Public Key Encryption for the Forgetful

Puwen Wei[1]* Yuliang Zheng[2]† Xiaoyun Wang[1,3]*

[1]Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China
pwei@sdu.edu.cn

[2]Department of Software and Information Systems,
The University of North Carolina at Charlotte, Charlotte, NC 28223, USA
yzheng@uncc.edu

[3]Institute for Advanced Study, Tsinghua University, Beijing 100084, China
xiaoyunwang@mail.tsinghua.edu.cn

April 6, 2011

Abstract

We investigate public key encryption that allows the originator of a ciphertext to retrieve a "forgotten" plaintext from the ciphertext. This type of public key encryption with "backward recovery" contrasts more widely analyzed public key encryption with "forward secrecy". We advocate that together they form the two sides of a whole coin, whereby offering complementary roles in data security, especially in cloud computing, 3G/4G communications and other emerging computing and communication platforms. We formalize the notion of public key encryption with backward recovery, and present two construction methods together with formal analyses of their security. The first method embodies a generic public key encryption scheme with backward recovery using the "encrypt then sign" paradigm, whereas the second method provides a more efficient scheme that is built on Hofheinz and Kiltz's public key encryption in conjunction with target collision resistant hashing. Security of the first method is proved in a two-user setting, whereas the second is in a more general multi-user setting.

1 Introduction

Forward security, a notion first proposed by Günther [10] in the context of key exchange, has been well studied during the past decade. It guarantees the security of past uses of a secret key. Notably, past ciphertexts associated with a forward secure encryption scheme cannot be decrypted by an adversary even if the adversary possesses the current decryption key. A corollary of forward security is that it is infeasible even for either a receiver or a sender to recover plaintexts from past ciphertexts. A related point is that with traditional public key encryption, a sender is in general not able to decrypt a ciphertext he sent earlier. At a first look it might sound strange that a sender wants to decrypt a past ciphertext, given that it was him who created the ciphertext from a plaintext in his possession in the first place. As will be shown below, with the advent of new generations of computing and communication systems, users are increasingly relying on external storage to maintain data, especially when the amount of data exceeds the capacity of their local

*Wei and Wang were supported by the National Natural Science Foundation of China under Grant No. 60525201 and the National Basic Research Program of China under Grant No. 2007CB807902

†Part of Zheng's work was done while visiting the Institute for Advanced Study at Tsinghua University, and Shandong University on a Changjiang Scholars program sponsored by the Chinese Ministry of Education and Li Ka Shing Foundation in Hong Kong.

storage. This trend entails the necessity of decrypting ciphertexts by the sender who has "forgotten" the associated plaintexts.

Let us first take a look at the emerging cloud computing platform. The deployment of 3G/4G and other new generations of communication systems, together with the availability of increasingly sophisticated smart phones and other handheld devices, is altering the traditional image of how people use mobile phones and access the Internet. Anecdotal evidence indicates that more and more people are bypassing traditional computers, and instead using a smart phone not only as a communication tool but also as an access point to data that is stored in servers residing in a communication infrastructure. This type of applications for outsourcing individual's computing needs mirrors cloud computing for businesses.

As an example, we consider a typical email communication system where a user relies on an email server, called a message storage and transfer agent or MSTA, for all communications. All his incoming messages are stored in an inbox folder on the MSTA, and an outgoing message is relayed by the MSTA to the MSTA of the intended recipient of the message, with a copy of the message being kept in the "Sent Mail" folder of the sender's MSTA. Clearly, due to its advantages in scalability and flexibility, public key encryption is a preferred method for secure email communication. Consider a situation where a verbatim copy of a public key encrypted message is kept on the MSTA. At a later stage the sender finds out that he is no longer in possession of the original message in the storage of his local or handheld device such as a netbook computer or a smart phone, and has to rely on the MSTA for the retrieval of the message. If the copy of the message on the MSTA is encrypted using a regular public key encryption technique, he will now be out of luck.

The issue discussed above could be addressed if the user keeps a copy of the original, unencrypted message on the MSTA. This, however, would require the modification of the user's email system, possibly deviating from current industrial standards for email protocols. Worse still, it would introduce new issues on the security of unencrypted messages kept on the server. Yet a further potential problem would be the requirement of storing two copies of a message, one encrypted and the other unencrypted, on the MSTA.

The above example highlights the need for a new type of public key encryption that allows the sender to decrypt a ciphertext at a later stage. We are not aware of any existing public key encryption technique that can be "tweaked" to fulfill the requirement. A further challenge for designing such a new type of public key encryption is a requirement related to the authenticity of a ciphertext. Specifically, the originator may need to be assured at a later stage that the ciphertext was indeed produced by himself earlier.

Our second example has do to with data centers or more generally, cloud computing. Businesses are increasingly relying on cloud computing to outsource data processing and storage, with the primary goal of reducing or eliminating in house technical support, cutting costs and increasing productivity. To that end, it is envisioned that a business would maintain no local copy of past data, relying instead on data centers in the cloud for the availability, security, integrity, consistency and freshness of the data. Among the myriad of technical and nontechnical issues that are yet to be fully addressed, storing data in an encrypted form is without doubt one of the techniques that should be in any toolkit for trusted cloud computing. It is not an over-statement to say that there are numerous possible ways to store encrypted data in remote servers. The simplest of these would be for a user to encrypt data, store the ciphertext in the cloud, and afterwards when he needs the data again, fetches the ciphertext from the cloud and decrypts it to the original data. The user can choose to use either a private key encryption algorithm or a public key encryption algorithm. When a private key encryption algorithm such as AES is used, both encryption and decryption can be done easily, even by a relatively low power device such as a smart phone. When a public key encryption algorithm is used instead, a problem similar to that of the mail transfer example arises. That is, the originator of a ciphertext may neither be able to decrypt the ciphertext not check its integrity. One might ask why one has to use public key encryption in such applications. The answer lies in the fact that modern applications are getting ever more complex, and often times encryption is applied as one of many intermediate stages of data processing. It is conceivable that public key encryption may be applied to provide data confidentiality in a large system for cloud computing.

These questions are a direct motivation for us to inquire into public key encryption that admits the backward recovery of plaintexts by their originator while at the same time ensuring that the originator can verify whether a ciphertext was produced by himself earlier.

Our result. Our first contribution is to formalize the notion of public key encryption with backward recovery (PKE-BR) as well as its associated security models. We then present a generic construction of PKE-BR using the "encrypt then sign" paradigm [2]. The basic idea underlying our construction is that an ephemeral key is encapsulated, called a key encapsulation mechanism or KEM, not only by a receiver's public key but also by a sender's, and the randomness re-use (RR) technique discussed in [4] is applied to reduce the bandwidth and computational cost. As for security analysis, we prove that our PKE-BR KEM is IND-CCA2 secure and existentially unforgeable in a two-user setting if the underlying two-receiver KEM is IND-CPA and the underlying signature is weakly unforgeable.

A downside of the generic construction is that it is not quite efficient due to the use of a signing procedure. It turns out that the "encrypt then sign" paradigm is in fact an overkill in the context of meeting the requirements of PKE-BR. One reason is that the receiver may not be always required to check the origin of a ciphertext. This happens in a situation where checking the origin of a ciphertext can be accomplished using an out-of-band method. For instance, a data center may process and store data received from legitimate users only, who are required to login and prove their identities before being allowed to use any service of the data center. Yet another reason is that the originator may not be willing to have his or her identity being tied to a ciphertext explicitly which is necessarily the case when a typical digital signature scheme is employed.

The above observations motivate us to design a more efficient PKE-BR scheme, by converting Hofheinz and Kiltz's public key encryption scheme [11] into a tag based KEM. Hofheinz and Kiltz's scheme, which is based on factoring in the standard model, is very efficient and requires only about two exponentiations for encryption and roughly one exponentiation for decryption. As was already discussed by Hofheinz and Kiltz in [11], an interesting property of the scheme is that the RSA modulus N can be shared among many users. This property allows the application of our backward recovery technique in the resulting scheme. Furthermore, we observe that a target collision resistant hash function suffices to guarantee the integrity of a ciphertext and a message authentication code can be applied to allow the sender to verify the origin of a ciphertext. Proving the security of this efficient scheme, however, turns out to be more challenging. The main difficulty of the security proof lies in the fact that the setting of the simulated receiver's public key is related to the adversary's challenge tag and relying on the factoring assumption only turns out to be inadequate. Hence the security proof of our scheme cannot follow that of [11] directly. To overcome this problem, we resort to the decisional Diffie-Hellman assumption in addition to the factoring assumption, so that a simulated receiver's public key can be proven to be indistinguishable from a public key in a real scheme by any probabilistic polynomial time adversary.

Related work. While backward recovery was already mentioned as "past message recovery" in early work on signcryption [3] and such an interesting property is still retained in some signcryption schemes, such as those in [3][14], not all signcryption schemes have such a property, and more importantly, the concept and formal definition of backward recovery are yet to be further studied.

A somewhat related notion is multi-receiver signcryption [8][17][16] which can provide not only confidentiality and integrity but also authenticity and non-repudiation for multiple receivers. In fact, a sender in our generic construction plays the role of a receiver in a multi-receiver signcryption setting. Notice that the roles of the two "receivers" (the sender and the receiver) in our setting are not equal, which results in a critical difference between our second scheme and traditional multi-receiver signcryptions. That is, the receiver has to check the integrity of a ciphertext in its entirety, including both the sender's part and the receiver's part, while the sender does not have to check the receiver's part. Although the receiver in our second scheme cannot verify the origin of a ciphertext, the scheme is very much suitable for an application environment where the sender's identity can be verified easily by some out-of-band methods, say, login authentication of a data center or subscriber authentication of a cell phone network.

2 Preliminaries

Notations. $\mathbb{Z}_N$ denotes the set of integers modulo N and $|u|$ denotes the absolute value of u, where $u \in \mathbb{Z}_N$ is interpreted as a signed integer with $-(N-1)/2 \leq u \leq (N-1)/2$. $[N]$ denotes the set of integers $1, 2, ..., N$. PPT denotes probabilistic polynomial time. A real-valued function μ over integers is said to be negligible if it approaches 0 at a rate faster than the inverse of a polynomial over integers.

Target-collision resistant hash functions. A hash function $H : \{0,1\}^l \rightarrow \{0,1\}^{l_H}$ is (ε_{Hash}, t)-target collision resistant if no probabilistic t-polynomial time algorithm A can output a y such that $H(y) = H(x)$ and $y \neq x$ for a given $x \in \{0,1\}^l$ with a probability at least ε_{Hash}, when a security parameter (which is typically played by l_H) is sufficiently large. H is simply said to be target collision resistant if ε_{Hash} can be any inverse polynomial and t be any polynomial in the security parameter.

Decisional Diffie-Hellman (DDH) assumption in groups with composite order. Let $\mathcal{G}$ be an Abelian group with composite order $N = (P-1)(Q-1)$, where P and Q are large primes such that $|P| = |Q|$. Here, $|P|$ (or $|Q|$) denotes the binary length of P (or Q). $g \in \mathcal{G}$ is a generator of $\mathcal{G}$. The decisional Diffie-Hellman assumption states that, for any PPT algorithm A, there exists a negligible function μ such that for all sufficiently large $|Q|$

$$|\Pr[a, b \overset{R}{\leftarrow} \mathbb{Z}_N : A(g, g^a, g^b, g^{ab}) = 1] - \Pr[a, b, c \overset{R}{\leftarrow} \mathbb{Z}_N : A(g, g^a, g^b, g^c) = 1]| \leq \mu(|Q|)$$

where the probability is taken over the random choices of g, a, b, c and the coin-tosses of A.

In this paper, we consider the DDH assumption in the quadratic residue group QR_N, where $N = (2p+1)(2q+1)$ is a Blum integer such that p, q, $(2p+1)$ and $(2q+1)$ are all primes.

Blum-Blum-Shub (BBS) pseudorandom number generator [6]. Let

$$\text{BBS}_N(u) = (\text{LSB}_N(u), \text{LSB}_N(u^2), ..., \text{LSB}_N(u^{2^{l_k-1}})) \in \{0,1\}^{l_k}$$

denote the BBS pseudorandom number generator, where $\text{LSB}_N(u)$ denotes the least significant bit of $u \mod N$. BBS is pseudorandom if factoring Blum integer N is hard. (For details, please refer to Theorem 2 of [11].) The pseudorandomness of BBS is defined by a pseudorandomness test described as follows.

PRNG experiment for BBS generator [11]. For an algorithm D, define the advantage of D as

$$Adv_D^{BBS}(k) = \frac{1}{2}|\Pr[D(N, z, BBS_N(u)) = 1] - \Pr[D(N, z, U_{\{0,1\}^{l_k}}) = 1]|,$$

where N is a Blum integer, $u \in QR_N$ is uniformly chosen, $z = u^{2^{l_k}}$, and $U_{\{0,1\}^{l_k}} \in \{0,1\}^{l_k}$ is independently and uniformly chosen.

The algorithm D is said to (t, ε)-break BBS if D's running time is at most t and $Adv_D^{BBS}(k) \geq \varepsilon$.

Weak unforgeability of digital signature. It is defined by the following game between a challenger and an adversary.

1. The adversary sends the challenger a list of messages $\mathcal{M} = \{m_1, ..., m_n\}$.

2. The challenger generates a private/public key pair (sk_{Sig}, pk_{Sig}) and signs each m_i using sk_{Sig} for $i = 1$ to n. The corresponding signature list is $Sig = \{(m_1, \sigma_1), ..., (m_n, \sigma_n)\}$. Then pk_{Sig} and Sig are sent back to the adversary.

3. The adversary outputs a pair (m^*, σ^*).

We say the adversary wins the game if $(m^*, \sigma^*) \notin Sig$ and σ^* can be verified to be a valid signature for m^*. A signature scheme is weakly unforgeable if the probability that the adversary wins the game is negligible.

The above game captures the existential unforgeability with respect to weak chosen-message attacks (or generic chosen message attacks [9]) against digital signature, where the generation of the message list $\mathcal{M}$ does not depend on the public key pk_{Sig}. We follow the definition of weak unforgeability in [12] except that in [12], the adversary is said to win the game if $m^* \notin \{m_1, ..., m_n\}$ and σ^* can be verified to be a valid signature for m^*.

3 Security requirements and models

3.1 Security requirements

To illustrate the security requirements of PKE-BR, we take the mail transfer as an example and consider a communication model of three communicating parties, including the message storage and transfer agent (MSTA), Alice (a sender) and Bob (a receiver). Note that this communication model also includes the case of a data center, where the data center plays both the role of the MSTA and that of the receiver. The main procedure of a PKE-BR scheme can be divided into the following phases.

1. **Encryption**. Alice encrypts a plaintext (or mail) m under Bob's public key, and passes the resultant ciphertext c over to her local MSTA.
2. **Delivery**. Upon receiving c, MSTA saves c in Alice's "sent mail folder" and delivers c to Bob. (Actually, c is delivered to Bob's local MSTA. Bob receives c from his local MSTA.)
3. **Decryption**. Upon receiving the ciphertext c, Bob can decrypt c using his private key.
4. **Recovery**. When requested by Alice, her MSTA retrieves c and returns it back to Alice. Alice can recover m from c using her own private key.

From the above procedure, one can see that when a weak public key encryption scheme is applied, m may be potentially leaked and/or modified during any of the following three phases: (1) transmission from Alice to Bob, (2) residing in the storage device of a MSTA, and (3) transmission from the MSTA back to Alice. This justifies the following three security requirements for PKE-BR:

- Confidentiality of the plaintext m. For the confidentiality, we adopt the notion of indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) [15].
- Integrity of the ciphertext c. Bob the receiver can be assured that the ciphertext c is not modified during transmission from Alice to Bob.
- PKE-BR authenticity. Alice the sender can be assured that a ciphertext c' received and a plaintext m' recovered from c' are both initially produced by/originated from herself.

The first requirement is a standard security requirement for any encryption scheme. The second one can be achieved by checking the integrity of the ciphertext. (Note that the first requirement does not necessarily imply the second.) To fulfill the third security requirement, a primitive which enables Alice to verify the authenticity of a ciphertext, such as a public key signature or a message authentication code, may be employed.

3.2 Key encapsulation with backward recovery

It is known that a key encapsulation mechanism or KEM can be converted to a hybrid encryption [1]. Similar techniques can be applied to construct a PKE-BR scheme from a KEM-BR (KEM with backward recovery). Hence, our focus will be on KEM-BR. In this section, we describe a public key KEM-BR, which consists of a tuple of five algorithms.

1. Common Parameter Generation
   - A probabilistic common parameter generation algorithm, *Com*. It takes as input a security parameter 1^k and returns the common parameter I.
2. Key Generation
   - A probabilistic sender key generation algorithm Key_S. It takes as input the common parameter I and outputs a private/public key pair (sk_S, pk_S) for the sender.
   - A probabilistic receiver key generation algorithm Key_R. It takes as input the common parameter I and outputs a private/public key pair (sk_R, pk_R) for the receiver.
3. Symmetric Key Generation and Encapsulation
   - A probabilistic symmetric key generation and encapsulation algorithm, KGE. It takes the common parameter I, the sender's private/public key pair (sk_S, pk_S) and the receiver's public key pk_R as input and outputs the symmetric key K, the state information s and the encapsulated value C_{KGE}. KGE consists of the following two algorithms.
     (a) A probabilistic symmetric key generation algorithm, *Sym*. It takes I as input, and outputs a symmetric key K and some state information denoted by s.
     (b) A probabilistic key encapsulation algorithm, *Encap*. It takes the state information s, the sender's private/public key (sk_S, pk_S) and the receiver's public key pk_R as input, and returns an encapsulated value C_{KGE}. More precisely, *Encap* consists of two algorithms, a key encapsulation algorithm KE and a key authentication algorithm KA which function as follows:
       - KE takes the state information s and the receiver's public key pk_R as input and returns E_1.
       - KE takes the state information s and the sender's public key pk_S as input and returns E_2. Denote (E_1, E_2) by E.
       - KA takes (E, sk_S, pk_S) as input and returns $\sigma = (\sigma_1, \sigma_2)$, where σ_1 provides integrity and σ_2 provides authenticity, respectively.

       Encap outputs $C_{KGE} = (E, \sigma) = (C_R, C_S)$, where $C_R = (E_1, \sigma_1)$ is intended for the receiver and $C_S = (E_2, \sigma_2)$ for the sender.
4. Decapsulation
   - A deterministic decapsulation algorithm, *Decap*. It takes the sender's public key pk_S, the receiver's private/public key pair (sk_R, pk_R) and C_{KGE} as input, and returns either a symmetric key K or a unique error symbol $\perp$. *Decap* consists of two algorithms, an integrity check algorithm IC and a key decapsulation algorithm KD whose functions are described below:
     - IC takes the sender's public key pk_S, the receiver's private/public key pair (sk_R, pk_R) and C_{KGE} as input, and outputs either "OK" or $\perp$.
     - If IC does not return $\perp$, KD takes as input the receiver's private/public key pair (sk_R, pk_R) and C_{KGE}, and outputs a symmetric key K.
5. Backward Recovery
   - A deterministic backward recovery algorithm *Recov*. It takes sk_S, pk_S, pk_R and C_{KGE} as input and returns $\perp$ or a symmetric key K. *Recov* consists of two algorithms KAC and KD. The former is for checking key authenticity and the latter for key decapsulation.
     - KAC takes the sender's private/public key pair (sk_S, pk_S), the receiver's public key pk_R and C_{KGE} as input, and outputs either "OK" or $\perp$.

– If KAC does not return $\perp$, KD takes as input the sender's private/public key pair (sk_S, pk_S), the receiver's public key pk_R and C_{KGE}, and outputs a symmetric key K.

Remark. Note that the description of KEM-BR is based on the model for signcryptions [5][7]. Compared with signcryptions [5][7], in our model authenticity and integrity checks, however, are separated. That is, the receiver only has to execute integrity check, while the sender has to execute authenticity check. A further difference is that the generation of a symmetric key (ephemeral key) K does not involve the receiver's (or sender's) private key in our model. Hence, the adversary does not have to query the symmetric key generation "oracle" in the following security models.

3.3 Security models

3.3.1 IND-CCA2 security for KEM-BR in the two-user setting

The IND-CCA2 game for KEM-BR is played by two parties, an adversary and a challenger.

1. The challenger generates a common parameter $I \leftarrow Com(1^k)$, a sender's private/public key pair $(sk_S, pk_S) \leftarrow Key_S(I)$ and a receiver's key pair $(sk_R, pk_R) \leftarrow Key_R(I)$.

2. Given (I, pk_S, pk_R), the adversary A can make the following three kinds of queries to the corresponding oracles.

   - The encapsulation query q_{Encap}. A forwards the public key pk_R as the encapsulation query q_{Encap} to the encapsulation oracle $\mathcal{O}_{Encap}$. Upon receiving q_{Encap}, $\mathcal{O}_{Encap}$ runs $KGE(I, sk_S, pk_S, pk_R)$ and returns C_{KGE}.
   - The decapsulation query q_{Decap}. A forwards (pk_R, C_{KGE}) as the decapsulation query q_{Decap} to the decapsulation oracle $\mathcal{O}_{Decap}$. Upon receiving q_{Decap}, $\mathcal{O}_{Decap}$ returns $Decap$ $(pk_S,\ sk_R,\ pk_R, C_{KGE})$.
   - The recovery query q_{Recov}. A forwards C_{KGE} as the recovery query q_{Recov} to the recovery oracle $\mathcal{O}_{Recov}$. Upon receiving q_{Recov}, $\mathcal{O}_{Recov}$ returns $Recov(sk_S, pk_S, pk_R,\ C_{KGE})$.

3. The challenger computes $(K_0, s) \overset{R}{\leftarrow} Sym(I)$ and the challenging encapsulation $C^*_{KGE} \leftarrow$ $Encap$ (sk_S, pk_S, pk_R, s). The challenger then generates a random symmetric key K_1 in the range of Sym and a random bit $b \in \{0, 1\}$. It sends (K_b, C^*_{KGE}) back to A.

4. A can forward the same kinds of queries as previously, except C^*_{KGE} in the decapsulation query q_{Decap} and the recovery query q_{Recov}.

5. A terminates by returning a guess b' for the value of b.

We say that the adversary wins the above game if $b' = b$. The advantage of A is defined as

$$|\Pr[A \text{ wins the game}] - 1/2|.$$

KEM-BR is IND-CCA2 secure if, for any PPT adversary A, the advantage of A is negligible with respect to the security parameter 1^k.

3.3.2 BR unforgeability in the two-user setting

For the authenticity and the integrity of KEM-BR, we require that it should be infeasible for an adversary to forge a valid C_{KGE}. Note that the receiver's key pair is fixed before the adversary issues any query in the two-user setting. The attack game of unforgeability goes as follows.

1. The challenger first generates $I \overset{R}{\leftarrow} Com(1^k)$, $(sk_S, pk_S) \overset{R}{\leftarrow} Key_S(I)$ and $(sk_R, pk_R) \overset{R}{\leftarrow}$ $Key_R(I)$. It then passes (I, pk_S, sk_R, pk_R) over to the adversary.

2. The adversary A is given access to $\mathcal{O}_{Encap}$ and $\mathcal{O}_{Recov}$ as defined in the IND-CCA2 game. A terminates by outputting C_{KGE}.

We say that the adversary wins the above game if the following conditions hold.

- C_{KGE} is not returned by $\mathcal{O}_{Encap}$ with pk_R as q_{Encap}.
- $\perp \not\leftarrow KAC(sk_S, pk_S, pk_R, C_{KGE})$ or $\perp \not\leftarrow IC(pk_S, sk_R, pk_R, C_{KGE})$.

Security model in the multi-user setting. The corresponding security model in the multi-user setting for KEM-BR is similar to that in the two-user setting, except that the adversary can issue any public key pk as q_{Encap} and pk may be generated by the adversary at his will.

4 A generic construction

In this section, we show how to construct a secure KEM-BR in the two-user setting from an IND-CPA KEM and a weakly unforgeable signature. The generic construction, which is based on the "encrypt then sign" techniques [2], goes as follows.

1. Using a randomness reusing (RR) technique, an IND-CPA KEM can be converted to an IND-CPA two-receiver KEM[1]. The two "receivers" are the receiver R and the sender S. (Notice that, not all the IND-CPA KEM can be converted to an IND-CPA two-receiver KEM using RR [4].) Let $KEM_2 = (Com_2, Key_2, KGE_2, KD_2)$ denote the resulting two-receiver KEM and $C = (C_R, C_S)$ denote the output of $KGE_2(pk_S, pk_R)$.

2. Apply a weakly unforgeable signature Sig to C and output $(C, Sig(sk_{Sig}, C))$, where sk_{Sig} denotes the sender's signing key. We note that Sig plays the role of KA in the KEM-BR model.

3. For decapsulation and backward recovery, $Sig(sk_{Sig}, C)$ should be verified first using the sender's public key for the signature. If the signature is valid, decapsulating C is carried out as in the two-receiver KEM. That is, the receiver runs $KD_2(sk_R, C)$ for decapsulation and the sender runs $KD_2(sk_S, C)$ for backward recovery. As for the security of the resulting KEM-BR, we have the following Theorems 1 and 2.

Theorem 1 *KEM-BR is IND-CCA2 secure in the two-user setting if the underlying two-receiver KEM is IND-CPA and the signature is weakly unforgeable.*

Proof. If there exists a PPT algorithm A that breaks the IND-CCA2 security of KEM-BR in the two-user setting, we show that one can use A to construct a PPT algorithm B to break the IND-CPA security of the underlying two-receiver KEM or the weak unforgeability of the signature.

Given the sender's public key pk_S and the receiver's public key pk_R, B generates a set of ciphertexts $List = \{C_1, C_2, ..., C_n\}$ using $KGE_2(pk_S, pk_R)$ and stores the corresponding key list $\mathcal{K} = \{K_1, ..., K_n\}$, where n is the maximum number of encapsulation queries issued by A. When receiving the challenge (K_b^*, C^*) of the IND-CPA game, B sends $List$ together with the challenge ciphertext C^* to the challenger of the underlying signature $Challenger_{Sig}$. $Challenger_{Sig}$ runs the key generation algorithm of the underlying signature scheme to get the public/secret key pair (pk_{Sig}, sk_{Sig}), and returns the corresponding signature list $List_{Sig} = \{\sigma_1, ..., \sigma_n\}$ and σ^* for the messages in $List$ and C^*. Upon receiving $List_{Sig}$ and σ^*, B sets the ciphertext list $\mathcal{C} = \{(C_1, \sigma_1), ..., (C_n, \sigma_n)\}$ and the queried list $\mathcal{Q} = \{\emptyset\}$, and defines the challenge of the IND-CCA game as $(K_b^*; (C^*, \sigma^*))$. Then B defines the sender's public key of KEM-BR as (pk_S, pk_{Sig}) and the receiver's public key as pk_R, and simulates the environment of an IND-CCA2 game of KEM-BR for A as follows.

In the first phase of the game, A can make the following three kinds of queries.

- The encapsulation query q_{Encap}. When A forwards q_{Encap}, B randomly chooses a ciphertext $(C_i, \sigma_i) \in \mathcal{C}/\mathcal{Q}$, adds (C_i, σ_i) to $\mathcal{Q}$, and sends (C_i, σ_i) to A.

[1]See Appendix A for two-receiver KEM

- The decapsulation query q_{Decap}. On receiving $q_{Decap} = (C, \sigma)$, B checks the validity of σ. There are two cases that we have to take into account.
  - If σ is not valid, B returns $\perp$.
  - If σ is valid, B checks whether $(C, \sigma) \in \mathcal{C}$. If $(C, \sigma) \in \mathcal{C}$, B returns the corresponding key in $\mathcal{K}$. If $(C, \sigma) \notin \mathcal{C}$ and $(C, \sigma) \neq (C^*, \sigma^*)$, B outputs (C, σ) as a valid forgery for the underlying signature and terminates. Otherwise, $(C, \sigma) = (C^*, \sigma^*)$ and B terminates by outputting "failure".
- The recovery query q_{Recov}. B acts in the same way as what he does in the decapsulation query.

B sends the challenge $(K_b^*; (C^*, \sigma^*))$ to A. In the second phase, A can forward the same kinds of queries as previously, except $(C, \sigma) = (C^*, \sigma^*)$ in the key decapsulation query and in the recovery query. Finally, A outputs a bit b', which is also the output of B.

If B does not terminates during the decapsulation query or the recovery query, B perfectly simulates the KEM-BR scheme in the two-user setting for A. According to the assumption that A outputs the right b' with a non-negligible advantage, B outputs the right b' in the above IND-CPA game for the underlying two-receiver KEM with a non-negligible advantage. If B terminates and outputs (C, σ), σ is a valid forgery for the underlying signature. Note that the probability that C^* appears in the queries of the first phase is negligible, since C^* is independent of $List$ and $List_{Sig}$. □

Theorem 2 *KEM-BR is existentially unforgeable in the two-user setting if the underlying signature scheme is weakly unforgeable.*

Proof. If there exists a PPT adversary A that can break the unforgeability of KEM-BR, we can use A as an oracle to construct a PPT adversary B that breaks the weak unforgeability of the underlying signature.

1. B generates the valid common parameter I and the public/secret encryption keys (pk_S, sk_S, pk_R, sk_R) for both the sender and the receiver of the underlying KEM. Using the underlying KEM, B computes a list of ciphertexts $List = \{C_1, ..., C_n\}$, where n denotes the maximum number of the encapsulation queries made by A. Send $List$ to the challenger as the messages that will be signed.

2. The challenger runs the key generation algorithm of the underlying signature scheme to get private/public key pair (sk_{Sig}, pk_{Sig}), and returns the corresponding signatures list $List_{Sig} = \{\sigma_1, ..., \sigma_n\}$ for the messages in $List$.

3. Upon receiving $List_{Sig}$ and pk_{Sig}, B defines the sender's public key of KEM-BR as (pk_S, pk_{Sig}) and the receiver's public key as pk_R, computes the ciphertext list $\mathcal{C} = \{(C_1, \sigma_1), ..., (C_n, \sigma_n)\}$ and sets the queried list $\mathcal{Q} = \{\emptyset\}$. Then B can answer the encapsulation queries from A as in the proof of Theorem 1. Finally, A outputs a valid forgery (C^*, σ^*) with a non-negligible probability. As a result, B can output σ^* as the forgery for the underlying signature. Note that the success probability of B is the same as that of A.

This completes the proof for Theorem 2. □

PKE-BR in the multi-user setting. An intuitive method of constructing PKE-BR scheme in the multi-user setting from KEM-BR is as follows.

1. Convert the KEM-BR scheme to a Tag-KEM using methods of [1]. More precisely, the signature of KEM-BR is computed as $Sig(sk_{Sig}, C, \tau)$, where τ is a tag. To prove the CCA2 security of the resulting Tag-KEM, the signature Sig is required to be existentially unforgeable against chosen message attack instead of weakly unforgeable.

2. Replace the tag with an IND-CCA2 secure DEM secure against passive attacks. The resulting scheme is a CCA2 Tag-KEM/DEM in the two-user setting.

3. Apply a generic transformation outlined in [2] to convert the scheme for the two-user setting to one for the multi-user setting. That is, the sender's public key is included in DEM as part of a plaintext and the receiver's public key is included in the signature, e.g., $Sig(sk_{Sig}, C, \tau, pk_R)$ where $\tau = DEM(K, m, pk_S)$. Notice that the underlying encryption DEM should be IND-CCA2 secure in order to prevent the adversary from modifying $DEM(K, m, pk_S)$ for a new related ciphertext [2], say, $DEM(K, m, pk')$.

We emphasize that since the above transformation is an intuitive one, rigorous security proofs of the resulting scheme need to be further investigated when concrete KEM/DEM and signature schemes are used to instantiate the construction.

5 An efficient construction — Tag based KEM-BR

To construct a more efficient KEM-BR, we slightly relax the previous security requirements for KEM-BR: the sender only has to check the integrity and authenticity of the sender's part of a ciphertext. In other words, during the backward recovery phase, the sender runs KAC on input $(sk_S, pk_S, pk_R, C_{Recov})$, where $C_{Recov} = C_S$ denotes the sender's part of C_{KGE}. With the modified security requirement, we can provide a concrete tag based KEM-BR (TBR) which is provably secure in the multi-user setting. The corresponding security model of TBR in the multi-user setting is similar to that of KEM-BR, except that the adversary is able to forward any public key as q_{Encap}, and the tag τ is included in the computation of KGE. In addition, the adversary can choose any tag during the challenge phase. More details of the security model follow.

5.1 Security model of TBR in the multi-user setting

IND-CCA2. The IND-CCA2 game for TBR is played by two parties, the adversary and the challenger.

1. The challenger generates a common parameter $I \leftarrow Com(1^k)$, a sender's private/public key pair $(sk_S, pk_S) \leftarrow Key_S(I)$ and a receiver's key pair $(sk_R, pk_R) \leftarrow Key_R(I)$.

2. The adversary A is given (I, pk_S, pk_R) and can make the following three kinds of queries.

   - The encapsulation query q_{Encap}. A forwards (pk, τ) as the encapsulation query q_{Encap} to the oracle $\mathcal{O}_{Encap}$, where the public key pk can be generated by the adversary and τ is the tag. Upon receiving q_{Encap}, the oracle $\mathcal{O}_{Encap}$ runs $KGE(I, sk_S, pk_S, pk, \tau)$ and returns C_{KGE}.
   - The decapsulation query q_{Decap}. A forwards (pk, C_{KGE}, τ) as the decapsulation query q_{Decap} to the decapsulation oracle $\mathcal{O}_{Decap}$. Upon receiving q_{Decap}, $\mathcal{O}_{Decap}$ returns $Decap(pk, sk_R, pk_R, C_{KGE}, \tau)$.
   - The recovery query q_{Recov}. A forwards (C_{Recov}, τ) as the decapsulation query q_{Recov} to the recovery oracle $\mathcal{O}_{Recov}$, where C_{Recov} is part of C_{KGE}. Upon receiving q_{Recov}, $\mathcal{O}_{Recov}$ returns $Recov(sk_S, pk_S, C_{Recov}, \tau)$.

3. A forwards τ^* to the challenger.

4. The challenger computes $(K_0, s) \overset{R}{\leftarrow} Sym(I)$, generates a random symmetric key K_1 in the range of Sym and a random bit $b \in \{0, 1\}$. Then, the challenger computes the encapsulation $C^*_{KGE} \leftarrow Encap\ (sk_S, pk_S, pk_R, s, \tau^*)$ and sends (K_b, C^*_{KGE}) back to A.

   A can make the same kinds of queries as previously, except that it cannot make $(pk_R, C^*_{KGE}, \tau^*)$ as a decapsulation query q_{Decap} or (C^*_{Recov}, τ^*) as a recovery query q_{Recov}, where C^*_{Recov} is part of C^*_{KGE}.

5. A terminates by returning a guess b' for the value of b.

We say the adversary wins the above game if $b' = b$.

Unforgeability. For the TBR scheme, we only require that it should be infeasible for an adversary to forge a valid C_{Recov}. The adversary can choose the receiver to which the adversary wishes to forge. The attack game of unforgeability runs as follows.

1. The challenger generates the common parameter $I \overset{R}{\leftarrow} Com(1^k)$ and a sender key pair $(sk_S, pk_S) \overset{R}{\leftarrow} Key_S(I)$. Send I and pk_S to the adversary.

2. The adversary A can forward encapsulation queries and recovery queries as defined in the IND-CCA2 game. A terminates by outputting a fixed receiver key pair (pk_R, sk_R) and C_{Recov}.

We say the adversary wins the above game if C_{Recov} is not returned as part of C_{KGE} by $\mathcal{O}_{Encap}$ with pk_R as q_{Encap}, and $\perp \not\leftarrow KAC(sk_S, pk_S, pk_R, C_{Recov})$.

5.2 Tag based KEM-BR scheme (TBR)

Our TBR is based on the Hofheinz-Kiltz KEM [11] and a message authentication code (MAC). In our construction, the validity of MAC can be checked by the sender only. Furthermore, we take advantage of a target collision resistant hash function to guarantee that the receiver can check the integrity of the whole ciphertext. The TBR scheme is described below.

Common Parameter Generation. A security parameter 1^k, a target collision resistant hash function H with l_H-bit output, and a BBS pseudorandom number generator BBS_N with l_K-bit output. A MAC that is existentially unforgeable against chosen message attack. $N = (2p+1)(2q+1)$ is a Blum integer such that p and q are two primes and $|p| = |q| = k$. g is the generator of QR_N. Note that the factorization of N is kept secret.

Key Generation. The sender's private/public key pair is (sk_S, pk_S) where sk_S=$(x_S,\ sk_{MAC})$, $pk_S = y_S$ such that $x_S \overset{R}{\leftarrow} [(N-1)/4]$, $y_S = g^{x_S \cdot 2^{l_K+l_H}}$ and sk_{MAC} is the key of MAC. The receiver's private/public key pair is (sk_R, pk_R), where $sk_R = x_R$, $pk_R = y_R$ such that $x_R \overset{R}{\leftarrow} [(N-1)/4]$, $y_R = g^{x_R \cdot 2^{l_K+l_H}}$.

Symmetric Key Generation and Encapsulation by sender. $KGE(I,\ sk_S,\ pk_S, pk_R, \tau)$

1. Choose at random $r \in [(N-1)/4]$ and compute $U = g^{r \cdot 2^{l_K+l_H}} \mod N$ and $K = BBS_N(g^{r \cdot 2^{l_H}})$.

2. Compute $V_1 = |(g^{v_1} y_R)^r| \mod N$, $V_2 = |(g^{v_2} y_S)^r| \mod N$ and $\tau_{MAC} = MAC(sk_{MAC}, U, V_2, \tau)$, where $v_1 = H(U, V_2, \tau, \tau_{MAC})$, $v_2 = H(U, \tau)$.

Output $C = (U, V_1, V_2, \tau_{MAC})$.

Decapsulation by receiver. $Decap(sk_R, C, \tau)$
Given $C = (U, V_1, V_2, \tau_{MAC})$ and τ,

1. Check $(V_1^2)^{2^{l_K+l_H}} \overset{?}{=} (U^2)^{v_1 + x_R 2^{l_K+l_H}} \mod N$, where $v_1 = H(U, V_2, \tau, \tau_{MAC})$. If the equation holds, it outputs "OK"; otherwise, it outputs $\perp$ and terminates.

2. Compute $a, b, c \in \mathbb{Z}$ such that $2^c = \gcd(v_1, 2^{l_K+l_H}) = av_1 + b2^{l_K+l_H}$. Then compute a symmetric key $K = BBS_N(((V_1^2)^a \cdot (U^2)^{b-ax_R})^{2^{l_H-c-1}})$ which is the output.

Backward Recovery by sender. $Recov(sk_S, C_{Recov}, \tau)$
Given $C_{Recov} = (U, V_2, \tau_{MAC})$ and τ,

1. Check $MAC(sk_{MAC}, U, V_2, \tau) \overset{?}{=} \tau_{MAC}$. If the equation holds, $Recov$ returns "OK"; otherwise, it returns $\perp$ and terminates.

2. Compute a, b and c such that $2^c = \gcd(v_2, 2^{l_K+l_H}) = av_2 + b2^{l_K+l_H}$. Then compute $K = BBS(((V_2^2)^a \cdot (U^2)^{b-ax_S})^{2^{l_H-c-1}})$ which is the output.

Correctness. Since the correctness of the computation of K in *Decap* and *Recov* can be verified in a similar way as in [11], we only show that both $(V_1^2)^{2^{l_K+l_H}} = (U^2)^{v_1+x_R 2^{l_K+l_H}} \mod N$ and $K = BBS_N(((V_1^2)^a \cdot (U^2)^{b-ax_R})^{2^{l_H-c-1}})$ hold for a valid ciphertext C.

Given a valid ciphertext $C = (U, V_1, V_2, \tau_{MAC})$, we have

$$\begin{aligned}(V_1^2)^{2^{l_K+l_H}} &= ((g^{v_1} y_R)^r)^{2 \cdot 2^{l_K+l_H}} \\ &= (g^{v_1+x_R 2^{l_K+l_H}})^{r \cdot 2 \cdot 2^{l_K+l_H}} \\ &= (g^{r \cdot 2^{l_H+l_K}})^{2 \cdot (v_1+x_R \cdot 2^{l_K+l_H})} \\ &= (U^2)^{v_1+x_R \cdot 2^{l_K+l_H}}\end{aligned}$$

where $v_1 = H(U, V_2, \tau, \tau_{MAC})$, and

$$\begin{aligned}(V_1^2)^a \cdot (U^2)^{b-ax_R} &= (g^{v_1+x_R \cdot 2^{l_K+l_H}})^{2ra} \cdot (g^{2^{l_K+l_H}})^{2rb-2rax_R} \\ &= g^{(v_1+x_R \cdot 2^{l_K+l_H})2ra - x_R 2^{l_K+l_H} 2ra + 2^{l_K+l_H} 2rb} \\ &= g^{2r(av_1+b2^{l_K+l_H})} \\ &= g^{2^{c+1} r},\end{aligned}$$

Hence $BBS(((V_2^2)^a \cdot (U^2)^{b-ax_S})^{2^{l_H-c-1}}) = BBS((g^{2^{c+1}r})^{2^{l_H-c-1}}) = BBS(g^{r \cdot 2^{l_H}}) = K$.

5.3 IND-CCA2 security of TBR

Theorem 3 *TBR is IND-CCA2 secure if BBS is pseudorandom, MAC is existentially unforgeable against chosen message attack and DDH assumption holds in QR_N.*

Proof. If there exists a PPT adversary A to break the IND-CCA2 security of TBR, we may construct a PPT adversary D to break the security of BBS as in [11]. However, the proof technique in [11] cannot be applied to the construction of D directly, since the setting of the simulated receiver's public key is related to the adversary A's challenge tag τ^*. That is, D cannot compute the simulated public key without τ^*, while A forwards τ^* only after receiving the simulated public key. To solve the above problem, we introduce a new game, called Game 2, which is similar to the standard IND-CCA2 game, except that the challenger randomly chooses τ^{**} and compute the challenge ciphertext using τ^{**} instead of τ^*. Then, we show that the adversary's view in Game 1 and Game 2 are indistinguishable. Therefore, to prove the security of TBR in the standard IND-CCA2 game (Game 1), we only have to prove its security in Game 2.

Game 1. This is the same as the standard IND-CCA2 game for TBR. The challenger picks at random $r^* \in [(N-1)/4]$ and computes a challenge ciphertext $(K_b; C^*) = (K_b; U^*, V_1^*, V_2^*, \tau_{MAC}^*)$, where $U^* = g^{r^* \cdot 2^{l_K+l_H}}$, $V_1^* = |(g^{v_1^*} \cdot y_R)^{r^*}|$, $V_2^* = |(g^{v_2^*} \cdot y_S)^{r^*}|$, $v_1^* = H(U^*, V_2^*, \tau^*, \tau_{MAC}^*)$, $v_2^* = H(U^*, \tau^*)$, $\tau_{MAC}^* = MAC(sk_{MAC}, U^*, V_2^*, \tau^*)$ and τ^* is a challenge tag chosen by the adversary.

Game 2. Game 2 is similar to Game 1, except that here the challenger picks at ransom τ^{**} and τ_{MAC}^{**}, and computes $V_1^{**} = |(g^{v_1^{**}} \cdot y_R)^{r^*}|$, $V_2^{**} = |(g^{v_2^{**}} \cdot y_S)^{r^*}|$, where $v_1^{**} = H(U^*, V_2^{**}, \tau^{**}, \tau_{MAC}^{**})$, $v_2^{**} = H(U^*, \tau^{**})$. The challenge ciphertext is $(K_b; C^*) = (K_b; U^*, V_1^{**}, V_2^{**}, \tau_{MAC}^*)$, where $\tau_{MAC}^* = MAC(sk_{MAC}, U^*, V_2^{**}, \tau^*)$ and τ^* is chosen by the adversary as in Game 1.

Note that the only difference between Game 1 and Game 2 is the challenge ciphertexts. Hence, in order to prove the indistinguishability between Game 1 and Game 2, we only have to prove that $(V_1^*, V_2^*, \tau_{MAC}^*)$ in Game 1 and $(V_1^{**}, V_2^{**}, \tau_{MAC}^*)$ in Game 2 are indistinguishable. To that end, we need the following claim.

Claim 1 *Let g_1 and g_2 be the generators of QR_N and $f_{p,q}(\cdot,\cdot,\cdot,\cdot)$ be a PPT computable function whose output is an element in QR_N. (p, q) is the auxiliary input of f. Assume DDH assumption* *holds in QR_N, it is infeasible for any PPT adversary A' to distinguish between tuple T_0 and tuple T_1*

$$\begin{aligned} T_0 &= \{g_1, g_2, g_1^r, g_2^r \cdot f_{p,q}(g_1, g_2, g_1^r, \tau_0)\} \\ T_1 &= \{g_1, g_2, g_1^r, g_2^r \cdot f_{p,q}(g_1, g_2, g_1^r, \tau_1)\} \end{aligned}$$

where r is a random element in $[(N-1)/4]$, τ_1 is chosen uniformly at random, and τ_0 is generated by the adversary.

Proof. In Claim 1, we implicitly define the following game between the challenger and the adversary A'. Given (N, g_1, g_2, g_1^r), the adversary A' computes τ_0 and sends τ_0 to the challenger. Then the challenger returns T_b, where $b \overset{R}{\leftarrow} \{0,1\}$. A' aims to tell whether $b = 0$ or 1. More details are described in the following.

We construct a series of tuples to show the indistinguishability between T_0 and T_1.

1. Tuple 1 $= \{g_1, g_2, g_1^r, g_2^r\}$, where r is a random element in $[|QR_N|]$.
2. Tuple 2 $= \{g_1, g_2, g_1^r, Q\}$, where Q is a random element in QR_N.

   Let ε_{DDH} be the advantage with which A' can solve the DDH problem. Tuple 1 and Tuple 2 can be distinguished with advantage at most ε_{DDH}, if DDH assumption holds. That is,

   $$|\Pr[A'(\text{Tuple 1}) = 1] - \Pr[A'(\text{Tuple 2}) = 1]| \leq \varepsilon_{DDH}. \tag{1}$$

   where ε_{DDH} is negligible.
3. Tuple 3 $= \{g_1, g_2, g_1^r, Q \cdot f_{p,q}(g_1, g_2, g_1^r, \tau_0)\}$, where τ_0 is generated by the adversary A' on input (N, g_1, g_2, g_1^r).
4. Tuple 4 $= \{g_1, g_2, g_1^r, Q \cdot f_{p,q}(g_1, g_2, g_1^r, \tau_1)\}$, where τ_1 is chosen randomly in QR_N.

   The distribution of Tuple 2, Tuple 3 and Tuple 4 are identical, since Q is a random element in QR_N. We have

   $$\Pr[A'(\text{Tuple 2}) = 1] = \Pr[A'(\text{Tuple 3}) = 1] = \Pr[A'(\text{Tuple 4}) = 1]$$
5. Tuple 5 $= \{g_1, g_2, g_1^r, g_2^r \cdot f_{p,q}(g_1, g_2, g_1^r, \tau_0)\}$
6. Tuple 6 $= \{g_1, g_2, g_1^r, g_2^r \cdot f_{p,q}(g_1, g_2, g_1^r, \tau_1)\}$

   Since $|\Pr[A'(\text{Tuple 1}) = 1] - \Pr[A'(\text{Tuple 2}) = 1]| \leq \varepsilon_{DDH}$, we have

   $$|\Pr[A'(\text{Tuple 3}) = 1] - \Pr[A'(\text{Tuple 5}) = 1]| \leq \varepsilon_{DDH}. \tag{2}$$

   Similarly, $|\Pr[A'(\text{Tuple 4}) = 1] - \Pr[A'(\text{Tuple 6}) = 1]| \leq \varepsilon_{DDH}$.

   Therefore, we have

   $$\begin{aligned} & |\Pr[A'(\text{Tuple 5}) = 1 - A'(\text{Tuple 6}) = 1]| \\ \leq\ & |\Pr[A'(\text{Tuple 5}) = 1] - \Pr[A'(\text{Tuple 3}) = 1]| + \\ & \quad |\Pr[A'(\text{Tuple 6}) = 1] - \Pr[A'(\text{Tuple 3}) = 1]| \\ \leq\ & |\Pr[A'(\text{Tuple 5}) = 1] - \Pr[A'(\text{Tuple 3}) = 1]| + \\ & \quad |\Pr[A'(\text{Tuple 6}) = 1] - \Pr[A'(\text{Tuple 4}) = 1]| \\ \leq\ & 2\varepsilon_{DDH} \end{aligned} \tag{3}$$

   Next, we consider the indistinguishability between T_0 and Tuple 5. Conditioned on that $r \in [|QR_N|]$, T_0 and Tuple 5 are identically distributed. That is,

   $$\Pr[A'(T_0) = 1 | r \in [|QR_N|]] = \Pr[A'(\text{Tuple 5}) = 1 | r \in [|QR_N|]] \tag{4}$$

Hence,

$$|\Pr[A'(T_0)=1]-\Pr[A'(\text{Tuple 5})=1]| \le \Pr[r \notin [|QR_N|]] \tag{5}$$

where $\Pr[r \notin [|QR_N|]]$ denotes that $r \in [(N-1)/4]$ but $r > |QR_N|$.

Since $|QR_N| = pq = (2p+1)(2q+1)/4 - (2p+2q+1)/4 = (N-1)/4 - (p+q)/2$, we have $\Pr[r \notin [|QR_N|]] = ((p+q)/2)/((N-1)/4) \le 2^{-k+1}$. Likewise, we have

$$|\Pr[A'(T_1)=1]-\Pr[A'(\text{Tuple 6})=1]| \le 2^{-k+1} \tag{6}$$

Using 5, 6 and 3, we get $|\Pr[A'(T_1)=1]-\Pr[A'(T_0)=1]| \le 2^{-k+2}+2\varepsilon_{DDH}$.

The proof of Claim 1 is complete. □

The following Claim 2 can be proven in a way analogous to the proof for Claim 1.

Claim 2 *Let g_1, g_2 and g_3 be the generators of QR_N and $f'_{p,q}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ be a PPT computable function whose output is an element in QR_N. (p,q) is the auxiliary input of f'. Assume DDH assumption holds in QR_N, it is infeasible for any PPT adversary A' to distinguish between tuple T'_0 and tuple T'_1*

$$\begin{aligned} T'_0 &= \{g_1,g_2,g_3,g_1^r,g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_0)\} \\ T'_1 &= \{g_1,g_2,g_3,g_1^r,g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_1)\} \end{aligned}$$

where r is a random element in $[(N-1)/4]$, τ_1 is chosen randomly, and τ_0 is generated by the adversary.

To prove this claim, we notice that if there exists a PPT algorithm A' which can distinguish T'_0 and T'_1, we can then construct from A' a new PPT algorithm A to distinguish T_0 and T_1, which contradicts Claim 1. Details of the proof are similar to that for Claim 1 and hence are omitted.

Claim 3 *Let g_1, g_2 and g_3 be the generators of QR_N and $f_{p,q}(\cdot,\cdot,\cdot,\cdot)$ and $f'_{p,q}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ be two PPT computable functions whose output is an element in QR_N. (p,q) is the auxiliary input of f and f'. If*

- *T_0 and T_1 can be distinguished with advantage at most $2^{-k+2}+2\varepsilon_{DDH}$.*
- *T'_0 and T'_1 can be distinguished with advantage at most $2^{-k+2}+2\varepsilon_{DDH}$*

then T''_0 and T''_1 can be distinguished with advantage at most $2^{-k+2}+2\varepsilon_{DDH}$, where

$$\begin{aligned} T_0 &= \{g_1,g_2,g_1^r,g_2^r\cdot f_{p,q}(g_1,g_2,g_1^r,\tau_0)\}, \\ T_1 &= \{g_1,g_2,g_1^r,g_2^r\cdot f_{p,q}(g_1,g_2,g_1^r,\tau_1)\}, \\ T'_0 &= \{g_1,g_2,g_3,g_1^r,g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_0)\}, \\ T'_1 &= \{g_1,g_2,g_3,g_1^r,g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_1)\}, \\ T''_0 &= \{g_1,g_2,g_3,g_1^r,g_2^r\cdot f_{p,q}(g_1,g_2,g_1^r,\tau_0),g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_0)\}, \\ T''_1 &= \{g_1,g_2,g_3,g_1^r,g_2^r\cdot f_{p,q}(g_1,g_2,g_1^r,\tau_1),g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_1)\}, \end{aligned}$$

r is a random element in $[(N-1)/4]$, τ_0 is generated by the adversary A and τ_1 is chosen randomly.

To prove Claim 3, we note that if Claim 3 does not hold, we can easily construct an efficient algorithm to distinguish T_0 and T_1 (or T'_0 and T'_1), which contradicts Claim 1 (or Claim 2).

Claim 3 implies the indistinguishability between (V_1^*, V_2^*) and (V_1^{**}, V_2^{**}), which also implies the indistinguishability between $(V_1^*, V_2^*, \tau^*_{MAC})$ and $(V_1^{**}, V_2^{**}, \tau^*_{MAC})$. (Otherwise, MAC will serve as an efficient distinguishing algorithm.) That is, we can set $g_1 = g^{2^{l_K+l_H}}$, $g_2 = y_S$, $g_3 = y_R$, $g_1^r = U^*$, $\tau_0 = (\tau^*, \tau^*_{MAC})$, $\tau_1 = (\tau^{**}, \tau^{**}_{MAC})$ and

$$\begin{aligned} &|g_2^r\cdot f_{p,q}(g_1,g_2,g_1^r,\tau_0)| = |y_S^{r^*}\cdot(g^{H(U^*,\tau^*)})^{r^*}| = |(g^{v_2^*}\cdot y_S)^{r^*}| = V_2^*, \\ &|g_2^r\cdot f_{p,q}(g_1,g_2,g_1^r,\tau_1)| = |y_S^{r^*}\cdot(g^{H(U^*,\tau^{**})})^{r^*}| = |(g^{v_2^{**}}\cdot y_S)^{r^*}| = V_2^{**}, \\ &|g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_0)| = |y_R^{r^*}\cdot(g^{H(U^*,V_2^*,\tau^*,\tau^*_{MAC})})^{r^*}| = |(g^{v_1^*}\cdot y_R)^{r^*}| = V_1^*, \\ &|g_3^r\cdot f'_{p,q}(g_1,g_2,g_3,g_1^r,g_2^r,\tau_1)| = |y_R^{r^*}\cdot(g^{H(U^*,V_2^{**},\tau^{**},\tau^{**}_{MAC})})^{r^*}| = |(g^{v_1^{**}}\cdot y_R)^{r^*}| = V_1^{**}. \end{aligned}$$

Due to Claim 3, we have that the challenge ciphertexts of Game 1 and Game 2 can be distinguished with advantage at most $2^{-k+2} + 2\varepsilon_{DDH}$. Hence, we have

Claim 4 *$|\Pr[Game\ 1] - \Pr[Game\ 2]| \leq 2^{-k+2} + 2\varepsilon_{DDH}$, where $\Pr[Game\ i]$ denotes the probability that the adversary wins Game i, for $i = 1$ and 2.*

Next, we prove the IND-CCA2 security of TBR in Game 2. If there exists an adversary A which can break the IND-CCA2 security of TBR in Game 2, we can construct a BBS distinguisher D to break the BBS generator security. Given (N, z, W), the aim of D is to distinguish whether W is a pseudorandom string generated by $BBS_N(z^{2^{-l_K}})$ or a random string in $\{0,1\}^{l_K}$. Actually, D can set the receiver's public key and the challenge ciphertext by selecting τ^{**} and τ^{**}_{MAC} randomly. More precisely, D can set $U^* = z$, $K = W$, $V_1^{**} = |U^{*\beta_1}|$, $V_2^{**} = |U^{*\beta_2}|$, $v_1^{**} = H(U^*, V_2^{**}, \tau^{**}, \tau^{**}_{MAC})$, $v_2^{**} = H(U^*, \tau^{**})$, $y_R = g^{\beta_1 \cdot 2^{l_K + l_H} - v_1^{**}}$, $y_S = g^{\beta_2 \cdot 2^{l_K + l_H} - v_2^{**}}$, where $\beta_1 \overset{R}{\leftarrow} [(N-1)/4]$, $\beta_2 \overset{R}{\leftarrow} [(N-1)/4]$, τ^{**} and τ^{**}_{MAC} are chosen randomly by D. After receiving τ^* from the adversary, D can compute $\tau^*_{MAC} = MAC(sk_{MAC}, U^*, V_2^{**}, \tau^*)$, where sk_{MAC} is generated by D. Hence, the challenge ciphertext is $(U^*, V_1^{**}, V_2^{**}, \tau^*, \tau^*_{MAC})$. The remaining proof is similar to that of Theorem 3 of [11], except that, for the recovery query q_{Recov}, we have to consider the probability that the adversary can forge a valid MAC. According to Theorem 3 of [11] [2], we have

Claim 5 *$|\Pr[Game\ 2] - 1/2| \leq 2^{-k+3} + \varepsilon_{Hash} + \varepsilon_{BBS} + \varepsilon_{MAC}$, where ε_{Hash} denotes the probability that the target collision happens, ε_{BBS} denotes the advantage that the BBS output can be distinguished from the random string, and ε_{MAC} denotes the probability that the adversary can output a forgery for MAC.*

Due to Claim 4 and Claim 5, we get

$$|\Pr[Game\ 1] - 1/2| \leq 3 \cdot 2^{-k+2} + 2\varepsilon_{DDH} + \varepsilon_{Hash} + \varepsilon_{BBS} + \varepsilon_{MAC}$$

which completes the proof of Theorem 3. □

5.4 Unforgeability

Theorem 4 *TBR is existentially unforgeable if the underlying MAC is existentially unforgeable against chosen message attack.*

The theorem can be proved by contradiction, namely if there exists an efficient algorithm A that breaks the unforgeability of TBR, we can then construct an efficient algorithm B to break the security of the underlying MAC. Descriptions of the proof are straightforward and hence are omitted.

5.5 Implementation

For implementation, H and $HMAC$ can be instantiated with SHA-256 and HMAC-SHA-256, respectively, and AES can be applied for data encryption, where the symmetric key K is of length 128. The computational cost of the decapsulation (or the backward recovery) of TBR is similar to that of the decapsulation of the original Hofheinz-Kiltz encryption scheme [11]. For encapsulation, our scheme requires about one more full exponentiation than that of [11], due to the computation of V_2. However, more than 60% computation of the encapsulation can be processed offline. That is, $P_1 = g^r$, $P_2 = y_S^r$ and $U = P_1^{2^{l_H + l_K}}$ can be precomputed. On input the receiver's public key y_R and the plaintext, compute $V_2 = |P_1^{v_2} P_2|$ and $V_1 = |P_1^{v_1} y_R^r|$, where v_1 and v_2 are very small exponents. More precisely, assume that one regular exponentiation with an exponent of length l requires $1.5l$ modular multiplications and l_N, which is the binary length of N, is 2048. The offline computation of encapsulation requires about $3l_N + l_H + l_K = 6528$ multiplications; the online computation requires about $1.5l_N + 3l_H = 3840$ multiplications.

[2] Theorem 3 of [11] states that $\varepsilon_{KEM} \leq 2^{-k+3} + \varepsilon_{Hash} + \varepsilon_{BBS}$, where ε_{KEM} denotes the advantage that the adversary breaks the security of KEM. Notice that the definition of the advantage is $|\Pr[A \text{ wins the game}] - 1/2|$.

Acknowledgement

We would like to thank Lynn Batten for her invaluable comments.

References

[1] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup. Tag-KEM/DEM: a new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology-EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146. Springer-Verlag, 2005.

[2] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Advances in Cryptology-EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.

[3] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. *Journal of Cryptology*, 20(2):203–235, 2007.

[4] M. Bellare, A. Boldyreva, and J. Staddon. Multi-recipient encryption schemes: Security notions and randomness re-use. In *Public Key Cryptography 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer-Verlag, 2003.

[5] T. E. Bjϕrstad and A. W. Dent. Building better signcryption schemes with Tag-KEMs. In *Public Key Cryptography 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 491–507. Springer-Verlag, 2006.

[6] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudorandom number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[7] A. W. Dent. Hybrid signcryption schemes with insider security. In *Proceedings of 10th Australasian Conference on Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 253–266. Springer-Verlag, 2005.

[8] S. Duan and Z. Cao. Efficient and provably secure multi-receiver identity-based signcryption. In *Information Security and Privacy-ACISP 2006*, volume 4058 of *Lecture Notes In Computer Science*, pages 195–206. Springer-Verlag, 2006.

[9] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptively chosen message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[10] C. G. Günther. An identity-based key-exchange protocol. In *Advances in Cryptology-Eurocrypt 1989*, volume 434 of *Lecture Notes In Computer Science*, pages 29–37. Springer-Verlag, 1989.

[11] D. Hofheinz and E. Kiltz. Practical chosen ciphertext secure encryption from factoring. In *Advances in Cryptology-EUROCRYPT 2009*, volume 5479 of *Lecture Notes In Computer Science*, pages 313–332, Heidelberg, 2009. Springer-Verlag.

[12] S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *Advances in Cryptology-Crypto 2009*, volume 5677 of *Lecture Notes In Computer Science*, pages 654–670. Springer-Verlag, 2009.

[13] K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Public Key Cryptography 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 2002.

[14] B. Libert and J.-J. Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes In Computer Science*, pages 187–200. Springer-Verlag, 2004.

[15] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology-CRYPTO 91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, London, UK, 1991. Springer-Verlag.

[16] S. S. D. Selvi, S. S. Vivek, and C. P. Rangan. A note on the certificateless multi-receiver signcryption scheme. http://eprint.iacr.org/2009/308.pdf.

[17] S. S. D. Selvi, S. S. Vivek, D. Shukla, and C. P. Rangan. Efficient and provably secure certificateless multi-receiver signcryption. In *Provable Security 2008*, volume 5324 of *Lecture Notes In Computer Science*, pages 52–67. Springer-Verlag, 2008.

A Two-receiver KEM

Two-receiver KEM. We briefly describe the two-receiver KEM scheme, which is a special case of the multi-receiver encryption [13]. A two-receiver KEM consists of four PPT algorithms, which are the common parameter generation algorithm Com_2, the key generation algorithm Key_2, the symmetric key generation and encapsulation algorithm KGE_2 and the key decapsulation algorithm KD_2. Com_2 on inputs security parameter 1^k outputs the common parameter I. Key_2 on inputs the common parameter I outputs private/public key pair (sk, pk). Suppose the two receivers are $User_S$ and $User_R$, who have private/public key pair (sk_S, pk_S) and (sk_R, pk_R), respectively. KGE_2 on inputs two receivers' public key (pk_S, pk_R) outputs ciphertext $C = (C_S, C_R)$, where C_S is for receiver $User_S$ and C_R is for receiver $User_R$. KD_2 on inputs (sk_i, C) outputs the ephemeral key K or the error symbol $\perp$, where $i = S$ or R. More precisely, receiver $User_i$ uses a function $TAKE_i$ that on input C outputs C_i and computes K using sk_i and C_i.

IND-CCA2 security of two-receiver KEM. The IND-CCA2 game of a two-receiver KEM is played by two parties, the challenger and the adversary. The game is described as follows.

1. The challenger generates $I \leftarrow Com_2(1^k)$. It runs $Key_2(I)$ and outputs two private/public key pair (sk_S, pk_S) and (sk_R, pk_R). Send (pk_S, pk_R) to the adversary.
2. The adversary is given access to the decapsulation oracles $\mathcal{O}_{sk_S}$ and $\mathcal{O}_{sk_R}$, where $\mathcal{O}_{sk_i}$ on inputs C returns $KD_2(sk_i, C)$, for $i = S$ or R.
3. The challenger computes $(K_0^*, C^*) \leftarrow KGE_2(pk_S, pk_R)$, (where $C^* = (C_S^*, C_R^*)$,) generates a random symmetric key K_1^* and sends (K_b^*, C^*) to the adversary, where $b \overset{R}{\leftarrow} \{0, 1\}$.
4. The adversary can make the decapsulation queries C as in Step 2, except that $C_i^* \neq TAKE_i(C)$ for $i = S$ and R. Finally, the adversary terminates by returning a bit b'.

The adversary wins the game if $b = b'$. A two-receiver KEM is IND-CCA2 secure if, for any PPT adversary, $|Pr[b = b'] - 1/2|$ is negligible with respect to the security parameter 1^k.

The IND-CPA security of two-receiver KEM is defined similarly except that the adversary cannot have access to the decapsulation oracles.