Generic Constructions of Stateful Public Key Encryption and Their Applications

Joonsang Baek, Jianying Zhou, and Feng Bao

Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
{jsbaek, jyzhou, baofeng}@i2r.a-star.edu.sg

Abstract. We present generic constructions of stateful public key encryption (StPE). We build several new StPE schemes and explain existing ones using our generic constructions. Of the new StPE schemes, two schemes are built using the "identity-based technique" whereby one can construct public key encryption (PKE) schemes secure against chosen ciphertext attack in the standard model from identity-based encryption (IBE) schemes. These StPE schemes provide a positive answer to Bellare et al.'s open question on whether stateful variants of PKE schemes derived from IBE schemes exist.

1 Introduction

The main goal of the stateful public key encryption (StPE) schemes proposed by Bellare, Kohno and Shoup [6] is to reduce the cost of public key encryption by allowing a sender to maintain *state* that is reused across different encryptions. For example, one can obtain a stateful version of the ElGamal encryption in which a message M is encrypted to $(g^r, g^{rx}M)$ for public key g^x by maintaining the random value r and its corresponding value g^r as state so that g^r does not need to be computed each time. (Note, however, that much more is involved in the analysis of this scheme.)

Reducing the computational cost of public key encryption is of particular importance for low-power mobile devices where computational resources are constrained (such as PDA and mobile phones) or sensors communicating with the relatively powerful servers or base stations [24,15,12]. Due to the efficiency gained from maintaining state, StPE schemes have potential to be employed in these settings. But, even in the environments that provide reasonable amount of computational resources, it is preferable to speed up public key operation, which is often more expensive than symmetric key operation, for overall system performance.

The approach that Bellare et al. [6] adopt to construct StPE schemes is to convert specific public key encryption schemes such as DHIES [1] and Kurosawa and Desmedt's hybrid encryption scheme [20] into StPE schemes. However, for the practical reasons that speeding up public key operations is of great importance for the system performance, and new and more efficient computational

S.M. Bellovin et al. (Eds.): ACNS 2008, LNCS 5037, pp. 75–93, 2008.

primitives may emerge in the future, it is desirable to have some generic methods to construct StPE schemes. – The application of our generic construction to Kiltz's [18] new key encapsulation mechanism, which is presented in 4.3, is a good example of this argument.

Our Contributions. Regarding the issues discussed earlier we make the following contributions in this paper:

1. We formalize the concept of "partitioned" key encapsulation mechanism (PKEM) which is a special case of key encapsulation mechanism (KEM) [14] but turns out to encompass many existing schemes. Apart from the security against chosen ciphertext attack (IND-CCA) of KEM, we define some additional security properties that we require in our constructions of StPE. – See Section 2.2.
2. We present two generic constructions of StPE using PKEM and symmetric encryption. The first construction is shown to meet the strong security requirement defined in [6] in the known secret key (KSK) model without the random oracles [8]. The second construction is also shown to meet the security requirement in the unknown secret key (USK) model depending on the random oracles. – See Section 3.
3. We build several StPE schemes using the proposed generic constructions. Of these schemes, two are derived from the public key encryption (PKE) schemes constructed following the paradigm of converting identity-based encryption (IBE) into IND-CCA secure public key encryption (PKE) [13] in the *standard* model. We note that Bellare et al. [6] asked whether such StPE schemes exist. – See Section 4.

Related Work. Since Bellare et al. proposed the concept of StPE, to our knowledge, there has been few research work directly related to StPE. In their recent paper [25], Sarkar and Chaterjee discuss possible relation between the symmetric encryption they use for their generic construction of PKE from IBE and the symmetric encryption used in StPE. Other related work include the reuse of the randomness in the multi-receiver public key encryption, proposed by Kurosawa [19] and further formalized by Bellare, Boldyreva and Staddon [4]. (Readers are referred to [6] for detailed discussions on the relationship between StPE and the randomness reuse in the multi-receiver PKE.)

Organization of This Paper. In Section 2, we give definitions of all the building blocks we need in this paper. We then describe our generic constructions of StPE and give security analysis of them in Section 3. In Section 4, we provide new StPE schemes derived from our generic constructions.

2 Building Blocks

2.1 Stateful Public Key Encryption

In this subsection we review the definitions of StPE and its security as given in [6].

Definition 1 (StPE). A stateful public key encryption scheme, denoted StPE, consists of the following algorithms:

- StPE.Setup: Taking 1^λ for a security parameter $\lambda \in \mathbb{Z}_{\geq 0}$ as input, this algorithm generates a system parameter sp which includes λ. We write $sp \leftarrow \mathsf{StPE.Setup}(1^\lambda)$.
- StPE.KG: Taking sp as input, this algorithm generates a private/public key pair (sk, pk). We write $(sk, pk) \leftarrow \mathsf{StPE.KG}(sp)$.
- StPE.PKCk: Taking sp and pk as input, this algorithm returns 1 if the public key pk is valid and 0 otherwise. We write $\delta \leftarrow \mathsf{StPE.PKCk}(sp, pk)$, where $\delta \in \{0, 1\}$.
- StPE.NwSt: Taking sp as input, this algorithm generates a new state. We write $st \leftarrow \mathsf{StPE.NwSt}(sp)$.
- StPE.Enc: Taking sp, pk, st and a plaintext M as input, this algorithm outputs a ciphertext C and state st which may be different from the state provided as input to this algorithm. We write $(C, st) \leftarrow \mathsf{StPE.Enc}(sp, pk, st, M)$.
- StPE.Dec: Taking sp, sk and C as input, this deterministic algorithm outputs M which is either a plaintext or $\perp$ (meaning "reject") message. We write $M \leftarrow \mathsf{StPE.Dec}(sp, sk, C)$.

We impose a usual consistency condition on StPE: For any sp output by StPE.Setup, (sk, pk) generated by StPE.KG and st output by either StPE.NwSt or StPE.Enc, if (C, st) is an output of $\mathsf{StPE.Enc}(sp, pk, st, M)$, $\mathsf{StPE.Dec}(sp, sk, C) = M$.

We remark that the state generated by StPE.NwSt algorithm, the state provided as input to the StPE.Enc algorithm and the state output by StPE.Enc algorithm can all be different from each other. Note that StPE.PKCk is a public key verification algorithm that checks the validity of the given public key. The level of the validity check we require in this paper is the same as that of the simple public key checking mechanisms, eg. checking whether some component of the given public key belongs to the underlying (mathematical) group, which are already exercised in practice [21,17].

We now review the definition of chosen ciphertext security for StPE schemes as defined in [6].

Definition 2 (IND-CCA of StPE). Let StPE be a StPE scheme. Consider a game played with an attacker A:

Phase 1: The game computes $sp \leftarrow \mathsf{StPE.Setup}(1^\lambda)$, $(pk_1, sk_1) \leftarrow \mathsf{StPE.KG}(sp)$ and $st \leftarrow \mathsf{StPE.NwSt}(sp)$. (Note that (sk_1, pk_1) is the private/public key pair of the honest receiver R_1.) The game sends (sp, pk_1) to A.
Phase 2: A outputs public keys $pk_2, \ldots, pk_n$ of receivers $R_2, \ldots, R_n$ respectively, all of which are in the range of $\mathsf{KG}(sp)$. (Note that A may or may not know the private keys corresponding to the public keys $pk_2, \ldots, pk_n$.)
Phase 3: A issues a number of (but polynomially many) queries, each of which is responded by the game. The type of each query and the action taken by the game are described as follows:

- A challenge query (M_0, M_1) such that $|M_0| = |M_1|$: The game picks $\beta \stackrel{R}{\leftarrow} \{0,1\}$ (Throughout this paper, we denote by $s \stackrel{R}{\leftarrow} S$ the assignment of a uniformly and independently distributed random element from the set S to the variable s), computes $(C^*, st) \leftarrow \mathsf{StPE.Enc}(sp, pk_1, st, M_\beta)$, where st denotes current state, and sends C^* to A.
- Encryption queries, each of which is denoted by (i,M) where $i \in \{1,\ldots,n\}$: The game computes $(C, st) \leftarrow \mathsf{StPE.Enc}(sp, pk_i, st, M)$, where st denotes current state, and sends C to A.
- Decryption queries, each of which is denoted by $C \neq C^*$: The game computes $\mathsf{StPE.Dec}(sp, sk_1, C)$ and sends the resulting decapsulation (key or $\perp$ ("Reject")) to A.

Phase 4: A outputs its guess $\beta' \in \{0,1\}$.

We define A's advantage by $\mathbf{Adv}^{\text{IND-CCA}}_{A,\mathsf{StPE}}(\lambda) = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|$.

The chosen ciphertext security of StPE defined above can be considered in the *KSK* (Known Secret Key) or the *USK* (Unknown Secret Key) models [6]. In the KSK model, we assume that the attacker A possesses the corresponding private (secret) keys $sk_2 \ldots, sk_n$ of the public keys it outputs in Phase 2 of the attack game. On the other hand, in the USK model, we do not need this assumption. – Namely, in the KSK model, it is likely that the CA (Certificate Authority) is required to perform a proof of knowledge protocol to confirm whether users have corresponding private keys of their public keys while in the USK model, StPE.PKCk should be run (by the game) to check whether the public keys are valid.

2.2 Partitioned Key Encapsulation Mechanism

In this subsection we define a new primitive called "partitioned key encapsulation mechanism (PKEM)" which is a special type of normal KEM [14]. Speaking informally, PKEM has a property that a part of ciphertext, which does not explicitly depend on the given public key (but depends on the system parameter as will be defined below), can be "partitioned" from other parts of ciphertext. Though this property seems somewhat special, we show in the later section that many KEM schemes are in fact PKEM.

Definition 3 (PKEM). A partitioned KEM scheme, which we simply denote by PKEM, consists of the following algorithms.

- PKEM.Setup: Taking 1^λ for a security parameter $\lambda \in \mathbb{Z}_{\geq 0}$ as input, this algorithm generates a system parameter sp which includes λ. sp also defines the key space $\mathcal{K}_K$. We write $sp \leftarrow \mathsf{PKEM.Setup}(1^\lambda)$.
- PKEM.KG: Taking sp as input, this algorithm generates a private/public key pair (sk, pk). We write $(sk, pk) \leftarrow \mathsf{PKEM.KG}(sp)$.
- PKEM.Encap1 : Taking sp as input, this algorithm generates the first ciphertext vector ψ and state information ω which includes the internal randomness used to generate ψ. We write $(\omega, \psi) \leftarrow \mathsf{PKEM.Encap1}(sp)$.

- PKEM.Encap2 : Taking sp, ω, pk and ψ as input, this algorithm generates the second ciphertext vector σ and a key K. We write $(\sigma,K) \leftarrow \mathsf{PKEM.Encap2}(sp, pk, \omega, \psi)$.
- PKEM.Decap : Taking sp, sk, ψ and σ as input, this algorithm outputs either the session key K or the special symbol $\perp$. We write $K \leftarrow \mathsf{PKEM.Decap}(sp, sk, \psi, \sigma)$.

We impose a consitency condition on PKEM: For any sp output by PKEM.Setup, (sk, pk) generated by PKEM.KG, if (ω, ψ) and (σ, K) are outputs of $\mathsf{PKEM.Encap1}(sp)$ and $\mathsf{PKEM.Encap2}(sp, pk, \omega, \psi)$ respectively, $\mathsf{PKEM.Decap}(sp, sk, \psi, \sigma) = K$. We also impose the following *non-triviality* condition on (ψ, σ) and K: $\psi \neq \varepsilon$ and $K \neq \varepsilon$ but σ can be ε, where ε denotes empty string.

Note in the above definition that for the sake of convenience, we separate PKEM.Setup from the *KEM.KeyGen()* algorithm given in [14] which generates both the system parameter and the public key together. Note also that the IND-CCA definition for PKEM is essentially no different from the usual IND-CCA definition for KEM [14] as an attacker does not get the state information ω during the attack. For completeness, however, we define IND-CCA of PKEMs.

Definition 4 (IND-CCA of PKEM). Let PKEM be a PKEM scheme. Consider a game played with an attacker A:

Phase 1: The game computes $sp \leftarrow \mathsf{PKEM.Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{PKEM.KG}(sp)$ and gives (sp, pk) to A.
Phase 2: A issues decapsulation queries, each of which is denoted by (ψ, σ). On receiving (ψ, σ), the game computes $\mathsf{PKEM.Decap}(sp, sk, \psi, \sigma)$ and gives the resulting decapsulation K (which can be $\perp$) to A.
Phase 3: The game subsequently computes $(\omega^*, \psi^*) \leftarrow \mathsf{PKEM.Encap1}(sp)$ and $(\sigma^*, K_1^*) \leftarrow \mathsf{PKEM.Encap2}(sp, pk, \omega^*, \psi^*)$. It also picks K_0^* at random from the key space $\mathcal{K}_K$. The game then picks $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$ and gives $(\psi^*, \sigma^*, K_b^*)$ to A.
Phase 4: A issues decapsulation queries, each of which is denoted by (ψ, σ). A restriction here is that $(\psi, \sigma) \neq (\psi^*, \sigma^*)$. On receiving (ψ, σ), the game computes
$\mathsf{PKEM.Decap}(sp, sk, \psi)$ and gives the resulting decapsulation K (which can be $\perp$) to A. At the end of this phase, A outputs its guess $b' \in \{0,1\}$.

We define A's advantage by $\mathbf{Adv}^{\text{IND-CCA}}_{A,\mathsf{PKEM}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$.

Proving the security of our generic construction of StPE in the KSK model given in Section 3.1 requires us to define a new property of PKEM, which we call "*reproducibility*". Informally, this means that there exists a polynomial-time algorithm that, given a PKEM-ciphertext created under some public key and state information, and some other public/private key pair, produces another PKEM-ciphertext valid under the other public key and *the same state information* as the given ciphertext. (We note that a similar notion has been considered in [4] in the context of multi-receiver PKE.)

Intuitively, the reason why we require reproducibility is as follows. Given the specific state information, the adversary in the IND-CCA game of StPE (Definition 2) can come up with public keys of receivers other than the target receiver (denoted R_1) and produce ciphertexts associated with these public keys and the given state. In the KSK model, since the adversary is assumed to know private keys, each of which corresponds to each public key, he can produce such ciphertexts by himself. Now, a formal definition follows.

Definition 5 (Reproducibility of PKEM). Let PKEM be a PKEM scheme. Consider a game played with an algorithm R:

> **Phase 1:** The game computes $sp \leftarrow \mathsf{PKEM.Setup}(1^\lambda)$, $(sk, pk) \leftarrow \mathsf{PKEM.KG}(sp)$, $(\omega, \psi) \leftarrow \mathsf{PKEM.Encap1}(sp)$, $(\sigma, K) \leftarrow \mathsf{PKEM.Encap2}(sp, pk, \omega, \psi)$ and $(sk', pk') \leftarrow \mathsf{PKEM.KG}(sp)$. It gives $(sp, pk, \psi, \sigma, sk', pk')$ to R.
> **Phase 2:** R outputs (σ', K').

We define R's advantage by

$$\mathbf{Adv}^{\text{PKEM-Repr}}_{R,\mathsf{PKEM}}(\lambda) = \Pr[(\sigma', K') = \mathsf{PKEM.Encap2}(sp, pk', \omega, \psi)].$$

We say that the PKEM scheme is reproducible if $\mathbf{Adv}^{\text{PKEM-Repr}}_{R,\mathsf{PKEM}}(\lambda) = 1$.

Finally we define another type of security of PKEM, which is an extension of one-wayness (OW) under key checking attack (KCA) defined in [2]. (Note that KCA can be considered as a KEM version of the plaintext checking attack (PCA) defined in [23].) Our extension strengthens the OW-KCA of [2] in such a way that an attacker can freely choose a public key and include it to the "key checking" query. A formal definition, which we call "OW-EKCA (extended key checking attack)" is as follows.

Definition 6 (OW-EKCA of PKEM). Let PKEM be a PKEM scheme. Consider a game played with an attacker A:

> **Phase 1:** The game computes $sp \leftarrow \mathsf{PKEM.Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{PKEM.KG}(sp)$, $(\omega^*, \psi^*) \leftarrow \mathsf{PKEM.Encap1}(sp)$ and $(\sigma^*, K^*) \leftarrow \mathsf{PKEM.Encap2}(sp, pk, \omega^*, \psi^*)$, and gives $(sp, pk, \psi^*, \sigma^*)$ to A.
> **Phase 2:** A issues key checking queries, each of which is denoted by $(pk', \psi', \sigma', K')$. On receiving it, the game checks whether (ψ', σ') encapsulates K' or not with respect to pk'. If it is, the game returns 1, and 0 otherwise.
> – We write this checking procedure as $\mathsf{EKCO}(pk', \psi', \sigma', K')$, which returns 1 if (ψ', σ') encapsulates K' under the key pk' and 0 otherwise.
> **Phase 3:** A outputs its guess K.

We define A's advantage by $\mathbf{Adv}^{\text{OW-EKCA}}_{A,\mathsf{PKEM}}(\lambda) = \Pr[K = K^*]$.

2.3 Symmetric Encryption

To construct IND-CCA secure StPE schemes, we need a somewhat strong symmetric encryption scheme. Note that in the usual KEM/DEM framework (DEM: Data Encapsulation Mechanism) for hybrid encryption [14], it is sufficient that the underlying symmetric encryption is IND-CCA in the weak sense that the attacker does not issue queries to the encryption oracle. In contrast, we need symmetric encryption secure against CCA attack in which the attacker does issue encryption queries. – A formal definition of IND-CCA for symmetric encryption can naturally be defined and can easily be found in the literature including [14].

We remark that as mentioned in [6], the symmetric encryption schemes meeting the IND-CCA definition can in fact be easily constructed, eg. using the encrypt-then-mac composition [7] with an AES mode of operation (such as CBC) and a MAC (such as CBC-MAC or HMAC [5]).

3 Our Constructions

3.1 Construction in the KSK Model

Description. We assume that a PKEM scheme PKEM and a symmetric encryption scheme SYM are "compatible" meaning that the key space $\mathcal{K}_K$ of PKEM is the same as the key space $\mathcal{K}_D$ of SYM. We use these schemes as building blocks to construct a stateful encryption scheme StPE. Below, we describe each sub-algorithm of StPE.

StPE.Setup is the same as PKEM.Setup, which outputs system parameter sp. Likewise, StPE.KG is the same as PKEM.KG, which outputs (sk, pk), a private/public key pair. StPE.PKCk simply returns 1 (and does nothing else) as the KSK model implies that any public keys in this system are generated correctly following the algorithm StPE.KG. (Namely the entity that has generated a public key must know the corresponding private key.)

In our construction of stateful encryption, we assume that only two types of state exist. The first type of state is produced by StPE.NwSt, which simply returns the output of PKEM.Encap1 on input sp. This state is kept unchanged until StPE.NwSt is invoked again to produce fresh state of the first type. The second type of state is produced by the algorithm StPE.Enc, which appends the first type of state output by StPE.NwSt to pk (provided as input to StPE.Enc) and the output of PKEM.Encap2. (Note here that PKEM.Encap2 takes (sp, pk), the state output by StPE.NwSt and a plaintext M as input.) We also assume that pk and the output of PKEM.Encap2 of the second type of state is modified only by StPE.Enc. In what follows, we give algorithmic descriptions of StPE.NwSt, StPE.Enc and StPE.Dec.

Note that by the assumptions stated earlier, there are the following cases for st provided as input to StPE.Enc to become: 1) (ω, ψ) which is output of StPE.NwSt; 2) $(\omega, \psi, pk', \sigma', K')$ where σ' and K' are the outputs of PKEM.Encap2, both of which are created under the public key pk' different from the public key pk provided as input to StPE.Enc; and 3)$(\omega, \psi, pk, \sigma, K)$ where σ and K are created

under the public key pk provided as input to StPE.Enc. Note also that for state $st = (\omega, \psi)$ generated by the algorithm StPE.NwSt, $[\mathsf{StPE.Enc}(sp, pk, st, M)]_C = [\mathsf{StPE.Enc}(sp, pk, st', M)]_C$ for any st' output by StPE.Enc before StPE.NwSt is invoked to generate new state (different from st). Here, "$[\mathsf{StPE.Enc}(\cdots)]_C$" denotes the ciphertext part of an output of StPE.Enc. – This property is used crucially to prove the security of the proposed construction.

StPE.NwSt(sp)
  $(\omega, \psi) \leftarrow$ PKEM.Encap1(sp)
  $st \leftarrow (\omega, \psi)$
  Return st
StPE.Enc(sp, pk, st, M)
  If st is of the form (ω, ψ) or of the form $(\omega, \psi, pk', \sigma', K')$ such that $pk' \neq pk$ then
    $(\sigma, K) \leftarrow$ PKEM.Encap2(sp, pk, ω, ψ)
  Else
    Parse st as $(\omega, \psi, pk, \sigma, K)$
  $e \xleftarrow{R}$ SYM.Enc(K, M)
  $C \leftarrow (\psi, \sigma, e)$
  $st \leftarrow (\omega, \psi, pk, \sigma, K)$
  Return (C, st)
StPE.Dec(sp, sk, C)
  Parse C as (ψ, σ, e)
  $K \leftarrow$ PKEM.Decap(sp, sk, ψ, σ)
  If $K = \perp$ then return $\perp$
  Else return SYM.Dec(K, e)

We remark that the StPE.Enc algorithm becomes *highly efficient* when a sender sends encryptions to a single receiver: If the sender wants to send encryptions of $M_1, \ldots, M_n$ to the same receiver whose public key is pk, he does not have to run PKEM.Encap2 and PKEM.Key for each plaintext M_i for $i = 1 \ldots, n$ but just runs them once at the beginning and then only runs SYM.Enc on input (K, M_i) afterwards.

Security Analysis of Generic Construction. Our generic construction of StPE seems to be reminiscent of the KEM/DEM paradigm of constructing hybrid encryption given in [14]. However, the *PKEM-reproducibility* that we defined in the previous section (Definition 5) and the "reuse" of a ciphertext and its corresponding key of the PKEM scheme for the multiple encryptions without compromising confidentiality due to the strong security of the SYM scheme are the features that distinguish our generic construction of StPE from the KEM/DEM framework for constructing normal hybrid encryption. We now prove the following theorem.

Theorem 1. *In the KSK model, the proposed generic stateful public key encryption scheme* StPE *is IND-CCA secure if the underlying PKEM scheme* PKEM *is IND-CCA secure and reproducible, and the underlying symmetric encryption scheme* SYM *is IND-CCA secure. More precisely, we have*

$$\mathbf{Adv}^{IND\text{-}CCA}_{A,\mathsf{StPE}}(\lambda) \leq 2\mathbf{Adv}^{IND\text{-}CCA}_{B_1,\mathsf{PKEM}}(\lambda) + \mathbf{Adv}^{IND\text{-}CCA}_{B_2,\mathsf{SYM}}(\lambda),$$

where λ denotes the security parameter; A, B_1 and B_2 denote the corresponding attackers.

Proof. The proof uses the technique of sequence of games [26].

- Game $\mathsf{G_0}$: This game is identical to the IND-CCA game played by an attacker A against StPE. (Readers are referred to Definition 2.) We repeat this game to clean up the notations. Let sp be a system parameter. Let pk_1 and sk_1 be public and private keys of the honest receiver respectively. Let $pk_2, \ldots, pk_n$ be the public keys output by A. Let $st = (\omega^*, \psi^*)$, where $(\omega^*, \psi^*) \leftarrow \mathsf{PKEM.Encap1}(sp)$, be the sender's state generated by StPE.NwSt, fixed throughout each game. We denote a challenge ciphertext by $C^* = (\psi^*, \sigma^*, e^*)$, where $(\sigma^*, K_1^*) \leftarrow \mathsf{PKEM.Encap2}(sp, pk_1, \omega^*, \psi^*)$ and $e^* \xleftarrow{\mathrm{R}} \mathsf{SYM.Enc}(K_1^*, M_\beta)$ for $\beta \xleftarrow{\mathrm{R}} \{0,1\}$.
  Now, observe that we can assume that A does not make encryption queries of the form (i, M) for $i = 2, \ldots, n$. The reason is that since A is assumed to know sk_i corresponding to its public key pk_i following the KSK model, by the *reproducibility*, it, given (ψ^*, σ^*, pk_1), can compute (σ_i, K_i) such that $(\sigma_i, K_i) \leftarrow \mathsf{PKEM.Encap2}(sp, pk_i, \omega^*, \psi^*)$ for $i = 2, \ldots, n$. Consequently, it can compute $e_i \xleftarrow{\mathrm{R}} \mathsf{SYM.Enc}(K_i, M)$ and can create ciphertext $C_i = (\psi^*, \sigma_i, e_i)$ for all $i = 2, \ldots, n$.
  We denote by S_0 the event $\beta' = \beta$, where β' is a bit output by A at the end of the game. (We use a similar notation $S_1, S_2, \ldots$ for all modified games $\mathsf{G_1}, \mathsf{G_2}, \ldots$ respectively). Since $\mathsf{G_0}$ is the same as the real attack game of the IND-CCA of StPE, we have
- Game $\mathsf{G_1}$: In this game, we modify the generation of e^* (a component of challenge ciphertext) of the previous game as follows: $e^* \xleftarrow{\mathrm{R}} \mathsf{SYM.Enc}(K_0^*, M_\beta)$, where K_0^* is the key chosen at random from $\mathcal{K}_K (= \mathcal{K}_D)$ and $\beta \xleftarrow{\mathrm{R}} \{0,1\}$.
  Now, in the following, we construct an oracle machine B_1 that breaks IND-CCA of PKEM using A as a subroutine.
  Algorithm $B_1(sp, pk_1)$
  Give (sp, pk_1) to A
  If A issues a challenge query (M_0, M_1) such that $|M_0| = |M_1|$ then get a challenge ciphertext/key pair $(\psi^*, \sigma^*, K_b^*)$ where $b \xleftarrow{\mathrm{R}} \{0,1\}$ from the challenger; $\beta \xleftarrow{\mathrm{R}} \{0,1\}$; $e^* \xleftarrow{\mathrm{R}} \mathsf{SYM.Enc}(K_b^*, M_\beta)$;
  $C^* \leftarrow (\psi^*, \sigma^*, e^*)$; Give C^* to A
  If A issues an encryption query $(1, M)$ then
  $e \xleftarrow{\mathrm{R}} \mathsf{SYM.Enc}(K_b^*, M)$; $C \leftarrow (\psi^*, \sigma^*, e)$; Give C to A.

If A issues a decryption query $C \neq C^*$, where $C = (\psi, \sigma, e)$, then
If $(\psi, \sigma) \neq (\psi^*, \sigma^*)$ then query (ψ, σ) to the
challenger to get $K = \mathsf{Decap}(sp, sk_1, \psi, \sigma)$
If $K \neq \perp$ then return $\mathsf{SYM.Dec}(K, e)$
Else return $\perp$
Else return $\mathsf{SYM.Dec}(K_b^*, e)$
If A outputs β' such that $\beta' = \beta$ then return $b' = 1$ (b' is B_1's guess on b)
Else return $b' = 0$

First, assume that $b = 1$ in the above construction. In this case, note that K_1^* is the right key of PKEM. Importantly, note also that $[\mathsf{StPE.Enc}(sp, pk, st, M)]_C = [\mathsf{StPE.Enc}(sp, pk, st', M)]_C$ for any st' produced by StPE.Enc before StPE.NwSt is invoked to produce new state different from st. (Recall that "$[\mathsf{StPE.Enc}(\cdots)]_C$" denotes the ciphertext part of an output of StPE.Enc.) Hence, the ciphertexts (ψ^*, σ^*, e^*) and (ψ^*, σ^*, e) provided as responses to A's challenge and encryption queries respectively and those in the real attack game (which is Game $\mathsf{G_0}$) are distributed identically. Decryptions are also perfectly simulated. Consequently, B_1 creates the same environment as Game $\mathsf{G_0}$ in which A outputs its guess β'. Hence, we have $\Pr[S_0] = \Pr[\beta' = \beta] = \Pr[b' = 1 | b = 1]$. Next, assume that $b = 0$ in the above construction. Note that in this case, B_1 creates the same environment as this game (Game $\mathsf{G_1}$) in which K_0^* is the key chosen at random from $\mathcal{K}_K (= \mathcal{K}_D)$. Hence, we have $\Pr[S_1] = \Pr[\beta' = \beta] = \Pr[b' = 1 | b = 0]$. Thus, we obtain

$$
\begin{aligned}
|\Pr[S_0] - \Pr[S_1]| &= |\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]| \\
&= 2\Big(|\frac{1}{2}\Pr[b' = 1 | b = 1] - \frac{1}{2}\Pr[b' = 1 | b = 0]|\Big) \\
&= 2\Big(|\frac{1}{2}\Pr[b' = 1 | b = 1] + \frac{1}{2}(1 - \Pr[b' = 1 | b = 0]) - \frac{1}{2}|\Big) \\
&= 2\Big(|\frac{1}{2}\Pr[b' = 1 | b = 1] + \frac{1}{2}\Pr[b' = 0 | b = 0] - \frac{1}{2}|\Big) \\
&= 2\mathbf{Adv}^{\text{IND-CCA}}_{B_1, \mathsf{PKEM}}(\lambda).
\end{aligned}
$$

Now, in the following, we construct an oracle machine B_2 that breaks IND-CCA of SYM using the attacker A as a subroutine.
Algorithm $B_2(\lambda)$

Generate sp, sk_1 and pk_1; Give (sp, pk_1) to A
$(\omega^*, \psi^*) \leftarrow \mathsf{PKEM.Encap1}(sp)$;
$(\sigma^*, K^*) \leftarrow \mathsf{PKEM.Encap2}(sp, pk_i, \omega^*, \psi^*)$;
If A issues a challenge query (M_0, M_1) such that $|M_0| = |M_1|$ then
query (M_0, M_1) to the challenger to get $e^* \stackrel{\mathrm{R}}{\leftarrow} \mathsf{SYM.Enc}(K_0^*, M_\beta)$
where $\beta \stackrel{\mathrm{R}}{\leftarrow} \{0, 1\}$; $C^* \leftarrow (\psi^*, \sigma^*, e^*)$; Give C^* to A.
If A issues an encryption query $(1, M)$ then
query M to the challenger to get $e \stackrel{\mathrm{R}}{\leftarrow} \mathsf{Enc}(K_0^*, M)$;

$C \leftarrow (\psi^*, \sigma^*, e)$; Give C to A.
If A issues a decryption query $C \neq C^*$, where $C = (\psi, \sigma, e)$, then
If $(\psi, \sigma) \neq (\psi^*, \sigma^*)$ then
$K \leftarrow \mathsf{PKEM.Decap}(sp, sk_1, \psi, \sigma)$
If $K \neq \perp$ then return $\mathsf{SYM.Dec}(K, e)$
Else return $\perp$
Else query e (which must be different from e^*) to the challenger to get $d = \mathsf{SYM.Dec}(K_0^*, e)$; Return d
If A outputs β' then return β'

Observe that in the above algorithm B_2, A is essentially conducting chosen ciphertext attack on the symmetric encryption scheme SYM. Thus we have

$$\left| \Pr[S_1] - \frac{1}{2} \right| \leq \mathbf{Adv}^{\text{IND-CCA}}_{B_2,\mathsf{SYM}}(\lambda).$$

3.2 Construction in the USK Model

Description. Let H be a random oracle [8], whose range (output-space) is the same as the key space $\mathcal{K}_D$ of the symmetric encryption scheme SYM. Assume that there exists an algorithm PKV which checks whether public key of the given PKEM scheme PKEM is valid. We note that PKV is not an algorithm for "proving" the possession of the corresponding private key but a simple mechanism for validating keys or domain parameters by, eg. showing they belong to the output space of the key generation algorithm, as described in the public key cryptography standard such as P1363 [21]. (Readers are particularly referred to Section D.3.3 of the P1363 specification.)

Using the PKEM and SYM schemes and the random oracle [8] H as building blocks we construct another stateful encryption scheme StPE as follows. StPE.Setup is the same as PKEM.Setup, which outputs sp. Also, StPE.KG is the same as PKEM.KG, which outputs (sk, pk). We assume here that sk includes pk. StPE.PKCk runs the algorithm PKV to check whether a given public key pk is in $\{\mathsf{PKEM.KG}(sp)\}$.

Like the construction of stateful encryption in the KSK model presented in the previous section, we assume that there exist only two types of state. The first type of state is produced by StPE.NwSt, which simply returns the output of PKEM.Encap1 on input sp. This state is kept unchanged until StPE.NwSt is invoked again to produce fresh state of the first type. The algorithm StPE.Enc produces the second type of state by appending the first type of state output by StPE.NwSt to pk (provided as input to StPE.Enc) and the session key $\tilde{K}$ output by H. (Note here that H takes as input the part of the state output by StPE.NwSt, pk and the output of PKEM.Encap2.) We also assume that pk and the output of PKEM.Encap2 of the second type of state is modified only by StPE.Enc.

Security Analysis of Generic Construction. We now state the following theorem regarding the security of the construction presented above. – Due to the page limit, the proof will be provided in the full version of this paper.

$\mathsf{StPE.NwSt}(sp)$
  $(\omega, \psi) \leftarrow \mathsf{PKEM.Encap1}(sp)$
  $st \leftarrow (\omega, \psi)$
  Return st
$\mathsf{StPE.Enc}(sp, pk, st, M)$
  If st is of the form (ω, ψ) or of the form $(\omega, \psi, pk', \sigma', \tilde{K}')$ such that $pk' \neq pk$ then
    $(\sigma, K) \leftarrow \mathsf{PKEM.Encap2}(sp, pk, \omega, \psi)$
    $\tilde{K} \leftarrow \mathsf{H}(\psi, pk, \sigma, K)$
  Else
    Parse st as $(\omega, \psi, pk, \sigma, \tilde{K})$
  $e \stackrel{\mathrm{R}}{\leftarrow} \mathsf{SYM.Enc}(\tilde{K}, M)$
  $C \leftarrow (\psi, \sigma, e)$
  $st \leftarrow (\omega, \psi, pk, \sigma, \tilde{K})$
  Return (C, st)
$\mathsf{StPE.Dec}(sp, sk, C)$
  Parse C as (ψ, σ, e)
  $K \leftarrow \mathsf{PKEM.Decap}(sp, sk, \psi)$
  If $K = \perp$ then return $\perp$
  Else
    $\tilde{K} \leftarrow \mathsf{H}(\psi, pk, \sigma, K)$
    Return $\mathsf{SYM.Dec}(\tilde{K}, e)$

Theorem 2. *In the USK model, the proposed generic stateful public key encryption scheme* StPE *described above is IND-CCA secure if the underlying hash function* H *is modeled as random oracle; the underlying* PKEM *scheme is OW-EKCA secure; and the underlying* SYM *scheme is IND-CCA secure. More precisely, we have*

$$\mathbf{Adv}^{IND\text{-}CCA}_{A,\mathsf{StPE}}(\lambda) \leq \mathbf{Adv}^{OW\text{-}EKCA}_{B_1,\mathsf{PKEM}}(\lambda) + \mathbf{Adv}^{IND\text{-}CCA}_{B_2,\mathsf{SYM}}(\lambda),$$

where λ denotes the security parameter; A, B_1 and B_2 denote the corresponding attackers as defined in Section 2.

4 Applications

4.1 A StPE Scheme Based on the Identity-Based Technique by Boyen, Mei and Waters

One of interesting applications of our generic constructions is to build StPE based on the *identity-based technique* which converts IBE schemes into (possibly efficient) IND-CCA secure PKE schemes in the standard model [13]. (Recall that Bellare et al. [6] asked whether the PKE schemes constructed in this way have stateful variants.) Among various identity-based techniques available in

the literature, we select, due to their efficiency, one of Boyen, Mei and Waters' presented in [11], and Boneh and Katz's one presented in [10], both of which are essentially yielded from the Boneh-Boyen selective-ID IBE [9].

We now describe a PKEM scheme derived from Boyen et al.'s KEM scheme in [11], which we denote by BMW. Readers are referred to Appendix 4.2 for the application of our generic construction of StPE to Boneh and Katz's identity-based technique [10].

Description of BMW *PKEM.* By simply rearranging Boyen et al.'s KEM scheme, one can obtain BMW as follows. On input 1^λ, BMW.Setup picks groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ of prime order q, generated by g and h respectively. It then constructs a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. It picks a collision resistant hash function $H_s : \mathbb{G} \to \mathbb{Z}_q$. Finally it returns $sp = (\lambda, q, g, h, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, H_s)$. BMW.KG($sp$) selects $\alpha \xleftarrow{R} \mathbb{Z}_q$ and $l \leftarrow h^\alpha$, and computes $Z \leftarrow e(g, l)$. (Note that $l \in \hat{\mathbb{G}}$.) It then picks $x \xleftarrow{R} \mathbb{Z}_q$ and $y \xleftarrow{R} \mathbb{Z}_q$, and computes $u \leftarrow g^x$ and $v \leftarrow g^y$. It chooses a random seed s and returns $sk = (pk, l, x, y)$ and $pk = (s, Z, u, v)$. The rest of the algorithms are described as follows:

BMW.Encap1(sp)	BMW.Encap2(sp, pk, r, ψ)	BMW.Decap(sp, sk, ψ, σ)
$r \xleftarrow{R} \mathbb{Z}_q$; $\psi \leftarrow g^r$	$w \leftarrow H_s(\psi)$	$w \leftarrow H_s(\psi)$
Return (r, ψ)	$\sigma \leftarrow u^r v^{rw}$; $K \leftarrow Z^r$	$\bar{w} \leftarrow x + yw \pmod q$
	Return (σ, K)	If $\psi^{\bar{w}} = \sigma$ then
		$K \leftarrow e(\psi, l)$; Return K
		Else return $\perp$

Note that in the above description r denotes state information (represented by ω in Definition 3). Hereinafter, we use r to denote state information.

StPE from BMW *PKEM.* In [11], the BMW scheme is shown to be IND-CCA secure assuming that the Decisional Bilinear Diffie-Hellman (DBDH) problem is intractable. We now prove that it satisfies reproducibility (Definition 5) as well.

Lemma 1. BMW *PKEM is reproducible.*

Proof. Let $sp = (\lambda, q, g, h, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, H)$ be a system parameter. Let $sk = (l, x, y)$ and $pk = (s, Z, u, v)$, where $l = h^\alpha$ for random $\alpha \in \mathbb{Z}_q$, $u = g^x$ and $v = g^y$, be private and public keys respectively. Suppose that another private/public key pair (sk', pk') such that $sk' = (l', x', y')$ and $pk' = (s', Z', u', v')$, where $l = h^{\alpha'}$ for random $\alpha' \in \mathbb{Z}_q$, $u' = g^{x'}$ and $v' = g^{y'}$, is generated. Also, let $(\psi, \sigma) = (g^r, u^r v^{rw})$ for random $r \in \mathbb{Z}_q$, where $w = H_s(\psi)$, and $K = Z^r$.

Given $(sp, pk, \psi, \sigma, sk', pk')$, one can compute $w' \leftarrow H_{s'}(\psi)$; $\sigma' \leftarrow \psi^{x'+y'w'}$; $K' \leftarrow e(\psi, l')$ and outputs (σ', K'). Note that $\psi^{\bar{w}'} = g^{r\bar{w}'} = g^{r(x'+y'w')} = \psi^{x'+y'w'} = \sigma'$ assuming that $\bar{w}' = x' + y'w' \pmod q$. Note also that $(g, \psi, u' v'^{w'}, \sigma')$ is a Diffie-Hellman tuple, and that $K' = e(\psi, l') = e(g, l')^r = Z'^r$. Thus, $(\sigma', K') =$ PKEM.Encap2(sp, pk', r, ψ).

Now, assume that the key space of $\mathcal{K}_K$ of the BMW scheme is the same as that of the symmetric encryption scheme, which we denote by SYM. (Note that this can easily be achieved by providing the key K to the key derivation function (KDF) [14]. If the KDF is secure in the sense of "indistinguishability" as defined in [14] and the original KEM is IND-CCA, the resulting KEM scheme is also IND-CCA, which can easily be shown.) Then, if the SYM scheme is IND-CCA secure, by the result of Theorem 1, the StPE scheme based on BMW PKEM is IND-CCA secure in the KSK model without the random oracles.

4.2 A StPE Scheme Based on the Identity-Based Technique by Boneh and Katz

Another StPE scheme based on the identity-based technique can be built using our generic construction. This time the underlying PKEM scheme is derived from Boneh and Katz's [10] identity-based technique. By BK, we denote this PKEM scheme.

Description of BK *PKEM.* On input 1^λ, BK.Setup first picks groups $\mathbb{G}$ and $\mathbb{G}_1$ of prime order q, where $\mathbb{G}$ is generated by g. It then constructs a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. It selects a pseudorandom generator $G : \mathbb{G}_1 \to \{0,1\}^*$, a second-preimage resistant hash function $H : \{0,1\}^{448} \to \{0,1\}^{128}$ and a message authentication scheme MAC=$(\mathcal{T}, \mathcal{V})$. – $\mathcal{T}$ and $\mathcal{V}$ are tagging and verification algorithms respectively. Finally it returns $sp = (\lambda, q, g, \mathbb{G}, \mathbb{G}_1, e, e(g,g), G, H, \mathsf{MAC})$. BK.KG($sp$) picks $\alpha_1 \xleftarrow{R} \mathbb{Z}_q$, $\alpha_2 \xleftarrow{R} \mathbb{Z}_q$ and $x \xleftarrow{R} \mathbb{Z}_q$; computes $g_1 \leftarrow g^{\alpha_1}$, $g_2 \leftarrow g^{\alpha_2}$, $g_3 \leftarrow g^x$ and $Z \leftarrow e(g,g)^{\alpha_1 x}$. It chooses a hash function h from a family of pairwise independent hash functions. It returns $sk = (\alpha_1, \alpha_2, x, h)$ and $pk = (g_1, g_2, g_3, Z, h)$. The rest of the algorithms are described as follows:

BK.Encap1(sp)	BK.Encap2(sp, pk, r, ψ)	BK.Decap(sp, sk, ψ, σ)
$r \xleftarrow{R} \mathbb{Z}_q$; $\psi \leftarrow g^r$ Return (r, ψ)	$s \xleftarrow{R} \{0,1\}^{448}$ $k_1 \leftarrow h(s)$; $K \xleftarrow{R} \mathcal{K}_K$ $\rho \leftarrow H(s)$; $\theta \leftarrow g_2^r g_3^{r\rho}$ $\phi \leftarrow G(Z^r) \oplus (K \| s)$ $\tau \leftarrow \mathcal{T}(k_1, (\psi, \theta, \phi))$ $\sigma \leftarrow (\rho, \theta, \phi, \tau)$ Return (σ, K)	Parse σ as $(\rho, \theta, \phi, \tau)$; $t \xleftarrow{R} \mathbb{Z}_q$; $(K \| s) \leftarrow \phi \oplus G(e(\psi^{\alpha_1 x + t(\alpha_2 + x\rho)} \theta^{-t}, g))$ $k_1 \leftarrow h(s)$ If $\mathcal{V}(k_1, (\psi, \theta, \phi), \tau) = 1$ and $H(s) = \rho$ then return K Else return $\perp$

StPE from BK *KEM.* We first prove that BK has the reproducibility (Definition 5).

Lemma 2. BK *PKEM is reproducible.*

Proof. Let $sp = (\lambda, q, g, \mathbb{G}, \mathbb{G}_1, e, e(g,g), G, H, \mathsf{MAC})$ be a system parameter as defined earlier. Let $sk = (\alpha_1, \alpha_2, x, h)$ and $pk = (g_1, g_2, g_3, Z, h)$, where $g_1 = g^{\alpha_1}$, $g_2 = g^{\alpha_2}$, $g_3 = g^x$, $Z = e(g,g)^{\alpha_1 x}$ and h is drawn from a family of pairwise

independent hash functions, be private and public keys respectively. Suppose that another private/public key pair (sk', pk') such that $sk' = (\alpha'_1, \alpha'_2, x', h')$ and $pk' = (g'_1, g'_2, g'_3, Z', h')$, where $g'_1 = g^{\alpha'_1}$, $g'_2 = g^{\alpha'_2}$, $g'_3 = g^{x'}$, $Z' = e(g,g)^{\alpha'_1 x'}$ and h' is drawn from a family of pairwise independent hash functions, is generated. Also, let $\psi = g^r$ and $\sigma = (\rho, \theta, \phi, \tau)$, where $\rho = H(s)$, $\theta = g_2^r g_3^{r\rho}$, $\phi = G(Z^r) \oplus (K||s)$ and $\tau = \mathcal{T}(k_1, (\psi, \theta, \phi))$, where $k_1 = h(s)$, for random $r \in \mathbb{Z}_q$, $s \in \{0,1\}^{448}$ and $K \in \mathcal{K}_K$.

Given $(sp, pk, \psi, \sigma, sk', pk')$, one can compute $s' \stackrel{R}{\leftarrow} \{0,1\}^{448}$; $\rho' \leftarrow H(s')$; $\theta' \leftarrow \psi^{\alpha'_2} \psi^{x'\rho'}$; $K' \stackrel{R}{\leftarrow} \mathcal{K}_K$; $\phi' \leftarrow G(e(\psi, g)^{\alpha'_1 x'}) \oplus (K'||s')$; $k'_1 \leftarrow h'(s')$; $\tau' \leftarrow \mathcal{T}(k'_1, (\psi, \theta', \phi'))$ and output $\sigma' = (\rho', \theta', \tau')$ and K'.

Note that $\theta' = \psi^{\alpha'_2} \psi^{x'\rho'} = g^{r\alpha'_2} g^{rx'H(s')} = (g'_2)^r (g'_3)^{rH(s')}$. Note also that $e(\psi, g'_1)^{x'} = e(g, g'_1)^{rx'} = Z'^r$. It is clear that τ' is valid. Thus, $(\sigma', K') = \mathsf{BK.Encap2}(sp, pk', r, \psi)$.

Note that the above BK PKEM scheme is derived simply from the Boneh and Katz's PKE scheme (converted from selective-ID IBE) by providing a random key instead of a plaintext as input to the encryption algorithm (and separating the ciphertext part which depends on the system parameter only). One can easily show that if Boneh and Katz's PKE scheme is IND-CCA secure relative to the DBDH problem, which is actually shown in [10], the BK KEM scheme is also IND-CCA secure assuming that the DBDH problem is hard. Now, assume that the key space of $\mathcal{K}_K$ of the BK scheme is the same as that of the symmetric encryption scheme SYM. Then, if the SYM scheme is IND-CCA secure, by the result of Theorem 1, the StPE scheme from BK PKEM is IND-CCA secure in the KSK model without random oracles.

4.3 A StPE Scheme from Kiltz's KEM Scheme

One can also apply the generic construction presented in Section 3.1 to the KEM scheme proposed by Kiltz [18] very recently. Next, we describe the PKEM version of this scheme, which we denote by KI.

Description of KI *PKEM.* On input 1^λ, KI.Setup picks a group $\mathbb{G}$ of prime order q, generated by g. It then picks a target-collision resistant hash function $\mathsf{H} : \mathbb{G} \to \mathbb{Z}_q$ and a key derivation function KDF. Finally it returns $sp = (\lambda, q, g, \mathbb{G}, \mathsf{H}, \mathsf{KDF})$. KI.KG$(sp)$ picks $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and $y \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and computes $u \leftarrow g^x$ and $v \leftarrow g^y$. It returns $sk = (x, y)$ and $pk = (u, v)$. The rest of the algorithms are described as follows:

KI.Encap1(sp)	KI.Encap2(sp, pk, r, ψ)	KI.Decap(sp, sk, ψ, σ)
$r \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$; $\psi \leftarrow g^r$;	$t \leftarrow \mathsf{H}(\psi)$; $\sigma \leftarrow (u^t v)^r$	$t \leftarrow \mathsf{H}(\psi)$
Return (r, ψ)	$K \leftarrow \mathsf{KDF}(u^r)$	$K \leftarrow \mathsf{KDF}(\psi^x)$
	Return (σ, K)	If $\psi^{xt+y} = \sigma$
		Return K
		Else return $\perp$

StPE from KI *PKEM.* We prove that the above KI PKEM scheme satisfies the reproducibility.

Lemma 3. KI *PKEM is reproducible.*

Proof. Let $sp = (\lambda, q, g, \mathbb{G}, \mathsf{H}, \mathsf{KDF})$ be a system parameter. Let $sk = (x, y)$ and $pk = (u, v)$, where $u = g^x$ and $v = g^y$. Suppose that another private/public key pair (sk', pk') such that $sk' = (x', y')$ and $pk' = (u', v')$, where $u' = g^{x'}$ and $v' = g^{y'}$, is generated. Also, let $(\psi, \sigma) = (g^r, (u^t v)^r)$, where $t = \mathsf{H}(\psi)$.

Given $(sp, pk, \psi, \sigma, sk', pk')$, one can compute $\sigma' \leftarrow \psi^{x't+y'}; K' \leftarrow \mathsf{KDF}(\psi^{x'})$, where $t = \mathsf{H}(\psi)$, and output (σ', K'). Note that $\sigma' = g^{r(x't+y')} = (g^{x't} g^{y'})^r = (u'^t v')^r$. Thus, $(\sigma', K') = \mathsf{PKEM.Encap2}(sp, pk', r, \psi)$.

It is shown in [18] that the above KI PKEM scheme is IND-CCA secure assuming that the Gap Hashed Diffie-Hellman (GHDH) problem is hard and the underlying hash function H is target-collision resistant.

Thus, assuming that the key space of $\mathcal{K}_K$ of the KD scheme is the same as that of the symmetric encryption scheme SYM and the SYM scheme is IND-CCA secure, the StPE scheme based on KI PKEM is IND-CCA secure in the KSK model without the random oracles, by the result of Theorem 1.

4.4 A StPE Scheme from the Diffie-Hellman KEM Scheme

The stateful version of DHIES [1] proposed in [6] can actually be explained using our generic construction presented in Section 3.2. We now present a PKEM version of the Diffie-Hellman KEM, which we denote by DH as follows.

Description of DH *PKEM.* On input 1^λ, DH.Setup picks a group $\mathbb{G}$ of prime order q, generated by g. It then returns $sp = (\lambda, q, g, \mathbb{G})$. DH.KG($sp$) picks $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and computes $y \leftarrow g^x$. It returns $sk = x$ and $pk = y$. The rest of the algorithms are described as follows:

DH.Encap1(sp)	DH.Encap2(sp, pk, r, ψ)	DH.Decap(sp, sk, ψ, σ)
$r \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$; $\psi \leftarrow g^r$	$\sigma \leftarrow \varepsilon$ (empty string)	$K \leftarrow \psi^x$
Return (r, ψ)	$K \leftarrow y^r$	Return K
	Return (σ, K)	

StPE from DH *PKEM.* We prove that the above DH PKEM scheme is OW-EKCA secure (Definition 6).

Lemma 4. DH *PKEM is OW-EKCA secure assuming that Gap Diffie-Hellman (GDH) problem [22] is intractable.*

Proof. Assume that an GDH attacker B is given an instance $(\lambda, q, \mathbb{G}, g, g^a, g^b)$. B sets $sp = (\lambda, q, g, \mathbb{G})$, $pk = y = g^b$, $\psi^* = g^a$ and $\sigma^* = \varepsilon$ (empty string). B gives $(sp, pk, \psi^*, \sigma^*)$ to an OW-EKCA attacker A. Whenever A issues an EKCA query $(pk', \psi', \sigma', K')$, B forwards the query to its DDH oracle and sends back the

response $\mathsf{DDH}_g(pk', \psi', K')$ to A. (Note here that the DDH oracle $\mathsf{DDH}_g(\cdot,\cdot,\cdot)$ returns 1 if a given tuple is (g^u, g^v, g^{uv}) for $u, v \in \mathbb{Z}_q^*$ and returns 0 otherwise.) When A outputs it guess $\bar{K}$, B returns it.

Hence, assuming that the key space of $\mathcal{K}_K$ of the DH scheme is the same as that of the symmetric encryption scheme SYM and the SYM scheme is IND-CCA secure, the StPE scheme based on DH is IND-CCA secure in the USK and random oracle model, by the result of Theorem 2.

5 Discussions

It is clear that the StPE schemes built from BMW PKEM in the Section 4.1 and BK PKEM in Appendix 4.2 are more efficient than the original PKE schemes based on the identity-based techniques presented in [11] and [10] respectively since the PKEM ciphertext (ψ, σ) and the key K are *reused* across encryptions directed to one receiver whose public key is pk. The proven security of these schemes ensures that reusing one instance of internal randomness appeared in the "state" across multiple receivers do not compromise the confidentiality. Moreover, the proof of security does not depend on the random oracles.

We note that the stateful version of the Kurosawa-Desmedt PKE scheme presented in [6] cannot be explained using our generic construction since the underlying PKEM defined in the same way as [20] is not IND-CCA secure as shown in [16]. However, we remark that by defining an extension of our approach based PKEM called "Tag-PKEM" (similar to Tag-KEM [3]), which provides a tag as input to PKEM.Encap2, one could analyze the stateful version of the Kurosawa-Desmedt PKE scheme in [6]. But we realize that a definition of IND-CCA of Tag-PKEM needs to allow an attacker to have access to a new kind of encapsulation oracle which returns outputs of PKEM.Encap2 computed under the *same internal state* but different tags. Note that this oracle is required to simulate the encryption oracle of StPE, when it is queried by $(1, M_1), \ldots, (1, M_n)$, whose responses should be encryptions of $M_1, \ldots, M_n$ under the same state. (In the IND-CCA definition of *stateless* PKE, this special type of oracle is not required.) Consequently, one cannot "reuse" the results of the security analysis of various Tag-KEMs available in the literature, as they need to be analyzed under this new security definition which is stronger than the normal IND-CCA definition of Tag-KEM given in [3].

Finally, we remark that along with the efficiency issue pointed out in [6], there is also a technical reason why constructing stateful versions of the RSA (or possibly usual integer factorization based schemes) is not very feasible: The reproducibility of PKEM (Definition 5) or OW-EKCA (Definition 6) is indeed strong property that many integer factorization based encryption schemes in which different receivers should have different modulus to guaranteer security are not likely to satisfy. Nevertheless, more elaboration on this issues would be an interesting area of research.

Acknowledgement

The authors are grateful to the anonymous referees of ACNS '08 for their helpful comments. This work is partially funded by the European Union project SMEPP-033563.

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. Abe, M.: Combining Encryption and Proof of Knowledge in the Random Oracle Model. Comput. J. 47(1), 58–70 (2004)
3. Abe, M., Genaro, R., Kurosawa, K.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM, Cryptology ePrint Archive: Report, 2005/027 (2005) (This is the full version of their Eurocrypt 2005 paper with the same title)
4. Bellare, M., Boldyreva, A., Staddon, J.: Randomness Re-use in Multi-recipient Encryption Schemeas. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2002)
5. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
6. Bellare, M., Kohno, T., Shoup, V.: Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation. In: ACM-CCS 2006, pp. 380–389. ACM Press, New York (2006)
7. Bellare, M., Namprepre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
8. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM-CCS 1993, pp. 62–73. ACM Press, New York (1993)
9. Boneh, D., Boyen, X.: Efficinet Selective-ID Secure Identity-Based Encryption without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
10. Boneh, D., Katz, J.: Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
11. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-Based Techniques. In: ACM-CCS 2005, pp. 320–329. ACM Press, New York (2005)
12. Bresson, E., Chevassut, O., Essiari, A., Pointcheval, D.: Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In: IFIP-TC6 International Conference on Mobile and Wireless Communications Networks, pp. 59–62. World Scientific Publishing, Singapore (2003)
13. Canetti, R., Halevi, S., Katz, J.: Chosen Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
14. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal of Computing 33, 167–226 (2003)

15. Gaubatz, G., Kaps, J.-P., Sunar, B.: Public Key Cryptography in Sensor Networks Revisited. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313. Springer, Heidelberg (2005)
16. Herranz, J., Hofheinz, D., Kiltz, E.: The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure,Cryptology ePrint Archive, Report 2006/207 (2006)
17. ISO 18033-2, An Emerging Standard for Public-Key Encryption (2004)
18. Kiltz, E.: Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
19. Kurosawa, K.: Multi-recipient Public-Key Encryption with Shortened Ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
20. 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)
21. IEEE P1363, Standard Specifications For Public-Key Cryptography (2000)
22. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
23. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
24. Phan, T., Huang, L., Dulan, C.: Challenge: Integrating Mobile Wireless Devices Into the Computational Grid. In: MobiCom 2002, pp. 271–278. ACM Press, New York (2002)
25. Sarkar, P., Chatterjee, S.: New Generic Constructions of Public Key Encryption from Identity Based Encryption, Cryptology ePrint Archive: Report 2007/067 (2007)
26. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs, Cryptology ePrint Archive: Report 2004/332 (2004)