can be reused in subsequent encryptions for the same recipients. When a recipient, say U_1, is removed from the group of recipients, the sender can still reuse the remaining $(k_2, c_2), ..., (k_t, c_t)$ to generate a new DEM key k which will be no longer accessible to the removed recipient. An example is given in Table 1, which shows that when recipient U_1 is removed, the subsequent computational costs for the current recipients U_2,U_3,...,U_t in our MRES are less than that of

[1] Indeed, k is encrypted as a symmetric key for all recipients or reused as randomness in a ciphertext for a KEM.

the traditional method, e.g, the method specified in S/MIME. If on the other hand a new recipient U_{t+1} is added to the group, the sender needs to generate a new pair (k_{t+1}, c_{t+1}) for the new recipient only while keeping existing pairs $(k_1, c_1), (k_2, c_2), ..., (k_t, c_t)$ unchanged. These $t + 1$ pairs can then be used to generate a new DEM key k. By recycling (k_i, c_i)s, computational costs for each recipient can be reduced to merely two multiplications and one multiplicative inversion in $GF(2^{256})$.

Table 1. Efficiency comparison of encryption of our MRES with traditional method. RSA 2048, ElGamal 1024, ElGamal 2048,..., ECIES 256 are the public key encryption schemes used by recipient U_2,U_3,...,U_t, respectively. Inv-$GF(2^{256})$ denotes the multiplicative inversion in $GF(2^{256})$. Exp-xxx and ECMul-xxx denotes the exponentiation and the elliptic curve point multiplication on the corresponding groups, respectively.

	U_2 RSA 2048	U_3 ElGamal 1024	U_4 ElGamal 2048	...	U_t ECIES 256
Our MRES	1 Inv-$GF(2^{256})$	1 Inv-$GF(2^{256})$	1 Inv-$GF(2^{256})$	...	1 Inv-$GF(2^{256})$
Traditional method	1 Exp-2048	2 Exp-1024	2 Exp-2048	...	2 ECMul-256

2 Basic Definitions

Notation. If x is a string, then $|x|$ is the length of x. $x||y$ denotes the concatenation of x and y. If X is a set, then $x \xleftarrow{R} X$ denotes the operation of picking an element x of X uniformly at random. If A is an algorithm, then $z \leftarrow A(x, y, ...)$ denotes the operation of running A on input $(x, y, ...)$ with its output being saved in z.

Key Encapsulation Mechanism. A key encapsulation mechanism (KEM) consists of three algorithms $KEM.Gen$, $KEM.Enc$ and $KEM.Dec$. $KEM.Gen$ takes as input a security parameter 1^λ and outputs a public/secret key pair (pk, sk). $KEM.Enc$ takes as input a public key pk and outputs a ciphertext c and an ephemeral key k. Let $\{0,1\}^{\lambda_k}$ denote the KEM's key space, where λ_k is a polynomial in security parameter λ. $KEM.Dec$ takes as input a secret key sk and a ciphertext c, and outputs the ephemeral key k or a special failure symbol "$\perp$".

A standard notion of security for a KEM is indistinguishability security against adaptive chosen ciphertext attack, or IND-CCA2 security. This notion is defined by a two-party game played between a challenger and an adversary A. Throughout the game, the adversary can query a KEM oracle $\mathcal{O}_{KEM}$, which upon a decapsulation query c returns $KEM.Dec(sk, c)$.

$\mathbf{Game}_{KEM}^{IND-CCA2}$:
$(pk, sk) \leftarrow KEM.Gen(1^{\lambda})$
$(c^*, k_0^*) \leftarrow KEM.Enc(pk)$, $k_1^* \overset{R}{\leftarrow} \{0,1\}^{\lambda_k}$, $b \overset{R}{\leftarrow} \{0,1\}$
$b' \leftarrow A^{\mathcal{O}_{KEM}}(pk, c^*, k_b^*)$

Note that after receiving (c^*, k_b^*), the adversary can query $\mathcal{O}_{KEM}$ with any ciphertext c with the restriction that $c \neq c^*$. We say that A wins the game if $b = b'$. A KEM is said to be ϵ_{KEM}-IND-CCA2 secure if $|Pr[b = b'] - 1/2| \leq \epsilon_{KEM}$ for any probabilistic polynomial time (PPT) adversary, where ϵ_{KEM} is a negligible function in security parameter λ.

Data Encapsulation Mechanism. A data encapsulation mechanism (DEM) is composed of two algorithms, these being an encryption algorithm $DEM.Enc$ and a decryption algorithm $DEM.Dec$. $DEM.Enc$ takes as input a symmetric key k and a plaintext m and outputs a ciphertext c. The key space for the algorithm is denoted by $\{0,1\}^{\lambda_{Enc}}$, where λ_{Enc} is a polynomial in the security parameter λ. $DEM.Dec$ takes as input a symmetric key k and a ciphertext c and outputs m.

For the security of DEM, we adopt the notion of one-time symmetric-key encryption against passive attack (IND-OPA) [15] defined by the following game involving an adversary A:

$\mathbf{Game}_{DEM}^{IND-OPA}$:
A chooses a pair of plaintexts (m_0, m_1), where $|m_0| = |m_1|$
$k \overset{R}{\leftarrow} \{0,1\}^{\lambda_{Enc}}$, $c^* \leftarrow DEM.Enc(k, m_b)$, $b \overset{R}{\leftarrow} \{0,1\}$
$b' \leftarrow A(c^*)$

We say that the adversary A wins the game if $b = b'$. An one-time symmetric-key encryption is said to be ϵ_{DEM}-IND-OPA secure if $|Pr[b = b'] - 1/2| \leq \epsilon_{DEM}$ for any PPT adversary, where ϵ_{DEM} is a negligible function in λ.

One-Time Signature. An one-time signature $Sig = (Gen_{Sig}, Sign, Vrfy)$ requires a signer generate new verification/signing keys each time when he signs a message. It consists of three algorithms specified below. Gen_{Sig} takes as input a security parameter 1^{λ} and outputs a verification/signing key pair (vk, sk_{Sig}). $Sign$ takes as input sk_{Sig} and a message m and outputs a signature σ. $Vrfy$ takes as input vk, m and σ and outputs "1" if σ is valid; otherwise, "0".

The security of Sig required in this paper is the strong unforgeability under chosen message attack (SU-CMA), which is defined by the following game.

$\mathbf{Game}_{Sig}^{SU-CMA}$:
$(vk, sk_{Sig}) \leftarrow Gen_{Sig}(1^{\lambda})$
$(m^*, \sigma^*) \leftarrow A^{\mathcal{O}_{Sign}}(vk)$

Notice that A can make at most one query m to the signing oracle $\mathcal{O}_{Sign}$, which returns the corresponding signature $\sigma = Sign(sk_{Sig}, m)$. We say the adversary wins the game if $Vrfy(vk, m^*, \sigma^*) = 1$ and $(m^*, \sigma^*) \neq (m, \sigma)$. Sig is said to be

ϵ_{Sig}-SU-CMA secure if $\Pr[A \text{ wins}] \leq \epsilon_{Sig}$ for any PPT adversary, where ϵ_{Sig} is a negligible function in λ.

A Useful Lemma

Lemma 1. *[16] Let A_1, A_2, B be events defined over a probability space such that $\Pr[A_1 \bigcap \overline{B}] = \Pr[A_2 \bigcap \overline{B}]$. Then we have $|\Pr[A_1] - \Pr[A_2]| \leq \Pr[B]$.*

3 Security Model

In this section, we define a multi-recipient encryption scheme with sender recovery (MRES-SR) together with a security model for MRES-SR.

Let U_i denote the i-th recipient, for $1 \leq i \leq t$. A MRES-SR is a tuple of algorithms $(Gen_{MR}, Enc_{MR}, Dec_{MR}, Rec_{MR})$ defined as follows:

- A probabilistic key generation algorithm Gen_{MR}, which in turn consists of sub-algorithms $Gen_{MR}.S$ and $Gen_{MR}.U_i$, for all $1 \leq i \leq t$.
  - $Gen_{MR}.S$ takes as input the security parameter 1^{λ} and outputs a sender's secret recovery key sk_{rcv}.
  - $Gen_{MR}.U_i$ takes as input the security parameter $1^{\lambda^{(i)}}$ and outputs a recipient U_i's public/secret key pair (pk_i, sk_i).
- An encryption algorithm Enc_{MR}, which takes as input the sender's secret recovery key sk_{rcv}, the recipients' public keys $\mathbf{pk} = (pk_1, pk_2, ..., pk_t)$ and a plaintext m, outputs a ciphertext c_{MR}.
- A decryption algorithm Dec_{MR} (of U_i), which takes as input the recipient's secret key sk_i and a ciphertext c_{MR}, outputs the corresponding plaintext m or the error symbol "$\perp$".
- A recovery algorithm Rec_{MR}, which takes as input sk_{rcv} and c_{MR}, outputs the corresponding plaintext m or the error symbol "$\perp$".

Remark. Since recipients may use different types of public key encryption algorithms, descriptions of Enc_{MR} (or Dec_{MR}) are dependent on specific recipients' algorithms. For simplicity, we use the same notation Enc_{MR} (Dec_{MR}) for all recipients.

IND-CCA2 Security of MRES-SR. The security of MRES-SR is defined by an IND-CCA2 game $\mathbf{Game}_{MR}^{IND-CCA2}$ played between an adversary A and a challenger. During the game, the adversary A has access to three oracles: (1) an encryption oracle $\mathcal{O}_{Enc}$, which upon an encryption query $q_{Enc} = (\mathbf{pk}', m)$ returns a ciphertext $c_{MR} = Enc_{MR}(sk_{rcv}, \mathbf{pk}', m)$. (2) a decryption oracle $\mathcal{O}_{Dec\ i}$ for recipient U_i, which upon a decryption query $q_{Dec} = c_{MR}$ returns $Dec_{MR}(sk_i, c_{MR})$. (3) a recovery oracle $\mathcal{O}_{Rec}$, which upon a recovery query $q_{Rec} = c_{MR}$ returns $Rec_{MR}(sk_{rcv}, c_{MR})$. $\mathbf{Game}_{MR}^{IND-CCA2}$ proceeds as follows.

- The challenger runs Gen_{MR} to generate a target sender's secret recovery key sk_{rcv} and target recipient U_i's public/secret key pair (pk_i, sk_i), for $1 \leq i \leq t$. Let $\mathbf{pk} = (pk_1, pk_2, ..., pk_t)$.

- The adversary A generates a pair of plaintexts (m_0, m_1) such that $|m_0| = |m_1|$, and sends (m_0, m_1) to the challenger. The challenger returns a target ciphertext $c^*_{MR} \leftarrow Enc_{MR}(sk_{rcv}, \mathbf{pk}, m_b)$, where $b \overset{R}{\leftarrow} \{0, 1\}$.
- Finally, A terminates by returning a guess b'.

Notice that after the challenge phase, the adversary A can still query all the three oracles $\mathcal{O}_{Enc}$, $\mathcal{O}_{Dec\ i}$ and $\mathcal{O}_{Rec}$ with any input, provided that $q_{Dec} \neq c^*_{MR}$ and $q_{Rec} \neq c^*_{MR}$.

The above game models a sender transmitting a message to recipients U_1,...,U_t, who have public keys pk_1,...,pk_t, respectively. The adversary A can make query $q_{Enc} = (\mathbf{pk}', m)$ with any public keys $\mathbf{pk}'$, which captures the security of MRES-SR under maliciously chosen recipients' public keys. Unlike a traditional security model for MRES, queries q_{Enc} and q_{Rec} should be considered in our security model since the sender's secret key sk_{rcv} is taken as part of the inputs to Enc_{MR} and Rec_{MR}.

We say that the adversary A wins the above game if $b' = b$. MRES-SR is said to be ϵ-IND-CCA2 secure if, for any PPT adversary A, $|\Pr[A \text{ wins}] - 1/2| \leq \epsilon$.

4 Constructions

The main idea of constructing a MRES-SR is that the sender runs t different recipients' KEMs to generate t different ephemeral keys and then maps those keys to the same random string, which is used to generate a symmetric key for a DEM. During the decryption phase, a ciphertext for the DEM can be decrypted using any of t ephemeral keys. Hence, the function which maps t strings to one string plays a central role in our MRES-SR. Let us first show how to realize this function efficiently.

4.1 Map t Strings to One String

We follow ideas presented in [13] to employ universal hash function families UH_{CA} with t-collision accessibility property. Such functions can map t different strings, say $k_0, k_1, ..., k_{t-1} \in GF(2^{2\lambda})$, to the same string, say $k_{CA} \in GF(2^{\lambda})$. To construct UH_{CA} such that $UH_{CA}(k_0) = UH_{CA}(k_1) = ... = UH_{CA}(k_{t-1}) = k_{CA}$, we choose w_0,...,w_{t-1} and k_{CA} at random from $GF(2^{\lambda})$ and solve the following set of linear equations for $a_0, a_1, ..., a_{t-1} \in GF(2^{2\lambda})$:

$$\begin{cases} w_0||k_{CA} = a_0 + a_1 k_0 + a_2 k_0^2 + ... + a_{t-1} k_0^{t-1} \\ w_1||k_{CA} = a_0 + a_1 k_1 + a_2 k_1^2 + ... + a_{t-1} k_1^{t-1} \\ \qquad\qquad \vdots \\ w_{t-1}||k_{CA} = a_0 + a_1 k_{t-1} + a_2 k_{t-1}^2 + ... + a_{t-1} k_{t-1}^{t-1} \end{cases}$$

The description of UH_{CA} can be completely specified by $(a_0, a_1, ..., a_{t-1})$ and the output of $UH_{CA}(k_i)$ is k_{CA}, which is part of $a_0 + a_1 k_i + a_2 k_i^2 + ... + a_{t-1} k_i^{t-1}$.

Notice that the computational cost for solving the above equations grows as a cubic function of t, the number of recipients. This does not really scale well

for a large t, say when $t > 10,000$. To overcome this problem, we considered an iterated version of UH_{CA}, which adopts a MD like structure and uses UH_{CA} with 2-collision accessibility as a building block. (Other types of structures such as tree structure can also realize the "collision" of different strings using UH_{CA} with 2-collision accessibility.) The resulting construction is illustrated in Fig. 1, where $k_{CA,0}$, k_1,...,k_{t-1} are mapped to $k_{CA,t-1}$. In an iteration, $k_{CA,i-1}$ and k_i are mapped to the same string $k_{CA,i}$ by UH_{CA} with 2-collision accessibility. For a large t, the generation of $t-1$ UH_{CA} with 2-collision accessibility is significantly more efficient than that of UH_{CA} with t-collision accessibility.

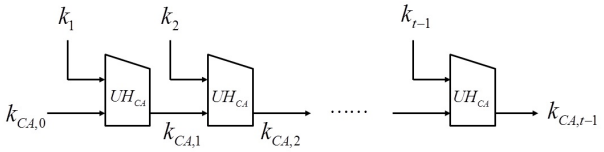

Fig. 1. UH_{CA} with MD like structure

We note that $k_{CA,i}$ the output of one UH_{CA} cannot be taken as input directly to generate the UH_{CA} of the next iteration, since the output length of UH_{CA} is shorter than its input length. Hence, additional randomness should be added to generate the next UH_{CA}. To that end, we resort to the verification key of one-time signature and hash functions, which is shown in the construction of MRES-SR.

A Useful Claim. Suppose the description $(a_0, a_1) \in \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$ of a UH_{CA} with 2-collision accessibility is generated by (1),

$$\begin{cases} w_0 || k_{CA} = a_0 + a_1 s_0 \\ w_1 || k_{CA} = a_0 + a_1 s_1 \end{cases} \tag{1}$$

and $(w_0, w_1, k_{CA}, s_0, s_1) \in \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_4} \times \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$ is said to be a valid tuple if $s_0 \neq s_1$, where $\lambda_1 = \lambda_3 + \lambda_4$. The following claim, which is proved in [11], is a very useful property of UH_{CA} with 2-collision accessibility.

Claim 1. *[11] Consider the following ensemble,*

$$W_\beta = \{(a_0^{(1)}, a_1^{(1)}, s_1^{(1)}), (a_0^{(2)}, a_1^{(2)}, s_1^{(2)}), ..., (a_0^{(N-1)}, a_1^{(N-1)}, s_1^{(N-1)}), (a_0^*, a_1^*, s_1^*)\},$$

where $(a_0^{(i)}, a_1^{(i)}, s_1^{(i)}), 1 \leq i \leq N-1$, *are generated by (1) using random and valid tuples and* $\beta \stackrel{R}{\leftarrow} \{0,1\}$. *If* $\beta = 0$, *then* (a_0^*, a_1^*, s_1^*) *is generated by* UH_{CA} *using a random and valid tuple* $(w_0^*, w_1^*, k_{CA}^*, s_0^*, s_1^*) \in \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_4} \times \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$. *Otherwise,* (a_0^*, a_1^*, s_1^*) *is chosen uniformly at random from* $\{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$. *Then the statistical difference between* W_0 *and* W_1 *is at most* $\frac{1}{2^{\lambda_3 - 1}}$.

4.2 Basic Multi-recipient Encryption with Sender Recovery

In this section we describe the construction of a basic MRES-SR, which will be further optimized in the next section. Since ephemeral keys generated by different KEMs may be of different lengths or on different groups, we cannot use those keys to generate the description of UH_{CA} directly. To solve the problem, we prepare two hash functions $H_0 : \{0,1\}^* \to \{0,1\}^{2\lambda_1}$, $H_1 : \{0,1\}^* \to \{0,1\}^{\lambda_1+\lambda_3}$, and a key derivation function $H_{KDF} : \{0,1\}^* \to \{0,1\}^{\lambda_k}$ to generate UH_{CA}. To realize the sender recovery property, the sender's recovery key together with other ephemeral keys are mapped to the same string, which is the symmetric key of a DEM. An one-time signature $Sig = (Gen_{Sig}, Sign, Vrfy)$ is applied to thwart adaptively chosen ciphertext attacks.

The generation of UH_{CA} must take into account a verification key vk for an one-time signature in addition to $(k_i, k_{CA,i-1})$. The advantage of adding vk is two fold. First, vk is used to generate the description of UH_{CA} and the sender can recover the ephemeral key $k_{CA,t}$ by vk. Second, the ephemeral keys and the ciphertexts of KEM can be reused thanks to a fresh vk, which will be explained later. More details of the basic MRES-SR are described below and Fig 2 illustrates the structure of the basic MRES-SR.

- Key generation Gen_{MR}
  - $Gen_{MR}.U_i$: Each recipient U_i, where $1 \le i \le t$, runs his key generation algorithm $KEM.Gen$ to generate a public/secret key pair (pk_i, sk_i). Let $\{0,1\}^{\lambda^{(i)}_{KEM}}$ denote U_i's KEM keyspace. Let $\mathbf{pk} = (pk_1, pk_2, ..., pk_t)$.
  - $Gen_{MR}.S$: The sender chooses at random $sk_{rcv} \in \{0,1\}^{\lambda_1}$ as her secret key for the recovery of a message from a ciphertext.
- Encryption $Enc_{MR}(sk_{rcv}, \mathbf{pk}, m)$ by the sender
  1. $(vk, sk_{Sig}) \leftarrow Gen_{Sig}(1^\lambda)$. Set $k_{CA,0} = sk_{rcv}$.
     For $i = 1, ..., t$, do
     (a) $(k_i, c_i) \leftarrow KEM.Enc(pk_i)$, $w_{0,i-1}||s_{0,i-1}||k_{CA,i} \leftarrow H_0(k_{CA,i-1}||vk)$ and $w_{1,i}||s_{1,i} \leftarrow H_1(k_i||vk)$. If $s_{1,i} = s_{0,i-1}$, compute $(k_i, c_i) \leftarrow KEM.Enc(pk_i)$ and $w_{1,i}||s_{1,i} \leftarrow H_1(k_i||vk)$ until $s_{1,i} \neq s_{0,i-1}$. Note that the lengths of $w_{j,i}$, $s_{j,i}$ and $k_{CA,i}$ are λ_3, λ_1 and λ_4, respectively.
     (b) Solve the system of linear equations below for $(a_{0,i}, a_{1,i})$.

$$\begin{cases} w_{0,i-1}||k_{CA,i} = a_{0,i} + a_{1,i}s_{0,i-1} \\ w_{1,i}||k_{CA,i} = a_{0,i} + a_{1,i}s_{1,i} \end{cases}$$

     Let $\mathbf{a} = \{(a_{0,i}, a_{1,i})\}$ and $\mathbf{c}_{KEM} = (c_1, c_2, ..., c_t)$, where $\{(a_{0,i}, a_{1,i})\}$ denotes an ordered list of $(a_{0,1}, a_{1,1}), ..., (a_{0,t}, a_{1,t})$.
  2. $k_{Enc} \leftarrow H_{KDF}(k_{CA,t}||\mathbf{a})$ and $c_{DEM} \leftarrow DEM.Enc(k_{Enc}, m)$. $tag \leftarrow Sign(sk_{Sig}, \mathbf{a}||\mathbf{c})$, where $\mathbf{c} = (\mathbf{c}_{KEM}, c_{DEM})$.
  3. Output $\mathbf{c}_{MR} = (vk, \mathbf{a}, \mathbf{c}, tag)$.
- Decryption $Dec_{MR}(sk_i, \mathbf{c}_{MR})$ by a recipient U_i
  1. If $Vrfy(vk, \mathbf{a}||\mathbf{c}, tag) \neq 1$, output $\perp$ and halt.
  2. Let $k_i \leftarrow KEM.Dec(sk_i, c_i)$. If $k_i = \perp$, output $\perp$ and halt.

3. Let $w_{1,i}||s_{1,i} \leftarrow H_1(k_i||vk)$ and $w'_{1,i}||k_{CA,i} \leftarrow a_{0,i}+a_{1,i}s_{1,i}$. If $w'_{1,i} \neq w_{1,i}$, output $\perp$ and halt.
4. If $i = t$, go to step 5. Otherwise, for $j = i+1, ..., t$, do
   - $w_{0,j-1}||s_{0,j-1}||k_{CA,j} \leftarrow H_0(k_{CA,j-1}||vk)$.
5. $k_{Enc} \leftarrow H_{KDF}(k_{CA,t}||\mathbf{a})$, $m' \leftarrow DEM.Dec(k_{Enc}, c_{DEM})$.
6. Output m'.

– Recovery $Rec_{MR}(sk_{rcv}, \mathbf{c}_{MR})$ by the sender
1. If $Vrfy(vk, \mathbf{a}||\mathbf{c}, tag) \neq 1$, output $\perp$ and halt.
2. Let $w_{0,0}||s_{0,0}||k_{CA,1} \leftarrow H_0(sk_{rcv}||vk)$ and $w'_{0,0}||k'_{CA,1} \leftarrow a_{0,1} + a_{1,1}s_{0,0}$. If $w'_{0,0}||k'_{CA,1} \neq w_{0,0}||k_{CA,1}$, output $\perp$ and halt.
3. For $j = 2, ..., t$, do
   - $w_{0,j-1}||s_{0,j-1}||k_{CA,j} \leftarrow H_0(k_{CA,j-1}||vk)$.
4. $k_{Enc} \leftarrow H_{KDF}(k_{CA,t}||\mathbf{a})$, $m' \leftarrow DEM.Dec(k_{Enc}, c_{DEM})$.
5. Output m'.

Note that $KEM.Gen$, $KEM.Enc$ and $KEM.Dec$ could be different for each

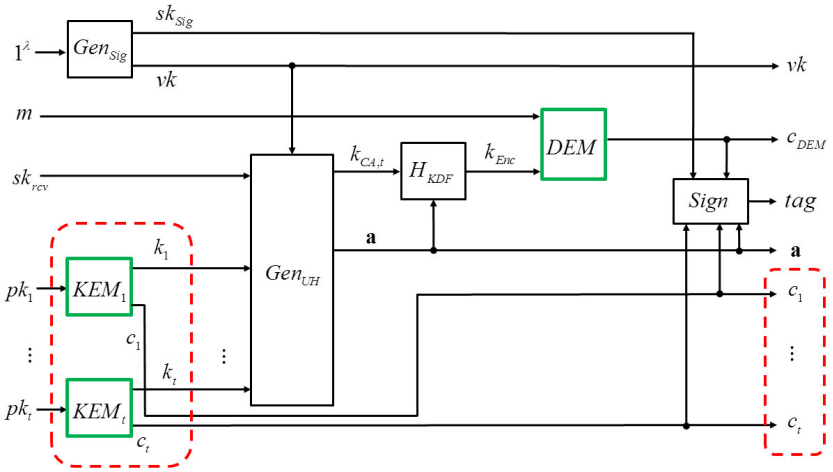

Fig. 2. Encrypting with MRES-SR, where Gen_{UH} denotes the generation of $\mathbf{a}$ and $k_{CA,t}$

recipient, being determined by the specific public key cryptosystem used by the recipient. For simplicity, we use the same notations. On the other hand, $DEM.Dec$ and the verification algorithm $Vrfy$ for the one-time signature are the same across all recipients. In order to decrypt correctly, each recipient U_i needs to know the "position" $(c_i, a_{0,i}, a_{1,i})$ in $\mathbf{c}_{MR}$, which is usually an implicit requirement for multi-recipient encryption.

MRES-SR with Tree Structure. Computational costs for recipients are not the same in that U_{i-j} has to compute j more hashing operations than does U_i, for $0 \leq j < i \leq t$. Although the differences may be immaterial in most applications, we do need to keep in mind that the depth of the hash chain increases linearly with the number of recipients. This problem can be alleviated by a logarithmic factor by the use of the tree structured construction. More precisely, a tree can be defined recursively starting at leaf nodes: sk_{rcv}, k_1,...,k_t are set to leaf nodes $k_{CA,0}^{[0]}$,...,$k_{CA,0}^{[t]}$, respectively, and each pair of nodes, say $k_{CA,0}^{[i]}$ and $k_{CA,0}^{[i+1]}$, for $i = 0, 2, 4, ...$, are mapped to their parent node, say $k_{CA,1}^{[i]}$, using UH_{CA}. If the number of nodes on a level is odd, the rightmost node on that level is set to his parent node directly. Finally, sk_{rcv}, k_1,...,k_t are mapped to the same root node, which can be generated using any one of the leaf nodes and the corresponding leaf to root path. Security of the tree structured MRES-SR can be proven by following a similar analysis to that for the basic MRES-SR shown in section 4.3. Details of the proof will be provided in the full version of the paper.

4.3 Security Analysis

Theorem 1. *The basic MRES-SR is ϵ-IND-CCA2 secure in the random oracle model if recipient U_i's KEM is $\epsilon_{KEM}^{(i)}$-IND-CCA2 secure, for $1 \leq i \leq t$, DEM is ϵ_{DEM}-IND-OPA secure and Sig is ϵ_{Sig}-SU-CMA secure where $\epsilon \leq 2t\epsilon_{KEM} + \epsilon_{DEM} + \epsilon_{Sig} + (\frac{1}{2^{\lambda_1-1}} + \frac{1}{2^{\lambda_k-1}})N_{RO} + (\frac{1}{2^{\lambda_1-2}} + \frac{1}{2^{\lambda_3-1}})N_{Rec} + \frac{1}{2^{\lambda_1}} + \frac{1}{2^{\lambda_3-1}}$, $\epsilon_{KEM} = max\{\epsilon_{KEM}^{(1)}, ..., \epsilon_{KEM}^{(t)}\}$, N_{Rec} denotes an upper bound on the number of recovery queries, $N_{RO} = max\{N_0, N_1\}$, and N_0 and N_1 denote upper bounds on the numbers of queries on H_0 and H_1, respectively.*

Proof. In order to show that any PPT adversary can win the IND-CCA2 game of MRES-SR with a negligible advantage only, we introduce three new games described below.

- Game 0 is the same as the original IND-CCA2 game of MRES-SR. During the challenge phase, the target ciphertext $\mathbf{c}_{MR}^* = (vk^*, \mathbf{a}^*, \mathbf{c}^*, tag^*)$ is computed as follows.
  - $(vk^*, sk_{Sig}^*) \leftarrow Gen_{Sig}(1^\lambda)$, $k_{CA,0}^* \leftarrow sk_{rcv}$.
    For $i = 1, ..., t$, do
    1. $(k_i^*, c_i^*) \leftarrow KEM.Enc(pk_i)$, $w_{0,i-1}^* || s_{0,i-1}^* || k_{CA,i}^* \leftarrow H_0(k_{CA,i-1}^* || vk^*)$ and $w_{1,i}^* || s_{1,i}^* \leftarrow H_1(k_i^* || vk^*)$ such that $s_{1,i}^* \neq s_{0,i-1}^*$.
    2. Solve the system of linear equations below for $(a_{0,i}^*, a_{1,i}^*)$.

$$\begin{cases} w_{0,i-1}^* || k_{CA,i}^* = a_{0,i}^* + a_{1,i}^* s_{0,i-1}^* \\ w_{1,i}^* || k_{CA,i}^* = a_{0,i}^* + a_{1,i}^* s_{1,i}^* \end{cases}$$

    Let $\mathbf{a}^* = \{(a_{0,i}^*, a_{1,i}^*)\}$ and $\mathbf{c}_{KEM}^* = (c_1^*, c_2^*, ..., c_t^*)$.
  - $k_{Enc}^* \leftarrow H_{KDF}(k_{CA,t}^* || \mathbf{a}^*)$, $c_{DEM}^* \leftarrow DEM.Enc(k_{Enc}^*, m_b)$ and $tag^* \leftarrow Sign(sk_{Sig}^*, \mathbf{a}^* || \mathbf{c}^*)$, where $\mathbf{c}^* = (\mathbf{c}_{KEM}^*, c_{DEM}^*)$.
- Game 1 is similar to Game 0, but with the following differences:

- At the beginning of the game, the challenger computes part of the target ciphertext as follows: $(vk^*, sk^*_{Sig}) \leftarrow Gen_{Sig}(1^\lambda)$, $k^*_{CA,0} \leftarrow sk_{rcv}$, $(k^*_i, c^*_i) \leftarrow KEM.Enc(pk_i)$ and $k^{**}_i \xleftarrow{R} \{0,1\}^{\lambda^{(i)}_{KEM}}$, for $1 \le i \le t$. Then let $\mathbf{c}^*_{KEM} = (c^*_1, c^*_2, ..., c^*_t)$.
  For $i = 1, ..., t$, do
  1. $w^*_{0,i-1}||s^*_{0,i-1}||k^*_{CA,i} \leftarrow H_0(k^*_{CA,i-1}||vk^*)$ and $w^{**}_{1,i}||s^{**}_{1,i} \leftarrow H_1(k^{**}_i||vk^*)$ such that $s^{**}_{1,i} \neq s^*_{0,i-1}$.
  2. Solve the system of linear equations below for $(a^{**}_{0,i}, a^{**}_{1,i})$.

$$\begin{cases} w^*_{0,i-1}||k^*_{CA,i} = a^{**}_{0,i} + a^{**}_{1,i} s^*_{0,i-1} \\ w^{**}_{1,i}||k^*_{CA,i} = a^{**}_{0,i} + a^{**}_{1,i} s^{**}_{1,i} \end{cases}$$

  Let $k^{**}_{Enc} \leftarrow H_{KDF}(k^*_{CA,t}||\mathbf{a}^{**})$, where $\mathbf{a}^{**} = \{(a^{**}_{0,i}, a^{**}_{1,i})\}$.
- In the challenge phase, $c^{**}_{DEM} \leftarrow DEM.Enc(k^{**}_{Enc}, m_b)$ and $tag^{**} \leftarrow Sign(sk^*_{Sig}, \mathbf{a}^{**}||\mathbf{c}^{**})$, where $\mathbf{c}^{**} = (\mathbf{c}^*_{KEM}, c^{**}_{DEM})$. The target ciphertext of Game 1 is $\mathbf{c}^{**}_{MR} = (vk^*, \mathbf{a}^{**}, \mathbf{c}^{**}, tag^{**})$.
- After the challenge phase, $\mathcal{O}_{Dec\ j}$ returns $\perp$ for decryption queries $c_{MR} = (vk, \mathbf{a}, (c_1, ..., c_t, c_{DEM}), tag)$ satisfying $c_j = c^*_j$, for $j \in \{1, ..., t\}$.

– Game 2 is similar to Game 1 except that the challenger randomly chooses $(a^+_{0,1}, a^+_{1,1})$ and k^+_{Enc} in place of $(a^{**}_{0,1}, a^{**}_{1,1})$ and k^{**}_{Enc}, respectively. Let $\mathbf{a}^+ = \{(a^+_{0,1}, a^+_{1,1}), (a^{**}_{0,2}, a^{**}_{1,2}) \ , ..., (a^{**}_{0,t}, a^{**}_{1,t})\}$. In the challenge phase, $c^+_{DEM} \leftarrow DEM.Enc(k^+_{Enc}, m_b)$, and $tag^+ \leftarrow Sign(sk^*_{Sig}, \mathbf{a}^+||\mathbf{c}^+)$, where $\mathbf{c}^+ = (\mathbf{c}^*_{KEM}, c^+_{DEM})$. The target ciphertext is $c^+_{MR} = (vk^*, \mathbf{a}^+, \mathbf{c}^+, tag^+)$.

Claim 2. *If recipient U_i's KEM is $\epsilon^{(i)}_{KEM}$-IND-CCA2 secure, for $1 \le i \le t$, and Sig is ϵ_{Sig}-SU-CMA secure, then* $|\Pr[Game\ 0 = 1] - \Pr[Game\ 1 = 1]| \le \frac{1}{2^{\lambda_1}} + (t+1)\epsilon_{KEM} + \epsilon_{Sig}$, *where* $\epsilon_{KEM} = max\{\epsilon^{(1)}_{KEM}, \epsilon^{(2)}_{KEM}, ..., \epsilon^{(t)}_{KEM}\}$ *and* $\Pr[Game\ j = 1]$ *denotes the probability that the adversary wins in Game j.*

Proof. We proceed with a sequence of t games to show the indistinguishability between Game 0 and Game 1.

– Game $0^{(0)}$ is similar to Game 0, except that, after the challenge phase, $\mathcal{O}_{Dec\ j}$ returns $\perp$ for decryption queries $c_{MR} = (vk, \mathbf{a}, (c_1, ..., c_t, c_{DEM}), tag)$ satisfying $c_j = c^*_j$, for $j \in \{1, ..., t\}$.
– Game $0^{(1)}$ is similar to Game $0^{(0)}$, except for the following modifications. At the beginning of the game, the challenger computes part of the target ciphertext: Compute vk^* and $\mathbf{c}^*_{KEM}$ as in the challenge phase. Then, let $k^{**}_1 \xleftarrow{R} \{0,1\}^{\lambda^{(1)}_{KEM}}$. k^{**}_1 instead of k^*_1 will be used to compute the target ciphertext.
  ⋮
– Game $0^{(i)}$ is similar to Game $0^{(i-1)}$, but at the beginning of the game, $k^{**}_i \xleftarrow{R} \{0,1\}^{\lambda^{(i)}_{KEM}}$ and k^{**}_i instead of k^*_i will be used to compute the target ciphertext.
  ⋮

– Game $0^{(t)}$ is similar to Game $0^{(t-1)}$, but at the beginning of the game, $k_t^{**} \stackrel{R}{\leftarrow} \{0,1\}^{\lambda_{KEM}^{(t)}}$ and k_t^{**} instead of k_t^* will be used to compute the target ciphertext. Indeed, $\Pr[Game\ 0^{(t)} = 1] = \Pr[Game\ 1 = 1]$.

We call c_{MR} is valid if c_{MR} can pass the validity check in step 1, 2 and 3 of decryption. Notice that A may query $\mathcal{O}_{Dec\ j}$ with a valid $c_{MR} = (vk, \mathbf{a}, (c_1, ..., c_t, c_{DEM}), tag)$ such that $c_{MR} \neq c_{MR}^*$ and $c_j = c_j^*$. Let $VALID$ denote such an event. If $VALID$ does not happen, Game 0 and Game $0^{(0)}$ are the same. That is, $\Pr[Game\ 0 = 1 | \overline{VALID}] = \Pr[Game\ 0^{(0)} = 1 | \overline{VALID}]$. Therefore,

$$\Pr[Game\ 0 = 1] - \Pr[Game\ 0^{(0)} = 1] \leq \Pr[VALID].$$

Next, we show $\Pr[VALID]$ is negligible by considering two cases below.

– $vk = vk^*$. If this case happens with a non-negligible probability, we can construct an algorithm to break the security of the underlying one-time signature and we have $\Pr[VALID] \leq \epsilon_{Sig}$.
– $vk \neq vk^*$.
  • A presents queries $k_j^{*real}||vk$ to H_1 where k_j^{*real} is the real ephemeral key corresponding to c_j^*. If this case happens with a non-negligible probability, we can construct an algorithm to break the IND-CCA security of one of t underlying KEMs. Hence, $\Pr[VALID] \leq t\epsilon_{KEM}$, where $\epsilon_{KEM} = max\{\epsilon_{KEM}^{(1)}, \epsilon_{KEM}^{(2)}, ..., \epsilon_{KEM}^{(t)}\}$.
  • A did not make queries $k_j^{*real}||vk$ to oracle H_1. Since the output of $H_1(k_j^{*real}||vk)$ is distributed uniformly over the range of H_1, the probability that the ciphertext can pass the validity check in step 3 of decryption is at most $\frac{1}{2^{\lambda_1}}$.

Therefore we have $\Pr[VALID] \leq \frac{1}{2^{\lambda_1}} + t\epsilon_{KEM} + \epsilon_{Sig}$.

Indistinguishability between Game $0^{(i-1)}$ and Game $0^{(i)}$ relies on the IND-CCA2 security of U_i's KEM, for $i = 1, ..., t$. It is not difficult to prove that $|\Pr[Game\ 0^{(i-1)}] - \Pr[Game\ 0^{(i)}]| \leq \epsilon_{KEM}^{(i)}$ and $|\Pr[Game\ 0^{(0)}] - \Pr[Game\ 0^{(t)}]| \leq t\epsilon_{KEM}$. Therefore, $|\Pr[Game\ 0 = 1] - \Pr[Game\ 1 = 1]| \leq |\Pr[Game\ 0 = 1] - \Pr[Game\ 0^{(0)} = 1]| + |\Pr[Game\ 0^{(0)} = 1] - \Pr[Game\ 0^{(t)} = 1]| \leq \frac{1}{2^{\lambda_1}} + 2t\epsilon_{KEM} + \epsilon_{Sig}$.

Claim 3. *$|\Pr[Game\ 1 = 1] - \Pr[Game\ 2 = 1]| \leq (\frac{1}{2^{\lambda_1 - 1}} + \frac{1}{2^{\lambda_k - 1}})N_{RO} + (\frac{1}{2^{\lambda_1 - 2}} + \frac{1}{2^{\lambda_3 - 1}})N_{Rec} + \frac{1}{2^{\lambda_3 - 1}}$, where $N_{RO} = max\{N_0, N_1\}$ and N_0 and N_1 denote upper bounds on the numbers of queries on H_0 and H_1, respectively.*

Sketch of Proof. Indistinguishability between Game 1 and Game 2 is closely related to Claim 1. Assume that there exists a PPT adversary A_{12} such that $|\Pr[Game\ 1 = 1] - \Pr[Game\ 2 = 1]|$ is non-negligible, we show how to construct a PPT algorithm M_{UH}, which takes as input

$$W_\beta = \{(a_0^{(1)}, a_1^{(1)}, s_1^{(1)}), (a_0^{(2)}, a_1^{(2)}, s_1^{(2)}), ..., (a_0^{(N-1)}, a_1^{(N-1)}, s_1^{(N-1)}), (a_0^*, a_1^*, s_1^*)\}$$

and outputs β with a non-negligible advantage.

The main idea of the proof is similar to that of [11], where there is only one UH_{CA}. Notice that if the inputs to a random oracle $H(\cdot)$ are different, the corresponding outputs are uniformly and independently distributed. Due to the unforgeability of the one-time signature, vk which is part of the inputs to H_0 and H_1 are usually different in each encryption and the corresponding outputs $w_{0,i-1}||s_{0,i-1}||k_{CA,i}$ and $w_{1,i}||s_{1,i}$ are uniformly and independently distributed. Therefore, (a_0, a_1)s in W_β can be considered as parts of the ciphertexts in the simulated game. However, our scheme uses t UH_{CA}s with the MD like iterated structure. To compute a simulated target ciphertext, the first iteration of UH_{CA} is generated using (a_0^*, a_1^*, s_1^*) and the remaining $t-1$ iterations can be generated following the idea of a length-extension attack. More details of the proof are omitted due to lack of space.

Claim 4. $|\Pr[Game\ 2 = 1] - 1/2| \leq \epsilon_{Sig} + \epsilon_{DEM}$ *if the underlying DEM is* ϵ_{DEM}*-IND-OPA secure and Sig is* ϵ_{Sig}*-SU-CMA secure.*

Proof. If there exists a PPT adversary, say A_2, which can win Game 2 with a non-negligible advantage, we show how to construct an algorithm M_{DEM} to break the IND-OPA security of the underlying DEM.

M_{DEM} sets parameters and answers queries $q_{Enc}, q_{Dec}, q_{Rec}$ from A_2 as in Game 2, except that when A_2 submits $q_{Dec} = (vk^*, \mathbf{a}', \mathbf{c}', tag')$ or $q_{Rec} = (vk^*, \mathbf{a}', \mathbf{c}', tag')$, M_{DEM} responds with "$\perp$". However, M_{DEM}'s answers may be wrong when such queries are valid. Denote such an event by Bad_2. When receiving (m_0, m_1) from A_2, M_{DEM} sends (m_0, m_1) to the challenger of IND-OPA game and gets the answer c^+_{DEM}. M_{DEM} sets the target ciphertext to $(vk^*, \mathbf{a}^+, (\mathbf{c}^*_{KEM}, c^+_{DEM}), tag^+)$. Finally, A_2 will terminates with output a bit, which is also the output of M_{DEM}. Notice that M_{DEM} perfectly simulates Game 2 for A_2 if Bad_2 does not happen. That is, $\Pr[Game\ 2 = 1 \cap \overline{Bad_2}] = \Pr[M_{DEM}\ \text{wins} \cap \overline{Bad_2}]$. Hence, we have $|\Pr[Game\ 2 = 1] - \frac{1}{2}| \leq |\Pr[Game\ 2 = 1] - \Pr[M_{DEM}\ \text{wins}]| + |\Pr[M_{DEM}\ \text{wins}] - \frac{1}{2}| \leq \Pr[Bad_2] + \epsilon_{DEM}$, where the last inequality follows from Lemma 1.

In fact, $\Pr[Bad_2]$ is negligible, which relies on the unforgeability of one-time signature. That is, $\Pr[Bad_2] \leq \epsilon_{Sig}$. Therefore, $\left|\Pr[Game\ 2 = 1] - \frac{1}{2}\right| \leq \epsilon_{Sig} + \epsilon_{DEM}$, which completes the proof of Claim 4.

From Claims 2, 3 and 4, it follows that $|\Pr[Game\ 0 = 1] - \frac{1}{2}| \leq |\Pr[Game\ 0 = 1] - \Pr[Game\ 1 = 1]| + |\Pr[Game\ 1 = 1] - \Pr[Game\ 2 = 1]| + |\Pr[Game\ 2 = 1] - \frac{1}{2}| \leq 2t\epsilon_{KEM} + \epsilon_{DEM} + \epsilon_{Sig} + (\frac{1}{2^{\lambda_1 - 1}} + \frac{1}{2^{\lambda_k - 1}})N_{RO} + (\frac{1}{2^{\lambda_1 - 2}} + \frac{1}{2^{\lambda_3 - 1}})N_{Rec} + \frac{1}{2^{\lambda_1}} + \frac{1}{2^{\lambda_3 - 1}}$, which completes the proof of Theorem 1.

4.4 Stateful MRES-SR

We find that $(k_1, c_1), ..., (k_t, c_t)$ can be cached as part of the state information for the scheme and reused later for improved efficiency. Specifically, the sender can use the same (k_i, c_i) for each recipient U_i when running Enc_{MR}. As a result, the main cost of computing $\mathbf{c}_{KEM}$, e.g. exponentiations in large cyclic groups, is

minimized for subsequent applications of the encryption algorithm. As a specific example, if we set $\lambda_1 = 256$ and $\lambda_3 = \lambda_4 = 128$, the main cost of a subsequent encryption for the same recipients is dominated by $2t$ multiplications, t inversions in $GF(2^{256})$ and a symmetric encryption operation.

Notice that the verification key vk is usually different in each encryption if the underlying one-time signature is unforgeable. Thanks to the randomness of vk, $\mathbf{a}$ and k_{Enc} are fresh each time when a new message is encrypted, even with the fixed state. Additionally, even if the group of recipients is changed, most part of the state could be still reused. As an example, if U_1 leaves the group of recipients, $(k_2, c_2), ..., (k_t, c_t)$ can still be reused and U_1 is denied the ability of decrypting a new ciphertext due to the use of a fresh symmetric key. On the other hand, if a new recipient U_{t+1} joins the group, the sender can generate a new pair (k_{t+1}, c_{t+1}) for the new recipient and then update the state to $(k_1, c_1), ..., (k_t, c_t), (k_{t+1}, c_{t+1})$.

Security Analysis. The security model for MRES-SR is carried over to the stateful version of our MRES-SR, where the encryption oracle computes the ciphertexts using a fixed state. The method of security proof of the basic MRES-SR, which is shown in section 4.3, still holds for the stateful MRES-SR.

The main difference between proofs of the stateful MRES-SR and the basic MRES-SR is that the sender uses the fixed state $(k_1^*, c_1^*), ..., (k_t^*, c_t^*)$ to answer the encryption queries from the beginning of Game 0, Game 1 and Game 2. (Note that $k_1^*, ..., k_t^*$ in Game 1 and Game 2 are strings chosen at random.) Only minor modifications need to be made in the proof of Claim 2. In Game $0^{(0)}$ of the proof of Claim 2, $\mathcal{O}_{Dec\ j}$ returns $\perp$ for decryption queries $c_{MR} = (vk, \mathbf{a}, (c_1, ..., c_t, c_{DEM}), tag)$ satisfying that $c_j = c_j^*$. However, in the game of stateful MRES-SR, it is possible for the adversary to make valid decryption queries c_{MR} such that $c_{MR} \neq c_{MR}^*$ and $c_j = c_j^*$. For instance, such c_{MR} could be returned by $\mathcal{O}_{Enc}$ since the state c_j^* is fixed. To reduce the possibility of returning $\perp$ for a valid ciphertext, a list $\mathcal{L}_{Enc}$ is used to record the encryption queries and the corresponding answers from $\mathcal{O}_{Enc}$. The modified description of Game $0^{(0)}$ are as follows. During the game, $\mathcal{O}_{Dec\ j}$ returns $\perp$ for decryption queries $c_{MR} = (vk, \mathbf{a}, (c_1, ..., c_t, c_{DEM}), tag)$ satisfying that c_{MR} is not in $\mathcal{L}_{Enc}$ and $c_j = c_j^*$. In addition, Game $0^{(i)}$ is similar to Game $0^{(i-1)}$, for $i = 1, ..., t$, except that a random k_i^{**} instead of k_i^* is used to answer the encryption queries and compute the target ciphertext. For the proof of $|\Pr[Game\ 0^{(i-1)}] - \Pr[Game\ 0^{(i)}]| \leq \epsilon_{KEM}^{(i)}$, the simulator can use $\mathcal{L}_{Enc}$ to answer the decryption queries c_{MR}s which are returned by $\mathcal{O}_{Enc}$.

5 Anonymous Multi-recipient Encryption

In this section, we show how our MRES-SR can be easily converted to an anonymous MRES-SR, denoted by ANOMRES-SR. Let $\mathcal{U} = \{U_1, U_2, ..., U_t\}$ be the universe of all legitimate recipients and $\mathbf{pk} = (pk_1, pk_2, ..., pk_t)$ be an ordered list of recipients' public keys. Let S, a subset of $\mathcal{U}$, be a group of recipients

to whom a sender intends to send a message. Anonymity of recipients can be accomplished by sending the message in an encrypted form to all legitimate recipients in such a way that only intended recipients can decrypt the ciphertext correctly. Specifically, the sender takes as input to the encryption algorithm sk_{rcv}, $\mathbf{pk}$ and m together with S the set of all intended recipients. During the encryption phase, if U_i is a legitimate recipient but not an intended one for this particular message m, that is $U_i \in \mathcal{U}$ and $U_i \notin S$, a randomly chosen k_i in place of the "real" k_i generated by $KEM.Enc(pk_i)$ is used to generate a ciphertext. As a result, the unintended recipient $U_i \notin S$ is denied the ability to decrypt the ciphertext. Decryption and recovery are the same as that of MRES-SR. More details of the modified encryption algorithm follow.

- Encryption $Enc_{MR}(sk_{rcv}, \mathbf{pk}, S, m)$ by the sender
  1. $(vk, sk_{Sig}) \leftarrow Gen_{Sig}(1^{\lambda})$. Set $k_{CA,0} = sk_{rcv}$.
     For $i = 1, ..., t$, do
     (a) $(k_i, c_i) \leftarrow KEM.Enc(pk_i)$. If $U_i \notin S$, replace k_i with a random element in the keyspace of KEM. Compute $w_{0,i-1}||s_{0,i-1}||k_{CA,i} \leftarrow H_0(k_{CA,i-1}||vk)$ and $w_{1,i}||s_{1,i} \leftarrow H_1(k_i||vk)$. If $s_{1,i} = s_{0,i-1}$, compute $(k_i, c_i) \leftarrow KEM.Enc(pk_i)$ and $w_{1,i}||s_{1,i} \leftarrow H_1(k_i||vk)$ until $s_{1,i} \neq s_{0,i-1}$.

The remaining steps of encryption are the same as that of MRES-SR.

The IND-CCA2 security of ANOMRES-SR can be proven in a similar way to that of MRES-SR and we only discuss the anonymity of ANOMRES-SR.

5.1 Analysis of Anonymity

Since the public/secret key pairs of MRES-SR are generated by recipients rather than a key generation center as in broadcast encryption, the adversary may choose public keys of his liking to compromise anonymity. In order to capture such attacks, in our security model we allow recipients' public keys to be generated by the adversary. What follows is our new model for anonymity described in its entirety.

- The challenger generates the sender's recovery key sk_{rcv} and a target recipient U_i's public/secret key pair (pk_i, sk_i), for $1 \leq i \leq t$. Let $S_t = \{U_1, U_2, ..., U_t\}$ and $S_c = \{\phi\}$ denote the set of the target recipients and the set of corrupted recipients, respectively. Let $\mathbf{pk} = (pk_1, pk_2, ..., pk_t)$. $\mathbf{pk}$ is sent to an adversary A who has access to four oracles described below.
  - A private key extraction oracle $\mathcal{O}_{Key}$, which upon a key extraction query $q_{Key} = pk_i$ returns sk_i, where $pk_i \in \mathbf{pk}$. If pk_i is queried, add U_i to S_c.
  - An encryption oracle $\mathcal{O}_{Enc}$, which upon an encryption query $q_{Enc} = (\mathbf{pk}', S, m)$ returns a ciphertext $c_{MR} = Enc_{MR}(sk_{rcv}, \mathbf{pk}', S, m)$. Public keys in $\mathbf{pk}'$ could be generated by the adversary.

- A decryption oracle $\mathcal{O}_{Dec\ i}$ for recipient $U_i \in S_t$, which upon a decryption query $q_{Dec} = c_{MR}$ returns $Dec_{MR}(sk_i, c_{MR})$.
- A recovery oracle $\mathcal{O}_{Rec}$, which upon a recovery query $q_{Rec} = c_{MR}$ returns $Rec_{MR}(sk_{rcv}, c_{MR})$.

– A chooses a message m, $\mathbf{pk}^*$ and two distinct sets $S_0 \subseteq \mathcal{U}^*$ and $S_1 \subseteq \mathcal{U}^*$ such that $(S_0 \backslash S_1) \bigcup (S_1 \backslash S_0) \subseteq S_t \backslash S_c$, where $\mathcal{U}^* = \{U_1^*, U_2^*, ..., U_t^*\}$ is the set of recipients corresponding to $\mathbf{pk}^*$. The challenger returns a target ciphertext $c_{MR}^* \leftarrow Enc_{MR}(sk_{rcv}, \mathbf{pk}^*, S_b, m)$, where $b \overset{R}{\leftarrow} \{0, 1\}$.

– A can make queries as described above, except that, for $U_i \in (S_0 \backslash S_1) \bigcup (S_1 \backslash S_0)$, A cannot query $\mathcal{O}_{Dec\ i}$ on $q_{Dec} = c_{MR}^*$ or query $\mathcal{O}_{Key}$ on $q_{Key} = pk_i$. Finally, the adversary terminates by returning a guess b'.

MRES-SR is said to be ϵ_{ANO}-anonymous if $|\Pr[Game_{ANO}^{b=0} = 1] - \Pr[Game_{ANO}^{b=1} = 1]| \leq \epsilon_{ANO}$, where $Game_{ANO}^{b=0} = 1$ ($Game_{ANO}^{b=1} = 1$) denotes the event that $b = b'$ when $b = 0$ ($b = 1$).

Theorem 2. *ANOMRES-SR is ϵ_{ANO}-anonymous if recipient U_i's KEM is $\epsilon_{KEM}^{(i)}$-IND-CCA2 secure, for $1 \leq i \leq t$, and Sig is ϵ_{Sig}-SU-CMA secure, where $\epsilon_{ANO} \leq \frac{2}{2^{\lambda_1}} + 3t\epsilon_{KEM} + 2\epsilon_{sig}$.*

Theorem 2 can be proven using similar methods in [12] and Claim 2. Descriptions of the proof are omitted due to space limitations.

6 Concluding Remarks

We have proposed an efficient method of constructing a (single message) multi-recipient encryption scheme in a heterogeneous setting, which offers an efficient solution to secure data sharing in a cloud computing environment. The resulting scheme can be used in a stateful manner, achieving significant savings in computation when multiple messages are sent to the same group of recipients. In addition, our scheme enjoys the sender recovery property and can be adapted to offer anonymity of recipients. One of the main techniques we use is the function which maps t strings to one string. A direction for future research is to identify more efficient ways to map t strings to one string without random oracles.

Acknowledgements. This work has been supported by 973 program (No. 2013CB834205), National Natural Science Foundation of China (No. 61103237, No. 61272035), Research Fund for the Doctoral Program of Higher Education of China (No. 20100131120015), Outstanding Young Scientists Foundation Grant of Shandong Province (No. BS2012DX018), Program for New Century Excellent Talents in University of China (No. NCET-13-0350) and Interdisciplinary Research Foundation of Shandong University (No.2012JC018).

References

1. RFC 3851: Secure/multipurpose internet mail extensions (s/mime) version 3.1 message specification, `https://tools.ietf.org/html/rfc3851`
2. Kurosawa, K.: Multi-recipient public-key encryption with shortened ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
3. Smart, N.P.: Efficient key encapsulation to multiple parties. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 208–219. Springer, Heidelberg (2005)
4. Bellare, M., Boldyreva, A., Staddon, J.: Multi-recipient encryption schemes: Security notions and randomness re-use. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2002)
5. Bellare, M., Boldyreva, A., Kurosawa, K., Staddon, J.: Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. IEEE Transactions on Information Theory 53(11), 3927–3943 (2007)
6. Barbosa, M., Farshim, P.: Randomness reuse: Extensions and improvements. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 257–276. Springer, Heidelberg (2007)
7. Hiwatari, H., Tanaka, K., Asano, T., Sakumoto, K.: Multi-recipient public-key encryption from simulators in security proofs. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 293–308. Springer, Heidelberg (2009)
8. Zheng, Y.: Digital signcryption or how to achieve cost (Signature & encryption) << cost(Signature) + cost(Encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)
9. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
10. Wei, P., Zheng, Y., Wang, X.: Public key encryption for the forgetful. In: Naccache, D. (ed.) Cryphtography and Security: From Theory to Applications. LNCS, vol. 6805, pp. 185–206. Springer, Heidelberg (2012)
11. Wei, P., Zheng, Y.: Efficient public key encryption admitting decryption by sender. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 37–52. Springer, Heidelberg (2013)
12. Libert, B., Paterson, K.G., Quaglia, E.A.: Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 206–224. Springer, Heidelberg (2012)
13. Zheng, Y., Hardjono, T., Pieprzyk, J.: The sibling intractable function family (SIFF): Notion, construction and applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science E76-A(1), 4–13 (1993)
14. Bellare, M., Kohno, T., Shoup, V.: Stateful public-key cryptosystems: how to encrypt with one 160-bit exponentiation. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 380–389. ACM, New York (2006)
15. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
16. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)