Efficient Public Key Encryption Admitting Decryption by Sender*

Puwen Wei[1] and Yuliang Zheng[2]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, School of Mathematics, Shandong University, Jinan 250100, China
pwei@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 investigates public key encryption that has a desirable feature of allowing the sender of a ciphertext to recover the original plaintext from the ciphertext without relying on a recipient's private decryption key (PKE-SR). We propose two efficient methods for converting KEM/DEM to PKE-SR. The first method, called pre-KEM seeding, can be applied to a large class of KEM/DEM constructions including those based on the discrete logarithm problem. The second method, called post-KEM converging, is more powerful and can be employed to convert any secure KEM/DEM into a secure PKE-SR. Post-KEM converging takes advantages of an interesting property, called collision accessibility, of sibling intractable hashing. For both methods, added costs in ciphertext length and computation are minimal, making them a particularly attractive "drop-in" replacement in applications where plaintexts need to be recovered efficiently by the sender alone.

Keywords: public key encryption, backward recovery, recovery by sender, KEM, DEM.

1 Introduction

Public key encryption admitting message recovery by sender enjoys a useful feature by which the originator of a ciphertext can retrieve the "forgotten" plaintext from the ciphertext, without relying on the private decryption key of the intended recipient. This notion was first introduced by Wei, Zheng and Wang in [12] and called public key encryption with backward recovery in that paper.

* Supported by NSFC Projects (No. 61133013, No. 61070244 and No. 61103237), Research Fund for the Doctoral Program of Higher Education of China (No. 20100131120015) and Interdisciplinary Research Found of Shandong University (No. 2012JC018). Part of the second author's work was done while visiting Shandong University on a Changjiang Scholars program sponsored by the Chinese Ministry of Education and Li Ka Shing Foundation in Hong Kong.

C. Mitchell and S. De Capitani di Vimercati (Eds.): EuroPKI 2012, LNCS 7868, pp. 37–52, 2013.

In this paper we continue this line of research. As backward recovery is implied by decryption by sender, in this paper we will instead use the term of public key encryption with sender recovery, or PKE-SR for short.

One can think of many practical applications of PKE-SR, thanks to its property of allowing the sender to decrypt a ciphertext by herself alone without the need to use a recipient's private decryption key. One example of such applications is secure email communication. Consider a situation where Alice the sender encrypts a message m under the public key of Bob the receiver and passes the resultant ciphertext c to Bob while keeping an identical copy of the ciphertext in a "Sent" folder on Alice's email server which may physically reside in an insecure computing "cloud". We note that other than the ciphertext, Alice may not keep an additional copy of the original message m. At a later time, Alice realizes that she needs to access m which lies in the "Sent" folder albeit in an encrypted form. By the virtue of traditional public key encryption, the only way to get m back is for Alice to ask Bob to decrypt the ciphertext with Bob's decryption key, which may be impractical or undesirable from either Alice or Bob's point of view. This dilemma is readily avoided if Alice and Bob employ PKE-SR or a public key encryption technique that admits of decryption by sender.

In [12], Wei et al. define a security model for PKE-SR and present two methods of constructing PKE-SR from the framework of key-encapsulation mechanism (KEM) and data encapsulation mechanism (DEM) [5]. The first method in [12] is a general one using the "encrypt then sign" paradigm, whereas the second method is more efficient, being based on a concrete public key encryption scheme by Hofheinz and Kiltz [8]. A common thread underlying both methods is the use of an ephemeral key, which is the symmetric key for DEM and encapsulated not only by a receiver's public key but also by a sender's. These two methods are not quite practical due to the fact that at least one additional group element needs to be added to a ciphertext. The inefficiency of the two earlier constructions stems also from an explicit but somewhat excessive requirement of ciphertext authenticity, which allows the sender to check whether the ciphertext is generated by herself. In this paper we remove this requirement for ciphertext authenticity.

A further difference between this paper and [12] lies in the fact that solutions in [12] are more akin to multi-recipient encryption in which the sender is included as one of the recipients. In this paper we address a more challenging problem, namely to search for a solution that is not only efficient but also imposes no requirement for the sender to possess a public/secret key pair.

Our Contributions. Since confidentiality is the basic requirement of any public key encryption, we focus on the confidentiality and the recovery property of public key encryption with sender recovery, which leads us to a simplified security model that does not require ciphertext authenticity. We provide two general methods for translating KEM/DEM to PKE-SR so that the efficiency of a resultant PKE-SR is comparable to that of the original KEM/DEM. With both methods, while the receiver of a message is obviously required to have a public/secret key pair, the sender is required to possess a secret recovery key

only. In practice, the sender may have to memorize a password only which can be used to derive the secret recovery key.

The first method is based on the idea of seeding random numbers used in many public key encryption techniques with pseudo-random numbers being derived from the sender's recovery key. In comparison, the second method focuses on the output of a KEM mechanism. It purposefully causes a collision between the sender's recovery key and a key generated by a KEM mechanism, whereby a second decryption path is created so that the sender alone can recover the original message. Neither method requires additional encryption (or encapsulation) of an ephemeral key for the sender.

2 Preliminaries

Notation. If $\mathcal{X}$ is a set then $x \xleftarrow{R} \mathcal{X}$ denotes that x is chosen uniformly at random from $\mathcal{X}$. $x||y$ denotes the concatenation of x and y. 1^λ denotes the string consisting of λ consecutive "1" bits. In this paper λ acts as a security parameter. A function in the security parameter λ is said to be negligible if it becomes smaller than the inverse of any polynomial in λ as λ becomes large. We use $\perp$ as a special symbol for an algorithm to indicate its failure in operations.

Pseudorandom Function. A pseudorandom function (PRF) [7] is an efficiently computable function $F : \mathcal{K}_{PRF} \times \mathcal{X} \to \mathcal{Y}$ such that no efficient adversary can distinguish the function from a truly random function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the domain, $\mathcal{Y}$ is the range of the functions, $\mathcal{K}_{PRF}$ is a key space defined by the security parameter λ and elements of $\mathcal{K}_{PRF}$ act as indices for instances of F. The security of PRF is defined by the following experiment between a challenger and an adversary [7] (see also [3]).

1. The challenger picks a random bit $b \xleftarrow{R} \{0,1\}$. If $b = 0$, the challenger chooses a random key $k \in \mathcal{K}_{PRF}$ and sets $f(\cdot) = F(k, \cdot)$. Otherwise, the challenger chooses a random function $f : \mathcal{X} \to \mathcal{Y}$.
2. The adversary can adaptively sends queries $x_1, ..., x_q \in \mathcal{X}$ to the challenger and the challenger responds with $f(x_1), ..., f(x_q)$.
3. Finally, the adversary outputs $b' \in \{0,1\}$.

Let $Exp^{PRF} = 1 (Exp^{Rand} = 1)$ denote the event that the adversary outputs 1 when $b = 0$ ($b = 1$). We say that F is ϵ_{PRF}-secure if for any probabilistic polynomial time (PPT) adversary, we have $|\Pr[Exp^{PRF} = 1] - \Pr[Exp^{Rand} = 1]| \leq \epsilon_{PRF}$ for a negligible function ϵ_{PRF} of the security parameter λ.

Message Authentication Code (MAC). In this paper, a MAC is a function that maps a key and a message (of arbitrary length) to a tag. Formally it is a function $MAC : \mathcal{K}_{MAC} \times \{0,1\}^* \to \mathcal{T}$, whose key space $\mathcal{K}_{MAC}$ is defined by the security parameter λ. The security of MAC can also be defined as a game between a challenger and an adversary. The challenger chooses a random key $k_{MAC} \xleftarrow{R} \mathcal{K}_{MAC}$. The adversary can adaptively make polynomially many

queries $m_1, m_2, ...$ to the challenger. The latter returns $t_1 = MAC(k_{MAC}, m_1)$, $t_2 = MAC(k_{MAC}, m_2)$, ... Finally, the adversary outputs a pair (m^*, t^*). We say that the adversary wins the game if $MAC(k_{MAC}, m^*) = t^*$ and t^* was never returned by the challenger in response to query m^*. MAC is said to be ϵ_{MAC}-unforgeable if for any PPT adversary, we have Pr[The adversary wins] $\leq \epsilon_{MAC}$ for a negligible function ϵ_{MAC} of λ.

IND-CCA2 Security of KEM. A key encapsulation mechanism, denoted by $KEM = (KEM.Gen, KEM.Enc, KEM.Dec)$, consists of three algorithms.

- a key generation algorithm, $KEM.Gen$, which takes as input a security parameter 1^λ, outputs a public/secret key pair (pk, sk).
- an encapsulation algorithm, $KEM.Enc$, which takes as input a public key pk, outputs a ciphertext c and an ephemeral key k.
- a decapsulation algorithm, $KEM.Dec$, which takes as input a secret key sk and a ciphertext c, outputs the ephemeral key k or a special failure symbol "$\perp$".

Indistinguishability security against adaptive chosen ciphertext attack, or IND-CCA2 security for short, for KEM is defined once again by a two party game played between a challenger and an adversary. During the game, the adversary is granted access to an oracle $\mathcal{O}_{sk}$, which upon a decapsulation query c returns $KEM.Dec(sk, c)$.

1. The challenger runs $KEM.Gen(1^\lambda)$ to generate a public/secret key pair (pk, sk) of KEM. It then sends pk to the adversary.
2. The challenger computes $(c^*, k_0^*) = KEM.Enc(pk)$, generates a random symmetric key $k_1^* \stackrel{R}{\leftarrow} \{0,1\}^{\lambda_k}$ and sends (c^*, k_b^*) to the adversary, where $b \stackrel{R}{\leftarrow} \{0,1\}$. We assume that the KEM's key space is a collection of bitstrings of length λ_k with λ_k being a polynomial in λ.
3. The adversary can query the oracle $\mathcal{O}_{sk}$ with any ciphertext c provided that $c \neq c^*$. Finally, the adversary terminates by outputting a bit b'.

We say that the adversary wins the game if $b = b'$, and that KEM is ϵ_{KEM}-IND-CCA2 secure if, for any PPT adversary, $|Pr[b = b'] - 1/2| \leq \epsilon_{KEM}$ for a negligible function ϵ_{KEM} of the security parameter λ.

One-Time Symmetric-Key Encryption against Passive Attack. (IND-OPA). Let $DEM = (DEM.Enc, DEM.Dec)$ denote an one-time symmetric-key encryption. The encryption algorithm $DEM.Enc$ takes as input a symmetric key k and a plaintext m and outputs a ciphertext c, where k is a bit string of length λ_{Enc} which is a polynomial in λ. The decryption algorithm $DEM.Dec$ takes as input a symmetric key k and a ciphertext c and outputs m. The security of DEM against passive attack is defined by the following game [6].

1. An adversary chooses a pair of plaintexts (m_0, m_1), where m_0 and m_1 are of the same length, and gives them to an encryption oracle $\mathcal{O}_k$. $\mathcal{O}_k$ generates a random key k and a random bit $b \stackrel{R}{\leftarrow} \{0,1\}$, and returns $c^* = DEM.Enc(k, m_b)$.

2. The adversary terminates by outputting a bit b'.

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

As shown in [6], it is easy to build a symmetric key encryption scheme that achieves IND-OPA security using standard symmetric-key techniques. For example, one can expand an encryption key k using a pseudorandom bit generator to obtain an one-time pad k' of length $|m|$ and compute $c = m \oplus k'$ as a ciphertext. Following the Encrypt-then-MAC paradigm, an IND-OPA DEM can be converted into an (one-time-) IND-CCA secure DEM. [6] also shows that, an IND-CCA secure KEM combined with an (one-time-) IND-CCA secure DEM yields a IND-CCA secure PKE.

The following lemma [11] is simple but useful in the security proof of this paper.

Lemma 1. *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 Public Key Encryption Admitting Sender Recovery (PKE-SR)

We recall the definition of PKE-SR first introduced in [12] (where it was called public key encryption with backward recovery). PKE-SR consists of four algorithms:

- A probabilistic key generation algorithm Gen_{SR}, which consists of two sub-algorithms $Gen_{SR}.S$ and $Gen_{SR}.R$. $Gen_{SR}.S$ takes as input the security parameter 1^λ and outputs a sender's secret recovery key sk_{rcv}. $Gen_{SR}.R$ takes as input 1^λ and outputs a receiver's public/secret key pair (pk_R, sk_R).
- An encryption algorithm Enc_{SR}, which takes as input the sender's secret recovery key sk_{rcv}, the receiver's public key pk_R and a plaintext m, outputs a ciphertext c_{SR}.
- A decryption algorithm Dec_{SR}, which takes as input the receiver's secret key sk_R and the ciphertext c_{SR}, outputs the corresponding plaintext m or the error symbol "$\perp$".
- A recovery algorithm Rec_{SR}, which takes as input the sender's secret key sk_{rcv}, the receiver's public key pk_R and the ciphertext c_{SR}, outputs the corresponding plaintext m or the error symbol "$\perp$".

The security definition for PKE-SR originally introduced in [12] covered both confidentiality and authenticity. We feel that since the primary goal of public key encryption is about confidentiality, leaving authenticity out of the picture would simplify the construction and analysis of public key encryption with message recovery by sender. With that in mind, we describe a simplified definition for the IND-CCA2 security of PKE-SR following the standard two-party game approach.

IND-CCA2 Security of PKE-SR. The IND-CCA2 game for PKE-SR is played between an adversary A and a challenger. During the game, the adversary A has access to three types of oracles: (1) an encryption oracle $\mathcal{O}_{Enc}$, which upon an encryption query $q_{Enc} = (pk', m)$ returns a ciphertext $c_{SR} = Enc_{SR}(sk_{rcv}, pk', m)$. (2) a decryption oracle $\mathcal{O}_{Dec}$, which upon a decryption query $q_{Dec} = c_{SR}$ returns $Dec_{SR}(sk_R, c_{SR})$. (3) a recovery oracle $\mathcal{O}_{Rec}$, which upon a recovery query $q_{Rec} = (pk', c_{SR})$ returns $Rec_{SR}(sk_{rcv}, pk', c_{SR})$.

The IND-CCA2 game proceeds as follows.

1. The challenger runs $Gen_{SR}(1^\lambda)$ to generate a target sender's secret recovery key sk_{rcv} and a target receiver's public/secret key pair (pk_R, sk_R).
2. The adversary 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 $c^*_{SR} = Enc(sk_{rcv}, pk_R, m_b)$, where $b \overset{R}{\leftarrow} \{0, 1\}$.
3. The adversary A can query all the three oracles $\mathcal{O}_{Enc}$, $\mathcal{O}_{Dec}$ and $\mathcal{O}_{Rec}$ with any input, provided that $q_{Dec} \neq c^*_{SR}$ and $q_{Rec} \neq (pk_R, c^*_{SR})$. Finally, A terminates by returning a guess b'.

We say that A wins the above game if $b' = b$. The advantage of A is defined as $Adv^{CCA2}_{SR}(A) = |\Pr[A \text{ wins}] - 1/2|$. PKE-SR is said to be ϵ-IND-CCA2 secure if, for any PPT adversary A, $Adv^{CCA2}_{SR}(A) \leq \epsilon$ for a negligible ϵ.

4 PKE-SR Using Pre-KEM Seeding

Randomness plays an important role in the construction of secure public key encryption. It is known that IND-CPA and IND-CCA security of public key encryption rests on sufficiently good randomness. Leakage of randomness may lead to the disclosure of a plaintext from the ciphertext. For instance, consider a ciphertext of the ElGamal encryption scheme $c = (g^r, y^r m)$, where g denotes the generator of a cyclic group, r a random seed used during the encryption phase, y a receiver's public key and m a plaintext. Clearly, anyone who knows r and y can efficiently recover m from c without using the receiver's decryption key.

The above observation leads us to an efficient construction of public key encryption with message recovery by sender. First we formalize the notion of a retraceable KEM which forms the basis for the construction method for PKE-SR. A KEM $(KEM.Gen, KEM.Enc, KEM.Dec)$ is retraceable if it satisfies the following properties.

- The encapsulation algorithm, $KEM.Enc$, takes as input a random number r and a public key pk, outputs a ciphertext c and an ephemeral key k. Here, $KEM.Enc$ acts as a deterministic polynomial time algorithm in such a way that for each (pk, c) there is a unique r.
- There exists a deterministic polynomial time algorithm $KEM.Rec$ which takes as input (r,pk,c) and outputs the ephemeral key k.

A large class of KEM schemes satisfy the above retraceable properties, including those in [4][1][2][10][9][8]. Many PKE schemes such as those in [4][1][2] can be used as KEM by encrypting a random element as an ephemeral key, and these resultant KEM schemes too satisfy the retraceable properties.

4.1 Construction of PKE-SR Using Pre-KEM Seeding

Let F denote a PRF which maps $\{0,1\}^{\lambda_1} \times \mathcal{X}$ to $\mathcal{Y}$, where $\mathcal{X}$ denotes the set of $pk||\tau$, pk a receiver's public key, $\tau \in \{0,1\}^{\lambda_0}$, and $\mathcal{Y}$ the set of random seeds r used in $KEM.Enc$. Let MAC denote a message authentication code that maps $\{0,1\}^{\lambda_2} \times \{0,1\}^*$ to $\mathcal{T}$. Note that λ_0, λ_1 and λ_2 are polynomial functions of the security parameter λ. Let $(KEM.Gen, KEM.Enc, KEM.Dec)$ denote a retraceable KEM scheme and $(DEM.Enc, DEM.Dec)$ denote a one-time symmetric key encryption scheme. The basic idea of pre-KEM seeding is

Table 1. PKE-SR from pre-KEM seeding

- Key generation $Gen_{SR}(1^\lambda)$.
  - The receiver runs $KEM.Gen(1^\lambda)$ to generate his public/secret key pair (pk, sk).
  - The sender picks a random $sk_{rcv} \in \{0,1\}^{\lambda_1}$ as her secret recovery key.
- Encryption $Enc_{SR}(sk_{rcv}, pk, m)$ by sender.
  - Choose $\tau \xleftarrow{R} \{0,1\}^{\lambda_0}$ and compute $r = F(sk_{rcv}, pk||\tau)$.
  - Compute $(c_{KEM}, k_{Enc}||k_{MAC}) = KEM.Enc(r, pk)$ and $c_{DEM} = DEM.Enc(k_{Enc}, m)$. Denote $c = (c_{KEM}, c_{DEM})$.
  - Compute $tag = MAC(k_{MAC}, pk||\tau||c)$.
  - Output $c_{SR} = (\tau, c, tag)$.
- Decryption $Dec_{SR}(sk, c_{SR})$ by receiver.
  - Compute $k_{Enc}||k_{MAC} = KEM.Dec(sk, c_{KEM})$. If $k_{Enc}||k_{MAC} = \perp$, output $\perp$ and halt.
  - If $tag \neq MAC(k_{MAC}, pk||\tau||c)$, output $\perp$ and halt; Otherwise, output $m' = DEM.Dec(k_{Enc}, c_{DEM})$.
- Recovery $Rec_{SR}(sk_{rcv}, pk, c_{SR})$ by sender.
  - Compute $r = F(sk_{rcv}, pk||\tau)$ and $k_{Enc}||k_{MAC} = KEM.Rec(r, pk, c_{KEM})$.
  - If $tag \neq MAC(k_{MAC}, pk||\tau||c)$, output $\perp$ and halt; Otherwise, output $m' = DEM.Dec(k_{Enc}, c_{DEM})$.

to let the sender choose a random $\tau \in \{0,1\}^{\lambda_0}$ and generate $r = F(sk_{rcv}, pk||\tau)$, which will be used as the random seed of KEM to create a symmetric key for DEM. τ is included as part of the ciphertext so that the sender can recreate r by computing $F(sk_{rcv}, pk||\tau)$. As the KEM has the retraceable properties, the sender can then obtain the ephemeral key using $KEM.Rec(r, pk, c_{KEM})$. To prevent an adversary from getting help through a recovery query (pk', c_{SR}), we need a MAC to provide validity check for the whole ciphertext. The resulting PKE-SR scheme is described in detail in Table 1 and Figure 1. Note that if the underlying KEM outputs a key whose length is not long enough for it to be split into two keys k_{Enc} and k_{MAC}, a secure pseudorandom number generator can be applied to extend it into a desired length prior to splitting.

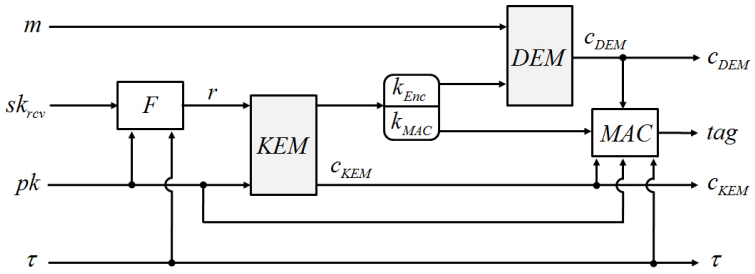

Fig. 1. PKE-SR from pre-KEM seeding

4.2 IND-CCA2 Security of PKE-SR from Pre-KEM Seeding

Theorem 1. *PKE-SR from pre-KEM seeding is ϵ-IND-CCA2 secure if KEM is retraceable and ϵ_{KEM}-IND-CCA2 secure, DEM is ϵ_{DEM}-IND-OPA secure, F is ϵ_{PRF}-secure and MAC is ϵ_{MAC}-unforgeable, where*

$$\epsilon \leq \epsilon_{PRF} + \epsilon_{KEM} + (N_{Dec} + N_{Rec})\epsilon_{MAC} + \epsilon_{DEM},$$

N_{Dec} and N_{Rec} denote upper bounds on the numbers of decryption queries and recovery queries, respectively.

Proof. We prove the theorem using the game hopping technique. We show that any PPT adversary can win the IND-CCA2 game, denoted by Game 0, with only a negligible advantage. To that end, we construct a sequence of three games Games 1, 2 and 3, and prove, consecutively, the indistinguishability between Games 0, 1, 2 and 3. More details are as follows.

- Game 0. This game is the standard IND-CCA2 game of PKE-SR. In the challenge phase, the adversary sends m_0 and m_1 to the challenger. The challenger randomly chooses $\tau^* \xleftarrow{R} \{0,1\}^{\lambda_0}$ and computes $r^* = F(sk_{rcv}, pk||\tau^*)$, $(c^*_{KEM}, k^*_{Enc}||k^*_{MAC}) = KEM.Enc(r^*, pk)$, $c^*_{DEM} = DEM.Enc(k^*_{Enc}, m_b)$ and $tag^* = MAC(k^*_{MAC}, pk||\tau^*||c^*)$, where $c^* = (c^*_{KEM}, c^*_{DEM})$. The target ciphertext is $c^*_{SR} = (\tau^*, c^*, tag^*)$.
- Game 1. Game 1 is similar to Game 0, except that the pseudorandom function F is replaced with a uniformly selected function $\phi : \mathcal{X} \to \mathcal{Y}$.
- Game 2. Game 2 is similar to Game 1, except that at the beginning of Game 2, the challenger randomly chooses $\tau^* \xleftarrow{R} \{0,1\}^{\lambda_0}$, $r^{**} \xleftarrow{R} \mathcal{Y}$, defines $\phi(pk||\tau^*) = r^{**}$ and computes $(c^{**}_{KEM}, k^{**}_{Enc}||k^{**}_{MAC}) = KEM.Enc(r^{**}, pk)$. In the challenge phase the challenger computes $c^{**}_{DEM} = DEM.Enc(k^{**}_{Enc}, m_b)$ and $tag^{**} = MAC(k^{**}_{MAC}, pk||\tau^*||c^{**})$, where $c^{**} = (c^{**}_{KEM}, c^{**}_{DEM})$. The target ciphertext is $c^{**}_{SR} = (\tau^*, c^{**}, tag^{**})$.
- Game 3. Game 3 is similar to Game 2, except that at the beginning of Game 3 the challenger randomly chooses $k^+_{Enc}||k^+_{MAC} \xleftarrow{R} \{0,1\}^{\lambda_k}$ and defines

$KEM.Enc(r^{**}, pk) = (c^{**}_{KEM}, k^{+}_{Enc}||k^{+}_{MAC})$ and $KEM.Dec(sk, c^{**}_{KEM}) = k^{+}_{Enc}||k^{+}_{MAC}$. In the challenge phase, the challenger computes $c^{+}_{DEM} = DEM.Enc(k^{+}_{Enc}, m_b)$ and $tag^{+} = MAC(k^{+}_{MAC}, pk||\tau^{*}||c^{+})$, where $c^{+} = (c^{**}_{KEM}, c^{+}_{DEM})$. The target ciphertext is $c^{+}_{SR} = (\tau^{*}, c^{+}, tag^{+})$.

Let Game i=1 denotes an event where the adversary wins in Game i. We have the following claims.

Claim 1. $|\Pr[Game\ 0 = 1] - \Pr[Game\ 1 = 1]| \leq \epsilon_{PRF}$ if F is ϵ_{PRF}-secure.

The proof of Claim 1 is straightforward and omitted.

Claim 2. $\Pr[Game\ 1 = 1] = \Pr[Game\ 2 = 1]$.

Proof. We note that the only difference between Game 1 and Game 2 is the fact that the output of $\phi(pk||\tau^{*})$ is set to r^{**}. Since ϕ is a real random function and r^{**} is randomly chosen, Game 1 and Game 2 are identical. □

Claim 3. $|\Pr[Game\ 2 = 1] - \Pr[Game\ 3 = 1]| \leq \epsilon_{KEM}$ if the underlying KEM is ϵ_{KEM}-IND-CCA2 secure.

Proof. Denote by A_{23} the adversary in the IND-CCA2 game. If $|\Pr[Game\ 2 = 1] - \Pr[Game\ 3 = 1]|$ is non-negligible, we can construct an efficient algorithm M_{KEM} to breach the security of the underlying KEM. More details follow.

The IND-CCA2 game of the underlying KEM is played between the challenger and M_{KEM}. Let (pk, sk) denote the public/secret key pair of the underlying KEM. pk is given to M_{KEM}. In the challenge phase the challenger sends the target ciphertext c^{**}_{KEM} and $k^{**+}_{Enc\ b}||k^{**+}_{MAC\ b}$ to M_{KEM}, where $b \xleftarrow{R} \{0,1\}$. Suppose the random seed used in generating c^{**}_{KEM} is r^{**}, which is unknown to M_{KEM}. Then M_{KEM} simulates the IND-CCA2 game for A_{23}. Let pk be the public key of PKE-SR from pre-KEM seeding. M_{KEM} chooses $\tau^{*} \xleftarrow{R} \{0,1\}^{\lambda_0}$ and ϕ, and defines $KEM.Enc(\phi(pk||\tau^{*}), pk) = (c^{**}_{KEM}, k^{**+}_{Enc\ b}||k^{**+}_{MAC\ b})$ and $KEM.Dec(sk, c^{**}_{KEM}) = k^{**+}_{Enc\ b}||k^{**+}_{MAC\ b}$. Note that sk and $\phi(pk||\tau^{*})$, which is set to r^{**}, are unknown to M_{KEM}. A_{23} can make three types of queries below:

- Encryption query with $q_{Enc} = (pk', m)$. M_{KEM} randomly chooses τ and sets $r = \phi(pk'||\tau)$, $(c_{KEM}, k_{Enc}||k_{MAC}) = KEM.Enc(r, pk')$, $c_{DEM} = DEM.Enc\ (k_{Enc}, m)$ and $tag = MAC(k_{MAC},\ pk'||\tau||c)$. M_{KEM} returns $c_{SR} = (\tau, c, tag)$. Note that if $\tau = \tau^{*}$ and $pk' = pk$, M_{KEM} does not know $\phi(pk||\tau^{*})$. In this case, M_{KEM} sets $c_{KEM} = c^{**}_{KEM}$ and computes c_{DEM} and tag using $k^{**+}_{Enc\ b}||k^{**+}_{MAC\ b}$ as in Game 3.
- Decryption query with $q_{Dec} = c_{SR} = (\tau, c, tag)$, where $c = (c_{KEM}, c_{DEM})$. M_{KEM} makes decapsulation query c_{KEM} to the challenger and gets the corresponding ephemeral key $k_{Enc}||k_{MAC}$. Using $k_{Enc}||k_{MAC}$, M_{KEM} can decrypt c_{DEM} and check whether $tag \stackrel{?}{=} MAC(k_{MAC}, pk||\tau||c)$. If $tag =$

$MAC(k_{MAC}, pk||\tau||c)$, M_{KEM} responds with $DEM.Dec(k_{Enc}, c_{DEM})$. Otherwise, $\perp$. Note that if $c_{KEM} = c^{**}_{KEM}$, M_{KEM} decrypts c_{DEM} and checks the validity of tag using $k^{***+}_{Enc\ b}||k^{***+}_{MAC\ b}$.

- Recovery query with $q_{Rec} = (pk', c_{SR}) = (pk', \tau, c, tag)$. If $pk' = pk$ and $\tau = \tau^*$, M_{KEM} checks the validity of tag and decrypts c_{DEM} using $k^{***+}_{Enc\ b}||k^{***+}_{MAC\ b}$; Otherwise, M_{KEM} computes $r = \phi(pk'||\tau)$, $k_{Enc}||k_{MAC} = KEM.Rec(r, pk, c_{KEM})$, and checks the validity of tag and decrypts c_{DEM} using $k_{Enc}||k_{MAC}$.

When receiving (m_0, m_1) from A_{23}, M_{KEM} sets $c^{***+}_{DEM} = DEM.Enc(k^{***+}_{Enc\ b}, m_{\tilde{b}})$ and $tag^{***+} = MAC(k^{***+}_{MAC\ b}, pk||\tau^*||c^{***+})$, where $c^{***+} = (c^{**}_{KEM}, c^{***+}_{DEM})$, $\tilde{b} \stackrel{R}{\leftarrow} \{0,1\}$. Hence, $c^{***+}_{SR} = (\tau^*, c^{***+}, tag^{***+})$. After the challenge phase, M_{KEM} can answer queries as described above, provided that $q_{Dec} \neq c^{***+}_{SR}$ and $q_{Rec} \neq (pk, c^{***+}_{SR})$. Finally, A_{23} outputs b'. If $b' = \tilde{b}$, M_{KEM} output 0; Otherwise, 1.

If $b = 0$, the above experiment is Game 2. If $b = 1$, the above experiment is Game 3. If $|\Pr[Game\ 2 = 1] - \Pr[Game\ 3 = 1]|$ is non-negligible, then M_{KEM} can efficiently break the IND-CCA2 security of the underlying KEM, from which it follows that $|\Pr[Game\ 2 = 1] - \Pr[Game\ 3 = 1]| \leq \epsilon_{KEM}$. □

Claim 4. $|\Pr[\text{Game } 3 = 1] - 1/2| \leq (N_{Dec} + N_{Rec})\epsilon_{MAC} + \epsilon_{DEM}$ if the underlying DEM is ϵ_{DEM}-IND-OPA secure and MAC is ϵ_{MAC}-unforgeable.

Proof. If there exists a PPT adversary A_3 which can win Game 3 with a non-negligible advantage, we can construct an algorithm M_{DEM} to break the IND-OPA security of the underlying DEM. The description of M_{DEM} is given below.

The IND-OPA game of the underlying DEM is played between the challenger and M_{DEM}. M_{DEM} sets the parameters of PKE-SR from pre-KEM seeding in the same manner as in Game 3 and simulates Game 3 for A_3 as follows.

- A_3 can query M_{DEM} with $q_{Enc}, q_{Dec}, q_{Rec}$ in the same manner as in Game 3, except that when $q_{Dec} = (\tau', (c^{**}_{KEM}, c'_{DEM}), tag')$ or $q_{Rec} = (pk, \tau', (c^{**}_{KEM}, c'_{DEM}), tag')$, M_{DEM} responds with "$\perp$". However, if $q_{Dec} = (\tau', (c^{**}_{KEM}, c'_{DEM}), tag')$ or $q_{Rec} = (pk, \tau', (c^{**}_{KEM}, c'_{DEM}), tag')$ are valid, M_{DEM} gives wrong answers. Denote such an event by Bad_3.
- In the challenge phase of IND-OPA game, M_{DEM} sends (m_0, m_1) which are chosen by A_3 to the challenger. The challenger responds with c^+_{DEM}. M_{DEM} sets the target ciphertext to $(\tau^*, c^+, tag^+) = (\tau^*, (c^{**}_{KEM}, c^+_{DEM}), tag^+)$, where c^{**}_{KEM} is computed as in Game 3, $tag^+ = MAC(k^+_{MAC}, pk||\tau^*||c^+)$ and k^+_{MAC} is randomly chosen by M_{DEM}.

Finally, the adversary outputs a guess for b, which is the output of M_{DEM}.

If Bad_3 does not happen, M_{DEM} perfectly simulates Game 3 for A_3 and the success probability of M_{DEM} is the same as that of A_3. Since $\Pr[Game\ 3 = 1 \cap \overline{Bad_3}] = \Pr[M_{DEM} \text{ wins} \cap \overline{Bad_3}]$ and $|\Pr[M_{DEM} \text{ wins}] - 1/2| \leq \epsilon_{DEM}$, we have

$$\begin{aligned}
&\left|\Pr[Game\ 3 = 1] - \frac{1}{2}\right| \\
&\leq |\Pr[Game\ 3 = 1] - \Pr[M_{DEM}\ \text{wins}]| + \left|\Pr[M_{DEM}\ \text{wins}] - \frac{1}{2}\right| \\
&\leq \Pr[Bad_3] + \epsilon_{DEM},
\end{aligned}$$

where the last inequality follows from Lemma 1.

It remains to show that $\Pr[Bad_3]$ is negligible. To that end, we construct a forging algorithm M_{MAC} as follows. M_{MAC} randomly chooses an index $j \in \{1, ..., N_{Dec} + N_{Rec}\}$. Next, M_{MAC} simulates Game 3 for the adversary A. If the j-th decryption/recovery query, which may be $q_{Dec-j} = (\tau_{(j)}, c_{(j)}, tag_{(j)})$ or $q_{Rec-j} = (pk', \tau_{(j)}, c_{(j)}, tag_{(j)})$, occurs before the challenge phase, M_{MAC} outputs $(pk'||\tau_{(j)}||c_{(j)}, tag_{(j)})$ and halts, where $pk' = pk$ if the j-th decapsulation/recovery query is q_{Dec-j}. Otherwise, in the challenge phase, A sends (m_0, m_1) to M_{MAC}. M_{MAC} computes (τ^*, c^+) as in Game 3 where $c^+ = (c^{**}_{KEM}, c^+_{DEM})$, and sends $pk||\tau^*||c^+$ to the challenger of the MAC, which returns tag^+. Then, M_{MAC} gives (τ^*, c^+, tag^+) to A and continues running A until A makes the j-th decapsulation/recovery query. M_{MAC} outputs $(pk'||\tau_{(j)}||c_{(j)}, tag_{(j)})$ and halts. Hence, the probability that M_{MAC} outputs a valid forgery is at least $\Pr[Bad_3]/(N_{Dec} + N_{Rec})$. If the underlying MAC is ϵ_{MAC}-unforgeable, we have $\Pr[Bad_3]/(N_{Dec} + N_{Rec}) \leq \epsilon_{MAC}$. That is, $\Pr[Bad_3] \leq (N_{Dec} + N_{Rec})\epsilon_{MAC}$.

Therefore, $\left|\Pr[Game\ 3 = 1] - \frac{1}{2}\right| \leq (N_{Dec} + N_{Rec})\epsilon_{MAC} + \epsilon_{DEM}$, which completes the proof of Claim 4. □

From Claims 1, 2, 3 and 4, we have

$$\begin{aligned}
&\left|\Pr[Game\ 0 = 1] - \frac{1}{2}\right| \\
&\leq |\Pr[Game\ 0 = 1] - \Pr[Game\ 1 = 1]| + |\Pr[Game\ 1 = 1] - \Pr[Game\ 2 = 1]| + \\
&\quad |\Pr[Game\ 2 = 1] - \Pr[Game\ 3 = 1]| + \left|\Pr[Game\ 3 = 1] - \frac{1}{2}\right| \\
&\leq \epsilon_{PRF} + \epsilon_{KEM} + (N_{Dec} + N_{Rec})\epsilon_{MAC} + \epsilon_{DEM}.
\end{aligned}$$

which completes the proof of Theorem 1. □

4.3 Instantiations

The underlying KEM of PKE-SR from pre-KEM seeding can be instantiated by many KEM schemes such as those in [6][9][8], and the underlying DEM can be instantiated by AES, which is assumed to be IND-OPA secure. Note that τ is an unpredictable nonce. It is part of a ciphertext and used to generated a fresh random seed r for the underlying KEM. In practice, τ could be used in conjunction with a public label L (e.g., the receiver's e-mail address, or any piece of information agreed upon between two communicating parties.)

PKE-SR Using PKE. As discussed before, some public key encryption (PKE) schemes are retraceable and can be used as KEM, e.g., [4][2], although it might be somewhat an "overkill", especially when the PKE is constructed out of KEM/DEM. Nevertheless for practical purposes, a PKE can be used to replace the underlying KEM in PKE-SR from pre-KEM seeding. A further example is based on DHIES [1]. It turns out that with DHIES, a somewhat more efficient PKE-SR can be built that requires only one additional element τ added to a ciphertext and one additional computation for PRF. Although the KEM part of DHIES is not IND-CCA2, we can still prove the IND-CCA2 security of the resultant PKE-SR under the oracle Diffie-Hellman assumption. Due to a lack of space, examples of instantiations based on [1][2][9] are deferred to the full paper.

5 PKE-SR Using Post-KEM Converging

In this section, we present a method of constructing PKE-SR, called post-KEM converging. The basic idea is from the sibling intractable function families with the collision accessibility property [13]. We first explore related properties of universal hash functions with collision accessibility in Section 5.1, then we show the details of the post-KEM converging method in Section 5.2

5.1 Universal Hash Function Families with Collision Accessibility Property

Loosely speaking, a universal hash function family with collision accessibility is one that has the property that, given a set of initial strings, the hash values of them can be made to collide with one another. A formal definition of k-collision accessibility follows.

Definition 1. *[13] Let $H = \bigcup H_\lambda$ be a family of functions that is polynomial time computable, samplable and maps $l(\lambda)$-bit input into $m(\lambda)$-bit output strings. H has the k-**collision accessibility property** if for all λ and for all $1 \leq i \leq k$, given a set $X = \{x_1, x_2, ..., x_i\}$ of i initial strings in $\{0,1\}^{l(\lambda)}$, it is possible in probabilistic polynomial time to select randomly and uniformly functions in H_λ^X, where $H_\lambda^X \subset H_\lambda$ is the set of all functions in H_λ that map $x_1, x_2, ..., x_i$ to the same strings in $\{0,1\}^{l(\lambda)}$.*

As an example, we show how to construct a universal hash function family with 2-collision accessibility.

Given $(s_1, s_2) \in \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$, randomly choose $(w_1, w_2, k_{CA}) \in \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_4}$, where $\lambda_1 = \lambda_3 + \lambda_4$. Note that operations in equations below are carried out over the finite field $GF(2^{\lambda_1})$. Solve the equations (1)(2) for $(a_1, a_2) \in \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$,

$$w_1 || k_{CA} = a_1 + a_2 s_1 \quad (1)$$

$$w_2 || k_{CA} = a_1 + a_2 s_2 \quad (2)$$

which can be also written as follows,

$$\begin{pmatrix} w_1||k_{CA} \\ w_2||k_{CA} \end{pmatrix} = \begin{pmatrix} 1 & s_1 \\ 1 & s_2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}. \quad (3)$$

Let $UH_{CA}(x)$ denote the resulting universal hash function, which takes $x \in \{0,1\}^{\lambda_1}$ as input and outputs the least significant λ_4 bits of $a_1 + a_2 x$. Hence, we have $UH_{CA}(s_1) = UH_{CA}(s_2) = k_{CA}$. (a_1, a_2) is called the description of UH_{CA}.

Note that a tuple $(w_1, w_2, k_{CA}, s_1, s_2)$ such that $s_1 \neq s_2$ ensures that Equation (3) has a unique solution. We say that such $(w_1, w_2, k_{CA}, s_1, s_2)$ is a valid tuple. An important observation is that when a valid tuple $(w_1, w_2, k_{CA}, s_1, s_2)$ is randomly chosen from $\in \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_4} \times \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$, we have the following claim on the distribution of (a_1, a_2).

Claim 5. If (a_1, a_2) is obtained from a random and valid tuple $(w_1, w_2, k_{CA}, s_1, s_2) \in \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_4} \times \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$, the statistical difference between (a_1, a_2) and a random pair in $\{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$ is less than $\frac{1}{2^{\lambda_3 - 1}}$.

The proof for Claim 5 is omitted due to a limitation in space. A consequence of Claim 5 is the following Claim 6.

Claim 6. Consider the following ensemble,

$$W_\beta = \{(a_1^{(1)}, a_2^{(1)}, s_1^{(1)}), (a_1^{(2)}, a_2^{(2)}, s_1^{(2)}), ..., (a_1^{(N-1)}, a_2^{(N-1)}, s_1^{(N-1)}), (a_1^*, a_2^*, s_1^*)\},$$

where $(a_1^{(i)}, a_2^{(i)}, s_1^{(i)}), 1 \leq i \leq N-1$, are generated by UH_{CA} using random and valid tuples and $\beta \xleftarrow{R} \{0,1\}$. If $\beta = 0$, then (a_1^*, a_2^*, s_1^*) is generated by UH_{CA} using a random and valid tuple $(w_1^*, w_2^*, k_{CA}^*, s_1^*, s_2^*)$. Otherwise, (a_1^*, a_2^*, 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 $\frac{1}{2^{\lambda_3 - 1}}$.

5.2 Construction of PKE-SR Using Post-KEM Converging

The core idea of post-KEM converging is to force an ephemeral key k_{KEM} from KEM and the sender's secret recovery key sk_{rcv} to collide by the use of UH_{CA}, by which we convert KEM/DEM into PKE-SR. To construct a full-fledged PKE-SR scheme, we need two one-way hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{\lambda_1 + \lambda_3}$, $H_2 : \{0,1\}^* \to \{0,1\}^{2\lambda_1}$ and a key derivation function $H_{KDF} : \{0,1\}^{2\lambda_1 + \lambda_4} \to \{0,1\}^{\lambda_k}$. More details about the scheme are given in Table 2 and Fig 2.

5.3 IND-CCA2 Security of PKE-SR from Post-KEM Converging

Theorem 2. *PKE-SR from post-KEM converging is ϵ-IND-CCA2 secure in the random oracle model if KEM is ϵ_{KEM}-IND-CCA2 secure, DEM is ϵ_{DEM}-IND-OPA secure and MAC is ϵ_{MAC}-unforgeable where*

$$\epsilon \leq \epsilon_{KEM} + \frac{N_{RO2}}{2^{\lambda_1 - 1}} + (\frac{1}{2^{\lambda_1 - 2}} + \frac{1}{2^{\lambda_3 - 1}}) N_{Rec} + \frac{1}{2^{\lambda_3 - 1}} + (N_{Dec} + N_{Rec}) \epsilon_{MAC} + \epsilon_{DEM},$$

Table 2. PKE-SR from post-KEM converging

- Key generation $Gen_{SR}(1^\lambda)$.
  - The receiver runs $KEM.Gen(1^\lambda)$ of the underlying KEM and generates his public/secret key pair (pk, sk).
  - The sender randomly chooses $sk_{rcv} \in \{0,1\}^{\lambda_1}$ as her secret key.
- Encryption $Enc_{SR}(sk_{rcv}, pk, m)$ by sender.
  - Randomly choose $\tau \xleftarrow{R} \{0,1\}^{\lambda_0}$. Compute $(c_{KEM}, k_{KEM}) = KEM.Enc(pk)$. Here, $KEM.Enc(\cdot)$ denotes a PPT algorithm.
  - Compute $w_1||s_1 = H_1(k_{KEM}||\tau)$ and $w_2||s_2||k_{CA} = H_2(sk_{rcv}||\tau)$, where $(w_1, w_2, k_{CA}, s_1, s_2) \in \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_3} \times \{0,1\}^{\lambda_4} \times \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$. If $s_1 = s_2$, run the first two steps of $Enc_{SR}(sk_{rcv}, pk, m)$ again until $s_1 \neq s_2$. Solve the equations below for (a_1, a_2).

$$w_1||k_{CA} = a_1 + a_2 s_1 \tag{4}$$

$$w_2||k_{CA} = a_1 + a_2 s_2 \tag{5}$$

  - Compute $k_{Enc}||k_{MAC} = H_{KDF}(k_{CA}||(a_1, a_2))$ and $c_{DEM} = DEM.Enc(k_{Enc}, m)$. Denote $c = (c_{KEM}, c_{DEM})$. Compute $tag = MAC(k_{MAC}, pk||\tau||(a_1, a_2)||c)$.
  - Output $c_{SR} = (\tau, (a_1, a_2), c, tag)$.
- Decryption $Dec_{SR}(sk, c_{SR})$ by receiver.
  - Compute $k_{KEM} = KEM.Dec(sk, c_{KEM})$. If $k_{KEM} = \perp$, output $\perp$ and halt. Compute $w_1||s_1 = H_1(k_{KEM}||\tau)$ and $w_1'||k_{CA} = a_1 + a_2 s_1$. If $w_1' \neq w_1$, output $\perp$ and halt. Compute $k_{Enc}||k_{MAC} = H_{KDF}(k_{CA}||(a_1, a_2))$.
  - If $tag \neq MAC(k_{MAC}, pk||\tau||(a_1, a_2)||c)$, output $\perp$ and halt; Otherwise, output $m' = DEM.Dec(k_{Enc}, c_{DEM})$.
- Recovery $Rec_{SR}(sk_{rcv}, pk, c_{SR})$ by sender.
  - Compute $w_2||s_2||k_{CA} = H_2(sk_{rcv}||\tau)$ and $w_2'||k_{CA}' = a_1 + a_2 s_2$. If $w_2'||k_{CA}' \neq w_2||k_{CA}$, output $\perp$ and halt. Compute $k_{Enc}||k_{MAC} = H_{KDF}(k_{CA}||(a_1, a_2))$.
  - If $tag \neq MAC(k_{MAC}, pk||\tau||(a_1, a_2)||c)$, output $\perp$ and halt; Otherwise, output $m' = DEM.Dec(k_{Enc}, c_{DEM})$.

Remark. In practice, KEM can be implemented in a multiplicative group whose elements are of 1024 bits or 2048 bits in length. We can set $\lambda_0 = \lambda_1 = 256$ and $\lambda_3 = \lambda_4 = 128$. Considering MAC is often applied to construct an IND-CCA2 KEM/DEM, the length of additional information for turning KEM/DEM into PKE-SR is 768-bit, which is shorter than that of a group element.

N_{RO2}, N_{Dec} and N_{Rec} denote upper bounds on the numbers of random oracle queries for H_2, decryption queries and recovery queries, respectively.

Due to space limitations, more details of the proof of Theorem 2 are deferred to the full paper.

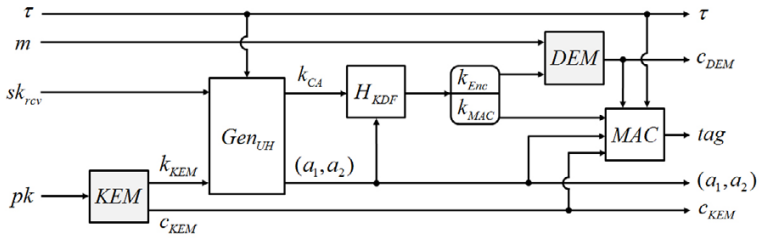

Fig. 2. PKE-SR from post-KEM converging. Gen_{UH} denotes the procedure that takes sk_{rcv}, τ and k_{KEM} as input and outputs a description (a_1, a_2) and k_{CA} is part of the hash value of a nonce and the sender's recovery key.

6 Conclusion

In this paper, we demonstrate two efficient methods for constructing secure public key encryption that admits decryption by sender, namely pre-KEM seeding and post-KEM converging. The pre-KEM seeding method can be applied to all KEMs with retraceable properties, whereas the post-KEM converging method makes use of universal hash function families with collision accessibility and can be applied to the entire class of KEM/DEM. Both constructions can achieve IND-CCA2 security.

Our final note is about the naive technique for constructing PKE-SR by encrypting an ephemeral key for KEM/DEM using a symmetric cipher under the sender's secret recovery key and then including the encrypted ephemeral key as part of the ciphertext. We find that this naive approach does not afford a rigorous security proof. More importantly, it does not scale well when there are a multiple number of receivers which will be the focus of our forthcoming paper.

Acknowledgements. We are grateful to Matt Franklin and Mingqiang Wang for invaluable discussions. We would also like to thank the anonymous reviewers for their helpful comments.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Baek, J., Zheng, Y.: Zheng and Seberry's public key encryption scheme revisited. International Journal of Information Security (IJIS) 2(1), 37–44 (2003)
3. Boneh, D., Montgomery, H.W., Raghunathan, A.: Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In: ACM Conference on Computer and Communications Security 2010, pp. 131–140 (2010)
4. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)

5. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
6. 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)
7. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM 34(4), 792–807 (1986)
8. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
9. Kiltz, E.: Chosen-ciphertext secure key encapsulation based on hashed gap decisional Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
10. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
11. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)
12. 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)
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)