Orthogonality between Key Privacy and Data Privacy, Revisited

Rui Zhang, Goichiro Hanaoka, and Hideki Imai

Research Center for Information Security (RCIS)
National Institute of Advanced Industrial Science and Technology (AIST)
{r-zhang,hanaoka-goichiro,h-imai}@aist.go.jp

Abstract. Key privacy is a notion regarding the privacy of the owner of a public key, which has important applications in building (receiver) anonymous channels, or privacy-enhanced authentication/signature schemes. Key privacy is considered to be an orthogonal (i.e., independent), notion from data privacy, while the key privacy of many public key encryption schemes has not been explored, though their data privacy is comparatively well understood. In this paper, we study key privacy of many practical encryption schemes and identify evidences that key privacy is not comparable to data privacy. We also formalize key privacy in the plaintext checking attack model and point out some generic transforms to enhance the key privacy of an encryption scheme. Interestingly, these well-known techniques have been used to enhance data security. Finally, we give detailed security analyses on the signed hashed-ElGamal encryption [27] in the random oracle model, of both key privacy and data security against chosen ciphertext attack. Again, this specific example supports our claim on the relation of two notions.

1 Introduction

Key privacy [4] is a notion regarding receiver anonymity, which captures an adversary's inability to learn any knowledge about the encryption key used for a given ciphertext. It has many applications, e.g., building an anonymous channel providing receiver anonymity. Also encryption schemes with key privacy are often used in generic constructions to build privacy-enhanced signature schemes.

Key privacy is considered as a totally different notion from data privacy, e.g., semantic security, and the authors of [4] mentioned that "*it is not hard to see that the goals of data privacy and key privacy are orthogonal*." They then explained this by showing some concrete encryption schemes which have data privacy but without key privacy, e.g., RSA-OAEP [8].

On the other hand, however, there exist some facts which seem to imply a strong relationship between key privacy and data privacy. Many natural semantically secure encryption schemes, e.g. ElGamal [12] and Cramer-Shoup [10], already have key privacy. Furthermore, Hayashi and Tanaka [18] pointed out, a well-known technique providing strong data privacy, i.e. [15], is also effective to upgrade the underlying scheme to have chosen ciphertext security, though their discussions are limited to *plaintext awareness* (PA) [5]. By "plaintext awareness", informally, it is required that an adversary should have already known corresponding plaintext before it queries a decryption oracle. PA has been a useful but very strong notion that helps to prove an encryption scheme has strong data privacy against adaptive chosen ciphertext attack (CCA) [20, 25].

In this paper, we reconsider the relationship between key privacy and data privacy from a different perspective, and confirm that these two notions are actually orthogonal. However, instead of merely giving counterexamples, we give clear evidence. We show, some cryptographic techniques for data privacy happen to be able to upgrade key privacy, but they don't provide key privacy. Towards this goal, as an intermediate step, we study key privacy in the plaintext checking attack model and show that orthogonality of the two notions still holds in this model. Then we prove that (1) key privacy and data privacy are orthogonal (as

pointed out by Bellare et al.), but (2) (natural) techniques (FOPKC [14], REACT [21]) for achieving data privacy against chosen-ciphertext attacks can also straightforwardly applied to obtain key privacy against the same attack (CCA). As another strong evidence for these claims, we give a security analysis of the signed hashed-ElGamal encryption [27] in the random oracle model, for both key privacy and data privacy. Existence of such schemes especially supports the second claim since this does not satisfy PA.

1.1 Our Contributions

Previous research [4] gives clear conclusions that the ElGamal encryption has key privacy against chosen plaintext attack and the Cramer-Shoup encryption has key privacy against chosen ciphertext attack, secure encryption[1], both under the decisional Diffie-Hellman (DDH) assumption. However, consider the DHIES [1] or ElGamal-REACT [21]. To prove data privacy of these schemes, one has to grant the simulator with access to a DDH oracle, furthermore, the access to the DDH oracle is essential for the proof [29]. Then it is already not obvious whether these useful schemes have key privacy in a group where DDH is easy, since no previous work has been done regarding key privacy under non-decisional versions of Diffie-Hellman assumptions.

On the other hand, consider some encryption schemes, such as the signed hashed-ElGamal encryption, in fact, this scheme is not plaintext aware, as we will prove later. Thus within known techniques, even if one proves the key privacy against chosen plaintext attack, to investigate key privacy against chosen ciphertext attack, one has to start from scratch. Then a natural question arises, *whether we can do it more intelligently when dealing with key privacy rather than these brute force treatments*? Previous studies didn't provide with us ready tools to study these schemes.

Our contribution is three-fold. We first give formal evidences that key privacy is orthogonal to data privacy. We then elaborate that well-known transforms to acquire chosen ciphertext security for data privacy, such as FOPKC [14] and REACT [22], are still effective to upgrade underlying schemes to have key privacy against chosen ciphertext attack. Finally, we give a specific security proof with exact security reductions on the data privacy and key privacy, namely indistinguishability against chosen ciphertext attack (IND-CCA) security and indistinguishability of keys against chosen ciphertext attack (IK-CCA) security of the signed hashed-ElGamal encryption. We further show that the signed hashed-ElGamal encryption is not plaintext aware.

As an independent interest, we formalize the notion of key privacy in the plaintext checking attack (PCA) model [21]. A plaintext checking attack is a chosen plaintext attack plus a plaintext checking oracle that decides whether a ciphertext encrypts a given plaintext.

2 Preliminary

In this section, we review some basic definitions and security notions of public key encryption and symmetric key encryption, signature schemes, some number theoretic assumptions as well as variants of the ElGamal encryption.

[1] In fact, these are the only two practical schemes whose key privacy are analyzed besides those based on RSA assumptions in [4] and those acquired via the second Fujisaki-Okamoto transform [14] in [18].

Notations. Define $x \leftarrow_R X$ as x being uniformly chosen from a finite set X. If A is an algorithm, $x \leftarrow A$ means that the output of A is x. When y is not a finite set nor an algorithm, $x \leftarrow y$ is an assignment operation. A function $f(k)$ is negligible, if for any constant c there exists $k_0 \in \mathbb{N}$, such that $f(k) < (1/k)^c$ for any $k > k_0$.

2.1 Public Key Encryption

A public key encryption scheme consists of three algorithms $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$. K is the randomized key generation algorithm, which takes a security parameter k, generates a pair of key (pk, sk), where pk is a public key and sk is a secret key. Denote $\mathcal{M}$ and C as the plaintext and ciphertext spaces, respectively. E is the possibly randomized encryption algorithm, which takes pk and a plaintext $m \in \mathcal{M}$ as input, together with internal random coin r, and outputs a ciphertext c as the output, denoted as $c \leftarrow \mathsf{E}(pk, m; r)$. D is the deterministic decryption algorithm, which takes sk and a ciphertext c as input, output the corresponding m, or "$\perp$" if decryption algorithm fails, denoted as $m \leftarrow \mathsf{D}(sk, c)$. We require that Π meets standard correctness requirement, namely, for all $(pk, sk) \leftarrow \mathsf{K}(1^k)$, and all $m \in \mathcal{M}$, $\mathsf{D}(sk, \mathsf{E}(pk, m; r)) = m$.

Indistinguishability. The widely accepted security notion for public key encryption is indistinguishability against adaptive chosen ciphertext attack (IND-CCA) [16, 11, 20, 25, 5].

Definition 1 (IND-CCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary. Let $\mathcal{DO}$ be a decryption oracle that replies the corresponding plaintext upon a query on ciphertext c. We define the advantage of $\mathcal{A}$ as*

$$Adv^{\text{ind-cca}}_{\Pi,\mathcal{A}}(k) = \Pr\left[b' = b \left| \begin{array}{l} (pk, sk) \leftarrow \mathsf{K}(1^k); (m_0, m_1, s) \leftarrow \mathcal{A}^{\mathcal{DO}}(pk); \\ b \leftarrow_R \{0,1\}; c^* \leftarrow \mathsf{E}(pk, m_b); b' \leftarrow \mathcal{A}^{\mathcal{DO}}(c^*, s) \end{array} \right. \right] - \frac{1}{2}$$

*where $\mathcal{A}$ is forbidden to query c^* at $\mathcal{DO}$. A scheme is (ϵ, t)-*IND-CCA *secure if any adversary with running time t, has advantage at most ϵ in winning the above game. Furthermore, we say a public key encryption scheme is* IND-CCA *secure if for any PPT adversary, ϵ is negligible.*

Key Privacy. We now review the security notion for key privacy, i.e., indistinguishability of keys against adaptive chosen ciphertext attack (IK-CCA) [4].

Definition 2 (IK-CCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary. Let $\mathcal{DO}$ be a decryption oracle that replies the corresponding plaintext upon a query on ciphertext c. We define the advantage of $\mathcal{A}$ as*

$$Adv^{\text{ik-cca}}_{\Pi,\mathcal{A}}(k) = \Pr\left[b' = b \left| \begin{array}{c} (pk_0, sk_0) \leftarrow \mathsf{K}(1^k); (pk_1, sk_1) \leftarrow \mathsf{K}(1^k); \\ (m^*, s) \leftarrow \mathcal{A}^{\mathcal{DO}}(pk_0, pk_1); b \leftarrow_R \{0,1\}; \\ c^* \leftarrow \mathsf{E}(pk_b, m^*); b' \leftarrow \mathcal{A}^{\mathcal{DO}}(c^*, s) \end{array} \right. \right] - \frac{1}{2}$$

*where $\mathcal{A}$ is forbidden to query c^* on $\mathcal{DO}$. A scheme is (ϵ, t)-*IK-CCA *secure if any adversary with running time t, has advantage at most ϵ in winning the above game. Furthermore, we say a public key encryption scheme is* IK-CCA *secure if for any PPT adversary, ϵ is negligible.*

For indistinguishability against chosen plaintext attack (IK-CPA), everything will be the same to the above experiment, except that $\mathcal{A}$ is not given access to $\mathcal{DO}$.

Plaintext Awareness. Informally, plaintext awareness [7, 5] states an intuition that if an adversary can produce a valid ciphertext, it should "have already known" the plaintext. Plaintext awareness (PA) is another important notion regarding encryption schemes secure against chosen ciphertext attack (CCA) and many encryption schemes (especially those with random oracles) are built on this idea. On the other hand, there are some attempts to formalize the notion in the standard model, e.g. [19, 6], but these definitions seem quite artificial. Moreover, PA is more like a means rather than a goal, since many useful encryption schemes are not plaintext aware [13].

To formulate the intuition, an adversary $\mathcal{B}$ is given a public key pk and access to a random oracle $\mathcal{G}$. Additionally, it is provided with a ciphertext generation oracle $\mathcal{E}^{\mathcal{G}}(pk)$ which with input pk, randomly chooses $m \in \mathcal{M}$, and outputs a ciphertext y. To be plaintext aware, $\mathcal{B}$ should necessarily infer m from y and oracle queries to $\mathcal{G}$.

Definition 3 (PA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme, let Γ be the set of all corresponding random functions, let $\mathcal{B}$ be an adversary, U be a plaintext extractor and $\mathcal{G}$ be a random oracle. $\mathcal{B}$ is additionally given access to an ciphertext generation oracle $\mathcal{E}^{\mathcal{G}}(pk)$. Denote the list* H *that contains all the random oracle queries from $\mathcal{B}$ to $\mathcal{G}$ and the list* C *that contains all the replies by $\mathcal{E}^{\mathcal{G}}(pk)$. We require the queries of $\mathcal{B}$ to $\mathcal{E}^{\mathcal{G}}(pk)$ is not included in* C *and interaction between $\mathcal{E}^{\mathcal{G}}(pk)$ and $\mathcal{G}$ is not included in* H*. For any $k \in \mathbb{N}$, define*

$$Suc^{\mathsf{pa}}_{\Pi,\mathcal{B},U}(k) = \Pr\left[U(\mathtt{H}, \mathtt{C}, y, pk) = \mathsf{D}^{\mathcal{G}}(sk, y) \,\middle|\, \begin{array}{c} \mathcal{G} \leftarrow_R \Gamma; (pk, sk) \leftarrow \mathsf{K}(1^k); \\ (\mathtt{H}, \mathtt{C}, y) \leftarrow \mathcal{B}^{\mathcal{G}, \mathcal{E}^{\mathcal{G}}(pk)} \end{array} \right]$$

The above definition is only defined in the single public key model, it is easily generalized to multiparty setting by modifying the key generation and oracle access for $\mathcal{B}$ in the above experiment, since the code of the extractor U can be used for multiple instances of the same encryption algorithm. A recent work [18] did this in more details, particularly with the FOCRYPTO transform [15].

Plaintext Checking Attack (PCA**).** Okamoto and Pointcheval [21] defined this attack model called plaintext checking attack (PCA), potentially motivated by an interesting property of some elliptic curves. In such an attack model, in addition to the public key, the adversary is given access to a plaintext checking oracle $\mathcal{PCO}$, which upon inputs of plaintext/ciphertext pair (m, c) and a public key pk, returns "yes" if $\mathsf{D}(sk, c) = m$ and "no" otherwise. This attack model is important and the security proof of many useful encryption scheme is based on this formulation [21, 1]. We note that in a single public key setting, usually, the public key pk need not to be part of the inputs to $\mathcal{PCO}$.

We review here the security notion of onewaysness of data privacy against plaintext checking attack. Informally, onewayness means that it is computationally infeasible to "undo" the encryption without knowing the secret key.

Definition 4 (OW-PCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary, and $\mathcal{PCO}$ be a plaintext checking oracle, We define the success probability of $\mathcal{A}$ as*

$$Suc^{\mathsf{ow\text{-}pca}}_{\Pi,\mathcal{A}}(k) = \Pr\left[m' = m^* \,\middle|\, \begin{array}{c} (pk, sk) \leftarrow \mathsf{K}(1^k); m^* \leftarrow_R \mathcal{M}; \\ c^* \leftarrow \mathsf{E}(pk, m^*); m' \leftarrow \mathcal{A}^{\mathcal{PCO}}(c^*, pk) \end{array} \right]$$

A scheme is (ϵ, t)-OW-PCA *secure if any adversary with running time t, succeeds with probability at most* ϵ *in winning the above game. Furthermore, we say a public key encryption scheme is* OW-PCA *secure if for any PPT adversary,* ϵ *is negligible.*

Remark 1. An important previous belief on PCA is that indistinguishability is *meaningless* in this model, whereas onewayness can be meaningful. To see this, since the adversary can query the plaintext checking oracle with its target ciphertext and one of its chosen message, thus easily breaks the indistinguishability of the scheme. However, we believe if ruling out this trivial attack, one can still define indistinguishability of data privacy in plaintext in the PCA model. The formulation of the security is quite straightforward and we omit here.

2.2 Digital Signature

A signature scheme Σ consists of three algorithms $\Sigma = (\mathsf{G}, \mathsf{S}, \mathsf{V})$. The randomized key generation algorithm G takes a security parameter k, and generates signing key $sigk$ and verification key vk. The possibly randomized signing algorithm S takes as inputs $sigk$ and $m \in \{0,1\}^*$, where m is a message, and outputs a signature σ. The deterministic verification algorithm V takes as inputs vk, m and σ, and outputs a symbol $\beta \in \{\mathtt{accept}, \mathtt{reject}\}$, denoted as $\beta \leftarrow \mathsf{V}(vk, m, \sigma)$. We require that for all $(sigk, vk)(= \mathsf{G}(1^k))$, all $m \in \{0,1\}^*$, $\mathtt{accept} = \mathsf{V}(vk, m, \mathsf{S}(sigk, m))$.

Unforgeability. Here, we mainly consider strong unforgeability, i.e., sUF-CMA [3], rather than the weaker version (UF-CMA) [17]. Let $\Sigma = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ be a signature scheme. Let $\mathcal{A}$ and k be an adversary and a security parameter, respectively.

Definition 5 (sUF-CMA). *Denote* $\{(\sigma_i, m_i)\}_{q_s}$ *as the set contains all* q_s *pairs of queries and replies between* $\mathcal{A}$ *and* $\mathcal{SO}$*, where* $\mathcal{SO}$ *is a signing oracle which for a given message m, returns a corresponding signature* σ*. The success probability of* $\mathcal{A}$ *is defined as*

$$Succ^{\text{suf-cma}}_{\mathcal{A},\Sigma}(k) = \Pr\left[\begin{array}{l|l} \mathsf{V}(vk, m^*, \sigma^*) = \mathtt{accept} & (vk, sigk) \leftarrow \mathsf{Gen}(1^k); \\ \wedge (\sigma^*, m^*) \notin \{(\sigma_i, m_i)\}_{q_s} & (\sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{SO}}(vk) \end{array}\right],$$

We say Σ *is* (t, ϵ)-sUF-CMA *secure if for any* $\mathcal{A}$ *in time bound t,* $\mathcal{A}$*'s success probability is at most* ϵ*. Especially, we say that* Σ *is* sUF-CMA *secure if* ϵ *is negligible.*

2.3 Number Theoretic Assumptions

We briefly review the discrete logarithm (DL), the computational Diffie-Hellman (CDH), the decisional Diffie-Hellman (DDH), and the gap Diffie-Hellman (GDH) assumptions.

Definition 6 (Assumptions). *Let* $\mathbb{G}$ *be a cyclic group of order q, where q is a large prime, let g be a generator of* $\mathbb{G}$ *and* $a, b, c \leftarrow_R \mathbb{Z}_q^*$*. The* (t, ϵ)-DL *assumption holds in* $\mathbb{G}$ *if for given* $(g, y) \in \mathbb{G}^2$*, no t-time algorithm finds* $x \in \mathbb{Z}_q$ *such that* $y = g^x$ *with probability at least* ϵ*. The* (t, ϵ)-CDH *assumption holds in* $\mathbb{G}$ *if for given* $(g, g^a, g^b) \in \mathbb{G}^3$*, no t-time algorithm finds* g^{ab} *with probability at least* ϵ*. The* (t, ϵ)-DDH *assumption holds in* $\mathbb{G}$ *if no t-time algorithm has with at least* ϵ *advantage, where for an algorithm* $\mathcal{A}$*,* $\mathcal{A}$*'s advantage* $\epsilon_{\mathcal{A}}$ *is defined as* $\epsilon_{\mathcal{A}} = \frac{1}{2}|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1]|$*. The* (t, ϵ)-GDH *assumption [22] holds in* $\mathbb{G}$ *if for given* $(g, g^a, g^b) \in \mathbb{G}^3$*, no t-time algorithm with polynomial bounded number of accesses to* $\mathcal{O}$ *finds* g^{ab} *with probability at least* ϵ*, where* $\mathcal{O}$ *is a decision oracle which for a given* $(g, g^a, g^b, T) \in \mathbb{G}^4$*, returns* 1 *if* $T = g^{ab}$ *or* 0 *otherwise.*

2.4 Some Variants of ElGamal Encryption

The signed ElGamal encryption [30, 27] was proposed as a highly secure upgrade to the plain ElGamal encryption [12], which combines the ElGamal encryption with Schnorr signature [26]. Because of its good homomorphic property, the signed ElGamal encryption is widely used to construct threshold encryptions, with various applications like mix-net and auctions schemes. On the other hand, it has been a long-standing open problem to prove the security of the signed ElGamal encryption scheme based on reasonable assumptions. Jacobsson and Schnorr provided a security analysis of the signed ElGamal assuming both random oracles [8] and generic groups [28]. Another natural generalized version of the signed ElGamal was proposed in [27], namely the signed hashed-ElGamal, however, even the data privacy of this variant has not been well studied except brief discussions mentioned in [27, 2]. We recall these schemes in Table 1. Let $\mathbb{G}$ be a group of prime order q, and let $g \in \mathbb{G}$ be a generator. Denote the public key as pk and the secret key as sk.

	Plain ElGamal	**Hashed-ElGamal**	**Signed Hashed-ElGamal**
K	$x \leftarrow_R \mathbb{Z}_q$, $y \leftarrow g^x$. Public key: $pk \leftarrow g, y$, secret key: $sk \leftarrow x$. The message space is $\mathbb{G}$.	Let $H : \mathbb{G} \to \{0,1\}^\ell$ be a hash function. Choose $x \leftarrow_R \mathbb{Z}_q$, $y \leftarrow g^x$. Public key: $pk \leftarrow (g, y, H)$, Secret key: $sk \leftarrow x$. The message space is $\{0,1\}^\ell$.	$x \leftarrow_R \mathbb{Z}_q$, $y \leftarrow g^x$. Let $H_1 : \mathbb{G} \leftarrow \{0,1\}^\ell$ and $H_2 : \mathbb{G}^2 \times \{0,1\}^\ell \to \mathbb{Z}_q$ be two hash functions. Public key: $pk \leftarrow (g, y, H_1, H_2)$, Secret key: $sk \leftarrow x$. The message space is $\{0,1\}^\ell$.
E	For $m \in \mathbb{G}$, $r \leftarrow_R \mathbb{Z}_q$, $c_1 = g^r$, $c_2 = m \cdot y^r$. Ciphertext: $c = (c_1, c_2)$.	For $m \in \{0,1\}^\ell$, $r \leftarrow_R \mathbb{Z}_q$, $c_1 = g^r$, $c_2 = m \oplus H(y^r)$. Ciphertext: $c = (c_1, c_2)$.	For $m \in \{0,1\}^\ell$, $r, s \leftarrow_R \mathbb{Z}_q$, $c_1 = g^r$, $c_2 = m \oplus H_1(y^r)$. $d = H_2(g^s, c_1, c_2)$ and $z = s + rd$. Ciphertext: $c = (c_1, c_2, d, z)$.
D	For $c' = (c'_1, c'_2)$, output $m' = c'_2 / {c'_1}^x$ as the plaintext.	For $c' = (c'_1, c'_2)$, output $m' = c_2 \oplus H({c'_1}^x)$ as the plaintext.	For $c' = (c'_1, c'_2, d', z')$, test if $H_2(g^z c_1^{-d'}, c'_1, c'_2) = d'$. If '`yes`', output $m' = c'_2 \oplus H_1({c'_1}^x)$ as the plaintext; otherwise output '$\perp$'.

Table 1. Some variants of the ElGamal encryption.

3 Key Privacy in the PCA Model

We sketch the intuition of the formalization before define key privacy in the PCA model, which is the first result of this work. We also give discussions on its reasonability, because this attack model appears to be stronger than CPA and weaker than CCA in the sense of key privacy. In the end of this session, we further give further evidence that key privacy is orthogonal to data privacy.

Recall that the plaintext checking attack was only defined in the single public key setting, we first has to extend it to multiple public key setting and give the corresponding definition. The key observation is that a query to the plaintext checking oracle should be paired with a public key. In fact, this also fits the single public key model. Since there is only one public key, the public key is usually omitted in each plaintext checking query.

On the other hand, to make the indistinguishability meaningful in PCA model, for the target ciphertext queried with one of the chosen public keys, the plaintext checking oracle returns a special symbol "$\perp$"

(cf. replayable chosen ciphertext attack RCCA [9]). With such limitation to an oracle access, the plaintext checking oracle may no longer trivially leak information regarding the public key.

Definition 7 (IK-PCA). *Let* $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be a public key encryption scheme. Let* $\mathcal{A}$ *be an adversary, and* $\mathcal{PCO}$ *be a plaintext checking oracle with limitations as discussed above. We define the advantage of* $\mathcal{A}$ *as*

$$Adv^{\text{ik-pca}}_{\Pi,\mathcal{A}}(k) = \Pr\left[b' = b \left| \begin{array}{c} (pk_0, sk_0) \leftarrow \mathsf{K}(1^k); (pk_1, sk_1) \leftarrow \mathsf{K}(1^k); \\ (m^*, s) \leftarrow \mathcal{A}^{\mathcal{PCO}}(pk_0, pk_1); b \leftarrow_R \{0,1\}; \\ c^* \leftarrow \mathsf{E}(pk_b, m^*); b' \leftarrow \mathcal{A}^{\mathcal{PCO}}(s) \end{array} \right. \right] - \frac{1}{2}$$

A scheme is (ϵ, t)-IK-PCA *secure if any adversary with running time* t*, has advantage at most* ϵ *in winning the above game. Furthermore, we say a public key encryption scheme is* IK-PCA *secure if for any PPT adversary,* ϵ *is negligible.*

For the security goal of key privacy, an adversary will not necessarily gain any direct "help" via its access to a plaintext checking oracle, thus plaintext checking attack (PCA) may be meaningful in this setting. However, the plaintext to query the plaintext checking oracle may depend on the public key, thus it may be a stronger attack even in the sense of key privacy. Based on above discussions we prove the following theorem:

Theorem 1. *There is a public key encryption scheme that is* IK-CPA *secure but not* IK-PCA *secure.*

Proof. Consider the plain ElGamal encryption. It is known from [4] that under the DDH assumption, it is IK-CPA. It is sufficient to show it is not IK-PCA.

To see this, we construct an IK-PCA adversary that defeats the ElGamal encryption. After setups, the IK-PCA adversary chooses $m^* \leftarrow_R \mathbb{G}$, and submits to the challenger. The challenger chooses $b \leftarrow_R \{0,1\}$, and returns the challenge ciphertext $c^* = (c_1, c_2) = (g^r, m^* \cdot y_b^r)$, where $r \leftarrow_R \mathbb{Z}_q$. Now the adversary just needs to query $(2 \cdot m^*, c')$ with $c' = (c_1, 2 \cdot c_2)$ to the plaintext checking oracle on public key y_1. If $b = 1$, the pair will be always valid and 0 otherwise, so the adversary always breaks the indistinguishability of keys of the scheme. □

One may argue that in the above attack we are tricky because if we assume the existence of plaintext checking oracle, the plain ElGamal encryption no longer has data privacy. However, we are considering key privacy, and a scheme with key privacy can in fact has no data privacy at all. To show that IK-PCA is a proper notion which lies between IK-CPA and IK-CCA, we present the following theorem:

Theorem 2. *There is a public key encryption scheme that is* IK-PCA *secure, but not* IK-CCA *secure.*

Proof. Consider the hashed-ElGamal encryption. We claim it is not IK-CCA secure. To see this, we construct an IK-CCA adversary as follows: After setup, the adversary is given the public key $pk_0 = (g, y_0, H_1)$ and $pk_1 = (g, y_1, H_2)$. [2] The adversary then chooses $m^* \leftarrow_R \{0,1\}^\ell$, and submits it to the challenger. The challenger chooses $b \leftarrow_R \{0,1\}$ and returns $c^* = (c_1, c_2) = (g^r, m^* \oplus H_b(y_b^r))$. Now the adversary just need to query (c', pk_1) to the decryption oracle, where $c' = (c_1, c_2 \oplus \delta)$ and $\delta \leftarrow_R \{0,1\}^\ell$. If the decryption oracle returns $m^* \oplus \delta$, then the adversary output "1"; otherwise "0".

[2] H_1 and H_2 can be of different implementations, as long as their input and output domain are the same.

It is easy to verify that c' is a valid ciphertext according to pk_b and $m^* \oplus \delta$. Therefore, we conclude the adversary guess the value of b with probability "1".

Next, we shall prove the hashed-ElGamal is IK-PCA under gap Diffie-Hellman (GDH) assumption. Here for simplicity, we consider the case that H_1 and H_2 are the same random oracle. The basic idea is that since the information of the public key can only be accessed via hash queries, thus any adversary that breaks the IK-PCA security of the scheme must have queried to the random oracle regarding the chosen public key. On the other hand, given a GDH problem instance, utilizing the random self-reducibility of the GDH problem, the simulator can create two independent public keys that distributes identically as the real attack.

We write above idea in more details. A GDH adversary $\mathcal{A}$ can be constructed as follows: for a given GDH instance $(A = g, B = g^a, C = g^b)$, where a, b are unknown, $\mathcal{A}$ then interacts with the IK-PCA adversary $\mathcal{B}$ as follows:

$\mathcal{A}$ gives $pk_0 = (A, B^{r_0}, H)$ and $pk_1 = (A, B^{r_1}, H)$ to $\mathcal{B}$ as two public keys, where $r_0, r_1 \leftarrow_R \mathbb{Z}_q$ and H is a random oracle controlled by $\mathcal{A}$. Since r_0 and r_1 are uniformly distributed in $\mathbb{Z}_q$, then the simulated public keys pk_0 and pk_1 are indistinguishable from the real game where $pk_0 = (A, y_0, H)$ and $pk_1 = (A, y_1, H)$. Moreover, $\mathcal{A}$ simulates the following oracles:

Random Oracle queries: $\mathcal{A}$ maintains a H-list with two entries (h_i, H_i). Where a query on h_i comes, $\mathcal{A}$ queries its own decision oracle whether (A, B, y_δ, h_i) $(\delta = 0, 1)$ is a Diffie-Hellman tuple. If "yes", $\mathcal{A}$ stops simulation and output h_i^{1/r_δ} as its answer. Otherwise, $\mathcal{A}$ searches H-list, if there exists such an entry, $\mathcal{A}$ returns the corresponding H_i. Otherwise, $\mathcal{A}$ then chooses $H_i \leftarrow_R \{0,1\}^\ell$, adds (h_i, H_i) to the list and returns H_i to $\mathcal{B}$.

Plaintext Checking queries: On a query (m, c) on public key pk_δ $(\delta \in \{0, 1\})$, where $c = (c_1, c_2)$, $\mathcal{A}$ searches in the H-list whether there exists an entry (h_i, H_i) such that (g, c_1, y_δ, h_i) is a Diffie-Hellman tuple (this can be done via querying its own decision oracle) and $c_2 = m \oplus H_i$. If both of above hold, return "yes". Otherwise "no".

Challenge query: When $\mathcal{B}$ signals a chosen message m^* to $\mathcal{A}$, on which it would like to be challenged, $\mathcal{A}$ chooses a bit $\beta \leftarrow_R \{0, 1\}$ and $R \leftarrow_R \{0, 1\}^\ell$, set $c^* = (B, m \oplus R)$ and returns the challenge c^* to $\mathcal{B}$.

Since the information of β can only be gained by querying g^{abr_β} to the random oracle. From above descriptions one easily verifies that $\mathcal{A}$ answers the random oracle, the plaintext checking and the challenge queries perfectly. Then $\mathcal{B}$'s success probability is exactly the same as the real attack. Combining all above discussions, $\mathcal{A}$ succeeds with probability at least $\epsilon_{\mathcal{B}}$. □

Furthermore, we would like to present an important claim of our paper, namely, data privacy and key privacy are totally *unrelated*.

Theorem 3. *There is a public key encryption scheme that is* IK-ATK *secure but not* IND-ATK *secure; there is a public key encryption scheme that is* IND-ATK *secure but not* IK-ATK *secure, where* ATK $\in$ {CPA, PCA, CCA}.

Proof. We only give sketch of the proof, since it is not difficult to write a long but more precise (though tedious) one. To see the first half of the statement, given an encryption scheme with both IND-ATK and IK-ATK security (in short IND-IK-ATK), e.g., the Cramer-Shoup [10], we can modify it to an IK-ATK secure encryption yet without data privacy. All will remain the same, except that for the encryption algorithm, we

append the plaintext m to the ciphertext c. Now it is easy to see, the modified encryption scheme is not even IND-CPA. On the other hand, since the new encryption algorithm does not produce ciphertexts that give more information on the public key than the original encryption scheme, it is still IK-ATK secure.

To see second half of the statement, we can do similar as above: for an IND-IK-ATK secure encryption scheme, just append the public key to the ciphertext. The decryption algorithm will check whether this public key is the correct public key corresponding to the secret key, otherwise the encryption scheme will be trivially malleable. It is easily verified then this modified encryption scheme has no key privacy but is still IND-ATK secure. □

4 Generic Transforms for Key Privacy

We here also discuss some generic transforms to enhance the chosen ciphertext security of an encryption scheme regarding key privacy. Informally and briefly speaking, for an IK-CPA secure encryption, to enhance it to an IK-CCA secure one, one just needs to additionally simulate the decryption oracle. This is quite difficult in the standard model, since the simulator has to do decryptions without the secret keys. The classic Naor-Yung paradigm is sure to apply here, as long as the new components introduced to the encryption scheme do not leak any information of the public key.

In the random oracle model, however, this can be done easily and efficiently. A recent work formulates the plaintext awareness in the two public key model and use the plaintext extractor provided via plaintext awareness to simulate the decryption oracle. Based on this idea, some famous generic transforms to enhance data privacy, like (two) Fujisaki-Okamoto (FO) [14, 15] and REACT [21] transforms are all capable to construct a plaintext extractor, which can be used to for simulation of the decryption oracle.

Recall the construction of such plaintext extractors in the original papers. It is not difficult to rewrite the proofs for the two public key setting just like what has been done in [18] via the FOCrypto transform [15]. In fact, the theorem attests exactly our claim that key privacy is unrelated to chosen ciphertext security.

Theorem 4. *The FOPKC and REACT transform are both capable to enhance* IK-CPA *security to* IK-CCA *security.*

Theorem 4 tells that FOPKC and REACT are both generic for key privacy enhancement, however, it is worth noting that "unnatural" implementations may still leak information of keys. In fact, our proofs require the symmetric key encryption should have key privacy. However, unlike public key encryption, the semantic security of implies its key privacy. For completeness, we review the model and the notion of key privacy for symmetric key encryption in Appendix A. The proof is given in Appendix B.

5 Analysis on Signed Hashed-ElGamal Encryption

In this section we give analyses on data privacy and key privacy of the signed hashed-ElGamal encryption. Since the data privacy (IND-CCA) has been studied [27, 2] before, however, without details, we give a brief but much specific proof. We focus on non-plaintext awareness and the key privacy properties of the scheme, namely we prove the following theorem.

Theorem 5. *The signed hashed-ElGamal is* IND-CCA *and* IK-CCA *secure and not PA.*

We split the proof into three parts, each shown by a lemma. The theorem then follows Lemma 1, 2 and 3. For data privacy, we first review a previous result on the security of the Schnorr signature in Proposition 1. Since the GDH assumption implies the DL assumption, combining Proposition 1 and Lemma 1, we conclude signed hashed-ElGamal is secure under the GDH assumption.

Proposition 1. *([24]) The Schnorr Signature is* (ϵ_s, t_1)-sUF-CMA *secure, assuming the discrete logarithm problem in G is* $(120686q_h t_1, 10(q_s+1)(q_s+q_h)/q)$*-hard, where* q_h *is number of random oracle queries,* q_s *is the number of signing queries and q is the order of subgroup G.*

Lemma 1. *The signed hashed-ElGamal is* $(\epsilon+\epsilon_s, (t_1+t_2+O(k)))$-IND-CCA *secure, assuming the* (ϵ, t_2)*-GDH assumptions holds and the Schnorr Signature is* (ϵ_s, t_1)-sUF-CMA *secure.*

Proof Ideas. To show a more general result, we present the following encryption scheme. For the hashed-ElGamal encryption, we append to it an $(\epsilon_s, 1, t_s)$-sUF-CMA secure signature scheme. The Schnorr signature satisfies such security dentition according to [23]. We describe the algorithms of such encryption scheme. The signature scheme used in signed hashed-ElGamal is a special case by using Schnorr Signature [26] and shown in Figure 1.

The key generation just runs the key generation algorithm of the hashed-ElGamal encryption, the public key and secret key are (pk, sk) as before. That is, $pk = (g, y, H)$ and $sk = (x, H)$, such that $y = g^x$. We note that additional hash functions may be used, then also include the hash functions into the public key.

The encryption algorithm first calls the key generation algorithm of the signature scheme, generates a (one-time) verification/signing key pair (vk, sk). Then it does just as the hash ElGamal encryption, except that it also bundles the public key of the signature scheme into the a hash function, namely, it produces $c = (c_1, c_2)$, where $c_1 = g^r$ is the same as before, however, $c_2 = m \oplus H(vk, y^r)$ (rather than $m \oplus H(y^r)$). Then a signature σ is produced on c. The ciphertext is (c, σ, vk). Note that if we use Schnorr signature, then we get a very efficient scheme: the verification key is unnecessary to be explicitly included as part of input for H and the ciphertext, since it is exactly c_1. In following proofs, we assume vk is explicitly indicated.

The decryption algorithm first verifies the signature σ is a correct signature on c according to vk. If this fails, it halts and output "$\perp$". Otherwise, it proceeds to undo the hashed-ElGamal encryption using the decryption algorithm of hashed-ElGamal encryption as shown in Table 1.

Proof. We next show the above encryption scheme is $(\epsilon+\epsilon_s, t_1+t_2+O(k))$-IND-CCA secure, if the (ϵ, q, t_1)-GDH assumption holds and the signature scheme used is $(\epsilon_s, 1, t_2)$-sUF-CMA secure. Towards the claim, suppose a GDH solver $\mathcal{A}$ interacts with an IND-CCA adversary $\mathcal{B}$ as follows.

Key Generation: $\mathcal{A}$ receives its GDH problem instance $(A = g, B = g^a, C = g^b)$, and sets $y = C^{r^*}$ and $pk = (g, y, H)$, where $r^* \leftarrow_R \{0,1\}^\ell$ and H is a random oracle controlled by $\mathcal{A}$. $\mathcal{A}$ gives pk to $\mathcal{B}$.

Random Oracle queries: $\mathcal{A}$ maintains a H-list with four entries (v_i, c_{1i}, h_i, H_i). Where a query comes for (v_i, h_i), $\mathcal{A}$ searches H-list, if there exists such an entry $(v_i, \cdot, h_i, H_i)$, where "$\cdot$" means "not of interest", $\mathcal{A}$ returns the corresponding H_i. Otherwise, $\mathcal{B}$ chooses $H_i \leftarrow_R \{0,1\}^\ell$ and adds (v_i, c_{1i}, h_i, H_i) to the list. Additionally, if there exists an entry with $(v_i, c_{1i}, *, H_i)$ in the list, where $*$ mains empty entry, and $(A, c_{1i}, C^{r^*}, h_i)$ is a Diffie-Hellman tuple, $\mathcal{A}$ completes the entry with (v_i, c_{1i}, h_i, H_i). In both cases, $\mathcal{A}$ returns H_i to $\mathcal{B}$.

Decryption queries: On a decryption query on (c, σ, vk), $\mathcal{A}$ first verifies σ is a valid signature on c under vk. If this fails, $\mathcal{A}$ simply returns "$\perp$". Otherwise, $\mathcal{A}$ searches for the H-list. $\mathcal{A}$ splits $c = (c_1, c_2)$, if there exists an entry with (vk, c_1, h_i, H_i), such that (g, c_1, y, h_i) is a DH-tuple, $\mathcal{A}$ then returns $H_i \oplus c_2$ as the plaintext. Otherwise, if (vk, c_1, h_i) is not queried before, $\mathcal{A}$ chooses $H_i \leftarrow_R \{0,1\}^\ell$ adds $(vk, c_{1i}, *, H_i)$ to H-list and returns $H_i \oplus c_2$ as the plaintext.

Challenge query: When $\mathcal{B}$ outputs a pair of chosen message (m_0, m_1), $\mathcal{A}$ chooses $\beta \leftarrow_R \{0,1\}$ and $R^* \leftarrow_R \{0,1\}^\ell$ and sets $c^* = (B, m_\beta \oplus R)$. $\mathcal{A}$ also queries its own signing oracle on c^* under vk^*. After the signing oracle O_s returns a signature σ^* on c^*, $\mathcal{A}$ forwards (c^*, σ^*, vk^*) to $\mathcal{B}$ as the challenge ciphertext. Additionally, $\mathcal{A}$ adds $(vk^*, B, *, R)$ to H-list.

After $\mathcal{B}$ submits a guess on β, $\mathcal{A}$ searches H-list for $(\cdot, B, h_i, \cdot)$ (recall "$\cdot$" means "not of interest"), such that $(A, B, C, h_i^{1/r^*})$ is a Diffie-Hellman tuple. If there does not exist such a tuple, $\mathcal{A}$ chooses $D \leftarrow_R \mathbb{G}$ and halts.

From above descriptions, we can verify the simulations of random oracle, decryption oracle and challenge oracle are perfect. Since the information of β is perfectly hiding without querying g^{abr^*} to the random oracle. Then if $\mathcal{B}$ gainst any advantage on guessing β correctly, $\mathcal{B}$ must have queried $h_i = g^{abr^*}$ already, except that $\mathcal{B}$ queries a decryption query on (c^*, σ') with $\sigma' \neq \sigma^*$, where σ' is a valid signature on c^* regarding vk^*, however, in this case, $\mathcal{B}$ has already broken the signature scheme, thus this event happens at most ϵ_s. Note that in above proof we only require $\mathcal{A}$ queries the signing oracle exactly once. This proves our claim on success probability of $\mathcal{A}$. The running time of $\mathcal{A}$ is verifiable from the above descriptions. □

Lemma 2. *The signed hashed-ElGamal is* $(\epsilon + \epsilon_s, (t_1 + t_2 + O(k)))$-IK-CCA *secure, assuming the* (ϵ, q, t_2)-*GDH assumptions holds and the Schnorr signature is* (ϵ_s, t_1)-sUF-CMA *secure.*

Proof Ideas. The idea here works as a combination of those of IK-PCA security of hashed-ElGamal and IND-CCA of signed hashed-ElGamal shown before. Basically $\mathcal{A}$ randomizes its GDH challenge into two public keys and interacts with a public key distinguisher $\mathcal{B}$. As the signature scheme does not leak information on the public key, the only way to distinguish which public key was used to compute the challenge is by directly querying some necessary information at the random oracle. Thus $\mathcal{A}$ can successfully extracts the answer for its own GDH problem.

Proof. $\mathcal{A}$ interacts with $\mathcal{B}$ in the following manner:

Key Generation: $\mathcal{A}$ receives its GDH problem instance $(A = g, B = g^a, C = g^b)$. $\mathcal{A}$ sets $pk_0 = (g, C^{r_0^*}, H)$ and $pk_1 = (g, C^{r_1^*}, H)$, where $r_0^*, r_1^* \leftarrow_R \{0,1\}^\ell$ and H is a random oracle controlled by $\mathcal{A}$ and gives pk_0 and pk_1 to $\mathcal{B}$.

Random Oracle queries: $\mathcal{A}$ maintains a H-list with four entries (v_i, c_{1i}, h_i, H_i). Where a query comes for (v_i, h_i), $\mathcal{A}$ searches H-list, if there exists such an entry with $(v_i, \cdot, h_i, H_i)$, $\mathcal{A}$ returns the corresponding H_i. Again "$\cdot$" means "not of interest". Otherwise, $\mathcal{B}$ chooses $H_i \leftarrow_R \{0,1\}^\ell$ and adds $(v_i, *, h_i, H_i)$ to the list. Here $*$ mean empty. Additionally, if there exists an entry with $(v_i, c_{1i}, *, H_i)$ in the list, where $*$ mains empty entry, and $(A, c_{1i}, C^{r_0^*}, h_i)$ or $(A, c_{1i}, C^{r_1^*}, h_i)$ is a Diffie-Hellman tuple, $\mathcal{A}$ completes the entry with (v_i, c_{1i}, h_i, H_i). In both cases, $\mathcal{A}$ returns H_i to $\mathcal{B}$.

Decryption queries: On a decryption query on (c, σ, vk) regarding pk_δ $(\delta \in \{0,1\})$, $\mathcal{A}$ first verifies σ is a valid signature on c under vk. If this fails, $\mathcal{A}$ simply returns "$\perp$". Otherwise, $\mathcal{A}$ splits $c_i = (c_{1i}, c_{2i})$,

searches in the H-list. If there exists an entry with (vk, c_{1i}, h_i, H_i), such that $(A, c_{1i}, C^{r^*_\delta}, h_i)$ is a DH-tuple, $\mathcal{A}$ then returns $H_i \oplus c_2$ as the plaintext. Otherwise, if (vk, c_{1i}, h_i) is not queried before, $\mathcal{A}$ chooses $H_i \leftarrow_R \{0,1\}^\ell$, adds $(vk, c_{1i}, *, H_i)$ to H-list and returns $H_i \oplus c_2$ as the plaintext.

Challenge query: When $\mathcal{B}$ outputs a chosen message m^*, $\mathcal{A}$ chooses $\beta \leftarrow_R \{0,1\}$ and $R^* \leftarrow_R \{0,1\}^\ell$ and sets $c^* = (B, m \oplus R)$. $\mathcal{A}$ also queries its own signing oracle on c^* under vk^*. After the signing oracle O_s returns a signature σ^* on c^*, $\mathcal{A}$ forwards (c^*, σ^*, vk^*) to $\mathcal{B}$ as the challenge ciphertext.

After $\mathcal{B}$ submits a guess on β, $\mathcal{A}$ searches H-list for $(\cdot, B, h_i, \cdot)$, such that $(A, B, C, h_i^{1/r^*_\beta})$ is a Diffie-Hellman tuple. If there does not exist such a tuple, $\mathcal{A}$ returns $D \leftarrow_R \mathbb{G}$ and halts.

With a similar discussion as before, we remark that the simulations of random oracle, decryption oracle and challenge oracle are again perfect. However, $\mathcal{A}$ may not utilize the hash queries submitted by $\mathcal{B}$, if (c^*, σ', vk^*) is submitted for decryption query and σ' is a valid signature on c^* under verification key vk^*. However, this event happens with probability at most ϵ_s, because the signature scheme is (ϵ_s, t_2)-sUF-CMA secure. (In above proof we only require $\mathcal{A}$ queries the signing oracle exactly once.) Thus $\mathcal{A}$ succeeds solving the GDH problem with probability at least $\epsilon - \epsilon_s$. The running time of $\mathcal{A}$ is $(t_1 + t_2 + O(k))$, which can be verified from the description of $\mathcal{A}$. □

Lemma 3. *The signed hashed-ElGamal is not PA.*

Proof Sketch. We give the idea of the proof here. An adversary $\mathcal{A}$ can produce a valid ciphertext $(c, \sigma) = (c_1, c_2, \sigma)$ where $c_1 = g^r$ and $c_2 \leftarrow_R \{0,1\}^\ell$, σ is a valid signature on c. Especially, $\mathcal{A}$ construct the ciphertext without querying $(vk, c_1, *)$ to the random oracle H. To extract the plaintext m, even if the plaintext extractor has all the random oracle queries of $\mathcal{A}$, since $(vk, c_1, *)$ hasn't been queried before, if the plaintext extractor can be constructed, i.e., $\mathsf{D}^H((sk, c_1, c_2, \sigma)$ can be constructed on the transcript given, but the existence of this knowledge extractor will contradicts the unpredictability of H.

Proof. For the signed hashed-ElGamal encryption, if the plaintext extractor can be constructed, that is, we can set $H(\cdot)$ to be $m \oplus c_2$. As mentioned above, H is a random oracle whose output is uniformly random, and c_2 is fixed already, thus m can only be extracted with probability at most $2^{-\ell}$, which is negligible. contradicting the definition of plaintext extractor. □

References

1. M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An Encryption Scheme Based on the Diffie-Hellman Problem. In *CT-RSA'01*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, 2001.
2. M. Abe. Combining Encryption and Proof of Knowledge in the Random Oracle Model. *The Computer Journal*, 47(1):58–70, 2004.
3. J.H. An, Y. Dodis, and Tal Rabin. On the Security of Joint Signature and Encryption. In *Eurocrypt'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.
4. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer-Verlag, 2001.
5. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public key encryption schemes. In *Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998.
6. M. Bellare and A. Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In *Asiacrypt'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer-Verlag, 2004.

7. M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *CCS'93*, pages 62–73. ACM Press, 1993.
8. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to encrypt with RSA. In *Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.
9. R. Canetti, H. Krawczyk, and J.B. Nielsen. Relaxing Chosen-Ciphertext Security. In *Crypto'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer-Verlag, 2003.
10. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.
11. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *STOC'91*, pages 542–552. ACM, 1991.
12. T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEE Transactions on Information Theory*, 31(4):469–472, 1985.
13. E. Fujisaki. Plaintext Simulatability. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E-89A(1):55–65, 2006.
14. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC'99*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer-Verlag, 1999.
15. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–544. Springer-Verlag, 1999.
16. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
17. S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
18. R. Hayashi and K. Tanaka. PA in the Two-Key Setting and a Generic Conversion for Encryption with Anonymity. In *ACISP'06*, volume 4058 of *Lecture Notes in Computer Science*, pages 271–282. Springer-Verlag, 2006.
19. J. Herzog, M. Liskov, and S. Micali. Plaintext Awareness via Key Registration. In *Crypto'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 548–564. Springer-Verlag, 2003.
20. M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *STOC'90*, pages 427–437. ACM, 1990.
21. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT-RSA'01*, volume 159-175 of *Lecture Notes in Computer Science*, pages 159–175. Springer-Verlag, 2001.
22. T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *PKC'01*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, 2001.
23. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.
24. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
25. C. Rackoff and D.R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto'91*, volume 567 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.
26. C.P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
27. C.P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In *Asiacrypt'00*, volume 1976 of *Leture Notes in Computer Science*. Springer-Verlag, 2000.
28. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1997.
29. R. Steinfeld, J. Baek, and Y. Zheng. On the Necessity of Strong Assumptions for the Security of a Class of Asymmetric Encryption Schemes. In *ACISP'02*, volume 2384 of *LNCS*, pages 241–256. Springer-Verlag, 2002.
30. Y. Tsiounis and M. Yung. On the Security of El Gamal based Encryption. In *PKC'98*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer-Verlag, 1998.

A Symmetric Key Encryption

A (deterministic) symmetric key encryption (SKE) consists of two algorithms $\Phi = (\mathsf{Enc}, \mathsf{Dec})$. The encryption algorithm Enc with input a secret key $w \in \{0,1\}^{\ell_0}$ and a plaintext $m \in \{0,1\}^{\ell_1}$, outputs a ciphertext c, denoted as $c \leftarrow \mathsf{Enc}(w, m)$. The decryption algorithm Dec with input $w \in \{0,1\}^{\ell_0}$ and the ciphertext c, outputs a plaintext $m \in \{0,1\}^{\ell_1}$, denoted as $m \leftarrow \mathsf{Dec}(w, c)$.

Definition 8 (Key Privacy for SKE). *Let $\Phi = (\mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme. Let $\mathcal{A}$ be an adversary and $\mathcal{EO}$ be an encryption oracle that on a plaintext query (w, β) where $\beta \in \{0,1\}$, returns a corresponding ciphertext c_β under the secret key sk_β. Denote the advantage of $\mathcal{A}$ in the following game:*

$$Adv^{\mathsf{ik\text{-}cpa}}_{\Phi,\mathcal{A}}(k) = \Pr\left[b' = b \;\middle|\; \begin{array}{c} w_0 \leftarrow_R \{0,1\}^{\ell_0}; w_1 \leftarrow_R \{0,1\}^{\ell_0}; \\ (m^*, s) \leftarrow \mathcal{A}^{\mathcal{EO}}(1^{\ell_1}); b \leftarrow_R \{0,1\}; \\ c^* \leftarrow \mathsf{Enc}(w_b, m^*); b' \leftarrow \mathcal{A}^{\mathcal{EO}}(c^*, s) \end{array} \right] - \frac{1}{2}$$

Φ is (ϵ, t)-IK-CPA secure if any adversary with running time t, gains advantage at most ϵ in winning the above game. Furthermore, we say a symmetric key encryption scheme is IK-CPA secure if for any PPT adversary, ϵ is negligible.

For a deterministic symmetric key encryption scheme, key privacy is quite related to the length of the secret key rather than the space of the randomness for a public key encryption scheme.

B Proof of Theorem 4

We first review the transforms (FOPKC and REACT) in Table 2 and Table 3 and analyze their functionalities on key privacy. Theorem 4 naturally follows Lemma 4 and Lemma 5.

Transform: FOPKC		
$\mathsf{K}'(1^k)$	$\mathsf{E}(pk, m)$	$\mathsf{D}(sk, c)$
$(pk', sk') \leftarrow_R \mathsf{K}'(1^k)$	$r \leftarrow_R \mathcal{R}$	$\bar{m} \leftarrow \mathsf{D}'(sk, c)$
$H \leftarrow \mathcal{H}$	$\bar{m} = m\|\|r$, s.t. $\bar{m} \in \mathcal{M}$	$\bar{m} = m\|\|r$
$pk = (pk', H)$	$c \leftarrow \mathsf{E}'(pk, \bar{m}; H(\bar{m}))$	check if $c = \mathsf{E}'(pk, \bar{m}; H(\bar{m}))$
$sk = sk'$		if "valid", return m
return (pk, sk)		return "$\perp$"

† $\Pi' = (\mathsf{K}', \mathsf{E}', \mathsf{D}')$ is an IK-CPA secure public key encryption scheme. $\mathcal{H}$ is the the family of hash functions. $\mathcal{R}$ and $\mathcal{M}$ are the spaces of random coins and plaintext for Π, respectively.

Table 2. Generic Transform for Key Privacy: FOPKC

Lemma 4. *Π' is $(\epsilon(1 - 1/2^{k_1})^{q_H}(1-\gamma)^q_d, t + 2q_H(T_{\Pi'} + O(k))$-IK-CCA secure, assuming Π is (ϵ, t)-IK-CPA secure and γ-uniform, where q_H and q_d are the maximum numbers of hash queries and decryption queries, k is the security parameter and $k_1 = |r|$.*

Proof. Since Π' is IK-CPA secure, Π is also IK-CPA secure. it is sufficient to show, it has also chosen ciphertext security. A simulator $\mathcal{B}$ against Π' interacts with an IK-CCA adversary $\mathcal{A}$ against Π as follows.

Key Generation Query $\mathcal{B}$ forwards its own public key pk_β for $\beta = 0, 1$ and H to $\mathcal{A}$. Especially, H is a function controlled by $\mathcal{B}$.

Hash Query $\mathcal{B}$ maintains a five-entry H-list, initially empty. For a query on $\bar{m} = m||r$, if there is already an entry of the form (pk_β, m, r, h, c) with $\beta \in \{0,1\}$ in the list, $\mathcal{B}$ returns h to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses randomly $h \leftarrow_R \mathcal{R}$, adds $(pk_\beta, \bar{m}, r, h, \mathsf{E}'(pk_\beta, \bar{m}; h))$ for $\beta = 0, 1$ (two new entries), to H-list and returns h to $\mathcal{A}$.

Decryption Query For a decryption query on (pk_β, c) with $\beta \in \{0,1\}$, $\mathcal{B}$ searches in H-list for the entry of the form (pk_β, m, r, h, c). If there is such an entry, $\mathcal{B}$ returns m to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns "$\perp$" to $\mathcal{A}$.

Challenge Query For a chosen plaintext m^* submitted by $\mathcal{A}$, $\mathcal{B}$ sets $\bar{m}^* = m^* \| r^*$, where $r \leftarrow_R \mathcal{R}$ is properly distributed. $\mathcal{B}$ then forwards $\bar{m}$ to its own challenger. After receiving a target ciphertext c^*, $\mathcal{B}$ also forwards c^* to $\mathcal{A}$.

When $\mathcal{A}$ finally halts and outputs a guess b', $\mathcal{B}$ outputs the same bit and terminates. If $\mathcal{A}$ doesn't halt in polynomial time, $\mathcal{B}$ stops interaction with $\mathcal{A}$, outputs a random bit and terminates.

From the description of $\mathcal{B}$, we conclude all the queries are perfect simulated except that (i) $\mathcal{B}$ is asked a hash query on $\bar{m}^*$ where $c^* = \mathsf{E}'(pk^*, \bar{m}^*; r^*)$ for pk^* and some r^*; and (ii) a ciphertext query c which is valid but not in H-list.

For the first case, denote $|r| = k_1$, we have the probability that $\mathcal{B}$ succeeds to answer q_H hash queries is $(1-1/2^{k_1})^{q_H}$. For the second case, for q_d decryption queries, $\mathcal{B}$ succeeds with probability $(1-\gamma)^{q_d}$, where Π' is γ-uniform. Furthermore, if $\mathcal{B}$ doesn't fail in simulation, $\mathcal{B}$ always has the same advantage of winning with $\mathcal{A}$. Observing case (i) and (ii) are independent events, we conclude $\mathcal{B}$'s advantage is $\epsilon(1-1/2^{k_1})^{q_H}(1-\gamma)^{q_d}$. The running time of $\mathcal{B}$ is easily verified from the description of $\mathcal{B}$. □

Transform: REACT		
$\mathsf{K}(1^k)$	$\mathsf{E}(pk, m)$	$\mathsf{D}(sk, c)$
$(pk, sk) \leftarrow \mathsf{K}'(1^k)$ $G \leftarrow_R \mathcal{G}$ $H \leftarrow_R \mathcal{H}$ $pk = (pk', G, H)$ $sk = sk'$ return (pk, sk)	$r \leftarrow_R \mathcal{M}$ $s \leftarrow_R \mathcal{R}$ $c_0 \leftarrow \mathsf{E}'(pk, r)$ $c_1 \leftarrow \mathsf{Enc}(G(r), m)$ $c_2 \leftarrow H(m, r, c_0, c_1)$ return $c = \langle c_0, c_1, c_2 \rangle$	$c = \langle c_0, c_1, c_2 \rangle$ $r \leftarrow \mathsf{D}'(sk, c_0)$ $m \leftarrow \mathsf{Dec}(G(r), c_1)$ check if $c_2 = H(m, r, c_0, c_1)$ if "valid", return m return "$\perp$"

† $\Pi' = (\mathsf{K}', \mathsf{E}', \mathsf{D}')$ is an IK-PCA and OW-PCA secure asymmetric key encryption and $\Phi = (\mathsf{Enc}, \mathsf{Dec})$ is an IK-CPA secure symmetric key encryption scheme with key space $\{0,1\}_1^\ell$. $\mathcal{R}$ and $\mathcal{M}$ are the spaces of random coins and plaintext for Π, respectively. $\mathcal{G}, \mathcal{H}$ is the the family of hash functions.

Table 3. Generic Transform for Key Privacy: REACT

Lemma 5. *Π' is a $(\epsilon_1 + \epsilon_2 + q_G \cdot 2^{-l_1} + q_d \cdot 2^{-l_2}, t_1 + t_2 + O((q_G + q_H)k))$-IK-CCA secure, assuming Π is (ϵ_1, t_1)-IK-PCA secure and Φ is (ϵ_2, t_2)-IK-CPA secure, where q_G and q_d are the numbers of G-oracle queries, and decryption queries.*

Proof. An adversary $\mathcal{B}$ against Π', with plaintext checking oracle access for Π', interacts with an IK-CCA adversary $\mathcal{A}$ against Π as follows.

Key Generation Query $\mathcal{B}$ forwards its own public key pk_β for $\beta = 0, 1$ to $\mathcal{A}$. Especially, G, H is a function controlled by $\mathcal{B}$.

G-Oracle Query $\mathcal{B}$ maintains a two-entry G-list, initially empty. On a query on k_1, if their is an entry of the form $(k_1, k_2$ in the list, returns k_2. Otherwise, chooses $k_2 \leftarrow_R \{0,1\}^{l_1}$, appends the entry (k_1, k_2) in H-list and returns k_2 to $\mathcal{A}$.

H-**Oracle Query** $\mathcal{B}$ maintains a five-entry H-list, initially empty. On a query on (k_1, m, c_0, c_1), if there is an entry of the form (k_1, k_2, c_0, c_1, h) in the list, $\mathcal{B}$ returns h to $\mathcal{A}$. Otherwise, $\mathcal{B}$ sets $h \leftarrow_R \{0,1\}^{l_2}$, appends the entry (k_1, m, c_0, c_1, h) to G-list and returns h to $\mathcal{A}$.

Decryption Query For a decryption query $c = \langle c_0, c_1, c_2 \rangle$, $\mathcal{B}$ checks if there is an entry (k_1, m, c_0, c_1, h) contains (c_0, c_1) queried to H-list before. $\mathcal{B}$ returns m, if the decision oracle tells that (m, c_0) is a correct ciphertext for Π' and there is an entry (k_1, k_2) in G-list such that $c_1 = \mathsf{Enc}(k_2, m)$. Otherwise, $\mathcal{B}$ returns "$\perp$" to $\mathcal{A}$.

Challenge Query When $\mathcal{A}$ outputs a plaintext m^* which it wishes to be challenged, $\mathcal{B}$ also outputs the same plaintext to its own challenger. After receiving a challenge c_0^*, $\mathcal{B}$ sets $c^* = \langle c_0^*, c_1^*, c_2^* \rangle$, where $c_1^* = \mathsf{Enc}(k^*, m^*)$, where $k^* \leftarrow_R \{0,1\}_1^\ell$ and $c_2^* \leftarrow_R \{0,1\}^{l_2}$. $\mathcal{B}$ forwards c^* to $\mathcal{A}$.

Finally, when $\mathcal{A}$ stops and outputs a bit b, $\mathcal{B}$ also outputs the same bit. If $\mathcal{A}$ doesn't stops in polynomial time, $\mathcal{B}$ outputs a random bit and terminates. This completes the description of $\mathcal{B}$.

Denote (k_1^*, k_2^*) is the variables used to build the challenge ciphertext $c^* = \langle c_0^*, c_1^*, c_2^* \rangle$. From above description, when $\mathcal{B}$ never answers wrongly upon an invalid ciphertext query. $\mathcal{B}$ may reject a correct ciphertext query. Denote this event as $\mathsf{Fail1}$, which happens at with probability $(1-2^{-l_1})^{q_G} + (1-2^{-l_2})^{q_d}$, where q_G and q_d is the number of G-oracle queries and decryption queries.

Additionally, $\mathcal{B}$ may fail when $\mathcal{B}$ is queried on k_1^* within G-oracle queries. Denote this event as $\mathsf{Fail2}$, which happens with probability at most $\epsilon_2 + (1-2^{-l_1})^{q_G}$.

Furthermore, denote the event $\mathcal{B}$ fails in the simulation as FailB. Since the information of the public key is only contained in c_0, we have if $\mathcal{B}$ doesn't fail, $\mathcal{A}$'s advantage is at most $\epsilon_{\mathcal{B}}$. Namely, $\epsilon_{\mathcal{B}} = \epsilon_{\mathcal{A}} \cdot \Pr[\neg\mathsf{FailB}] = \epsilon_{\mathcal{A}} \cdot \Pr[\neg(\mathsf{Fail1} \wedge \mathsf{Fail2})] \geq \epsilon_{\mathcal{A}} - \Pr[\mathsf{Fail1}] - \Pr[\mathsf{Fail2}]$.

Recall $\epsilon_{\mathcal{B}}$ is $\mathcal{B}$'s advantage against Π', we have $\epsilon_{\mathcal{A}} \leq \epsilon_1 + \epsilon_2 + q_G \cdot 2^{-l_1} + q_d \cdot 2^{-l_2}$, which is exactly what was claimed in Lemma 5. The claimed time bound can be easily verified from the description of $\mathcal{B}$. □