Hindawi
Security and Communication Networks
Volume 2021, Article ID 5895809, 16 pages
https://doi.org/10.1155/2021/5895809

Research Article

Selective-Opening Security for Public-Key Encryption in the Presence of Parameter Subversion

Burong Kang,[1,2,3] **Zhengan Huang**,[4] **and Lei Zhang**[1,2,3]

[1]*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China*
[2]*Guangxi Key Laboratory of Cryptography and Information Security, Guilin, Guangxi 541000, China*
[3]*Engineering Research Center of Software/Hardware Co-Design Technology and Application, Ministry of Education, East China Normal University, Shanghai 200062, China*
[4]*Peng Cheng Laboratory, Shenzhen 518000, China*

Correspondence should be addressed to Zhengan Huang; zhahuang.sjtu@gmail.com and Lei Zhang; leizhang@sei.ecnu.edu.cn

Received 21 July 2021; Accepted 15 September 2021; Published 23 October 2021

Academic Editor: Yong Yu

In public-key encryption (PKE), ciphertexts received by a receiver may be possibly correlated and the security of a PKE relies on honestly generated system parameters. Security against selective opening attacks (SOA) for PKE guarantees that even when an attacker has broken into a subset of honestly generated ciphertexts and opened them (i.e., seeing plaintexts and random bits), the unopened ciphertexts remain secure. While security against parameter subversion attacks (PSA) for PKE requires that even when the public system parameters are maliciously generated, a PKE scheme should be secure. In this paper, we initiate the study of PKE secure against both SOA and PSA. To capture SOA and PSA simultaneously, we formulate a new security notion called indistinguishability under selective opening attacks and parameter subversion attacks (IND-SO-PSA). Further, we define the lossy trapdoor function and all-but-many lossy trapdoor function in the presence of PSA (LTF-PSA and ABM-LTF-PSA correspondingly) and propose an instantiation with the efficiently-embeddable group (EG). Applying these new primitives, we construct a PKE scheme that is proven to be IND-SO-PSA secure.

1. Introduction

Public-key encryption (PKE) is a fundamental cryptographic primitive to achieve confidential communication from a sender to a receiver [1]. Traditional security requirement of PKE is indistinguishability under chosen plaintext/ciphertext attack (IND-CPA/CCA) security and only considers the single sender case, i.e., the communication is between a single sender and a single receiver. Whereas, many application scenarios, e.g., secure multiparty computation, secret sharing, cloud computing, are often multisender case [1–6]. These scenarios usually require a set of senders to generate their own messages (possibly correlated) encrypted with the same receiver's public key and may suffer from some special attacks. One of the mostly studied attacks is selective opening attacks (SOA) [7–18], in which an attacker may corrupt a subset of the senders to open their ciphertexts as well as the corresponding random bits used in the encryption algorithm. Under such an attack, the requirement to guarantee the security of the unopened ciphertexts is beyond the capability of traditional IND-CPA/CCA security [14, 15, 19]. SOA security [7] is a commonly secure property to capture the above attack.

Commonly, the security of PKE schemes depends on some honestly generated public system parameters, e.g., security parameter, primes, elliptic curves, and common reference string [20–24]. In fact, public system parameters are often specified in some standards [20, 23], e.g., NIST FIPS 186-4 (2013) [25, 26]. It is timesaving for the implementations of PKE systems to use the public system parameters. However, recent research results show that PKE schemes may suffer from parameter subversion attacks (PSA) which allow adversaries to fully control the public system parameters [20, 23, 27–31] and hence compromise

the security of the corresponding PKE schemes. For instance, the elliptic curve cryptosystem specified in NIST FIPS 186-4 standard has been analyzed to be insecure due to the malicious manipulation of the elliptic curves [23]. Thus, PKE schemes resistant to PSA should be developed. That is, a PKE scheme should provide usual security with trusted system parameters, but retain as much security as possible when the parameters are maliciously generated.

1.1. Related Work. SOA security was originally discussed by Dwork et al. [17] and first formalized by Bellare et al. [7]. In [7], SOA security was further classified into simulation-based SOA security (SIM-SO security) and the indistinguishability-based SOA security (IND-SO security) [7]. The formal requires that there exists a simulator which can compute the same output as the adversary without ever seeing any ciphertext at all. For IND-SO security, it requires that any adversary who is given a vector of ciphertexts and an opened subset of these ciphertexts cannot distinguish the unopened messages from fresh messages which are resampled from a message distribution [7]. According to whether a condition of the message distribution for the fresh messages is needed, there are weak IND-SO security where the message distribution is efficiently conditionally resamplable and full IND-SO security where the message distribution is an arbitrary one [14]. At present, few full IND-SO secure schemes are proposed. In the following, when we mention IND-SO security, it refers to the weak one. In [7], Bellare et al. showed that SIM-SO security is stronger than the weak IND-SO security. Whereas, it is showed that SIM-SO security is significantly harder to achieve [11, 18]. Thus, in this paper, we focus on the IND-SO security.

The existing PKE schemes which satisfy IND-SO security are realized based on lossy trapdoor function (LTF) which is firstly studied in [32] and originally motivated to construct PKE schemes secure against CCA attacks. The primal LTFs in [32] are all-but-one LTFs (ABO-LFT) and only suite for the settings in which there is no more than one challenge ciphertext. PKE schemes secure against SOA usually involve multiple challenge ciphertexts. All-but-N LTF (ABN-LTF) [16] allows N challenge ciphertexts. However, N has to be fixed at the construction time. In SOA security, an adversary may observe many ciphertexts (arbitrary number of challenge ciphertexts) and may open arbitrary number of them. All-but-many LTF (ABM-LTF) introduced by Hofheinz [33] could be viewed as a generalization of both ABO-LFT and ABN-LTF and may meet the above requirement.

Beside the SOA, PKE schemes also suffer from PSA. Such an attack is initially studied in [20, 21]. In [20], the security notion called indistinguishable encryptions under parameter subversion attacks (IND-PSA) is introduced to capture PSA, which guarantees that even though the public system parameter of a PKE scheme is chosen maliciously, the resulting ciphertext should be indistinguishable [20]. Based on a security tool referred to as an efficient embedding group (EG), a PKE scheme satisfying IND-PSA is proposed as well. Whereas, their scheme was only considered in the single sender case. In other words, the construction of the PKE scheme secure against the PSA in SOA setting is out of their consideration.

1.2. Our Contributions. Although there are already some research results for the SOA and PSA, so far, they are only considered separately. Our aim is to construct a PKE scheme secure against both of these attacks simultaneously.

To achieve this goal, we formalize a new security notion called indistinguishability under selective opening attacks and parameter subversion attacks (IND-SO-PSA). IND-SO-PSA security captures indistinguishability under parameter subversion attacks, selective opening attacks, and chosen ciphertext attacks. It guarantees that even though the system parameter of the PKE scheme is chosen maliciously and the adversary is allowed to open a subset of the challenge ciphertexts (i.e., seeing the corresponding plaintexts and randomness used during the encryption), the remainder of the plaintexts should be indistinguishable from freshly resampled ones.

PKE achieving IND-SO security and PKE satisfying IND-PSA are now independently realized based on ABM-LTFs and EGs, respectively. However, the existing ABM-LTFs are only designed for PKE secure against SOA and not applicable for the construction of a PKE scheme secure against PSA. On the contrary, it seems that it is not a trivial task to extend existing EGs to prevent SOA. Therefore, it would be interesting if we can merge these two tools so that the SOA and PSA can be prevented simultaneously. In order to achieve this intention, we define the lossy trapdoor function and all-but-many lossy trapdoor function in the presence of PSA (LTF-PSA and ABM-LTF-PSA correspondingly) and propose an instantiation with the efficiently-embeddable group (EG). With these new primitives and instantiation, we construct a novel PKE scheme that is proven to satisfy the IND-SO-PSA security.

1.3. Paper Organization. The rest of this paper is organized as follows. Some preliminaries are given in Section 2. The definitions of LTF-PSA and ABM-LTF-PSA primitives and security model of IND-SO-PSA are stated in Section 3. In Section 4, we propose the instantiation of ABM-LTF-PSA from an efficiently-embeddable group family. LTF-PSA can be similarly instantiated. Based on the newly constructed LTF-PSA and ABM-LTF-PSA, we construct a new PKE scheme and prove its security in Section 5 and Section 6, respectively. Section 7 is the conclusion.

2. Preliminary

2.1. Notions and Definitions. Throughout this paper, we use k to denote the security parameter. Let X be a finite set and A be an algorithm. We denote by $x \xleftarrow{\$} X$ the random selection of an element x from X and by $y \xleftarrow{\$} A(x_1, \ldots; r)$ the running of algorithm A with inputs $(x_1, \ldots; r)$ and output y. If a probabilistic algorithm's running time is polynomial in k, we call it probabilistic polynomial-time (PPT). For a string α, we denote by $|\alpha|$ the bit length of α. For a nonnegative integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$.

Assume $\mathbf{x} = (x^{(1)}, \ldots, x^{(n)})$ is an n-array vector, we denote by $x^{(i)}$ its ith component and by $|\mathbf{x}|$ its length. Define $\mathbf{X}^{n\times n}$ to be an $n \times n$ matrix, $\mathsf{Rank}(\mathbf{X})$ to be the rank of $\mathbf{X}$, $\mathbf{X}^T$ to be the transpose of $\mathbf{X}$, and $\mathbf{X}_{i,j}$ is the element of ith row and jth column, $1 \le i$ and $j \le n$. For two $n \times n$ matrixes $\mathbf{A}$ and $\mathbf{B}$, let $(\mathbf{A}|\mathbf{B})$ (i.e., an $n \times 2n$ matrix) be the concatenation of $\mathbf{A}$ and $\mathbf{B}$. Let $\mathbb{G}$ be an embedding cyclic group generated by generator g with order q. Given an $n \times n$ matrix $\mathbf{X}$ for $\mathbf{X}_{i,j} \in \mathbb{Z}_q$, the operation $\mathbf{Y} = g^{\mathbf{X}}$ is defined by computing $\mathbf{Y}_{i,j} = g^{\mathbf{X}_{i,j}}$ and setting the matrix $\mathbf{Y}$ to be $(\mathbf{Y}_{i,j})^{n\times n}$. For two random variables X and Y over finite set $\mathbb{S}$, their statistical distance is $\Delta(X, Y) = (1/2)\sum_{s\in\mathbb{S}}|\Pr[X = s] - \Pr[Y = s]|$.

Definition 1 (universal hash function). Let $l_1 = l_1(k)$, $l_2 = l_2(k)$ and $l_3 = l_3(k)$. A family of functions UHF_s: $\{0,1\}^{l_1} \longrightarrow \{0,1\}^{l_2}$ is called universal hash functions with index key $s \in \{0,1\}^{l_3}$, if for all $X, X' \in \{0,1\}^{l_1}$, with $X \neq X'$, we have $\Pr[\mathrm{UHF}_s(X) = \mathrm{UHF}_s(X')] \le 1/|l_1|$ over the random choice of function UHF_s.

Lemma 1 (adapted from [34]). *Let X and Y are random variables such that $X \in \{0,1\}^{l_1}$ and the average min-entropy of X given Y is at last k. Let UHF_s: $\{0,1\}^{l_1} \longrightarrow \{0,1\}^{l_2}$ be a family of universal hash functions, where $l_2 \le k - 2\log(1/\epsilon)$. Let $R \longleftarrow_{\$} \{0,1\}^{l_2}$, and it holds that $\Delta((UHF_s, UHF_s(X), Y), (UHF_s, R, Y)) \le \epsilon$.*

Definition 2 (pseudorandom function). Let $l_2 = l_2(k)$. A function family PRF_{sk}: $\{0,1\}^{l_2} \longrightarrow \{0,1\}^{l}$ with a key space $\{0,1\}^*$ is called the pseudorandom function if for any PPT adversary $\mathscr{A}$, $\mathscr{A}$'s advantage $\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}}(k) = |\Pr[\mathbf{G}^{\mathrm{PRF}}_{\mathscr{A}}(k) = 1] - 1/2|$ of winning the security game $\mathbf{G}^{\mathrm{PRF}}_{\mathscr{A}}(k)$ defined in Figure 1 is negligible.

Definition 3 (efficiently re-sampling). Let $n = n(k)$ and $\mathscr{D}$ be a joint distribution over $(\{0,1\}^k)^n$. We say that $\mathscr{D}$ is efficiently re-sampling if there is an algorithm $\mathsf{Resamp}_{\mathscr{D}}$ such that, for any $\mathcal{I} \subseteq [n]$ and any partial vector $(m'^{(i)})_{i\in\mathcal{I}} \in (\{0,1\}^k)^{|\mathcal{I}|}$, $\mathsf{Resamp}_{\mathscr{D}}((m'^{(i)})_{i\in\mathcal{I}})$ samples from the distribution $\mathscr{D}$, conditioned on $(m'^{(i)}) = (m^{(i)})$ for all $i \in \mathcal{I}$.

2.2. Efficiently-Embeddable Group Family. An efficiently-embeddable group family $\mathsf{EG} = (\mathsf{EG.P}, \mathsf{EG.G}, \mathsf{EG.S}, \mathsf{EG.E}, \mathsf{EG.I})$ consists of following algorithms [20].

Parameter generation algorithm $\mathsf{EG.P}(1^k)$ inputs the security parameter k and returns a public system parameter π. Group generation algorithm $\mathsf{EG.G}(1^k, \pi)$ inputs k and π and returns a tuple $\mathbb{G} = (G, q, g)$, where G is a cyclic group of prime order q and g is a generator of G. Sampling algorithm $\mathsf{EG.S}(1^k, \pi, \mathbb{G})$ is used to sample exponents for the group generator. It returns an exponent $y \in \mathbb{Z}_q$. Embedding algorithm $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, g^y)$ is used to embed group elements into $\mathsf{EG.ES}(k, \pi)$, where $\mathsf{EG.ES}(k, \pi)$ is a finite set called the embedding space [20]. That is, the output $\sigma \longleftarrow_{\$} \mathsf{EG.E}(1^k, \pi, \mathbb{G}, g^y)$ (i.e., an embedding) is in $\mathsf{EG.ES}(k, \pi)$. For such a construction, refer to [20] for the details. Inversion algorithm $\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \sigma)$ is a deterministic algorithm which is used to invert the embedding. It returns $\psi \longleftarrow \mathsf{EG.I}(1^k, \pi, \mathbb{G}, \sigma)$ in $\mathbb{G}$. This algorithm should satisfy correctness, i.e., $\Pr[\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \sigma) = g^y] \ge 1 - \mathsf{EG.ie}(k)$, for all $\pi \longleftarrow \mathsf{EG.P}(1^k)$ and $\mathbb{G} \longleftarrow \mathsf{EG.G}(1^k, \pi)$, where the probability is over $y \longleftarrow_{\$} \mathsf{EG.S}(1^k, \pi, \mathbb{G})$ and $\sigma \longleftarrow_{\$} \mathsf{EG.E}(1^k, \pi, \mathbb{G}, g^y)$. The function $\mathsf{EG.ie}(k)$: $\mathbb{N} \longrightarrow \mathbb{R}$ is defined to be the inversion error of EG.

As noted in [20], $\mathsf{EG} = (\mathsf{EG.P}, \mathsf{EG.G}, \mathsf{EG.S}, \mathsf{EG.E}, \mathsf{EG.I})$ should satisfy the embedding pseudorandomness under parameter subversion attack (EPR-PSA). We review the definition of it below:

Game $\mathbf{G}^{\mathrm{PRF}}_{\mathcal{A}}(k)$
$\mathrm{sk} \leftarrow_{\$} \{0,1\}^*$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\mathrm{FN}}(1^k)$
Return 1 if $b = b'$, and 0 otherwise

$\mathrm{FN}(x)$
If $b = 1$ then $S[x] \leftarrow \mathrm{PRF}_{\mathrm{sk}}(x)$
Else if $S[x] = \perp$ then $S[x] \leftarrow_{\$} \{0,1\}^l$
Return $S[x]$

Figure 1: Game-defining PRF security of a function family.

Definition 4 (embedding pseudorandomness [20]). EG holds embedding pseudorandomness under parameter subversion attack, if for any PPT adversary $\mathscr{A}$ against EG, $\mathscr{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}}(k) = \left|\Pr\left[\mathbf{G}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}}(k) = 1\right] - \frac{1}{2}\right|, \quad (1)$$

of winning the security game $\mathbf{G}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}}(k)$ defined in Figure 2 is negligible.

We note that, in $\mathbf{G}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}}(k)$, $\mathscr{A}$ may query the INIT oracle for only once with a parameter π^* chosen by himself. For a randomly-chosen coin $b \longleftarrow_{\$} \{0,1\}$ chosen by $\mathscr{C}$, $\mathscr{A}$ wins the game, if he can distinguish σ^* is generated by running algorithms EG.S and EG.E, or sampled uniformly from the embedding space $\mathsf{EG.ES}(k, \pi^*)$. In other words, $\mathscr{A}$ wins the game, if its guess b' is equal to b.

2.3. Public-Key Encryption. Let $\mathbb{M}$ be a message space, $\mathbb{C}$ be a ciphertext space, $\mathbb{PK}$ be a public key space, $\mathbb{SK}$ be a secret key space, and $\mathbb{R}$ be a randomness space. A public-key encryption (PKE) scheme $\mathsf{PKE} = (\mathsf{PKE.P}, \mathsf{PKE.Kg}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ consists of following algorithms.

The parameter generation algorithm PKE.P takes as input 1^k and returns a public system parameter π. The key generation algorithm PKE.Kg takes as input $1^k, \pi$ and returns a pair of public key and a secret key $(pk, sk) \in \mathbb{PK} \times \mathbb{SK}$. The encryption algorithm PKE.Enc takes as input $(1^k, \pi, pk, m; r)$ to encrypt a message $m \in \mathbb{M}$ under pk, π and randomness $r \in \mathbb{R}$, and outputs a ciphertext $c \in \mathbb{C}$. The decryption algorithm PKE.Dec is a deterministic

Game $\mathbf{G}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathcal{A}}(k)$

$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\mathrm{INIT}}(1^k)$
Return 1 if $b = b'$, and 0 otherwise

$\underline{\mathrm{INIT}(\pi^*)}$
$\mathbb{G} \leftarrow_{\$} \mathsf{EG.G}(1^k, \pi^*)$
If $(\mathbb{G} = \perp)$ then return $\perp$
$(\mathrm{G}, q, g) \leftarrow \mathbb{G}$
If $b = 1$ then
  $y \leftarrow_{\$} \mathsf{EG.S}(1^k, \pi^*, \mathbb{G})$
  $\sigma^* \leftarrow_{\$} \mathsf{EG.E}(1^k, \pi^*, \mathbb{G}, g^y)$
Else $\sigma^* \leftarrow_{\$} \mathsf{EG.ES}(k, \pi^*)$
Return $(\mathbb{G}, \sigma^*)$

Figure 2: Games-defining EPR-PSA security with respect to EG.

algorithm. It takes as input $(1^k, \pi, sk, c)$ and outputs a message m or an error symbol $\perp$ which means the ciphertext is invalid. We require that, for all $m \in \mathbb{M}$ and $(pk, sk) \in \mathbb{PK} \times \mathbb{SK}$, it holds that $\mathsf{PKE.Dec}(1^k, \pi, sk, (\mathsf{PKE.Enc}(1^k, \pi, pk, m; r)) = m$.

The above definition of PKE is suitable for single user (single message) encryption. In this paper, we consider the public-key encryption scheme in selective opening scenarios, i.e., the encryption in multiuser settings (multiple messages). We have to use the notion of vector-valued encryption [1, 35, 36]. That is, for vectors $\mathbf{m} = (m^{(i)})$, $\mathbf{r} = (r^{(i)})$, and $\mathbf{c} = (c^{(i)})$ with $i \in [n]$, we denote by $\mathbf{c} \longleftarrow_{\$} \mathsf{PKE.Enc}(1^k, \pi, pk, \mathbf{m}; \mathbf{r})$ the tuple $(\mathsf{PKE.Enc}(1^k, \pi, pk, m^{(1)}; r^{(1)}), \mathsf{PKE.Enc}(1^k, \pi, pk, m^{(2)}; r^{(2)}), \ldots, \mathsf{PKE.Enc}(1^k, \pi, pk, m^{(n)}; r^{(n)}))$, where n is the number of messages. The decryption algorithm is analogous.

To capture parameter subversion attacks, Auerbach et al. [20] formulated a new security notion called indistinguishability under parameter subversion attack (IND-PSA). IND-PSA is a different security notion from traditional IND-CCA security and guarantees that the ciphertexts are indistinguishable even when the system parameter is generated maliciously. We introduce the IND-PSA security below.

Definition 5 (IND-PSA security [20]). PKE holds indistinguishability under parameter subversion attack, if for any PPT adversary $\mathcal{A}$ against PKE, $\mathcal{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{IND-PSA}}_{\mathsf{PKE},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{G}^{\mathrm{IND-PSA}}_{\mathsf{PKE},\mathcal{A}}(k) = 1\right] - \frac{1}{2}\right|, \quad (2)$$

of winning the security game $\mathbf{G}^{\mathrm{IND-PSA}}_{\mathsf{PKE},\mathcal{A}}(k)$ defined in Figure 3 is negligible.

In game $\mathbf{G}^{\mathrm{IND-PSA}}_{\mathsf{PKE},\mathcal{A}}(k)$, the challenger $\mathcal{C}$ will toss a coin to determine a challenge bit $b \in \{0,1\}$. Then, $\mathcal{A}$ may query INIT oracle with a system parameter π^* chosen by himself to initialize a public-private key pair, and the public key pk is returned to $\mathcal{A}$. The ENC oracle may be queried with two challenge plaintexts m_0, m_1 ($|m_0| = |m_1|$) and return the challenge ciphertext c^* to $\mathcal{A}$. If $b = 0$, c^* is an encryption for m_0; otherwise, it is the encryption for m_1. The DEC oracle is not allowed for the challenge ciphertext c^*. We note that the INIT and ENC oracle can be queried for only once. Given the public key and challenge ciphertext, $\mathcal{A}$ outputs a guess bit b'. If $b' = b$, we say $\mathcal{A}$ wins the game, and the game will output 1 in this case.

The standard IND-CCA security of the public-key encryption scheme also does not generalize the security under selective opening attacks. To capture the security of the public-key encryption scheme under the selective opening attacks, new security notion called indistinguishability against selective opening attacks and chosen ciphertext attack (IND-SO-CCA) is proposed [16]. IND-SO-CCA security guarantees that any adversary cannot distinguish the messages corresponding to the unopened ciphertexts from a freshly re-sampled one according to a re-samplable distribution $\mathcal{D}$ conditioned on the opened messages. We introduce the IND-SO-CCA security below.

Definition 6 (IND-SO-CCA security [20]). PKE holds indistinguishability against selective opening attacks and chosen ciphertext attacks (IND-SO-CCA), if for any PPT adversary $\mathcal{A}$ against PKE, $\mathcal{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{IND-SO-CCA}}_{\mathsf{PKE},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{G}^{\mathrm{IND-SO-CCA}}_{\mathsf{PKE},\mathcal{A}}(k) = 1\right] - \frac{1}{2}\right|, \quad (3)$$

of winning the security game $\mathbf{G}^{\mathrm{IND-SO-CCA}}_{\mathsf{PKE},\mathcal{A}}(k)$ defined in Figure 4 is negligible.

At the beginning of the game $\mathbf{G}^{\mathrm{IND-SO-CCA}}_{\mathsf{PKE},\mathcal{A}}(k)$, the challenger $\mathcal{C}$ firstly tosses a coin to determine a challenge bit $b \in \{0,1\}$. Then, $\mathcal{A}$ firstly queries the INIT oracle. $\mathcal{C}$ will run the algorithms PKE.P and PKE.Kg to generate the public system parameter π and public-secret key pair (pk, sk) correspondingly and returns the public key pk to $\mathcal{A}$. $\mathcal{A}$ will output a message distribution $\mathcal{D}$ as well as a re-sampling algorithm $\mathsf{ReSamp}_{\mathcal{D}}$ to $\mathcal{C}$ after adaptively querying the DEC oracle. Subsequently, $\mathcal{C}$ uniformly samples a message vector $\mathbf{m}_0$ from $\mathcal{D}$ and a randomness vector $\mathbf{r} = (r^{(i)})_{i \in [n]}$ from a randomness space $(\mathbb{R})^n$. $\mathcal{C}$ returns the challenge ciphertext $\mathbf{c} = (c^{(i)})_{i \in [n]}$ to $\mathcal{A}$. After receiving $\mathbf{c}$, $\mathcal{A}$ adaptively queries the DEC oracle and chooses an arbitrary subset $\mathcal{I} \subseteq [n]$ to indicate which ciphertexts he wants to open. $\mathcal{I}$ is given to $\mathcal{C}$. Notice that any element of the challenge ciphertext $\mathbf{c}$ is not allowed for DEC oracle. After receiving $\mathcal{I}$, $\mathcal{C}$ will re-sample a message $\mathbf{m}_1 \longleftarrow_{\$} \mathsf{Resamp}_{\mathcal{D}}(\mathbf{m}_{\mathcal{I}})$ conditioned on the $(m^{(i)})_{i \in \mathcal{I}}$ of $\mathbf{m}_0$. $\mathcal{C}$ will send $(\mathbf{m}_b, (m^{(i)}, r^{(i)})_{i \in [\mathcal{I}]})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess bit b', if $b' = b$, we say $\mathcal{A}$ wins the game, and the game will output 1 in this case.

2.4. Chameleon Hash Function. A chameleon hash function $\mathsf{CHF} = (\mathsf{CHF.Kg}, \mathsf{CHF.Eval}, \mathsf{CHF.Equ})$ with domain CHF.Dom consists of the following algorithms. The key generation algorithm CHF.Kg takes as input 1^k and outputs a public key pk_{chf} and a trapdoor td_{chf}. The evaluation algorithm $\mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x; R)$ takes as input a public key pk_{chf}, a preimage $x \in \mathsf{CHF.Dom}$, and a randomness

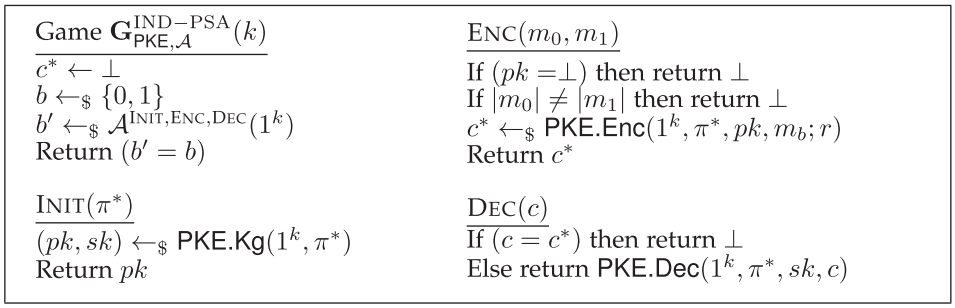

Figure 3: Game-defining IND-PSA security of PKE.

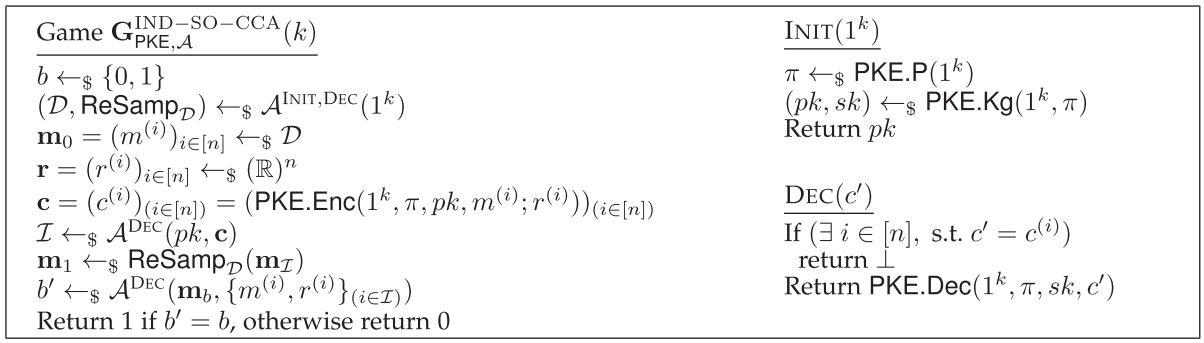

Figure 4: Game-defining IND-SO-CCA security of PKE.

$R \in \mathscr{R}_{\mathsf{chf}}$ where $\mathscr{R}_{\mathsf{chf}}$ is the randomness space of CHF and returns an image $y = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x; R) \in \mathsf{CHF.Ran}$, where CHF.Ran is the range of CHF. The equivocation algorithm $\mathsf{CHF.Equ}(td_{\mathsf{chf}}, x, x'; R)$ takes a trapdoor td_{chf}, R, x, and x' as input and outputs a randomness $R' \in \mathscr{R}_{\mathsf{chf}}$ such that $\mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x; R) = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x'; R')$. We require that, for any $x, x' \in \mathsf{CHF.Dom}$, if R is uniformly distributed, then so is $R' \in \mathscr{R}_{\mathsf{chf}}$.

A secure chameleon hash function $\mathsf{CHF} = (\mathsf{CHF.Kg}, \mathsf{CHF.Eval}, \mathsf{CHF.Equ})$ should satisfy the collision resistance property, which is introduced below.

Collision Resistance. Given a public key pk_{chf}, it is difficult for any PPT adversary to find (x, R, x', R') such that $\mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x; R) = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x'; R')$ for $x \neq x'$ without td_{chf}. Formally, for any PPT adversary $\mathscr{A}$, for random $(pk_{\mathsf{chf}}, td_{\mathsf{chf}}) \longleftarrow_{\$} \mathsf{CHF.Kg}(1^k)$, the advantage $\mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}}(k) = \Pr[x \neq x' \wedge \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x; R) = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, x'; R') \mid ((x, R), (x', R')) \longleftarrow \mathscr{A}(1^k, pk_{\mathsf{chf}})]$ is negligible.

2.5. Lossy Trapdoor Function. A lossy trapdoor function $\mathsf{LTF} = (\mathsf{LTF.IKg}, \mathsf{LTF.LKg}, \mathsf{LTF.Eval}, \mathsf{LTF.Inv})$ with domain LTF.Dom consists of the following algorithms [16, 32, 33, 37, 38].

(i) $\mathsf{LTF.IKg}(1^k)$: this is the injective key generation algorithm. On input 1^k, it outputs an injective evaluation key ek and a inversion key ik for an injective function.

(ii) $\mathsf{LTF.LKg}(1^k)$: this is the lossy key generation algorithm. On input 1^k, it outputs a lossy evaluation key lk for a lossy function. In this case, there is no inversion key.

(iii) $\mathsf{LTF.Eval}(ek, x)$: this is the evaluation algorithm. It takes the evaluation key ek and a preimage $x \in \mathsf{LTF.Dom}$ as input and returns an image $y = \mathsf{LTF.Eval}(ek, x) \in \mathsf{LTF.Ran}$, where LTF.Dom and LTF.Ran denote the domain and the range of LTF, respectively.

(iv) $\mathsf{LTF.Inv}(ik, y)$: this is the inversion algorithm. It takes as input the inversion key ik and an image $y \in \mathsf{LTF.Ran}$ and outputs $x \in \mathsf{LTF.Dom}$.

Moreover, for any LTF, the following property requirements should be satisfied.

Correctness. The above LTF algorithm should satisfy correctness, i.e., $\mathsf{LTF.Inv}(ik, \mathsf{LTF.Eval}(ek, x)) = x$ for all $(ek, ik) \longleftarrow_{\$} \mathsf{LTF.IKg}(1^k)$, $x \in \mathsf{LTF.Dom}$.

Lossiness. A lossy trapdoor function is l-lossy if for all possible $lk \longleftarrow_{\$} \mathsf{LTF.LKg}(1^k)$, the image set $\mathsf{LTF.Eval}(lk, \mathsf{LTF.Dom})$ is of size at most $|\mathsf{LTF.Dom}|/2^l$.

Indistinguishability. Let $\mathscr{A}$ be any PPT adversary against LTF. LTF holds indistinguishability if $\mathscr{A}$ cannot distinguish the injective evaluation key ek from the lossy evaluation key lk, i.e., $\mathscr{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{IND}}_{\mathsf{LTF},\mathscr{A}}(k) = \left|\Pr\left[\mathscr{A}\left(1^k, ek\right) = 1\right] - \Pr\left[\mathscr{A}\left(1^k, lk\right) = 1\right]\right|, \tag{4}$$

is negligible, where $(ek, ik) \longleftarrow_{\$} \mathsf{LTF.IKg}(1^k)$ and $lk \longleftarrow_{\$} \mathsf{LTF.LKg}(1^k)$.

2.6. All-but-Many Lossy Trapdoor Function. We now recall the definition of all-but-many lossy trapdoor function (ABM-LTF) which is originally introduced by Hofheinz [33]. In the following definition, we keep the same mechanism of tags as in [33]. In other words, each tag of ABM-LTF consists of two parts: the core part which is also called the primary part, and the auxiliary part which is commonly a random string. In addition, we note that we maintain the same definition of tag space introduced in [34]. That is, the tag space of ABM-LTF is divided into three disjoint subsets: a lossy tag set, an injective tag set, and an invalid tag set. We will give the concrete definition of ABM-LTF in the following.

An all-but-many lossy trapdoor function $\mathsf{ABM.LTF} = (\mathsf{ABM.Kg}, \mathsf{ABM.Eval}, \mathsf{ABM.Inv}, \mathsf{ABM.LTg})$ with domain $\mathsf{ABM.Dom}$ consists of the following algorithms.

(i) $\mathsf{ABM.Kg}(1^k)$: this is a key generation algorithm. On input 1^k, it outputs an injective evaluation key ek_{abm}, an inversion key ik_{abm}, and a tag key tk_{abm}. With the evaluation key ek_{abm}, we define a set $\mathcal{T} = \mathcal{T}_p \times \mathcal{T}_a$ which is a tag space. We note that the tag space $\mathcal{T}$ consists of three disjoint sets: the lossy tags $\mathcal{T}_{\mathsf{los}}$, the injective tags $\mathcal{T}_{\mathsf{inj}}$, and the invalid tags $\mathcal{T}_{\mathsf{inv}}$. It is easy to find that each of them is a subset of $\mathcal{T}$, i.e., $\mathcal{T}_{\mathsf{los}} \subseteq \mathcal{T}$, $\mathcal{T}_{\mathsf{inj}} \subseteq \mathcal{T}$ and $\mathcal{T}_{\mathsf{inv}} \subseteq \mathcal{T}$. Each tag is defined in the term of $t = (t_p, t_a)$, where $t_{\mathsf{p}} \in \mathcal{T}_{\mathsf{p}}$ is the core part and $t_a \in \mathcal{T}_a$ is the auxiliary part.

(ii) $\mathsf{ABM.Eval}(ek_{\mathsf{abm}}, t, x)$: this is the evaluation algorithm. It takes the evaluation key ek_{abm}, the tag t, and a preimage $x \in \mathsf{ABM.Dom}$ as input and returns $y = \mathsf{ABM.Eval}(ek_{\mathsf{abm}}, t, x) \in \mathsf{ABM.Ran}$, where $\mathsf{ABM.Dom}$ and $\mathsf{ABM.Ran}$ denote the domain and the range of $\mathsf{ABM.LTF}$, respectively.

(iii) $\mathsf{ABM.Inv}(ik_{\mathsf{abm}}, t, y)$: this is the inversion algorithm. It takes an inversion key ik_{abm}, a tag t, and an image y as input and outputs $x \in \mathsf{ABM.Dom}$.

(iv) $\mathsf{ABM.LTg}(tk_{\mathsf{abm}})$: this is the lossy tag generation algorithm. It takes the tag key tk_{abm} as input and outputs a lossy tag $t = (t_p, t_a)$.

Moreover, for any ABM-LTF, the following property requirements should be satisfied.

Correctness. $\mathsf{ABM.LTF}$ should satisfy correctness, i.e., $\mathsf{ABM.Inv}(\mathsf{ABM.Eval}(ek_{\mathsf{abm}}, t, x), ik_{\mathsf{abm}}, t) = x$ for all $(ek_{\mathsf{abm}}, ik_{\mathsf{abm}}, tk_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}(1^k)$, tag $t \in \mathcal{T}_{\mathsf{inj}}$, and $x \in \mathsf{ABM.Dom}$. This property is also referred to as invertibility.

Lossiness. $\mathsf{ABM.LTF}$ is l-lossy if the size of the image set $\mathsf{ABM.Eval}(ek_{\mathsf{abm}}, t, \mathsf{ABM.Dom})$ is at most $|\mathsf{ABM.Dom}|/2^l$, for all possible $(ek_{\mathsf{abm}}, ik_{\mathsf{abm}}, tk_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}(1^k)$ and lossy tag $t \in \mathcal{T}_{\mathsf{los}}$. l is called the lossiness of $\mathsf{ABM.LTF}$.

Indistinguishability. Let $\mathcal{A}$ be any PPT adversary against $\mathsf{ABM.LTF}$. $\mathsf{ABM.LTF}$ holds the indistinguishability between multiple lossy tags and random tags if $\mathcal{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{IND}}_{\mathsf{ABM.LTF},\mathcal{A}}(k) = \left|\Pr\left[\mathcal{A}^{\mathsf{ABM.LTg}(tk_{\mathsf{abm}},\cdot)}\left(1^k, ek_{\mathsf{abm}}\right) = 1\right]\Pr\left[\mathcal{A}^{\mathcal{O}_{\mathcal{T}}(\cdot)}\left(1^k, ek_{\mathsf{abm}}\right) = 1\right]\right|, \tag{5}$$

is negligible, where $(ek_{\mathsf{abm}}, tk_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}(1^k)$. The call to oracle $\mathsf{ABM.LTg}(tk_{\mathsf{abm}}, \cdot)$ returns a lossy tag, and $\mathcal{O}_{\mathcal{T}}(\cdot)$ is a random oracle which returns a uniform and independent tag from $\mathcal{T}$.

Evasiveness. This property requires that it is hard to compute noninjective tags for any adversary, even though it has been given multiple lossy tags. Let $\mathcal{A}$ be any PPT adversary against $\mathsf{ABM.LTF}$. $\mathsf{ABM.LTF}$ holds the evasiveness if $\mathcal{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{EVA}}_{\mathsf{ABM.LTF},\mathcal{A}}(k) = \Pr\left[\mathcal{A}^{\mathsf{ABM.LTg}(tk_{\mathsf{abm}},\cdot),\mathcal{O}(\cdot)}\left(1^k, ek_{\mathsf{abm}}\right) \in \mathcal{T}_{\mathsf{los}} \cup \mathcal{T}_{\mathsf{inv}}\right], \tag{6}$$

is negligible, where $(ek_{\mathsf{abm}}, tk_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}(1^k)$, and the oracle $\mathcal{O}(\cdot)$ is queried by $\mathcal{A}$ with input t which returns the answers "lossy/invalid" and "injective" which indicate the type of t.

3. Definitions of New Primitives and IND-SO-PSA Security

In this section, for both LTF and ABM-LTF, we extend their definitions to the PSA setting. We introduce the primitive of lossy trapdoor function in the presence of PSA (LTF-PSA) and the primitive of all-but-many lossy trapdoor function in the presence of PSA (ABM-LTF-PSA). Further, we introduce the definition of newly-proposed IND-SO-PSA security.

3.1. LTF-PSA Primitive. A lossy trapdoor function in the presence of PSA LTF' with domain $\mathsf{LTF.Dom}'$ consists of the algorithms $(\mathsf{LTF.P}', \mathsf{LTF.IKg}', \mathsf{LTF.LKg}', \mathsf{LTF.EVAL}', \mathsf{LTF.Inv}')$ specified as follows.

(i) $\mathsf{LTF.P}'(1^k)$: this is the system parameter generation algorithm. On input 1^k, it outputs a system parameter π.

(ii) $\mathsf{LTF.IKg}'(1^k, \pi)$: this is the injective key generation algorithm. On input 1^k and π, it outputs an injective evaluation key ek' and a inversion key ik' for an injective function.

(iii) $\mathsf{LTF.LKg}'(1^k, \pi)$: this is the lossy key generation algorithm. On input 1^k and π, it outputs a lossy evaluation key lk' for a lossy function. In this case, there is no inversion key.

(iv) $\mathsf{LTF.Eval}'(ek', x)$: this is the evaluation algorithm. It takes the evaluation key ek' and a preimage $x \in \mathsf{LTF.Dom}'$ as input and returns an image $y = \mathsf{LTF.Eval}'(ek', x) \in \mathsf{LTF.Ran}'$, where $\mathsf{LTF.Dom}'$ and $\mathsf{LTF.Ran}'$ denote the domain and range of LTF', respectively.

(v) $\mathsf{LTF.Inv}'(ik', y)$: this is the inversion algorithm. It takes as input the inversion key ik' and an image $y \in \mathsf{LTF.Ran}'$ and outputs $x \in \mathsf{LTF.Dom}'$.

In the following, we will give the formal definitions of security requirements for LTF'. The correctness and lossiness are almost the same as that of LTF. In particular, we introduce the notions of indistinguishability under PSA (IND-PSA2) for LTF'.

Correctness. The above LTF' should satisfy correctness, i.e., $\mathsf{LTF.Inv}'(ik', \mathsf{LTF.Eval}'(ek', x)) = x$ for all $\pi \longleftarrow_{\$} \mathsf{LTF.P}'(1^k)$, $(ek', ik') \longleftarrow_{\$} \mathsf{LTF.IKg}'(1^k, \pi)$, and $x \in \mathsf{LTF.Dom}'$.

Indistinguishability under PSA. Let $\mathscr{A}$ be any PPT adversary against LTF'. LTF' holds indistinguishability under PSA if $\mathscr{A}$ cannot distinguish of the injective evaluation key ek' from the lossy evaluation key lk', i.e., $\mathscr{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{IND-PSA2}}_{\mathsf{LTF}',\mathscr{A}}(k) = \left|\Pr\left[\mathscr{A}\left(1^k, ek'\right) = 1\right] - \Pr\left[\mathscr{A}\left(1^k, lk'\right) = 1\right]\right|, \tag{7}$$

is negligible, where $(ek', ik') \longleftarrow_{\$} \mathsf{LTF.LKg}'(1^k, \pi^*)$ and $lk' \longleftarrow_{\$} \mathsf{LTF.IKg}'(1^k, \pi^*)$, where π^* is chosen by $\mathscr{A}$ rather than from the $\mathsf{LTF.P}'$ algorithm.

Lossiness. A lossy trapdoor function is l-lossy if for all possible $\pi \longleftarrow_{\$} \mathsf{LTF.P}'(1^k)$ and $lk' \longleftarrow_{\$} \mathsf{LTF.LKg}'(1^k, \pi)$, the image set $\mathsf{LTF.Eval}'(lk', \mathsf{LTF.Dom}')$ is of size at most $|\mathsf{LTF.Dom}'|/2^l$.

3.2. ABM-LTF-PSA Primitive. An all-but-many lossy trapdoor function in the presence of PSA $\mathsf{ABM.LTF}' = (\mathsf{ABM.P}', \mathsf{ABM.Kg}', \mathsf{ABM.Eval}', \mathsf{ABM.Inv}', \mathsf{ABM.LTg}')$ with domain $\mathsf{ABM.Dom}'$ consists of the following algorithms:

(i) $\mathsf{ABM.P}'(1^k)$: this is the system parameter generation algorithm. On input 1^k, it outputs a system parameter π.

(ii) $\mathsf{ABM.Kg}'(1^k, \pi)$: on input 1^k and π, it outputs an injective evaluation key ek'_{abm}, an inversion key ik'_{abm}, and a tag key tk'_{abm}. We note that it has the tag space $\mathscr{T}' = \mathscr{T}'_{\mathsf{inj}} \times \mathscr{T}'_{\mathsf{los}} \times \mathscr{T}'_{\mathsf{inv}}$. Similar to that of ABM.LTF, the term of each tag for $\mathsf{ABM.LTF}'$ is also as $t' = (t'_p, t'_a)$, where $t'_p \in \mathscr{T}'_p$ is the core part and $t'_a \in \mathscr{T}'_a$ is the auxiliary part.

(iii) $\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', x')$: it takes the evaluation key ek'_{abm}, the tag t', and a preimage $x' \in \mathsf{ABM.Dom}'$ as input and returns $y' \in \mathsf{ABM.Ran}'$, where $\mathsf{ABM.Dom}'$ and $\mathsf{ABM.Ran}'$ be the domain and range of the $\mathsf{ABM.LTF}'$.

(iv) $\mathsf{ABM.Inv}'(ik'_{\mathsf{abm}}, t', y')$: it takes an inversion key ik'_{abm}, a tag t', and an image y' as input and outputs $x' \in \mathsf{ABM.Dom}'$.

(v) $\mathsf{ABM.LTg}'(tk'_{\mathsf{abm}})$: it takes the tag key tk'_{abm} as input and outputs a lossy tag $t' = (t'_{\mathsf{p}}, t'_{\mathsf{a}})$.

We give the formal definitions of security requirements for the all-but-many lossy trapdoor function in the presence of PSA in the following. The correctness and lossiness are almost the same as that of ABM.LTF. In particular, we introduce the notions of indistinguishability under PSA (IND-PSA3) and evasiveness under PSA (EVA-PSA) for $\mathsf{ABM.LTF}'$.

Correctness. $\mathsf{ABM.LTF}'$ should satisfy the correctness. That is, the equation should hold, $\mathsf{ABM.Inv}'(\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', x'), ik'_{\mathsf{abm}}, t') = x'$, for all $\pi \longleftarrow_{\$} \mathsf{ABM.P}'(1^k)$, $(ek'_{\mathsf{abm}}, ik'_{\mathsf{abm}}, tk'_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}'(1^k, \pi)$, tag $t' \in \mathscr{T}'_{\mathsf{inj}}$, and $x' \in \mathsf{ABM.Dom}'$. This property is also referred to as invertibility.

Lossiness. $\mathsf{ABM.LTF}'$ is l-lossy if the size of the image set $\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', \mathsf{ABM.Dom}')$ is at most $|\mathsf{ABM.Dom}'|/2^l$, for all $\pi \longleftarrow_{\$} \mathsf{ABM.P}'(1^k)$, $(ek'_{\mathsf{abm}}, ik'_{\mathsf{abm}}, tk'_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}'(1^k, \pi)$, and lossy tag $t' \in \mathscr{T}'_{\mathsf{los}}$. l is called the lossiness of $\mathsf{ABM.LTF}'$.

Indistinguishability under PSA. Let $\mathscr{A}$ be any PPT adversary against $\mathsf{ABM.LTF}'$. We say $\mathsf{ABM.LTF}'$ holds the indistinguishability under PSA between multiple lossy tags and random tags if $\mathscr{A}$'s the advantage

$$\mathbf{Adv}^{\mathrm{IND-PSA3}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k) = \left|\Pr\left[\mathscr{A}^{\mathsf{ABM.LTg}'\left(tk'_{\mathsf{abm}}, \cdot\right)}\left(1^k, ek'_{\mathsf{abm}}\right) = 1\right]\Pr\left[\mathscr{A}^{\mathcal{O}_{\mathscr{T}'}(\cdot)}\left(1^k, ek'_{\mathsf{abm}}\right) = 1\right]\right|, \tag{8}$$

is negligible, where $(ek'_{\mathsf{abm}}, tk'_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}'(1^k, \pi^*)$, π^* is chosen by $\mathscr{A}$ rather than from the $\mathsf{ABM.P}'$ algorithm. The call to oracle $\mathsf{ABM.LTg}'(tk'_{\mathsf{abm}}, \cdot)$ returns a lossy tag, and $\mathcal{O}_{\mathscr{T}'}(\cdot)$ is a random oracle returns a uniform and independent tag from $\mathscr{T}'$.

Evasiveness under PSA. This property requires that it is hard to compute noninjective tags for any adversary, even though it has been given multiple lossy tags. Let $\mathscr{A}$ be any PPT adversary against $\mathsf{ABM.LTF}'$. $\mathsf{ABM.LTF}'$ holds the evasiveness under PSA if $\mathscr{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k) = \Pr\left[\mathscr{A}^{\mathsf{ABM.LTg}'(tk'_{\mathsf{abm}},\cdot),\mathcal{O}(\cdot)}\left(1^k, ek'_{\mathsf{abm}}\right) \in \mathscr{T}'_{\mathsf{los}} \cup \mathscr{T}'_{\mathsf{inv}}\right], \tag{9}$$

is negligible, where $(ek'_{\mathsf{abm}}, tk'_{\mathsf{abm}}) \longleftarrow_{\$} \mathsf{ABM.Kg}'(1^k, \pi^*)$, π^* is chosen by $\mathscr{A}$ itself. The oracle $\mathcal{O}(\cdot)$ is queried by $\mathscr{A}$ with input t' which returns the answers "lossy/invalid" and "injective" which indicate the type of t'.

3.3. IND-SO-PSA Security. To capture the security of the public-key encryption scheme under SOA and PSA, we formulate a new security notion called indistinguishability under selective opening attacks and parameter subversion attacks (IND-SO-PSA). IND-SO-PSA security captures indistinguishability under parameter subversion attacks, selective opening attacks, and chosen ciphertext attacks. In other words, it guarantees that even though the system parameter of the PKE scheme is chosen maliciously and the adversary is allowed to open a subset of the challenge ciphertexts (i.e., seeing the corresponding plaintexts and randomness used during the encryption), the remainder of the plaintexts should be indistinguishable from freshly re-sampled ones. In the following, we give the formal definition of IND-SO-PSA security.

Definition 7 (IND-SO-PSA security). Let $\mathscr{A}$ be an adversary attacking PKE. We say that PKE holds the indistinguishability under selective opening attacks, parameter subversion attacks, and chosen ciphertext attacks (IND-SO-PSA security), if $\mathscr{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k), = \left|\Pr\left[\mathbf{G}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k) = 1\right] - \frac{1}{2}\right|, \tag{10}$$

of winning the security game $\mathbf{G}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k)$ defined in Figure 5 is negligible.

At the beginning of the game $\mathbf{G}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k)$, the challenger $\mathscr{C}$ firstly tosses a coin to determine a challenge bit $b \in \{0,1\}$. Then, $\mathscr{A}$ firstly queries the INIT oracle with the public system parameter π^* chosen by himself. $\mathscr{C}$ runs the algorithm $\mathsf{PKE.Kg}$ to generate public-secret key pair (pk, sk). pk is given to the adversary $\mathscr{A}$. $\mathscr{A}$ will output a message distribution $\mathscr{D}$ as well as a re-sampling algorithm $\mathsf{ReSamp}_{\mathscr{D}}$ after adaptively querying the DEC oracle. Subsequently, $\mathscr{C}$ samples uniformly a message vector $\mathbf{m}_0$ from $\mathscr{D}$ and a randomness vector $\mathbf{r} = (r^{(i)})_{i\in[n]}$ from a randomness space $(\mathbb{R})^n$. $\mathscr{C}$ returns the challenge ciphertext $\mathbf{c} = (c^{(i)})_{i\in[n]}$ to $\mathscr{A}$. After receiving $\mathbf{c}$, $\mathscr{A}$ adaptively queries the DEC oracle and chooses an arbitrary subset $\mathscr{I} \subseteq [n]$ to indicate which ciphertexts he wants to open. $\mathscr{I}$ is given to $\mathscr{C}$. Notice that any element of the challenge ciphertext $\mathbf{c}$ is not allowed for DEC oracle. After receiving $\mathscr{I}$, $\mathscr{C}$ will re-sample a message $\mathbf{m}_1 \longleftarrow_{\$} \mathsf{Resamp}_{\mathscr{D}}(\mathbf{m}_{\mathscr{I}})$ conditioned on the $(m^{(i)})_{i\in\mathscr{I}}$ of $\mathbf{m}_0$. $\mathscr{C}$ will send $(\mathbf{m}_b, (m^{(i)}, r^{(i)})_{i\in[,\mathscr{I}]})$ to $\mathscr{A}$. Finally, $\mathscr{A}$ outputs a guess bit b', and if $b' = b$, we say $\mathscr{A}$ wins the game, and the game will output 1 in this case.

4. Construction of ABM-LTF-PSA

In this section, based on the embeddable group that was introduced by Auerbach et al. in [20], we construct an ABM-LTF-PSA and prove that it satisfies the IND-PSA3 as well as the EVA-PSA without the random oracle. The formal guarantees that a lossy tag is computationally indistinguishable from a random one, even when the adversary is given access to the lossy tag generation oracle and could choose the system parameter by himself. The latter prevents the adversary who could choose the system parameter by himself from generating lossy tags. We note that ABM-LTF-PSA can be seen as a generic LTF-PSA, which means that it could be used as LTF-PSA. On the contrary, LTF-PSA could be constructed with the same way as the construction of ABM-LTF-PSA. Thus, we will not reconsider the instantiation of LTF-PSA here.

4.1. ABM-LTF-PSA from Embeddable Group Family. Let $\mathsf{CHF} = (\mathsf{CHF.Kg}, \mathsf{CHF.Eval}, \mathsf{CHF.Equ})$ be a chameleon hash function, $\mathscr{H}_1: \mathbb{G} \longrightarrow \{0,1\}^*$ be a hash function, and $\mathsf{EG} = (\mathsf{EG.P}, \mathsf{EG.G}, \mathsf{EG.S}, \mathsf{EG.E}, \mathsf{EG.I})$ be an embeddable group family (we have recalled in the Section 2.2) [20]. We assume that the multiplication and addition operations hold on $\mathsf{EG.ES}$. In addition, we note that the algorithm $\mathsf{EG.E}$ of EG is deterministic as the instantiations of EG in [20]. An ABM-LTF-PSA based on the embeddable group family is a tuple of algorithms $\mathsf{ABM.LTF}' = (\mathsf{ABM.P}', \mathsf{ABM.Kg}', \mathsf{ABM.Eval}', \mathsf{ABM.Inv}', \mathsf{ABM.LTg}')$ that specified as following.

(i) $\mathsf{ABM.P}'(1^k)$: taking as input the security parameter k, it runs the $\mathsf{EG.P}(1^k)$ algorithm of EG to generate a system parameter π.

(ii) $\mathsf{ABM.Kg}'(1^k, \pi)$: the key generation algorithm does the following steps:

(1) Generate a public key pk_{chf} and the corresponding trapdoor td_{chf} of the CHF using

Game $\mathbf{G}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathcal{A}}(k)$

$b \leftarrow_{\$} \{0,1\}$
$(\mathcal{D}, \mathsf{ReSamp}_{\mathcal{D}}) \leftarrow_{\$} \mathcal{A}^{\mathrm{INIT},\mathrm{DEC}}(1^k)$
$\mathbf{m}_0 = (m^{(i)})_{i\in[n]} \leftarrow_{\$} \mathcal{D}$
$\mathbf{r} = (r^{(i)})_{i\in[n]} \leftarrow_{\$} (\mathbb{R})^n$
$\mathbf{c} = (c^{(i)})_{(i\in[n])} = (\mathsf{PKE.Enc}(1^k, \pi^*, pk, m^{(i)}; r^{(i)}))_{(i\in[n])}$
$\mathcal{I} \leftarrow_{\$} \mathcal{A}^{\mathrm{DEC}}(pk, \mathbf{c})$
$\mathbf{m}_1 \leftarrow_{\$} \mathsf{ReSamp}_{\mathcal{D}}(\mathbf{m}_{\mathcal{I}})$
$b' \leftarrow_{\$} \mathcal{A}^{\mathrm{DEC}}(\mathbf{m}_b, \{m^{(i)}, r^{(i)}\}_{(i\in\mathcal{I})})$
Return 1 if $b' = b$, otherwise return 0

$\underline{\mathrm{INIT}(\pi^*)}$
$(pk, sk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^k, \pi^*)$
Return pk

$\underline{\mathrm{DEC}(c')}$
If $(\exists\ i \in [n],$ s.t. $c' = c^{(i)})$
return $\perp$
Return $\mathsf{PKE.Dec}(1^k, \pi^*, sk, c')$

Figure 5: Game-defining IND-SO-PSA security of PKE.

CHF.Kg. This CHF will be used in the following to compute tags of ABM.LTF'.

(2) Generate an embedding cyclic group $\mathbb{G} = (\mathrm{G}, q, g)$ by running $\mathsf{EG.G}(1^k, \pi)$.

(3) Let $n = n(k)$. Sample $\alpha_{ij} \in \mathbb{Z}_q^{n\times n}$, $\beta_{ij} \in \mathbb{Z}_q^{n\times n}$, and $\delta_{ij} \in \mathbb{Z}_q^{n\times n}$ by running $\mathsf{EG.S}(1^k, \pi, \mathbb{G})$ for $1 \le i$ and $j \le n$, and set three matrixes $\mathbf{S}_1 = (\alpha_{ij})^{n\times n}$, $\mathbf{S}_2 = (\beta_{ij})^{n\times n}$, and $\mathbf{S}_3 = (\delta_{ij})^{n\times n}$.

(4) Set matrixes $\mathbf{A} = (g^{\alpha_{ij}})^{n\times n}$, $\mathbf{B} = (g^{\beta_{ij}})^{n\times n}$, and $\mathbf{C} = (g^{\delta_{ij}})^{n\times n}$ for g generated in Step 2.

(5) Compute σ_{ij} by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, \mathbf{A}_{ij})$, θ_{ij} by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, \mathbf{B}_{ij})$, and τ_{ij} by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, \mathbf{C}_{ij})$ for $1 \le i$ and $j \le n$. We have σ_{ij}, θ_{ij}, and $\tau_{ij} \in \mathsf{EG.ES}(k, \pi)$. Set matrixes $\mathbf{E}_1 = (\sigma_{ij})^{n\times n}$, $\mathbf{E}_2 = (\theta_{ij})^{n\times n}$, and $\mathbf{E}_3 = (\tau_{ij})^{n\times n}$.

(6) Select a pseudorandom function PRF: $\{0,1\}^* \times \{0,1\}^{l_2} \longrightarrow \{0,1\}^l$. Run $\mathsf{EG.S}(1^k, \pi, \mathbb{G})$ to get vector $\mathbf{e} = (e^{(1)}, e^{(2)}, \ldots, e^{(n)}) \in \mathbb{Z}_q^{1\times n}$. Set vector $\mathbf{k} = (\kappa^{(1)}, \kappa^{(2)}, \ldots, \kappa^{(n)}) \in \{0,1\}^*$ as the key of the PRF, where $\kappa^{(i)} = \mathscr{H}_1(g^{e^{(i)}})$. Run τ_{ij} to compute $\kappa'^{(i)} \in \mathsf{EG.ES}(k, \pi)$ and get vector $\mathbf{k}' = (\kappa'^{(1)}, \kappa'^{(2)}, \ldots, \kappa'^{(n)})$ as the hiding vector of $\mathbf{k}$. We note that $\mathbf{k}$ should be kept secret and $\mathbf{k}'$ could be public.

(7) Select an universal hash function $\mathrm{UHF}_{\mathbf{s}}$: $\{0,1\}^{l_1} \times \{0,1\}^{l_3} \longrightarrow \{0,1\}^{l_2}$ with key $\mathbf{s} \in \{0,1\}^{l_3}$. Sample a matrix $\mathbf{H}^{n\times n}$ from the embedding space $\mathsf{EG.ES}(k, \pi)$, and we use $\mathbf{h}_i$ to denote the ith row of $\mathbf{H}$.

(8) Set the public evaluation key $ek'_{\mathsf{abm}} = (\mathbf{C}, \mathbf{E}_1, \mathbf{E}_2, pk_{\mathsf{chf}}, \mathbf{k}', \mathbf{s}, \mathbf{H}, \mathbb{G})$, private inversion key $ik'_{\mathsf{abm}} = (\mathbf{C}, \mathbf{S}_1, \mathbf{S}_2, \mathbf{k}, \mathbf{s}, \mathbf{H}, \mathbb{G})$, and lossy tag generation key $tk'_{\mathsf{abm}} = (\mathbf{E}_3, \mathbf{k}, \mathbf{s}, \mathbf{H}, td_{\mathsf{chf}}, \mathbb{G})$.

(iii) Tags: a tag in $\mathsf{ABM.LTF}'$ is defined in the form of $t' = (t'_p, t'_a)$, where $t'_a \in \{0,1\}^*$ is an auxiliary part, $t'_p = (\mathbf{D}, R_{\mathsf{chf}}) \in (\mathsf{EG.ES}(k, \pi)^{n\times n} \times \mathscr{R}_{\mathsf{chf}})$ is a primary part, and $R_{\mathsf{chf}} \in \mathscr{R}_{\mathsf{chf}}$ is a randomness for CHF. Set the tag space as $\mathscr{T}' = (\mathsf{EG.ES}(k, \pi)^{n\times n} \times \mathscr{R}_{\mathsf{chf}}) \times \{0,1\}^*$. With a tag $t' = ((\mathbf{D}, R_{\mathsf{chf}}), t_a)$, we compute $\varphi = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}, t'_a); R_{\mathsf{chf}}) \in \{0,1\}^{l_1}$. Let matrix $\mathbf{U} = \mathbf{D}\mathbf{E}_3 - \mathbf{F}$ where $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$ and $\mathbf{f}_i = \mathrm{PRF}(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)) \cdot \mathbf{h}_i$ is the ith row of $\mathbf{F}$ for $1 \le i \le n$. We define

$$t' \in \begin{cases} \mathscr{T}'_{\mathsf{inj}}, & \mathsf{Rank}(\mathbf{U}) = n; \\ \mathscr{T}'_{\mathsf{los}}, & \mathbf{U} = 0; \\ \mathscr{T}'_{\mathsf{inv}}, & \mathsf{Rank}(\mathbf{U}) \ne n \text{ and } \mathsf{Rank}(\mathbf{U}) \ne 0. \end{cases} \tag{11}$$

We say that t' is an injective tag if the rank of matrix $\mathbf{U}$ is n, i.e., $\mathsf{Rank}(\mathbf{U}) = n$; else, if $\mathbf{U} = 0$, we say that t' is a lossy tag; particularly, if the rank of matrix $\mathbf{U}$ is out of the above two cases, i.e., $\mathsf{Rank}(\mathbf{U}) \ne n$ and $\mathsf{Rank}(\mathbf{U}) \ne 0$, we say that t' is an invalid tag.

(iv) $\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', \mathbf{x})$: with a tag $t' = ((\mathbf{D}, R_{\mathsf{chf}}), t'_a)$ and the public evaluation key $ek'_{\mathsf{abm}} = (\mathbf{C}, \mathbf{E}_1, \mathbf{E}_2, pk_{\mathsf{chf}}, \mathbf{k}', \mathbf{s}, \mathbf{H}, \mathbb{G})$, for a vector of preimage input $\mathbf{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(n)}) \in \mathbb{Z}_q^{1\times n}$, the algorithm does the following:

(1) For tag t', compute $\varphi = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}, t'_a); R_{\mathsf{chf}}) \in \{0,1\}^{l_1}$. With $\mathbf{k}'$, recover the key vector $\mathbf{k} = (\kappa^{(1)}, \kappa^{(2)}, \ldots, \kappa^{(n)})$ of PRF as $\kappa^{(i)} = \mathscr{H}_1(g^{e^{(i)}})$, where $g^{e^{(i)}}$ is computed by running $\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \kappa'^{(i)})$.

(2) For matrixes $\mathbf{E}_1 = (\sigma_{ij})^{n\times n}$ and $\mathbf{E}_2 = (\theta_{ij})^{n\times n}$, compute $g^{\alpha_{ij}}$ and $g^{\beta_{ij}}$ by running $\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \mathbf{E}_{1ij})$ and $\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \mathbf{E}_{2ij})$ correspondingly. Obtain the matrixes $\mathbf{A} = (g^{\alpha_{ij}})^{n\times n}$ and $\mathbf{B} = (g^{\beta_{ij}})^{n\times n}$, where $1 \le i$ and $j \le n$. Randomly select μ and υ from $\mathbb{Z}_q$ by running $\mathsf{EG.S}(1^k, \pi, \mathbb{G})$, and compute $s_1 = g^{\mu}$, $s_2 = g^{\upsilon}$, and $\mathbf{E} = \mathbf{A}^{\mu} + \mathbf{B}^{\upsilon}$, i.e., $\mathbf{E}_{ij} = \mathbf{A}_{ij}{}^{\mu} + \mathbf{B}_{ij}{}^{\upsilon} = g^{\alpha_{ij}\mu} + g^{\beta_{ij}\upsilon}$. Compute ϖ_1 by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, s_1)$ and ϖ_2 by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, s_2)$, where ϖ_1 and $\varpi_2 \in \mathsf{EG.ES}(k, \pi)$.

(3) Compute τ_{ij} by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, \mathbf{C}_{ij})$ to obtain matrix $\mathbf{E}_3 = (\tau_{ij})^{n\times n}$ for $1 \le i$ and $j \le n$. Concatenating matrixes $\mathbf{E}$ and $\mathbf{U}$ to obtain the matrix $\mathbf{M}^{n\times 2n}$. That is, $\mathbf{M} = (\mathbf{E}|\mathbf{U}) \in \mathsf{EG.ES}(k, \pi)^{n\times 2n}$, where $\mathbf{U} = \mathbf{D}\mathbf{E}_3 - \mathbf{F}$, $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots,$

$\mathbf{f}_n)^T$, and $\mathbf{f}_i = \mathrm{PRF}(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)) \cdot \mathbf{h}_i$ for $1 \le i \le n$.

(4) Compute $\overline{x}^{(i)}$ by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, g^{x^{(i)}})$ to set a vector $\overline{\mathbf{x}} = (\overline{x}^{(1)}, \overline{x}^{(2)}, \ldots, \overline{x}^{(n)})$, where $\overline{x}^{(i)} \in \mathsf{EG.ES}(k, \pi), 1 \le i \le n$.

(5) Compute the following equation to get the image vector $\mathbf{y} \in \mathsf{EG.ES}(k, \pi)^{1\times 2n}$:

$$\mathbf{y} = \sum_{i=1}^{n} \mathrm{PRF}\left(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)\right) \cdot \overline{\mathbf{x}} \cdot \mathbf{M}. \tag{12}$$

(6) Output $(\mathbf{y}, \varpi_1, \varpi_2)$.

(v) $\mathsf{ABM.Inv}'(ik'_{\mathsf{abm}}, t', \mathbf{y}, \varpi_1, \varpi_2)$: given an inversion key $ik'_{\mathsf{abm}} = (\mathbf{C}, \mathbf{S}_1, \mathbf{S}_2, \mathbf{k}, \mathbf{s}, \mathbf{H}, \mathbb{G})$, a tag $t' = ((\mathbf{D}, R_{\mathsf{chf}}), t'_a)$, ϖ_1 and ϖ_2, and image vector $\mathbf{y}$, the algorithm $\mathsf{ABM.Inv}'$ computes $\mathbf{x}$ as follows:

(1) Compute s_1 and s_2 by running $\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \varpi_1)$ and $\mathsf{EG.I}(1^k, \pi, \mathbb{G}, \varpi_2)$. Compute $\mathbf{E}'_1 = s_1^{\mathbf{S}_1}$ and $\mathbf{E}'_2 = s_2^{\mathbf{S}_2}$, which means that $\mathbf{E}'_{1\,ij} = g^{\mu\alpha_{ij}}$ and $\mathbf{E}'_{2\,ij} = g^{\upsilon\beta_{ij}}$. Obtain the matrix $\mathbf{E}$ by computing $\mathbf{E} = \mathbf{E}'_1 + \mathbf{E}'_2$, that is, $\mathbf{E}_{ij} = g^{\mu\,\alpha_{ij}} + g^{\upsilon\,\beta_{ij}}$. Compute $\mathbf{M} = (\mathbf{E}|\mathbf{U}) \in \mathsf{EG.ES}(k, \pi)^{n\times 2n}$ as the algorithm $\mathsf{ABM.Eval}'$.

(2) For tag $t' = ((\mathbf{D}, R_{\mathsf{chf}}), t'_a)$, it first computes $\varphi = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}, t'_a); R_{\mathsf{chf}}) \in \{0,1\}^{l_1}$.

(3) With ik'_{abm}, it computes the following equation to get $\overline{\mathbf{x}}$:

$$\overline{\mathbf{x}} = \frac{\mathbf{y} \cdot \mathbf{M}^{-1}}{\sum_{i=1}^{n} \mathrm{PRF}\left(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)\right)}, \tag{13}$$

where $(\mathbf{M}^{-1})^{2n\times n}$ is the inverse matrix of $\mathbf{M}$.

(4) Finally, it computes and outputs the vector of preimage $\mathbf{x}$ by running $\mathsf{EG.I}(1^k, \pi, \overline{\mathbf{x}})$.

(vi) $\mathsf{ABM.LTg}'(tk'_{\mathsf{abm}})$: the lossy tag generation algorithm is given the lossy tag generation key tk'_{abm}. It does the following steps to compute a lossy tag $t' = (t'_p, t'_a)$:

(1) Randomly choose a tag $\widetilde{t'} = ((\widetilde{\mathbf{D}}, \widetilde{R}_{\mathsf{chf}}), \widetilde{t}'_a) \in \mathscr{T}'$, and then, compute the value $\varphi = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\widetilde{\mathbf{D}}, \widetilde{t}'_a), \widetilde{R}_{\mathsf{chf}})$.

(2) Solve for an appropriate $n \times n$ matrix $\mathbf{D} \in \mathsf{EG.ES}(k, \pi)^{n\times n}$ such that the equation $\mathbf{DE}_3 = \mathbf{F}$ is valid, where $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$, $\mathbf{f}_i = \mathrm{PRF}(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)) \cdot \mathbf{h}_i$, for $1 \le i \le n$.

(3) Compute the value R_{chf} from the equation $R_{\mathsf{chf}} = \mathsf{CHF.Equ}(td_{\mathsf{chf}}, (\widetilde{\mathbf{D}}, \widetilde{t}'_a), (\mathbf{D}, t'_a); \widetilde{R}_{\mathsf{chf}})$, and output a tag $t' = ((\mathbf{D}, R_{\mathsf{chf}}), t'_a)$. It is easy to check that the tag t' output by this algorithm is indeed a lossy one.

Correctness. We note that the correctness of the above ABM-LTF-PSA instantiation holds due to the following equation:

$$\mathbf{E} = \mathbf{A}^{\mu} + \mathbf{B}^{\upsilon} = s_1^{\mathbf{S}_1} + s_2^{\mathbf{S}_2} = \mathbf{E}'_1 + \mathbf{E}'_2 = \mathbf{E}. \tag{14}$$

Lossiness. For any lossy tag $t' \in \mathscr{T}'_{\mathsf{los}}$ in the above ABM-LTF-PSA instantiation, we note that its lossiness $l = \log_2(q^n/(n^2|\mathsf{EG.ES}(k, \pi)|))$. When $t' \in \mathscr{T}'_{\mathsf{los}}$, the matrix $\mathbf{U}$ is equal to 0, i.e., $\mathbf{U} = 0$, we have at most n^2 possible values of $\mathbf{M} = (\mathbf{E}|\mathbf{U}) \in \mathsf{EG.ES}(k, \pi)^{n\times 2n}$. Thus, the image set $\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', \mathsf{ABM.Dom}')$ has size $n^2|\mathsf{EG.ES}(k, \pi)|$. Meanwhile, the size of domain $\mathsf{ABM.Dom}'$ of the $\mathsf{ABM.Eval}'$ is q^n because the preimage of $\mathsf{ABM.LTF}'$ is $\mathbf{x} \in \mathbb{Z}_q^{1\times n}$. According to the definition of lossiness for $\mathsf{ABM.LTF}'$, we could obtain the lossiness of the above construction is $l = \log_2(q^n/(n^2|\mathsf{EG.ES}(k, \pi)|))$.

4.2. Proofs of Security

Lemma 2. *Let n is polynomial in k and q is exponential in the security parameter k. A matrix $\mathbf{X}$ randomly sampled from $\mathbb{Z}_q^{n\times n}$ will have* $\mathsf{Rank}(\mathbf{X}) = n$ *with all but negligible probability in k.*

Proof. The probability of $\mathsf{Rank}(\mathbf{X}) = n$ is equal to the probability that n columns of $\mathbf{X}$ are linearly independent. This means that there are $q^n - 1$ possibilities (i.e., the zero vector is removed) for the sample of the first column vector. For the sample of the second column vector, there are $q^n - q$ possibilities (i.e., the q vectors linearly dependent on the first vector should be removed). The rest may be deduced by analogy. Thus, the probability of $\mathsf{Rank}(\mathbf{X}) = n$ is

$$\begin{aligned}\frac{(q^n-1)\cdot(q^n-q)\cdot\left(q^n-q^2\right)\cdots\left(q^n-q^{n-1}\right)}{q^n\cdot q^n\cdots q^n} &= \prod_{i=1}^{n}\left(1-\frac{1}{q^i}\right)\\ &= 1-\frac{1}{q}\mathsf{func}(k),\end{aligned} \tag{15}$$

where $\mathsf{func}(k)$ is a function satisfying that $0 < \mathsf{func}(k) \le n(1 + 1/q + \cdots + 1/q^n)$. Because q is exponential in k, $(1/q)\mathsf{func}(k)$ is negligible in k. So, we could finally obtain that this probability is overwhelming. □

Theorem 1 (indistinguishability). *Let* EG *be an efficiently-embeddable group and* CHF *be a chameleon hash function. Let* $\mathsf{ABM.LTF}'$ *be the all-but-many lossy trapdoor function that we built above. For any PPT adversary $\mathscr{A}$ with advantage* $\mathbf{Adv}^{IND-PSA3}_{\mathsf{ABM.LTF}',\mathscr{A}}$, *there exists adversaries $\mathscr{A}_1$ and $\mathscr{A}_2$ such that*

$$\mathbf{Adv}^{\mathrm{IND-PSA3}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k) \le \left(2n^2 + n\right)\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}_1}(k) + n\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k). \tag{16}$$

Proof. We proceed with the proof by describing a sequence of games, i.e., **Game** 1 to **Game** 3. Let S_i be the event that the output of adversary $\mathscr{A}$ is 1 in **Game** i. In **Game** 1, all algorithms work exactly the same as the real scheme. $\mathscr{A}$ interacts with $\mathsf{ABM.LTg}'(tk'_{\mathsf{abm}}, \cdot)$, and the algorithm outputs lossy tags for adversary $\mathscr{A}$. We emphasis that the system parameter π^* involved in this game is chosen by $\mathscr{A}$. So, we obtain

$$\Pr[S_1] = \Pr\left[\mathscr{A}^{\mathsf{ABM.LTg}'(tk_{\mathsf{abm}}, \cdot)}\left(1^k, ek'_{\mathsf{abm}}\right) = 1\right]. \tag{17}$$

In **Game** 2, we modify the way of generating the public evaluation key $ek'_{\mathsf{abm}} = (\mathbf{C}, \mathbf{E}_1, \mathbf{E}_2, pk_{\mathsf{chf}}, \mathbf{k}', \mathbf{s}, \mathbf{H}, \mathbb{G})$. Specially, we choose the matrixes $\mathbf{E}_1$ and $\mathbf{E}_2$ and the vector $\mathbf{k}'$ uniformly at random from $\mathsf{EG.ES}(k, \pi^*)$. Since $\mathbf{E}_1$, $\mathbf{E}_2$, and $\mathbf{k}'$ do not influence the output distribution of $\mathsf{ABM.LTg}'$, by the EPR-PSA assumption of EG, this modification will not be noticed by $\mathscr{A}$. So, for an EPR-PSA adversary $\mathscr{A}_1$, we obtain

$$\left|\Pr[S_2] - \Pr[S_1]\right| \leq \left(2n^2 + n\right)\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}_1}(k). \tag{18}$$

In **Game** 3, the proceeding of algorithm $\mathsf{ABM.LTg}'$ is changed. Particularly, in the 2nd step of algorithm $\mathsf{ABM.LTg}'$, we randomly select a matrix $\mathbf{F} \longleftarrow_{\$} \mathsf{EG.ES}(k, \pi^*)^{n\times n}$ instead of computing a matrix $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$, in which $\mathbf{f}_i = \mathrm{PRF}(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)) \cdot \mathbf{h}_i$ with uniformly random φ for $1 \leq i \leq n$, as in **Game** 2. According to the PRF security (defined in Figure 1) of the pseudorandom function, for a PPT adversary $\mathscr{A}_2$ against PRF, we could obtain

$$\left|\Pr[S_3] - \Pr[S_2]\right| \leq n\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k). \tag{19}$$

On the contrary, in **Game** 3, the matrix $\mathbf{D}$ of t'_{p} is the solution of equation $\mathbf{DE}_3 = \mathbf{F} \mod q$ with random $\mathbf{F}$. If the rank of matrix $\mathbf{E}_3$ is n, then $\mathbf{D}$ will be random. According to Lemma 2 (same result holds for $\mathsf{EG.ES}(k, \pi)^{n\times n}$), we indeed have $\mathsf{Rank}(\mathbf{E}_3) = n$. This means that $\mathbf{D}$ is random. Thus, we could obtain result that the primary part $t'_{\mathsf{p}} = (\mathbf{D}, R_{\mathsf{chf}})$ of tag t' is random. Therefore, all tags generated in **Game** 3 are random tags. So, we obtain

$$\Pr[S_3] = \Pr\left[\mathscr{A}^{\mathcal{O}_{\mathcal{T}'}(\cdot)}\left(1^k, ek'_{\mathsf{abm}}\right) = 1\right]. \tag{20}$$

Summing up, we find that the advantage $\mathbf{Adv}^{\mathrm{IND-PSA3}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k)$ of adversary $\mathscr{A}$ is

$$\left|\Pr\left[\mathscr{A}^{\mathsf{ABM.LTg}'\left(tk'_{\mathsf{abm}},\cdot\right)}\left(1^k, ek'_{\mathsf{abm}}\right) = 1\right] - \Pr\left[\mathscr{A}^{\mathcal{O}_{\mathcal{T}'}(\cdot)}\left(1^k, ek'_{\mathsf{abm}}\right) = 1\right]\right| \leq \left(2n^2 + n\right)\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}_1}(k) + n\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k). \tag{21}$$

□

Theorem 2 (evasiveness). *Let* EG *be the efficiently-embeddable group and* CHF *be a chameleon hash function. Let* $\mathsf{ABM.LTF}'$ *be the all-but-many lossy trapdoor function that we built above. For any PPT adversary* $\mathscr{A}$ *with advantage* $\mathbf{Adv}^{EVA-PSA}_{\mathsf{ABM.LTF}',\mathscr{A}}$, *there exist adversaries* $\mathscr{A}_1$, $\mathscr{A}_2$, *and* $\mathscr{A}_3$ *such that*

$$\mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k) \leqslant \left(2n^2 + n\right)\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}_1}(k) + n\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k) + \mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}_3}(k) + \mathbf{neg}(k). \tag{22}$$

Proof. We proceed with the proof by describing a sequence of games, **Game** 1 to **Game** 4. Let S_{i} be the event that the output of adversary $\mathscr{A}$ is a lossy or invalid tag in **Game** i. We consider the following two types of tags output by $\mathscr{A}$. Specifically, for a tag $t' = ((\mathbf{D}^*, R^*_{\mathsf{chf}}), t'^*_a)$,

(i) Type 1: if $\varphi^* = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}^*, t'^*_a); R^*_{\mathsf{chf}})$ is also the chameleon hash output of some previously generated tag, then we refer to $t' = ((\mathbf{D}^*, R^*_{\mathsf{chf}}), t'^*_a)$ as a Type 1 tag.

(ii) Type 2: if $\varphi^* = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}^*, t'^*_a); R^*_{\mathsf{chf}})$ is not the chameleon hash output of any previously generated tag, then we refer to $t' = ((\mathbf{D}^*, R^*_{\mathsf{chf}}), t'^*_a)$ as a Type 2 tag.

We make an assumption, without loss of generality, that $\mathscr{A}$ obtains $n = n(k)$ lossy tags by querying the $\mathsf{ABM.LTg}'(tk'_{\mathsf{abm}})$ oracle. Let $(t'_i)_{i\in[n]} = ((\mathbf{D}_i, R_{\mathsf{chf}\,i}), t'_{ai})_{i\in[n]}$ denote these lossy tags. Then, $\mathscr{A}$ adaptively comes up with $n' = n'(k)$ tags $(t'^*_i)_{i\in[n']} = ((\mathbf{D}^*_i, R^*_{\mathsf{chf}i}), t'^*_{ai})_{i\in[n']}$ by querying the oracle $\mathcal{O}_{\mathcal{T}'}$ and gets answers "invalid" or "injective."

In **Game** 1, all algorithms work exactly the same as the real scheme. In particular, we note that the system parameter π^* involved in this game is chosen by $\mathscr{A}$. So, we have

$$\mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k) = \Pr[S_1]. \tag{23}$$

In **Game** 2, we modify the way of generating the public evaluation key $ek'_{\mathsf{abm}} = (\mathbf{C}, \mathbf{E}_1, \mathbf{E}_2, pk_{\mathsf{chf}}, \mathbf{k}', \mathbf{s}, \mathbf{H}, \mathbb{G})$. Specially, we choose the matrixes $\mathbf{E}_1$ and $\mathbf{E}_2$ and the vector $\mathbf{k}'$ uniformly at random from $\mathsf{EG.ES}(k, \pi^*)$. Since $\mathbf{E}_1$, $\mathbf{E}_2$, and $\mathbf{k}'$ do not influence the output distribution of $\mathsf{ABM.LTg}'$, by the EPR-PSA assumption of EG, this modification will not be noticed by $\mathscr{A}$. So, for an EPR-PSA adversary $\mathscr{A}_1$, we obtain

$$\left|\Pr[S_2] - \Pr[S_1]\right| \leq \left(2n^2 + n\right)\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}_1}(k). \tag{24}$$

In **Game** 3, we change the algorithm $\mathsf{ABM.LTg}'$. In particular, in the 2nd step of $\mathsf{ABM.LTg}'$, we randomly select a matrix $\mathbf{F} \longleftarrow_{\$} \mathsf{EG.ES}(k, \pi^*)^{n\times n}$ for any φ, instead of computing a matrix $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$, in which $\mathbf{f}_i = \mathrm{PRF}(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi)) \cdot \mathbf{h}_i$. For all queries $(t'^*_i)_{i\in[n']}$ to the $\mathcal{O}_{\mathcal{T}'}$ oracle, we return the answer "injective." In the following context, we will use the mathematical methods of induction to prove that the distinguishability of **Game** 2 and **Game** 3 implies that there is an adversary $\mathscr{A}_2$ against the pseudorandom function PRF.

(i) For the base step, suppose $n' = 1$ (the case $n' = 0$ is vacuous). In **Game** 2, $\mathcal{O}_{\mathcal{T}'}$ honestly computes $\mathbf{U} = \mathbf{DE}_3 - \mathbf{F}$, where $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$ and $\mathbf{f}_i = \mathrm{PRF}(\kappa^{(i)}, \mathrm{UHF}_{\mathbf{s}}(\varphi^*_1)) \cdot \mathbf{h}_i$, for $1 \leq i \leq n$. Then, it returns the answer "injective." The distributions of

$\mathbf{U}$ in **Game** 2 and **Game** 3 are correspondingly $\{\mathrm{PRF}(\kappa^{(j)}, \mathrm{UHF}_{\mathbf{s}}(\varphi_i^*))\}_{j\in[n]} \cup \{\mathrm{PRF}(\kappa^{(j)}, \mathrm{UHF}_{\mathbf{s}}(\varphi_1^*)\}$ and $\{\mathbf{F}_i \longleftarrow_{\$} \mathsf{EG.ES}(k, \pi^*)^{n\times n}\}_{i\in[n]} \cup \{\mathbf{F} \longleftarrow_{\$} \mathsf{EG.ES}(k\pi^*)^{n\times n}\}$. Hence, these two distributions are computationally distinguishable, according to the security of PRF, with the probability at most $n \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k)$ for a PRF adversary $\mathscr{A}_2$. In addition, since for φ_1^* from $t_i'^*$ in **Game** 3, the matrix $\mathbf{F}$ is random, so is the matrix $\mathbf{U}$, and the adversary is always given answers "injective" except with the negligible probability ε. Thus, it holds that $|\Pr[S_3] - \Pr[S_2]| \le n \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k) + n' \cdot \varepsilon$ when $n' = 1$.

(ii) Suppose that the above result $|\Pr[S_3] - \Pr[S_2]| \le n \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k) + n' \cdot \varepsilon$ holds for $\eta = n' - 1 \ge 1$. Accordingly, in **Game** 3, we simply answer "injective" without even looking at the query φ_i^*.

(iii) Now, we consider the case of n'. In **Game** 3, for tags $\{t_i'^*\}_{i\in[\eta+1]}$, we only focus on the n'th query tag $\{t_{\eta+1}'^*\}$. In **Game** 2, we honestly derived the "injective" answers for the previous η queries, and the n'th answer is computed as $\mathbf{U} = \mathbf{DE}_3 - \mathbf{F}$. Since PRF in **Game** 3 is only evaluated on $\{\mathrm{UHF}_{\mathbf{s}}(\varphi_i^*)_{i\in[n]}\} \cup \{\mathrm{UHF}_{\mathbf{s}}(\varphi_{\eta+1}^*)\}$ and since in **Game** 2 by inductive hypothesis the answers were all "injective," the inductive hypothesis still holds, i.e., $|\Pr[S_3] - \Pr[S_2]| \le n \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k) + (\eta + 1) \cdot \varepsilon$. As a result, for all $n' = n'(k)$, we set $\mathbf{neg}_1(k) = n' \cdot \varepsilon$, and it holds that

$$\left|\Pr[S_3] - \Pr[S_2]\right| \le n\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k) + \mathbf{neg}_1(k). \tag{25}$$

We note that tags $(t_i'^*)_{i\in[n']}$ in **Game** 3 are distributed as random tags.

In **Game** 4, the trapdoor td'_{chf} of chameleon hash function CHF is not available. Each primary part of tag t' is sampled uniformly at random, i.e., $(\mathbf{D}, R_{\mathsf{chf}}) \longleftarrow_{\$} (\mathsf{EG.ES}(k, \pi^*)^{n\times n} \times \mathscr{R}_{\mathsf{chf}})$. Thus, it holds that $\Pr[S_4] = \Pr[S_3]$. In addition, for any fresh φ, we randomly select a matrix $\mathbf{F} \longleftarrow_{\$} \mathsf{EG.ES}(k, \pi^*)^{n\times n}$, that is, there is not an adversary that will output Type 2 tags with at most a negligible probability $\mathbf{neg}_2(k)$, so we could get $\Pr[S_4, 2] \le \mathbf{neg}_2(k)$. The Type 1 outputs will break the collision resistance of the chameleon hash function CHF, so we have $\Pr[S_4, 1] \le \mathbf{Adv}^{\mathrm{COLL}}_{\mathscr{A}_3}(k)$ for some adversary $\mathscr{A}_3$. Therefore, we could obtain $\Pr[S_4] \le \mathbf{neg}_2(k) + \mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}_3}(k)$.

Summing up, let $\mathbf{neg}(k) = \mathbf{neg}_1(k) + \mathbf{neg}_2(k)$, and we find that the advantage of adversary $\mathscr{A}$ is

$$\mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}}(k) \le \left(2n^2 + n\right)\mathbf{Adv}^{\mathrm{EPR-PSA}}_{\mathsf{EG},\mathscr{A}_1}(k) + n\mathbf{Adv}^{\mathrm{PRF}}_{\mathscr{A}_2}(k) + \mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}_3}(k) + \mathbf{neg}(k). \tag{26}$$

□

5. Construction of IND-SO-PSA Secure PKE Scheme

Taking the constructions in [32–34, 39] as guidances, we propose an IND-SO-PSA secure PKE scheme with the newly-built ABM-LTF-PSA and LTF-PSA based on an efficiently-embeddable group family. Let $\mathsf{ABM.LTF}' = (\mathsf{ABM.P}', \mathsf{ABM.Kg}', \mathsf{ABM.Eval}', \mathsf{ABM.Inv}', \mathsf{ABM.LTg}')$ be an all-but-many lossy trapdoor function in the presence of PSA we constructed in Section 4, $\mathsf{LTF}' = (\mathsf{LTF.P}', \mathsf{LTF.IKg}', \mathsf{LTF.LKg}', \mathsf{LTF.Eval}', \mathsf{LTF.Inv}')$ be a lossy trapdoor function in the presence of PSA, and $\mathsf{EG} = (\mathsf{EG.P}, \mathsf{EG.G}, \mathsf{EG.S}, \mathsf{EG.EE}, \mathsf{EG.I})$ be the underlying embeddable group family of $\mathsf{ABM.LTF}'$ and LTF'. Let $\mathscr{H}_2$: $\mathsf{EG.ES}(k, \pi) \longrightarrow \{0,1\}^{l_1}$ be a hash function and $\mathsf{CHF} = (\mathsf{CHF.Kg}, \mathsf{CHF.Eval}, \mathsf{CHF.Equ})$ be a chameleon hash function.

The scheme $\mathsf{PKE} = (\mathsf{PKE.P}, \mathsf{PKE.Kg}, \mathsf{PKE.Enc}, \mathsf{PKE.DEC})$ will be built as follows.

(i) $\mathsf{PKE.P}(1^k)$: taking as input the security parameter k, it runs the $\mathsf{EG.P}(1^k)$ algorithm of EG to generate a system parameter π. We note that here π is also used as the system parameter of $\mathsf{ABM.LTF}'$ and LTF' in this construction.

(ii) $\mathsf{PKE.Kg}(1^k, \pi)$: the key generation algorithm does the following steps:

(1) Run algorithm $\mathsf{ABM.Kg}'(1^k, \pi)$ to get $(ek'_{\mathsf{abm}}, ik'_{\mathsf{abm}}, tk'_{\mathsf{abm}})$. Have the same embeddable group $\mathbb{G}$ with generator g and order q and the same public key pk_{chf} of CHF as generated in the step 1 and step 2 of algorithm $\mathsf{ABM.Kg}'$.

(2) Run algorithm $\mathsf{LTF.IKg}'(1^k, \pi)$ to get (ek', ik'). Set the public key as $pk = (ek'_{\mathsf{abm}}, ek', pk_{\mathsf{chf}})$ and the secret key as $sk = (ik'_{\mathsf{abm}}, ik')$. It finally outputs a pair of public and secret keys (pk, sk).

(iii) $\mathsf{PKE.Enc}(pk, m)$: with the public key $pk = (ek'_{\mathsf{abm}}, ek', pk_{\mathsf{chf}})$ and the message $m \in \{0,1\}^{l_2}$, the encryption algorithm $\mathsf{PKE.Enc}(pk, m)$ does the following:

(1) Randomly select a vector $\mathbf{r} = (r^{(1)}, r^{(2)}, \ldots, r^{(n)})$ from $\mathbb{Z}_q^{1\times n}$ by running $\mathsf{EG.S}(1^k, \pi, \mathbb{G})$.

(2) Compute $\overline{r}^{(i)}$ by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, g^{r^{(i)}})$ for $1 \le i \le n$, and set vector $\overline{\mathbf{r}} = (\overline{r}^{(1)}, \overline{r}^{(2)}, \ldots, \overline{r}^{(n)})$. That is, we have $\overline{r}^{(i)} \in \mathsf{EG.ES}(k, \pi)$.

(3) Randomly select an universal hash function $\mathrm{UHF}_{\mathbf{w}}$: $\{0,1\}^{l_1} \times \{0,1\}^{l_3} \longrightarrow \{0,1\}^{l_2}$ with index key $\mathbf{w} \in \{0,1\}^{l_3}$. Compute $\Theta = \mathscr{H}_2(\overline{\mathbf{r}})$, $\rho = \mathrm{UHF}_{\mathbf{w}}(\Theta) \oplus m$, and $\mathbf{y}_1 = \mathsf{LTF.Eval}'(ek', \mathbf{r})$.

We note that $\mathrm{UHF}_{\mathbf{w}}$ is not the same one used in the construction of ABM.LTF′.

(4) Set $t' = (t'_p, t'_a)$ for randomly-sampled $t'_p = (\mathbf{D}, R_{\mathsf{chf}})$ and $t'_a = (\mathrm{UHF}_{\mathbf{w}}, \rho, \mathbf{r})$; then, compute $\varphi = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}, t'_a, \mathbf{y}_1); R_{\mathsf{chf}})$. We note that CHF is the same one used in the construction of ABM.LTF′.

(5) Use φ as the input of the step 3 of the $\mathsf{ABM.Eval}'$ algorithm, and compute $\mathbf{y}_2 = \mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', \mathbf{r})$. Notice that $\mathbf{y}_2$ is a tuple $(\hat{\mathbf{y}}, \varpi_1, \varpi_2)$ as described in the step 6 of $\mathsf{ABM.Eval}'$. Output the ciphertext $\mathrm{C} = (\mathbf{y}_1, \mathbf{y}_2, t'_p, \mathrm{UHF}_{\mathbf{w}}, \varphi, \rho)$. Note that the randomness of this encryption algorithm contains $\mathbf{w}$ and t', and all components in t' are public except $\mathbf{r}$.

(iv) $\mathsf{PKE.Dec}(C, sk)$: after receiving the ciphertext $\mathrm{C} = (\mathbf{y}_1, \mathbf{y}_2, t'_p, \mathrm{UHF}_{\mathbf{w}}, \varphi, \rho)$, the decryption algorithm does the following:

(1) Run algorithm $\mathsf{LTF.Inv}'(ik', \mathbf{y}_1)$ to obtain the vector $\mathbf{r} = (r^{(1)}, r^{(2)}, \ldots, r^{(n)})$.

(2) Run algorithm $\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', \mathbf{r})$ to compute $\mathbf{y}'_2$ with the tag $t' = (t'_p, t'_a)$, where $t'_p = (\mathbf{D}, R_{\mathsf{chf}})$ and $t'_a = (\mathrm{UHF}_{\mathbf{w}}, \rho, \mathbf{r})$. Check that whether the equation $\mathbf{y}'_2 = \mathbf{y}_2$ holds. If not, then return $\perp$; otherwise, go to the next step.

(3) Compute $\varphi' = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}, t'_a, \mathbf{y}_1); R_{\mathsf{chf}})$. If $\varphi = \varphi'$, reject this ciphertext; otherwise, go to the next step.

(4) Compute $\bar{r}^{(i)}$ by running $\mathsf{EG.E}(1^k, \pi, \mathbb{G}, g^{r^{(i)}})$ for $1 \le i \le n$ to obtain vector $\bar{\mathbf{r}} = (\bar{r}^{(1)}, \bar{r}^{(2)}, \ldots, \bar{r}^{(n)})$. Compute $\Theta = \mathscr{H}_2(\bar{\mathbf{r}})$, and recover the message from $m = \mathrm{UHF}_{\mathbf{w}}(\Theta) \oplus \rho$.

6. Security Analysis

In this section, we analyse the IND-SO-PSA security of our PKE scheme constructed in Section 5. In particular, we prove that our scheme can satisfy the IND-SO-PSA security which is defined in Section 3.3. We will give more detailed proofs of security in the following. We note that the proof is processed without random oracle.

Theorem 3 (IND-SO-PSA). *Let* ABM.LTF′ *be the newly proposed all-but-many lossy trapdoor function from the embeddable group family scheme we built in Section 4,* LTF′ *be a lossy trapdoor function in the presence of PSA, and* CHF *be a chameleon hash function. For any PPT adversary* $\mathscr{A}$ *with advantage* $\mathbf{Adv}^{IND-SO-PSA}_{\mathsf{PKE},\mathscr{A}}(k)$*, there exist adversaries* $\mathscr{A}_1$, $\mathscr{A}_2$, $\mathscr{A}_3$, *and* $\mathscr{A}_4$ *such that*

$$\mathbf{Adv}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k) \leqslant \mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}_1}(k) + \mathbf{Adv}^{\mathrm{IND-PSA3}}_{\mathsf{ABM.LTF}',\mathscr{A}_2}(k) + \mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}_3}(k) + \mathbf{Adv}^{\mathrm{IND-PSA2}}_{\mathsf{LTF}',\mathscr{A}_4}(k) + \mathbf{neg}(k). \tag{27}$$

Proof. We firstly revisit the definition of the IND-SO-PSA security (shown in Figure 5). Suppose that we have n challenge ciphertexts, and the ith challenge ciphertext is denoted as $\mathrm{C}^{(i)} = (\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, t'^{(i)}_{\mathrm{p}}, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \varphi^{(i)}, \rho^{(i)})$ in which $t'^{(i)}_{\mathrm{p}} = (\mathbf{D}^{(i)}, R^{(i)}_{\mathsf{chf}})$. We proceed with the proof by describing a sequence of games, **Game** 1 to **Game** 6. Let S_i be the event that the output of adversary $\mathscr{A}$ is 1 in **Game** i. In **Game** 1, all algorithms work exactly the same as the game $\mathbf{G}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k)$. We note that the system parameter π^* involved in this game is chosen by $\mathscr{A}$, as shown in Figure 5. By the definition of IND-SO-PSA security, we have

$$\mathbf{Adv}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k) = \left|\Pr[S_1] - \frac{1}{2}\right|. \tag{28}$$

In **Game** 2, we note that the decryption queries whose element $\varphi^{(i)}$ has already existed in one of the challenge ciphertexts were rejected. That is, if adversary $\mathscr{A}$ queries decryption oracle with a ciphertext $\mathrm{C} = (\mathbf{y}_1, \mathbf{y}_2, t'_{\mathrm{p}}, \mathrm{UHF}_{\mathbf{w}}, \varphi^{(i)}, \rho)$ in which $\varphi^{(i)}$ is appeared in some $\mathrm{C}^{(i)} = (\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, t'^{(i)}_{\mathrm{p}}, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \varphi^{(i)}, \rho^{(i)})$, then it will be rejected, or the collision resistant property of CHF will be broken. We note that the only element that is not a part of the inputs of CHF in ciphertext C is $\mathbf{y}_2$. Let $t'_a = (\mathrm{UHF}_{\mathbf{w}}, \rho, \mathbf{r})$ and $t'^{(i)}_a = (\mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)}, \mathbf{r}^{(i)})$. In the following, we consider three cases:

(i) $\mathbf{y}_2 = \mathbf{y}_2^{(i)}$ and $(\mathbf{y}_1, t'_p, \mathrm{UHF}_{\mathbf{w}}, \rho) = (\mathbf{y}_1^{(i)}, t'^{(i)}_{\mathrm{p}}, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)})$: this is the case that the query ciphertext is precisely the ith challenge ciphertext. Thus, this decryption query should be rejected.

(ii) $\mathbf{y}_2 = \mathbf{y}_2^{(i)}$ and $(\mathbf{y}_1, t'_p, \mathrm{UHF}_{\mathbf{w}}, \rho) \neq (\mathbf{y}_1^{(i)}, t'^{(i)}_p, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)})$: to recover the encrypted message, the decryption algorithm has to verify the correctness of the tag of ABM.LTF′ by executing the equivocation algorithm of CHF. This would happen only if the following equation holds: $\mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}, t'_a, \mathbf{y}_1); R_{\mathsf{chf}}) = \mathsf{CHF.Eval}(pk_{\mathsf{chf}}, (\mathbf{D}^{(i)}, t^{(i)}_{\mathrm{a}}, \mathbf{y}_1^{(i)}); R^{(i)}_{\mathsf{chf}})$, which means that there exist a collision of chameleon hash function CHF.

(iii) $\mathbf{y}_2 \neq \mathbf{y}_2^{(i)}$: recall that $\varphi = \varphi^{(i)}$ is computed by an injective tag. We will consider the following cases. (1) $\mathbf{y}_1 = \mathbf{y}_1^{(i)}$ and $(t'_p, \mathrm{UHF}_{\mathbf{w}}, \rho) = (t'^{(i)}_p, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)})$: the output of this query in step 1 is $\mathbf{r}$. It is obvious that $\mathbf{r} = \mathbf{r}^{(i)}$. Thus, we can get $t'_a = t'^{(i)}_a$. In this case, there is indeed a collision $((\mathbf{D}, t'_a, \mathbf{y}_1); R_{\mathsf{chf}})$ and $((\mathbf{D}^{(i)}, t'^{(i)}_a, \mathbf{y}_1^{(i)}); R^{(i)}_{\mathsf{chf}})$ happening to CHF. (2) $\mathbf{y}_1 = \mathbf{y}_1^{(i)}$ and $(t'_p, \mathrm{UHF}_{\mathbf{w}}, \rho) \neq (t'^{(i)}_p, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)})$: similar to the above case, we must have $\mathbf{r} = \mathbf{r}^{(i)}$ and $t'_a = t'^{(i)}_a$. Then, the query will be rejected unless a collision happens to CHF. (3) $\mathbf{y}_1 \neq \mathbf{y}_1^{(i)}$ and $(t'_p, \mathrm{UHF}_{\mathbf{w}}, \rho) = (t'^{(i)}_p, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)})$: the output of this decryption query in the step 1 is $\mathbf{r}$, and we must

have $\mathbf{r} \neq \mathbf{r}^{(i)}$, and thus, $t_a' \neq t_a'^{(i)}$. Then, the query will be rejected unless an exact collision $((\mathbf{D}, t_a', \mathbf{y}_1); R_{\mathsf{chf}})$ and $((\mathbf{D}^{(i)}, t_a'^{(i)}, \mathbf{y}_1^{(i)}); R_{\mathsf{chf}}^{(i)})$ happens to CHF. (4) $\mathbf{y}_1 \neq \mathbf{y}_1^{(i)}$ and $(t_p', \mathrm{UHF}_{\mathbf{w}}, \rho) \neq (t_p'^{(i)}, \mathrm{UHF}_{\mathbf{w}^{(i)}}, \rho^{(i)})$: similar to the above case, we must have $\mathbf{r} \neq \mathbf{r}^{(i)}$ and $t_a' \neq t_a'^{(i)}$. Then, the query will be rejected unless a collision happens.

Therefore, **Game** 1 to **Game** 2 behaves the same unless the collision resistancy of the chameleon hashing is broken. Thus, for some adversary $\mathscr{A}_1$, it holds that

$$\left|\Pr[S_2] - \Pr[S_1]\right| \leq \mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}_1}(k). \tag{29}$$

In **Game** 3, we generate the lossy tags by running the $\mathsf{ABM.LTg}'$ for all challenge ciphertexts $\mathrm{C}^{(i)}$ for $i \in [n]$. Note that we allow the decryption query made with lossy tags in which $\varphi \neq \varphi^{(i)}$. According to the indistinguishability under PSA of the $\mathsf{ABM.LTF}'$, for some adversary $\mathscr{A}_2$, we have

$$\left|\Pr[S_3] - \Pr[S_2]\right| \leq \mathbf{Adv}^{\mathrm{IND-PSA3}}_{\mathsf{ABM.LTF}',\mathscr{A}_2}(k). \tag{30}$$

Recall that, in **Game** 3, we firstly run the algorithm $\mathsf{LTF.Inv}'(ik', \mathbf{y}_1)$ to obtain the vector $\mathbf{r}$. Then, we check that $\mathsf{ABM.Eval}'(ek'_{\mathsf{abm}}, t', \mathbf{r}) = \mathbf{y}_2$. If it is not, then reject. Now, in **Game** 4, with ik'_{abm}, we run the algorithm $\mathsf{ABM.Inv}'$ to recover $\mathbf{r}$. Particularly, in the step 3 of $\mathsf{ABM.Inv}'$, the value φ is the one contained in the ciphertext for the decryption query. According to the correctness of $\mathsf{ABM.LTF}'$ and LTF', **Game** 4 and **Game** 3 could have the same result unless the component φ of decryption query belonging to one of the challenge ciphertexts or the queries are made with lossy or invalid tags. We have to consider the previous situtation in **Game** 3. The latter would happen unless the evasiveness under the PSA property of $\mathsf{ABM.LTF}'$ is broken. Thus, for some adversary $\mathscr{A}_3$, we get

$$\left|\Pr[S_4] - \Pr[S_3]\right| \leq \mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}_3}(k). \tag{31}$$

In **Game** 5, a lossy evaluation key is generated for LTF'. So, for some adversary $\mathscr{A}_4$, we have

$$\left|\Pr[S_5] - \Pr[S_4]\right| \leq \mathbf{Adv}^{\mathrm{IND-PSA2}}_{\mathsf{LTF}',\mathscr{A}_4}(k). \tag{32}$$

In **Game** 6, the element ρ of each challenge ciphertext is set as $\rho = \gamma \oplus m$ where γ is selected randomly from $\{0,1\}^{l_2}$. As in **Game** 5, the $\mathbf{y}_2$ elements are computed by $\mathsf{ABM.LTF}'$ with lossy tags for all challenge ciphertexts. Thus, according to Lemma 1, we have

$$\left|\Pr[S_6] - \Pr[S_5]\right| \leq \mathbf{neg}(k). \tag{33}$$

In **Game** 6, since all challenge messages have been padded, the adversary $\mathscr{A}$ could get no information about them. The original message vector $\mathbf{m}_0$ and conditionally resampled message vector $\mathbf{m}_1$ come from the same distribution, so $\Pr[S_6] = 1/2$. Summing up, we could get that $\mathbf{Adv}^{\mathrm{IND-SO-PSA}}_{\mathsf{PKE},\mathscr{A}}(k) \leq \mathbf{Adv}^{\mathrm{COLL}}_{\mathsf{CHF},\mathscr{A}_1}(k) + \mathbf{Adv}^{\mathrm{IND-PSA3}}_{\mathsf{ABM.LTF}',\mathscr{A}_2}(k) + \mathbf{Adv}^{\mathrm{EVA-PSA}}_{\mathsf{ABM.LTF}',\mathscr{A}_3}(k) + \mathbf{Adv}^{\mathrm{IND-PSA2}}_{\mathsf{LTF}',\mathscr{A}_4}(k) + \mathbf{neg}(k)$. □

7. Conclusion

In this paper, to capture SOA and PSA simultaneously, we introduce a new security notion called IND-SO-PSA, which captures indistinguishability under parameter subversion attacks, selective opening attacks, and chosen ciphertext attacks. IND-SO-PSA security guarantees that even though the system parameter of the PKE scheme is chosen maliciously and the adversary is allowed to open a subset of the challenge ciphertexts (i.e., seeing the corresponding plaintexts and randomness used during the encryption), the remainder of the plaintexts should be indistinguishable from freshly re-sampled ones. In order to construct IND-SO-PSA secure PKE, we introduce two new primitives (i.e., LTF-PSA and ABM-LTF-PSA) and propose their instantiations with an efficiently-embeddable group family. Then, based on these new primitives, we construct a PKE scheme satisfying the newly-proposed IND-SO-PSA security.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China 2017YFB0802000, NSF of China under Grants 61972159 and 61572198, Open Research Fund of Engineering Research Center of Software/Hardware Co-Design Technology and Application, Ministry of Education (East China Normal University), the Fundamental Research Funds for the Central Universities, and Guangxi Key Laboratory of Cryptography and Information Security (no. GCIS202109).

References

[1] M. Bellare, A. Boldyreva, and S. Micali, "Public-key encryption in a multi-user setting: security proofs and improvements," in *Advances in Cryptology-EUROCRYPT 2000*, vol. 1807, pp. 259–274, Springer, Berlin, Germany, 2000.

[2] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 639–648, Philadelphia, PA, USA, May 1996.

[3] T. Chen, L. Zhang, K.-K. R. Choo, R. Zhang, and X. Meng, "Blockchain based key management scheme in fog-enabled IoT systems," *IEEE Internet of Things Journal*, vol. 8, 2021.

[4] L. Zhang, "Key management scheme for secure channel establishment in fog computing," *IEEE Transactions on Cloud Computing*, vol. 9, 2019.

[5] L. Zhang, H. Xiong, Q. Huang, J. Li, K.-K. R. Choo, and L. Jiangtao, "Cryptographic solutions for cloud storage:

challenges and research opportunities," *IEEE Transactions on Services Computing*, 2019.

[6] Y. Zhang, L. Zhang, D. Ni, K.-K. R. Choo, and B. Kang, "Secure, robust and flexible cooperative downloading scheme for highway VANETs," *IEEE Access*, vol. 9, pp. 5199–5211, 2020.

[7] M. Bellare, D. Hofheinz, and S. Yilek, "Possibility and impossibility results for encryption and commitment secure under selective opening," in *Advances in Cryptology-EUROCRYPT 2009*, vol. 5479, pp. 1–35, Springer, Berlin, Germany, 2009.

[8] S. Fehr, D. Hofheinz, E. Kiltz, and H. Wee, "Encryption schemes secure against chosen-ciphertext selective opening attacks," in *Advances in Cryptology-EUROCRYPT 2010*, vol. 6110, pp. 381–402, Springer, Berlin, Germany, 2010.

[9] D. Hofheinz and A. Rupp, "Standard versus selective opening security: separation and equivalence results," in *Theory of Cryptography-TCC 2014*, vol. 8349, pp. 591–615, Springer, Berlin, Germany, 2014.

[10] F. Heuer, T. Jager, S. Schäge, and E. Kiltz, "Selective opening security of practical public-key encryption schemes," *IET Information Security*, vol. 10, no. 6, pp. 304–318, 2016.

[11] V. T. Hoang, J. Katz, A. O'Neill, and M. Zaheri, "Selective-opening security in the presence of randomness failures," in *Advances in Cryptology-ASIACRYPT 2016*, vol. 10032, pp. 278–306, Springer, Berlin, Germany, 2016.

[12] L. Lyu, S. Liu, and S. Han, "Public-key encryption with tight simulation-based selective-opening security," *The Computer Journal*, vol. 61, no. 2, pp. 288–318, 2018.

[13] Z. Huang, S. Liu, and B. Qin, "Sender-equivocable encryption schemes secure against chosen-ciphertext attacks revisited," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 369–385, Springer, Nara, Japan, February 2013.

[14] F. Böhl, D. Hofheinz, and D. Kraschewski, "On definitions of selective opening security," in *Public-Key Cryptography-PKC 2012*, vol. 7293, pp. 522–539, Springer, Berlin, Germany, 2012.

[15] D. Hofheinz, V. Rao, and D. Wichs, "Standard security does not imply indistinguishability under selective opening," in *Theory of Cryptography-TCC 2016*, vol. 9986, pp. 121–145, Springer, Berlin, Germany, 2016.

[16] B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud, "Lossy encryption: constructions from general assumptions and efficient selective opening chosen ciphertext security," in *Advances in Cryptology-ASIACRYPT 2011*, vol. 7073, pp. 70–88, Lecture Notes in Computer Science, Springer, Berlin, Germany, 2011.

[17] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer, "Magic functions," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pp. 523–534, IEEE, New York, NY, USA, October 1999.

[18] D. Hofheinz, T. Jager, and A. Rupp, "Public-key encryption with simulation-based selective-opening security and compact ciphertexts," in *Theory of Cryptography-TCC 2016*, vol. 9986, pp. 146–168, Springer, Berlin, Germany, 2016.

[19] M. Bellare, R. Dowsley, B. Waters, and S. Yilek, "Standard security does not imply security against selective-opening," in *Advances in Cryptology-EUROCRYPT 2012*, vol. 7237, pp. 645–662, Springer, Berlin, Germany, 2012.

[20] B. Auerbach, M. Bellare, and E. Kiltz, "Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups," in *Public-Key Cryptography-PKC 2018*, vol. 10769, pp. 348–377, Springer, Berlin, Germany, 2018.

[21] M. Bellare, G. Fuchsbauer, and A. Scafuro, "Nizks with an untrusted crs: security in the face of parameter subversion," in *Advances in Cryptology-ASIACRYPT 2016*, vol. 10032, pp. 777–804, Springer, Berlin, Germany, 2016.

[22] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," in *Advances in Cryptology-ASIACRYPT 2001*, vol. 2248, pp. 566–582, Springer, Berlin, Germany, 2001.

[23] D. J. Bernstein, T. Chou, C. Chuengsatiansup et al., "How to manipulate curve standards: a white paper for the black hat http://bada55. cr. yp. to," in *Security Standardisation Research-SSR 2015*, vol. 9497, pp. 109–139, Springer, Berlin, Germany, 2015.

[24] D. J. Bernstein, T. Lange, G. Iliev, and Z. V. Georgi Iliev, "Safecurves: choosing safe curves for elliptic-curve cryptography," 2013, http://safecurves.cr.yp.to.

[25] F. Nist, *Fips 186-4–digital Signature Standard (DSS)*, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2013.

[26] M. Adalier, R. Azarderakhsh, D. Bernstein et al., *Public Comments Received on Fips 186-4: Digital Signature Standard (DSS)*, National Institute of Standards and Technology, Gaithersburg, MD, USA.

[27] B. Kang, X. Meng, L. Zhang, and Y. Sun, "Nonce-based key agreement protocol against bad randomness," *International Journal of Foundations of Computer Science*, vol. 30, no. 4, pp. 619–633, 2019.

[28] Q. Pei, B. Kang, L. Zhang, K.-K. R. Choo, Y. Zhang, and Y. Sun, "Secure and privacy-preserving 3D vehicle positioning schemes for vehicular ad hoc network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, pp. 1–12, 2018.

[29] X. Meng, L. Zhang, and B. Kang, "Fast secure and anonymous key agreement against bad randomness for cloud computing," *IEEE Transactions on Cloud Computing*, 2020.

[30] T. Baigneres, C. Delerablée, M. Finiasz, L. Goubin, T. Lepoint, and M. Rivain, "Trap me if you can-million dollar curve," *IACR Cryptol. ePrint Arch*, vol. 2015, p. 1249, 2015.

[31] Z. Huang, L. Zhang, X. Meng, and K.-K. R. Choo, "Key-free authentication protocol against subverted indoor smart devices for smart home," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1039–1047, 2019.

[32] C. Peikert and B. Waters, "Lossy trapdoor functions and their applications," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1803–1844, 2011.

[33] D. Hofheinz, "All-but-many lossy trapdoor functions," in *Advances in Cryptology-EUROCRYPT 2012*, vol. 7237, pp. 209–227, Springer, Berlin, Germany, 2012.

[34] X. Boyen and Q. Li, "All-but-many lossy trapdoor functions from lattices and applications," in *Advances in Cryptology-CRYPTO 2017*, vol. 10403, pp. 298–331, Springer, Berlin, Germany, 2017.

[35] M. Bellare, Z. Brakerski, M. Naor et al., "Hedged public-key encryption: how to protect against bad randomness," in *Advances in Cryptology-ASIACRYPT 2009*, vol. 5912, pp. 232–249, Springer, Berlin, Germany, 2009.

[36] A. Boldyreva, C. Patton, and T. Shrimpton, "Hedging public-key encryption in the real world," in *Advances in Cryptology-CRYPTO 2017*, vol. 10403, pp. 462–494, Springer, Berlin, Germany, 2017.

[37] B. Hemenway and R. Ostrovsky, "Extended-ddh and lossy trapdoor functions," in *Public Key Cryptography-PKC 2012*, vol. 7293, pp. 627–643, Springer, Berlin, Germany, 2012.

[38] B. Hemenway and R. Ostrovsky, "Building lossy trapdoor functions from lossy encryption," in *Advances in Cryptology-ASIACRYPT 2013*, vol. 8270, pp. 241–260, Springer, Berlin, Germany, 2013.

[39] N. Cao, Z. Cao, Z. Liu, X. Dong, and X. Zhao, "All-but-many lossy trapdoor functions under decisional rsa subgroup assumption and application," *The Computer Journal*, vol. 62, no. 8, pp. 1148–1157, 2019.