Received June 19, 2018, accepted July 25, 2018, date of publication August 7, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2864163

Consecutive Leakage-Resilient and Updatable Lossy Trapdoor Functions and Application in Sensitive Big-Data Environments

MINGWU ZHANG[1,2,3], JIAJUN HUANG[1], HUA SHEN[1], ZHE XIA[2], AND YONG DING[3], (Member, IEEE)

[1]School of Computers, Hubei University of Technology, Wuhan 430068, China
[2]Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, Wuhan 430070, China
[3]School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Mingwu Zhang (csmwzhang@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672010 and Grant 61702168, in part by the fund of Hubei Key Laboratory of Transportation Internet of Things under Grant WHUTIOT-2017B001, and in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201717.

ABSTRACT Lossy trapdoor functions (LTFs) are very useful tools in constructing complex cryptographic primitives in a black-box manner, such as injective trapdoor functions, collision-resistant hashes, CCA secure public-key encryption, and so on. However, the trapdoor is very sensitive in lossy trapdoor function systems, and the attacker can obtain partial sensitive information of trapdoor by the side-channel attacks, which leads to not only the *leakage of sensitive information* but also the *impossibility of provable security*. In this paper, we present the new model of *updatable lossy trapdoor functions in presence of consecutive and continual leakage-resilient*, to provide a more efficient mechanism in solving the sensitive trapdoor leakage problem in LTF systems. Our contribution has threefold: 1) we give the definition and model of consecutive and continual leakage-resilient LTFs, and provide the concrete construction to achieve the lossiness of 50%; 2) using the proposed LTF scheme as a primitive, we present a updatable public-key encryption in the presence of consecutive and continual leakage-resilience, in which the leakage of secret key can occur during the updates that can simulate the real leakage scenarios; and 3) We provide a secure application deployment in sensitive-data revealing environments that employ the proposed CCLR-PKE scheme as a building block, in which a side-channel analyzer might obtain some sensitive information by controlling the secret channel, watching the private memory and detecting the algorithm executing and so on.

INDEX TERMS Consecutive leakage, lossy trapdoor function, trapdoor update, leakage rate.

I. INTRODUCTION

A. BACKGROUNDS

The notion of lossy trapdoor functions (LTFs) was first proposed by Peikert and Waters in STOC'08 [17], in which there exist two modes in lossy trapdoor functions: *injective* mode and *lossy* mode. In the regular injective mode, computable functions are injective and invertible with a secret trapdoor. In the lossy mode, functions statistically lose information about their inputs. Moreover, the two modes are computationally indistinguishable.

LTFs can serve as black-box building blocks within more complex primitives, such as regular injective trapdoor functions, provably collision resistant hashing, and public-key encryption with chosen-ciphertext security etc [4], [5], [9], [17], [18], [23], [25]. In the injective trapdoor function $f(\cdot)$, a party with a trapdoor can invert the function. However, inversion should be infeasible for any attacker without the sensitive trapdoor [17], [18], [20]. Let $n(\lambda) = \mathsf{poly}(\lambda)$ represent the input length of the function and $\ell(\lambda) \leq n(\lambda)$ represent the lossiness of the collection. The residual leakage $r(\lambda) = n(\lambda) - \ell(\lambda)$. The *larger* lossiness of $r(\lambda)$, the *harder* inversion of function.

Actually, in the injective mode, the image of lossy trapdoor function can be efficiently inverted to obtain the pre-image using the sensitive trapdoor. Thus the trapdoor is very sensitive in the lossy trapdoor function system. However, the trapdoor are usually stored in the memory, and the attacker can gain the partial sensitive information of trapdoor by the side-channel attacks, which leads to not only the leakage of the information but the impossibility of provable security [16], [19], [23], [24], [26].

In order to keep the sensitive trapdoor in the memory secretely, we divide the memory into two parts: *public memory* and *private memory*. Public memory can store the public key, system parameters, and the inputs and outputs of the computation. Private memory is used to store the sensitive information such as secret key, secret randomness and seed, and the intermediate value in the computations. We allow the attacker to watch the contents of public memory, while is allowed to gain a limited amount of sensitive information in private memory for a period, which simulates the actual side-channel attacks.

To avoid the attacker to gain entire information in private memory, we introduce a refresh mechanism to update the secret information which is called *continual leakage resilience*(CLR) [23], [24]. In continual leakage-resilient schemes for trapdoor function, encryption or signature schemes, secret trapdoors/keys will be updated periodically and thus the attacker can only gain at most a bounded leakage between two updates, while keeping the public key same. When the secret trapdoors/keys are updated then the secret trapdoors/keys must be re-randomized to recover enough min-entropy. Otherwise, the attacker can obtain some future secret trapdoors/keys, bit-by-bit, via its leakage in each time period to obtain entire secret information [2], [3], [6], [7].

In order to simulate the leakage, we define a *leakage oracle*. The attacker can gain the (bounded) sensitive information by querying the leakage oracle. Each time, say in the i-th query, the attacker provides an efficiently computable leakage function f_i[1] whose output is at most μ-bit,[2] and the challenger chooses randomness r_i, updates the secret trapdoor/key from td_{i-1} to td_i, and gives the attacker the leakage response $\ell_i = f(td_{i-1})$.

In the *traditional continual leakage model* [6]–[8], [10], [11], [14], the leakage attack is applied on a single trapdoor/key, and the leakage oracle responds with $\ell_i = f_i(td_{i-1})$ (the input of leakage function is only associated with trapdoors or secret keys). In the *continual leak-on-update* model [1], [13], the leakage attack is applied on the current trapdoor/key and the randomness used for updating the trapdoor/key, i.e., the leakage oracle answer the leakage with $\ell_i = f_i(td_{i-1}, r_i)$. *Consecutive continual leakage* was first presented by Dachman-Soled *et al.* [12], in which the leakage is defined by two consecutive secret key, e.g., $\ell_i = f_i(td_{i-1}, td_i)$. Obviously, in the consecutive leakage model the attacker can query the secret trapdoor information about two consecutive trapdoors, i.e., previous (un-updated) trapdoor and updated trapdoor.

[1]During each time period, we allow the attacker to choose an arbitrary (efficiently computable) leakage function, and obtain as a result the leakage function applied to current state.

[2]This leakage is called bounded leakage, and the leakage bound is μ.

B. OUR CONTRIBUTION

Since the side-channel attacks arise as a huge threat for cryptographic schemes than previously realized, the emergence of leakage-resilient cryptography has led to constructions of many cryptographic primitives which can be proven secure even against attackers who obtain limited additional information about secret trapdoors/keys and other internal states. Our contribution in this work is listed as follows:

1) We give the definition and security model of *consecutive and continual leakage-resilient LTF* (namely, CCLR-LTF in short), in which the inputs of the update algorithm are associated with both previous leakage and current trapdoor which simulates the real trapdoor online update environments. We provide the concrete construction of CCLR-LTF scheme. Analysis indicates that our proposed scheme achieves approximate 12.5% leakage rate. Also, we present the performance analysis such as *leakage bound, leakage rate, lossiness*, and the sizes of *system parameters*, *trapdoor*, *evaluation key*, and *output of function* etc.
2) Using the proposed CCLR-LTF scheme as a primitive, we present a *updatable public-key encryption in the presence of consecutive and continual leakage-resilience*. In our updatable public-key encryption system, the leakage of secret key can occur during the updates. At the end of each time-period, we "update" or "refresh" the secret key, in which the update is a randomized procedure that takes as input a secret key sk corresponding to a public key pk, and outputs a uniformly random secret key.
3) We give a secure scenario that deploys the CCLR-PKE scheme as the building block. In our deployment, we allow a side-channel analyzer that can monitor the system to obtain some sensitive information by controlling the secret channel, watching the private memory and detecting the algorithm executing and so on.

C. APPLICATIONS IN SENSITIVE INFORMATION REVEALING ENVIRONMENTS

We show the practical applications in the presence of consecutive and continual sensitive leakage. Traditionally, in the distributed cloud systems, the key generator will distribute the secret key for the user via a secure channel, and the secret key will store in the private and secret memory so that the attacker can not read the memory. Also, after encrypted by a sender, the ciphertext will send to the receiver via a public channel. During the decryption, the receiver will perform the decryption algorithm using the secret key and the ciphertext as inputs. However, when the decryption algorithm are executing, the algorithm must be performed in a blackbox manner, i.e., the attacker can not watch or debug the performing since the secret key and intermediate states are sensitive.

In our sensitive information revealing environments, we assume that there exists a side-channel analyzer, which can monitor the system to obtain some sensitive information leakage. In this case, the side-channel analyzer is able to monitor the secret channel, watch the private memory and detect the executing algorithm etc. We give the restriction that the side-channel analyzer can obtain at most μ-bit sensitive information in one period, and at the end of this period the secret key will be updated.

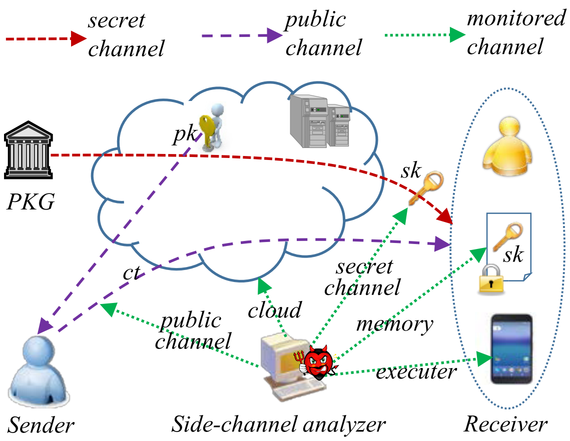

FIGURE 1. **Secure data transmit in side-channel attack environment.**

Fig. 1 demonstrates the scenario for our scheme in secure data transmit in side-channel attack environments, even the side-channel analyzer can obtain a limited sensitive information leakage by side-channel attacks. By the experiment test in Section III-C, we allow the side-channel analyzer to obtain approximative 45K-bit leakage in one update period in the setting $l = 96$, and we can achieve more allowable leakage when enlarge the parameter l.

D. PAPER ORGANIZATION

In Section II, we give the preliminaries and mathematical primitives. We present the concrete construction of CCLR-LTF and provide the security and performance analysis in Section III and, using CCLR-LTF as a primitive, we propose the updatable public-key encryption against consecutive and continual leakage-resilience in Section IV, respectively. Finally, we draw the conclusion in Section V.

II. DEFINITIONS AND PRELIMINARIES

Throughout of this paper, we use λ to denote the system security parameter. We say that a function $\beta(\lambda)$ is negligible in security parameter λ if for all polynomial ploy and sufficiently large $\beta(\lambda) \le 1/\mathsf{poly}(\lambda)$.

We use the bold caption to denote a matrix, and use $Rank_d(\mathbb{Z}_q^{n\times m})$ to denote a random n-by-m matrix over $\mathbb{Z}_q$ of rank d. We let $[n]$ to denote the set $\{1, 2, \cdots, n\}$, and $[n, m]$ to denote the set $\{n, n+1, \cdots, m\}$. For $\boldsymbol{Y} \in \mathbb{Z}_q^{n\times m}$, $g^{\boldsymbol{Y}}$ denotes $(g^{Y_{11}}, g^{Y_{12}}, \cdots, g^{Y_{nm}})$. For $r \in \mathbb{Z}_q$, two vectors $\boldsymbol{A} = (A_1, A_n \cdots, A_n) \in \mathbb{G}^n$, $\boldsymbol{B} = (B_1, B_2, \cdots, B_n) \in \mathbb{G}^n$, we denote $r\boldsymbol{A} = (A_1^r, A_2^r, \cdots, A_n^r)$ and $e(\boldsymbol{A}, \boldsymbol{B}) = \prod_{i=1}^n e(A_i, B_i)$, respectively.

If $x \in L$ the corresponding r is called a witness for x, and (X, L) forms a subset membership problem [21], [23].

Definition 1 (Statistical Distance): Let X and Y be two random variables in a finite set $\mathcal{Z}$. The statistical distance between X and Y is defined as:

$$\mathsf{SD}(X, Y) = \frac{1}{2}\sum_{z\in\mathcal{Z}}\Big|\Pr[X = z] - \Pr[Y = z]\Big| \qquad (1)$$

Definition 2 (Trapdoor Function): Let $n = n(\lambda) = \mathsf{poly}(\lambda)$ denote the input length of the trapdoor functions. A collection of *injective* trapdoor functions is given by a tuple of algorithms $\mathsf{TF} = (\mathsf{S}, \mathsf{F}, \mathsf{F}^{-1})$ having the following properties:

- (*Easy to sample, compute, and invert with trapdoor*): Algorithm S outputs (v, td) where v is a function index and td is its trapdoor. Algorithm $\mathsf{F}(V, \cdot)$ computes an injective function $f_V(x)$ over the domain $\{0, 1\}^n$ and algorithm $\mathsf{F}^{-1}(td, \cdot)$ computes $f_V^{-1}(\cdot)$ using a trapdoor td.
- (*Hard to invert without trapdoor*): for any $p.p.t$ inverter $\mathcal{I}$, the probability that $\mathcal{I}(f_V(x))$ outputs x is negligible, where the probability is taken over the choice of $(V, td) \leftarrow \mathsf{S}$, $x \leftarrow \{0, 1\}^n$ and $\mathcal{I}$'s randomness.

Definition 3 (Extended Diffie-Hellman Assumption): The extended Diffie-Hellman assumption states as: Given $(\mathbb{G}, q, g_1, g_2, \cdots, g_n)$, it is hard to distinguish the following two distributions:

$$\begin{pmatrix} g_1, & g_2, & \cdots, & g_n \\ g_1^r, & g_2^r, & \cdots, & g_n^r \end{pmatrix} \approx_c \begin{pmatrix} g_1, & g_2, & \cdots, & g_n \\ g_1^{r_1}, & g_2^{r_2}, & \cdots, & g_n^{r_n} \end{pmatrix} \qquad (2)$$

where $r, r_1, \cdots, r_n \in \mathbb{Z}_q$.

Clearly, for the exponent matrices formed in above distributions, the rank of valid extended Diffie-Hellman tuple is 1, and the rank for invalid tuple is 2.

Naor and Segev [21] indicated that the Diffie-Hellman assumption is equivalent to the assumption in distinguishing between an n-by-m matrix $\boldsymbol{X}$ with rank i and one with rank $j > i$ in the exponent of a generator g of group $\mathbb{G}$.

Definition 4 (Rank Hiding Assumption, RHA [7]): Let $\mathsf{Rank}_i(\mathbb{Z}_q^{n\times m})$ be the uniform distribution on all n-by-m matrices over $\mathbb{Z}_q$ of rank i. The rank hiding assumption requires that, for any $p.p.t$ attacker $\mathcal{A}$, we have

$$\Big|\Pr[\mathcal{A}((g, g^{\boldsymbol{X}}) : \boldsymbol{X} \leftarrow \mathsf{Rank}_i(\mathbb{Z}_q^{n\times m})) = 1] - \Pr[\mathcal{A}((g, g^{\boldsymbol{X}}) : \boldsymbol{X} \leftarrow \mathsf{Rank}_j(\mathbb{Z}_q^{n\times m})) = 1]\Big| \le \beta(\lambda) \qquad (3)$$

Remark 1: The Diffie-Hellman assumpiton is an instance of $\boldsymbol{X} \in \mathbb{Z}_q^{2\times 2}$ in distinguishing rank 1 and 2. Namely, in Diffie-Hellman assumption, the tuple is a valid DDH tuple if $\mathsf{Rank}(\boldsymbol{X}) = 1$ and is invalid tuple if $\mathsf{Rank}(\boldsymbol{X}) = 2$.

In this paper, we want to use the fact that under the rank-hiding assumption, random rank-2 matrices in the exponent are indistinguishable from random rank-3 matrices.

Definition 5 (Extended Rank Hiding Assumption, eRHA [1]): The extended rank hiding assumption implies that, for any $p.p.t$ attacker $\mathcal{A}$, we have

$$\Big| \Pr\big[\mathcal{A}\big((g, g^{\boldsymbol{X}}, \boldsymbol{V}_1, \cdots, \boldsymbol{V}_t) : \boldsymbol{X} \leftarrow \mathsf{Rank}_i(\mathbb{Z}_q^{n\times m}), \{\boldsymbol{V}_l\}_{l=1}^t \in \ker(\boldsymbol{X})\big) = 1\big] - \Pr\big[\mathcal{A}\big((g, g^{\boldsymbol{X}}, \boldsymbol{V}_1, \cdots, \boldsymbol{V}_t) : \boldsymbol{X} \leftarrow \mathsf{Rank}_j(\mathbb{Z}_q^{n\times m}), \{\boldsymbol{V}_l\}_{l=1}^t \in \ker(\boldsymbol{X})\big) = 1\big]\Big| \leq \beta(\lambda) \quad (4)$$

where $i, j, m, n \in \mathbb{N}$ s.t. $j > i$ and $t \leq \min\{n, m\} - \max\{i, j\}$.

Definition 6 (Strong Random Extractor [15]): A function $\mathsf{Ext} : \mathcal{X} \times \{0, 1\}^t \rightarrow \mathcal{Y}$ is an average-case (m, ϵ)-strong extractor if for all random variables (X, Z) such that $X \in \mathcal{X}$ and $\tilde{H}_\infty(X|Z) \geq m$, we have

$$\mathsf{SD}\big((\mathsf{Ext}(X, Seed), Seed, Z), (U_{\mathcal{Y}}, Seed, Z)\big) \leq \epsilon \quad (5)$$

where $Seed$ is uniform in $\{0, 1\}^t$ and $U_{\mathcal{Y}}$ is uniform over distribution $\mathcal{Y}$.

Lemma 1 (Generalized Crooked Leftover Hash Lemma [15]): Let the family $\mathcal{H} = \{H_k : \mathcal{X} \rightarrow \mathcal{Y}\}_{k\in\mathcal{K}}$ is a universal hash family. For any two random variables X, Z and $k \in \mathcal{K}$, we have

$$\mathsf{SD}\big((H_k(X), k, Z), (U_{\mathcal{Y}}, k, Z)\big) \leq \frac{1}{2}\sqrt{2^{-\tilde{H}_\infty(X|Z)}|\mathcal{Y}|} \quad (6)$$

Remark 2: The leftover hash lemma implies that any universal hash function is a good extractor: For two random variables X and Y, a family of universal hash functions $\{H_k : \mathcal{X} \rightarrow \mathcal{Y}\}_{k\in\mathcal{K}}$ is an average-case (m, ϵ)-strong extractor $\mathsf{Ext} : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{Y}$ as long as $\tilde{H}_\infty(X|Z) \geq m$ and $\log|\mathcal{Y}| \leq m - 2\log(1/\epsilon) + 2$.

It is straightforward to prove (via a hybrid argument) that statistical and computational indistinguishability are transitive under polynomially-many steps.

A. LEAKAGE-RESILIENT SUBSPACES

In order to obtain consecutively continual-leakage resilience, we consider the following two distributions of random subspaces are indistinguishable. i.e., for arbitrary and adaptively chosen functions f_i $(1 \leq i \leq n)$,

$$\big(\boldsymbol{X}, f_1(V_0, V_1), f_2(V_1, V_2), \cdots, f_n(V_{n-1}, V_n)\big) \approx \big(\boldsymbol{X}, f_1(U_0, U_1), f_2(U_1, U_2), \cdots, f_n(U_{n-1}, U_n)\big) \quad (7)$$

where V_is are the kernel of $\boldsymbol{X}$, and U_is are the uniformly selected vectors with the same length of V_is.

Note that every chosen function f_i can be determined after seeing the previous outputs of $f_1(\cdot), f_2(\cdot), \cdots, f_{i-1}(\cdot)$. We give the security for the above random subspace under extended rank hiding assumption defined in 5 when constructs the scheme in ECC groups.

Lemma 2 [1], [7], [21]: Let $n, l, t, d \in \mathbb{N}$, s.t. $n \geq l \geq 3d$, and q be a prime. Let $\boldsymbol{A} \in \mathbb{Z}_q^{t\times n}$, $\boldsymbol{X} \in \mathbb{Z}_q^{n\times l}$ s.t. $\boldsymbol{A} \cdot \boldsymbol{X} = \boldsymbol{0}$, and $\boldsymbol{U} \in \mathbb{Z}_q^{n\times d}$ be a kernel of matrix $\boldsymbol{A}$ (i.e., $\boldsymbol{A} \cdot \boldsymbol{U} = \boldsymbol{0}$). Let $\boldsymbol{T}, \boldsymbol{T}'$ be the matrices in $\mathbb{Z}_q^{l\times d}$ with degree d, i.e., $\boldsymbol{T}, \boldsymbol{T}' \leftarrow Rank_d(\mathbb{Z}_q^{l\times d})$. For any function $f : \mathbb{Z}_q^{t\times n} \times \mathbb{Z}_q^{n\times 2d} \rightarrow \boldsymbol{W}$, we have

$$\begin{cases} \mathsf{SD}\big((\boldsymbol{A}, \boldsymbol{X}, f(\boldsymbol{A}, \boldsymbol{XT}, \boldsymbol{XT}'), \boldsymbol{XT}'), (\boldsymbol{A}, \boldsymbol{X}, f(\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{XT}'), \boldsymbol{XT}')\big) \leq \epsilon \\ |\boldsymbol{W}| \leq (q-1) \cdot q^{l-3d-2} \cdot \epsilon^2 \end{cases} \quad (8)$$

We write the kernel of matrix $\boldsymbol{A}$ as $\ker(\boldsymbol{A})$. The above lemma 2 is a result of generalization of the Crooked Leftover Hash Lemma [15], [26].

Lemma 3 [12]: Let $H : \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ be a hash function family, and (K, Z) be joint random variables over distributions $(\mathcal{K}, \mathcal{Z})$ for the set $\mathcal{K}$ and some set $\mathcal{Z}$. Define the set

$$\Gamma = \Big\{(d, d', z) \in \mathcal{D} \times \mathcal{D} \times \mathcal{Z} : \mathsf{SD}((H_{K|Z=z}(d), H_{K|Z=z}(d')), (U_{|Z=z}, U'_{|Z=z}) > 0\Big\} \quad (9)$$

where $U_{|Z=z}$ and $U'_{|Z=z}$ denote two independent uniform distributions over $\mathcal{R}$ conditioned on $Z = z$, and $K|(Z = z)$ denotes the conditional distribution of K given $Z = z$.

Suppose D and D' are independent random variables over $\mathcal{D}$, (K, Z) are random variables over $\mathcal{D} \times \mathcal{Z}$ s.t. $\Pr[(D, D', Z) \in \Gamma] \leq \epsilon$. Then for any set $\mathcal{S}$ and any function $f : \mathcal{R} \times \mathcal{Z} \rightarrow \mathcal{S}$, we have

$$\mathsf{SD}\big((K, Z, f(H_K(D), Z)), (K, Z, f(U_{|Z}, Z))\big) \leq \frac{\sqrt{3|\mathcal{S}| \cdot \epsilon}}{2} \quad (10)$$

B. MODEL AND SECURITY OF CONSECUTIVE CONTINUAL-LEAKAGE RESILIENT AND UPDATABLE LOSSY TRAPDOOR FUNCTIONS

In this section, we first give the definition of lossy trapdoor functions with key update, and then provide the security requirements for the consecutive continual leakage resilience, whose framework is desecirbed in Fig. 2.

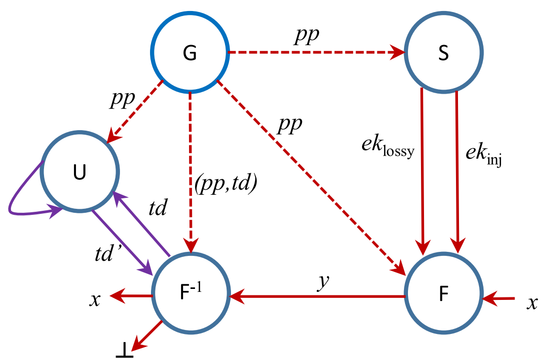

FIGURE 2. Framework of lossy trapdoor function with update.

Definition 7 (Lossy Trapdoor Functions with Key Update): A collection of (d, k)-lossy trapdoor functions with key update consists of five algorithms: CCLR-LTF=(G, S, F, F^{-1}, U), such that:

- $\mathsf{G}(1^\lambda)$: The parameter and trapdoor generation algorithm takes in the security parameter λ and outputs the public parameter pp and a trapdoor td.

- $\mathsf{S}(pp, b)$: The sample algorithm takes in the public parameter pp and a bit $b \in \{0, 1\}$, and samples an evaluation key ek. The evaluation key is called *injective* when $b = 1$ and *lossy* when $b = 0$.
- $\mathsf{F}(ek, \boldsymbol{x})$: The evaluation algorithm takes as input the evaluation key ek and an input $\boldsymbol{x} \in \{0, 1\}^n$, and outputs the image $\boldsymbol{y}$.
- $\mathsf{F}^{-1}(td, \boldsymbol{y})$: The inversion algorithm takes as input the image $\boldsymbol{y}$ and the trapdoor td, and outputs the pre-image $\boldsymbol{x} \in \{0, 1\}^n$ or a failure symbol $\perp$.
- $\mathsf{U}(td)$: The update algorithm takes as input the trapdoor td and outputs a *updated and re-randomized* trapdoor td'.

The CCLR-LTF scheme holds the following properties:

1) **Correctness of the evaluation**. A correct image $\boldsymbol{y}$ evaluated by an injective evaluation key can recover the pre-image $\boldsymbol{x}$ by a trapdoor: For all $(pp, td) \leftarrow \mathsf{G}(1^\lambda)$, $ek \leftarrow \mathsf{S}(pp, 1)$ and all $\boldsymbol{x} \in \{0, 1\}^n$, it requires that $\mathsf{F}^{-1}(td, \mathsf{F}(ek, \boldsymbol{x})) = \boldsymbol{x}$.
2) **Trapdoor sample**. It is easily to sample an injective function with the trapdoor, and however, it only can sample a lossy function publicly without trapdoor.
3) **Consistency of trapdoor update**. It requires that, for all pp and evaluation key ek, the updated trapdoor td' can also recover the pre-image $\boldsymbol{x}$ of $\boldsymbol{y}$ correctly geneated in the injective mode. i.e.,

$$\Pr\left[\mathsf{F}^{-1}(td', \boldsymbol{y}) = \boldsymbol{x} : \begin{array}{l} (pp, td) \leftarrow \mathsf{G}(1^\lambda), \\ ek \leftarrow \mathsf{S}(pp, 1), \\ td' \leftarrow \mathsf{U}(td), \\ \boldsymbol{x} \leftarrow \{0, 1\}^n, \\ \boldsymbol{y} \leftarrow \mathsf{F}(ek, \boldsymbol{x}) \end{array}\right] = 1 \quad (11)$$

4) **Injective/Lossy**. For any $ek \leftarrow \mathsf{S}(pp, 1)$ where the function $\mathsf{F}(ek, \cdot)$ works in the injective mode, and for any $ek \leftarrow \mathsf{S}(pp, 0)$ where the function $\mathsf{F}(ek, \cdot)$ works in the lossy mode. The image size of the lossy function $\mathsf{F}(ek, x)$ is at most 2^{n-k}.
When the evaluation $\mathsf{F}(ek, x)$ works in the injective mode, it requires that it can be inverted to the correct pre-image using either the trapdoor td or any of its polynomial many updated trapdoor td'.
5) **Hard to distinguish between injective from lossy**. Let a bit b be the flag to denote the mode. The output distributions of $\mathsf{S}(pp, b = 1)$ and $\mathsf{S}(pp, b = 0)$ are computationally indistinguishable even after seeing lots of updated trapdoors.

We consider the security model of updatable lossy trapdoor functions against consecutive and continual leakage as the following definition.

Definition 8 (Security of Updatable LTF in the Presence of Consecutive Continuous Leakage): A CCLR-LTF $= (\mathsf{G}, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1}, \mathsf{U})$ scheme is said to be consecutive continual μ-bit leakage-resilient (namely, μ-CCLR-LTF), if for any *p.p.t* attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a negligible advantage $\mathsf{Adv}_{\mathcal{A}}^{\mu\text{-CCLR}}(\lambda, b)$ in the security experiment, i.e.,

$$\left|2 \Pr\left[\mathsf{Adv}_{\mathcal{A}}^{\mu\text{-CCLR}}(\lambda, b) = 1\right] - 1\right| \leq \beta(\lambda) \quad (12)$$

where the interactive experiment is defined as follows:

$\mathbf{Exp}_{\mathcal{A}}^{\mu\text{-CCLR}}(\lambda, b)$:

1) $(pp, td_0) \leftarrow \mathsf{G}(1^\lambda)$.
2) $st_0 = \emptyset$.
3) **For** $i = 1, 2, \cdots, t$,
   where $t = \mathsf{poly}(\lambda)$ is polynomial in λ.
   $st_i \leftarrow \mathcal{A}_1^{\mathcal{O}_{f_i}(td_{i-1})}(pp, st_{i-1})$ s.t. $|\mathsf{leak}(td_{i-1})| \leq \mu$.
   $td_i \leftarrow \mathsf{U}(td_{i-1})$.
4) $b \leftarrow \{0, 1\}$.
5) $ek \leftarrow \mathsf{S}(pp, b)$.
6) $c' \leftarrow \mathcal{A}_2((st)_{i\in[t]}, ek)$.
7) **Output** $(b' = b)$.

III. CONSTRUCTION OF CONSECUTIVE CCLR-LTF

A. THE SCHEME

Let $\mathbb{G}$ and $\mathbb{G}_2$ be two multiplicative groups of prime order q such that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_2$. Let g be a generator of $\mathbb{G}$, and $e(g, g)$ be a generator of $\mathbb{G}_2$. The construction of CCLR-LTF$=(\mathsf{G}, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1}, \mathsf{U})$ is given as follows.

- $\mathsf{G}(1^\lambda)$:
  1) Run the bilinear group generator to create group parameter $sp = (\mathbb{G}, \mathbb{G}_2, q, g, e) \leftarrow \mathcal{G}(1^\lambda)$.
  2) Let $l \geq 2n \geq 7$. At random select $\boldsymbol{A} \in \mathbb{Z}_q^{2\times l}$.
  3) At random select a kernel $\boldsymbol{Y}$ of $\boldsymbol{A}$ (i.e., $\boldsymbol{Y} \leftarrow \ker(\boldsymbol{A})$), that is, $\boldsymbol{Y} \in \mathbb{Z}_q^{l\times 2}$ can be viewed as two random points in the kernel of $\boldsymbol{A}$ s.t. $\boldsymbol{A} \cdot \boldsymbol{Y} = \boldsymbol{0}$.
  4) Set public parameter $pp = (sp, g^{\boldsymbol{A}})$, and keep the trapdoor $td = g^{\boldsymbol{Y}}$.
- $\mathsf{S}(pp, b)$:
  1) Given $b \in \{0, 1\}$ and pp, and let

$$\boldsymbol{C} = \begin{pmatrix} b, & 0, & 0, & 0, & \cdots, & 0 \\ 0, & b, & 0, & 0, & \cdots, & 0 \\ 0, & 0, & b, & 0, & \cdots, & 0 \\ 0, & 0, & 0, & b, & \cdots, & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0, & 0, & 0, & \cdots, & b, & 0 \\ 0, & 0, & 0, & \cdots, & 0, & b \end{pmatrix}_{2n\times l}$$

  2) At random select $\boldsymbol{R} = (R_1, R_2) \in L$ with a witness $\boldsymbol{R} \in \mathbb{Z}_q^2$, where L is a language of decisional Diffie-Hellman problem.
  3) Compute $ek = g^{\boldsymbol{C}} + \boldsymbol{R}^\top \times (\underbrace{g^{\boldsymbol{A}}, g^{\boldsymbol{A}}, \cdots, g^{\boldsymbol{A}}}_{n})^\top$ in Eq. 13, as shown at the top of the next page.
  4) Output $ek = g^{\boldsymbol{V}}$.

Remark 3: It is easily to verify that, when $b = 1$ and $l \geq 2n$, it is in injective mode since the matrix $\boldsymbol{V}$ is full-rank, i.e., $Rank(\boldsymbol{V}) = 2n$. When $b = 0$, it is in lossy mode and the rank of matrix $\boldsymbol{V}$ of $2n \times l$ is n.

Remark 4: The lossiness is n in the lossy mode.

$$
\begin{aligned}
ek &= g^{\boldsymbol{V}} \\
&= g^{\boldsymbol{C}} + \boldsymbol{R}^\top \times (g^{\boldsymbol{A}}, \cdots, g^{\boldsymbol{A}})^\top \\
&= \begin{pmatrix} \\ g^{\boldsymbol{C}} \\ \\ \end{pmatrix}_{2n\times l} + \begin{pmatrix} R_1 \\ R_2 \\ R_1 \\ R_2 \\ \vdots \\ R_1 \\ R_2 \end{pmatrix}_{2n\times 1} \times \begin{pmatrix} g^{A_{11}}, & g^{A_{12}}, & \cdots, & g^{A_{1l}} \\ g^{A_{21}}, & g^{A_{22}}, & \cdots, & g^{A_{2l}} \\ g^{A_{11}}, & g^{A_{12}}, & \cdots, & g^{A_{1l}} \\ g^{A_{21}}, & g^{A_{22}}, & \cdots, & g^{A_{2l}} \\ \vdots & \vdots & \ddots & \vdots \\ g^{A_{11}}, & g^{A_{12}}, & \cdots, & g^{A_{1l}} \\ g^{A_{21}}, & g^{A_{22}}, & \cdots, & g^{A_{2l}} \end{pmatrix}_{2n\times l} \\
&= \begin{pmatrix} g^{b+R_1A_{11}}, & g^{R_1A_{12}}, & g^{R_1A_{13}}, & g^{R_1A_{14}}, & \cdots, & g^{R_1A_{1l}} \\ g^{R_2A_{21}}, & g^{b+R_2A_{22}}, & g^{R_1A_{23}}, & g^{R_1A_{24}}, & \cdots, & g^{R_2A_{2l}} \\ g^{R_1A_{11}}, & g^{R_1A_{12}}, & g^{b+R_1A_{13}}, & g^{R_1A_{14}}, & \cdots, & g^{R_1A_{1l}} \\ g^{R_2A_{21}}, & g^{b+R_2A_{22}}, & g^{R_1A_{23}}, & g^{b+R_1A_{24}}, & \cdots, & g^{R_2A_{2l}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ g^{R_1A_{11}}, & g^{R_1A_{12}}, & g^{R_1A_{13}}, & \cdots, & g^{b+R_1A_{1(l-1)}}, & g^{R_1A_{1l}} \\ g^{R_2A_{21}}, & g^{R_2A_{22}}, & g^{R_2A_{23}}, & \cdots, & g^{R_2A_{2(l-1)}}, & g^{b+R_2A_{2l}} \end{pmatrix}
\end{aligned} \tag{13}
$$

- $\mathsf{F}(ek, \boldsymbol{x})$: On input an evaluation key $ek = g^{\boldsymbol{V}}$ of function index $\boldsymbol{V}$ and an input $\boldsymbol{x} = x_1x_2\cdots x_n \in \{0,1\}^n$, and compute the image of $\boldsymbol{x}$ as $F_{\boldsymbol{V}}(x) = \boldsymbol{y} = (y_1, y_2, \cdots, y_n)$, where

$$y_i = \begin{cases} (g^{\boldsymbol{V}_{2i}}, g^{\boldsymbol{V}_{2i+1}}) & x_i = 0; \\ g^{\boldsymbol{U}} & x_i = 1. \end{cases} \tag{14}$$

  where $\boldsymbol{U} \in \mathbb{Z}_q^{2\times l}$ is a uniformly random matrix.
- $\mathsf{F}^{-1}(td, \boldsymbol{y})$:
  1) At first parse $td = g^{\boldsymbol{Y}}$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$.
  2) For $i \in [n]$, compute

  $$e(y_i^\top, td) = e(g, g)^{(\boldsymbol{V}_{2i}, \boldsymbol{V}_{2i+1})^\top \boldsymbol{Y}}$$

  Set $x_i = 0$ if $e(y_i^\top, td) = (\mathbf{1}_{\mathbb{G}_2}, \mathbf{1}_{\mathbb{G}_2})$ and set $x_i = 1$ otherwise.
  3) Output $\boldsymbol{x} = x_1x_2\cdots x_n$.
- $\mathsf{U}(td)$:
  1) Input a trapdoor $td = g^{\boldsymbol{Y}} \in \mathbb{G}^{l\times 2}$, at first sample a random full-rank matrix $\boldsymbol{R} \leftarrow Rank_2(\mathbb{Z}_q^{2\times 2})$ with degree 2.
  2) Compute $td' = g^{\boldsymbol{YR}}$.

Remark 5: The trapdoor update operation is performed by "*rotating*" the matrix $\boldsymbol{Y}$: Sample a 2×2 full rank matrix $\boldsymbol{R}$, and set the new trapdoor to $g^{\boldsymbol{YR}}$.

B. CORRECTNESS, CONSISTENCY AND SECURITY

We give the analysis of correctness, consistency and security for the scheme in this section.

- **Consistency of trapdoor update**. The updated trapdoor is $td' = g^{\boldsymbol{YR}}$. In the evaluation of $\mathsf{F}^{-1}(td', \boldsymbol{y})$ using the updated trapdoor td',

$$
\begin{aligned}
e(y_i^\top, td') &= e(g, g)^{(\boldsymbol{V}_{2i}, \boldsymbol{V}_{2i+1})^\top \boldsymbol{YR}} \\
&= e(g, g)^{(\boldsymbol{V}_{2i}, \boldsymbol{V}_{2i+1})^\top \boldsymbol{Y}\cdot\boldsymbol{R}} \\
&= e(y_i^\top, td) e(g, g)^{\boldsymbol{R}}
\end{aligned} \tag{15}
$$

  If the evaluation of $e(y_i^\top, td) = (\mathbf{1}_{\mathbb{G}_2}, \mathbf{1}_{\mathbb{G}_2})$, then $e(y_i^\top, td') = (1_{\mathbb{G}_2}, 1_{\mathbb{G}_2}) \times e(g, g)^{\boldsymbol{R}}$ is also the identity element $(1_{\mathbb{G}_2}, 1_{\mathbb{G}_2})$.
  At the same time, as $\boldsymbol{Y}$ is the kernel of $\boldsymbol{A}$, i.e., $\boldsymbol{Y} \leftarrow \ker(\boldsymbol{A})$ and $\boldsymbol{AY} = \mathbf{0}$.
  For updated $\boldsymbol{Y}' = \boldsymbol{YR}$, $\boldsymbol{AY}' = \boldsymbol{AYR} = \mathbf{0}$, which means that $\boldsymbol{Y}'$ is also the kernel of $\boldsymbol{A}$. Thus $\boldsymbol{Y}$ and $\boldsymbol{Y}'$ correspond the same public key $g^{\boldsymbol{A}}$.
- **Correctness of evaluations of F and F^{-1}**. Without loss of generality, we let $n = 1$ and thus,

$$
\begin{aligned}
&\log_g ek \\
&= \begin{pmatrix} b + R_1A_{11}, & R_1A_{12}, & R_1A_{13}, & \cdots, & R_1A_{1l} \\ R_2A_{21}, & b + R_2A_{22}, & R_1A_{23}, & \cdots, & R_2A_{2l} \end{pmatrix}
\end{aligned} \tag{16}
$$

  Consider the exponent of trapdoor td,

$$\log_g td = \boldsymbol{Y} = \ker(\boldsymbol{A}) \tag{17}$$

  The evaluation value of F is $\mathsf{F}(ek, 0) = ek$ for input $x = 0$, and the value is random when $x = 1$.
- **Indistinguishability of injective/lossy trapdoor**. It is hard to guess $b = 0$ or $b = 1$ given $(g_1^{r_1+b}, g_2^{r_1}, g_1^{r_2}, g_2^{r_2+b})$. It is easily to obtain that the security of deployed in ek scheme can be reduced into the linear assumption, and thus distinguishing lossy mode from injective mode is reduced to differ b in linear assumption defined in 3.

Lemma 4: For any $t \in \mathsf{poly}(\lambda)$, $\boldsymbol{R} \leftarrow \mathbb{Z}_q^2$, $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{2\times l}$ and $\boldsymbol{Y} \leftarrow \ker^2(\boldsymbol{A})$. For polynomial functions $f_1, f_2, \cdots, f_t$ where each $f_i : \mathbb{Z}_q^{l\times 2} \times \mathbb{Z}_q^{l\times 2} \rightarrow \{0,1\}^\mu$ that can be adaptively selected, i.e., f_i can be chosen after seeing the previous output values of $f_1(\cdot), f_2(\cdot), \cdots, f_{i-1}(\cdot)$. The following two distributions Γ_0 and Γ_1 are computationally distinguishable, i.e., $\mathsf{SD}(\Gamma_0, \Gamma_1) \leq \beta(\lambda)$, where

$$\Gamma_0 = \left(g, g^{\boldsymbol{A}}, g^{\boldsymbol{R}^\top \boldsymbol{A}}, f_1(td_0, td_1), \cdots, f_t(td_{t-1}, td_t)\right)$$
$$\Gamma_1 = \left(g, g^{\boldsymbol{A}}, g^{\boldsymbol{U}}, f_1(td_0, td_1), \cdots, f_t(td_{t-1}, td_t)\right) \quad (18)$$

where $td_0 = g^{\boldsymbol{Y}}$ and td_i is the updated trapdoor from td_{i-1} using random $\boldsymbol{R} \leftarrow Rank_2(\mathbb{Z}_q^{2\times 2})$ in the trapdoor update algorithm.

Intuitively, the distribution Γ_0 is the view of the attacker given an encryption of 0 as the challenge ciphertext and consecutive continual leakage of the trapdoor. Γ_1 is the same except the challenge ciphertext is an encryption of 1. Our goal is to indicate that no $p.p.t$ attacker can distinguish between them.

Theorem 1: Suppose that the decisional linear assumption holds, for every lï£¡ ≥ 7, the **CCLR-LTF** scheme is μ-bit leakage resilient against trapdoor consecutive and continual leakage, where

$$\begin{cases} \mu \leq (l/2-3)|q| - \omega(\lambda) & \text{trapdoor leakage bound} \\ \rho = \dfrac{\mu}{2|td|} = \dfrac{l-6}{8l} \approx 12.5\% & \text{trapdoor leakage rate} \\ lb = n - \omega(\lambda) & \text{lossiness bits in lossy mode} \end{cases} \quad (19)$$

Proof: At first, we set the trapdoor td as $td_i \leftarrow \ker^2(\boldsymbol{A})$, instead of using a rotation of the current trapdoor, the update procedure re-samples two random points in the kernel of $\boldsymbol{A}$. That is,

$$\Gamma'_b = \left(g, g^{\boldsymbol{A}}, g^{\boldsymbol{Z}}, f_1(td'_0, td'_1), \cdots, f_t(td'_{t-1}, td'_t)\right)$$

for $g^{\boldsymbol{Z}}$ is sampled either from $g^{\boldsymbol{R}^\top \boldsymbol{A}}$ or $g^{\boldsymbol{U}}$. Intuitively, the operations are computed in the exponent, so the attacker can not distinguish between the modified experiments from the original ones under the decisional linear assumption.

We continue to modify the Γ'_b into Γ''_b where

$$\Gamma''_b = \left(g, g^{\boldsymbol{A}}, g^{\boldsymbol{Z}}, f_1(g^{\boldsymbol{U}_0}, g^{\boldsymbol{U}_1}), \cdots, f_t(g^{\boldsymbol{U}_{t-1}}, g^{\boldsymbol{U}_t})\right)$$

where the distribution samples a random matrix $\boldsymbol{X} \in \mathbb{Z}_q^{l\times (l-3)}$ s.t. $\boldsymbol{AX} = \boldsymbol{0}$. It samples $\boldsymbol{U}_i = \boldsymbol{XT}_i$ for $\boldsymbol{T}_i \in Rank_2(\mathbb{Z}_q^{(l-3)\times 2})$. Finally, it samples $\boldsymbol{Z}$ either as $\boldsymbol{R}^\top \boldsymbol{A}$ or uniform random matrix. It is easily to demonstrate that $\mathsf{SD}(\Gamma''_0, \Gamma''_1) \leq \beta(\lambda)$, which means that Γ''_0 and Γ''_1 are indistinguishable. If the attacker $\mathcal{A}$ can distinguish Γ''_0 from Γ''_1 with non-negligible probability, then we can breaks the decisional linear assumption with the same probability.

We now calculate the performance of leakage bound μ, leakage rate ρ and lossiness bits lb in lossy mode. Let $\boldsymbol{X}$ be a random matrix in $\mathbb{Z}_q^{l\times (l-3)}$, and $\boldsymbol{T}$ and $\boldsymbol{T}'$ be the two matrices with rank 2 in $\mathbb{Z}_q^{(l-3)\times 2}$. For the consecutive continual leakage of trapdoor from leakage function $f_i(td_{i-1}, td_i)$, we define $L : \mathbb{Z}_q^{l\times 2} \times \mathbb{Z}_q^{l\times 2} \rightarrow \{0,1\}^{2\mu}$. Note that each leakage $f_i(td_{i-1})$ and $f_i(td_i)$ is at most μ-bit. Thus $2\mu = (l-6)|q| - \omega(\lambda)$, and $|L| \leq q^{l-6} \cdot \lambda^{-\omega(1)}$. The leakage bound is

$$\mu = \frac{(l-6)|q|}{2} - \omega(\lambda) \quad (20)$$

Then the leakage rate ρ,

$$\rho = \frac{\mu}{2|td|} = \frac{(l/2-3)|q|}{4l|q|} = \frac{1}{8} - \frac{3}{4l} \approx 12.5\% \quad (21)$$

In the construction of ek for lossy mode, i.e., $b = 0$, the rank of matrix $\boldsymbol{V}$ is n. We note that the matrix $\boldsymbol{V}$ is a $2n \times l$ and $l \geq 2n$. Thus the lossiness bits in lossy mode is $lb = 2n - n = n$. ■

Remark 6: It is easily to see that, the larger parameter l, the more allowable leakage bound and higher leakage rate.

Remark 7: Since $l \geq 2n$, the larger l, the more allowable lossiness of the scheme.

C. PERFORMANCE

We give the performance analysis of our scheme. We perform the performance under standard NIST AES-128 bits security. In this AES-128 security, for the bilinear groups, the size $|\mathbb{G}| = 1024$-bit and $|\mathbb{G}_2| = 2048$-bit.

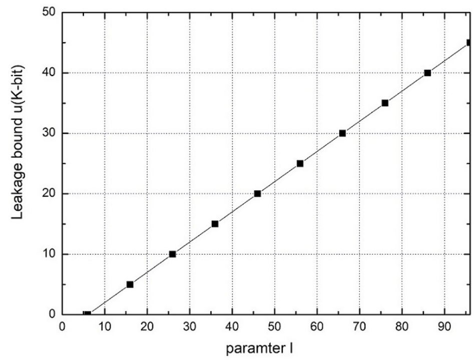

FIGURE 3. Leakage bound μ.

In our construction, we have $l \geq 7$. In order to obtain an optimized lossiness bits in lossy mode, we let $n = l/2$. Fig. 3 shows the relationship between leakage bound μ and parameter l. It indicates that the scheme can tolerate about 45K-bit trapdoor leakage when $l = 100$.

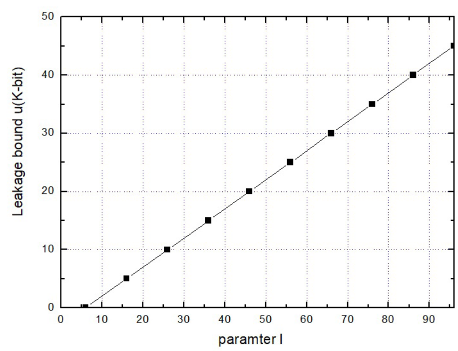

FIGURE 4. Leakage rate ρ.

Fig. 4 shows the leakage rate for our scheme. The theoretical leakage rate is at most 12.5%, and the experimental results demonstrate that the leakage rate is beyond 10% when $l \geq 30$.

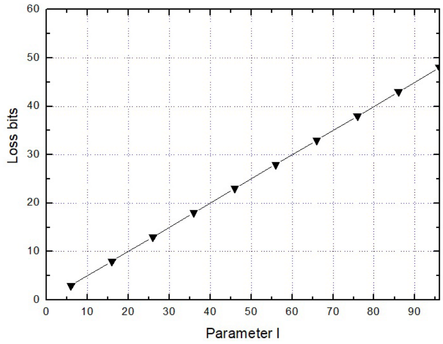

FIGURE 5. **Lossiness bits *lb*.**

Fig. 5 describes the relationship between lossiness bits in lossy mode of LTF and parameter l, and it shows that the lossiness is linear to the parameter l in the scheme.

Fig. 6 indicates the performance of pp, td and ek, and it is easily to see that the larger parameter l the larger sizes of pp, td and ek. Fig. 7 gives the evaluation size of $\boldsymbol{y}$ under the size n of input $\boldsymbol{x}$.

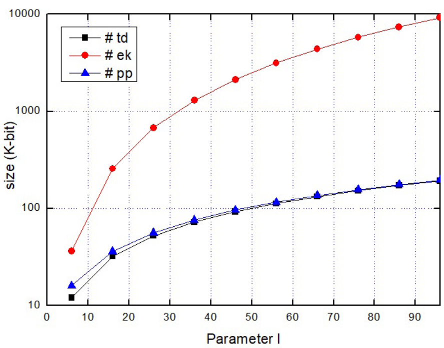

FIGURE 6. **Size of pp, td and ek.**

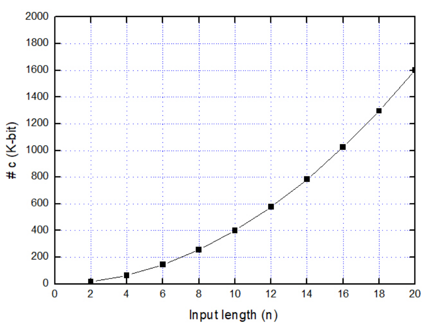

FIGURE 7. **Evaluation size under the input $x = \{0, 1\}^n$.**

IV. CONSTRUCTION OF CONSECUTIVE AND CONTINUAL LEAKAGE-RESILIENT PKE

In this section, we first give model and definition of consecutive and continual leakage resilient public-key encryption (namely, CCLR-PKE), and then present the concrete construction that employs the proposed CCLR-LTF scheme as a primitive.

A. MODEL OF CONSECUTIVE CLR ENCRYPTION

Definition 9 (Updatable Public-key Encryption): A updatable public-key encryption consists of four algorithms described as follows:

- KeyGen$(1^\lambda) \rightarrow (pk, sk_0)$: This algorithm takes as input the security parameter λ and outputs a public key pk and an initial secret key sk_0.
- Enc$(pk, m) \rightarrow ct$: This algorithm takes as input a public key pk and a message m and outputs a ciphertext ct.
- Dec$(sk_i, ct) \rightarrow m|\perp$: This algorithm takes as input a secret key sk_i and a ciphertext ct, and outputs a message m if decryption succeeds and a failure symbol $\perp$ otherwise.
- Upd$(sk_{i-1}) \rightarrow sk_i$: This algorithms takes as input a secret key sk_{i-1} and outputs a updated key sk_i corresponding to the same public key.

In the key-leakage resilient case, the attacker can launch a polynomial number of key-leakage queries. Each time, say in the i-th query, the attacker provides an efficiently computable leakage function f_i whose output is at most μ-bit, and the challenger $\mathcal{C}$ chooses a randomness r_i, updates the secret key from sk_{i-1} to sk_i, and answers the attacker the leakage output ℓ_i.

There have two types of models on continual leakage:

- **Traditional continual leakage model**. The leakage attack is applied on a single secret key sk_i, and the leakage answer is defined as $\ell_i = f_i\,(sk_{i-1})$.
- **Continual leak-on-update model**. The leakage attack is applied on the current secret key sk_{i-1} and the randomness r_i used for updating the secret key, i.e., $\ell_i = f_i\,(sk_{i-1}, r_i)$.

Definition 10 (Continual Leakage Resilience): A public key encryption scheme is said to be μ-continual leakage resilient (respectively, μ-CLR secure with leakage on key updates) if any $p.p.t$ attacker only has a negligible advantage wins the IND-CPA experiment even it has access to leakage oracle to gain at most μ-bit secret key.

Security notions in terms of interactive experiments involving an attacker algorithm $\mathcal{A}$. The view of the attacker in such an experiment is the ensemble of random variables, where each variable includes the random coins of $\mathcal{A}$ and all its inputs over the course of the experiment when run with security parameter λ.

Definition 11 (Consecutive CLR Encryption): Let PKE = (KeyGen, Enc, Dec, Upd) be a updatable public-key encryption scheme. The PKE scheme is said to be μ-leakage resilient against *consecutive* continual-leakage if any probabilistic polynomial-time attacker $\mathcal{A}$ only has a negligible advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CCLR}}(\lambda)$ in the interactive experiment game between a challenger $\mathcal{C}$ and an attacker A, described as follows:

- **Setup**: The challenger $\mathcal{C}$ calls PKE.KeyGen(1^λ) to generate the initial secret key sk_0 and the corresponding public key pk, and sends pk to the attacker $\mathcal{A}$. Note that no leakage is allowed in this phase.

- **Query**: The attakcer $\mathcal{A}$ launches an efficiently computable leakage function f_i whose output is bounded by a parameter μ. The challenger $\mathcal{C}$ updates the secret key (changing it from sk_{i-1} to sk_i), and then gives the leakage output $f_i(sk_{i-1}, sk_i)$ to the attacker. Actually, $\mathcal{A}$ can repeat this query for a bounded polynomial number of times.
- **Challenge**: $\mathcal{A}$ provides two message m_0 and m_1 as the challenged plaintexts. $\mathcal{C}$ tosses a random coin $b \in \{0, 1\}$, and then sends the encryption $\mathsf{Enc}(pk, m_b)$ of m_b as the challenge ciphertext to $\mathcal{A}$.
- **Response**: Finally, the attacker $\mathcal{A}$ outputs a bit b' as the guess of random coin b, and wins the game if $b' = b$.

Clearly, the leakage occurs during the refresh of the key, that is, the input of the leakage function is taken over the current key sk_{i-1} and the updated output key sk_i. In the above experiment, the attacker can only query the leakage of secret key. A public-key encryption scheme is said to be semantically secure in the presence of continual secret-key leakage if the advantage of the attacker $\mathcal{A}$ is negligible in security parameter λ.

B. TRANSFORMATION AND CONSTRUCTION FROM CCLR-LTF

Let CCLR-LTF $=$ $(\mathsf{G}, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1}, \mathsf{U})$ be a updatable lossy trapdoor function against consecutive and continual trapdoor leakage. A CCLR-PKE public-key encryption scheme is presented as follows.

- CCLR-PKE.KeyGen(1^λ): On input a security parameter 1^λ, call CCLR-LTF.G(1^λ) to generate (pp, td), and set $(pk = pp, sk = td)$.
- CCLR-PKE.Enc(pk, m):
  1) Calculate $ek \leftarrow$ CCLR-LTF.S$(pp, 1)$.
  2) Output $ct \leftarrow$ CCLR-LTF.F(ek, m).
- CCLR-PKE.Dec(sk, ct):
  Return $m \leftarrow$ CCLR-LTF.$\mathsf{F}^{-1}(td, ct)$.
- CCLR-PKE.Upd(sk): Output CCLR-LTF.U(sk).

Remark 8 In our transformation and construction, the ciphertext ct is created in injective mode, which means that the ciphertext is invertible under some secret key (trapdoor in LTF). The lossiness mode in algorithm Enc is only used to implement the security proof. Concretely, a valid ciphertext (injective mode) is indistinguishable from an invalid ciphertext (lossiness mode).

Assume that CCLR-LTF be a updatable lossy trapdoor function against consecutive and continual trapdoor leakage. The construction of CCLR-PKE is a semantically secure public-key encryption with secret key update in the presence of consecutive and continual secret-key leakage.

V. CONCLUSION

In this work, we provided the definition and security model of consecutive and continual leakage-resilient lossy trapdoor functions. The function family equipped with a update algorithm that takes as input both previous trapdoor leakage and current trapdoor, which can simulate the real online trapdoor update environments. We presented the concrete construction for the updatable lossy trapdoor function, and analyzed the leakage performance such as leakage bound, leakage rate, trapdoor lossiness etc.

Taking the proposed consecutive and continual leakage-resilient LTF as a building block, we presented a updatable public-key encryption in the presence of consecutive and continual leakage-resilience, in which the leakage of secret key can occur during the updates. Also, we gave a secure application deployment in sensitive-data revealing environments in which there exists a side-channel analyzer to obtain some sensitive information by monitoring the secret channel, watching the private memory and detecting the algorithm executing etc.

In our model, we only consider consecutive and continual leakage for a limited and bounded information, and the total leakage can beyond this bound counted for different leakage period. Whether there exists consecutive auxiliary input leakage and their construction is an open problem in the consecutive and continual leakage model.

REFERENCES

[1] S. Agrawal, Y. Dodis, V. Vaikuntanathan, and D. Wichs, "On continual leakage of discrete log representations," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 8270. Springer-Verlag, pp. 401–420, 2013.

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs, "Public-key encryption in the bounded-retrieval model," in *Advances Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 6110. Berlin, Germany: Springer, 2010, pp. 113–134.

[3] M. Bellare *et al.*, "Hedged public-key encryption: How to protect against bad randomness," in *Advances Cryptology ASIACRYPT* (Lecture Notes in Computer Science), vol. 5912. Berlin, Germany: Springer, 2009, pp. 232–249.

[4] A. Boldyreva, S. Fehr, and A. O'Neill, "On notions of security for deterministic encryption, and efficient constructions without random oracles," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, pp. 335–359, 2008.

[5] X. Boyen and B. Waters, "Shrinking the keys of discrete-log-type lossy trapdoor functions," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2010.

[6] E. Boyle, S. Goldwasser, A. Jain, and Y. T. Kalai, "Multiparty computation secure against continual memory leakage," in *Proc. STOC*, 2012, pp. 1235–1254.

[7] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan, "Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage," in *Proc. FOCS*, 2010, pp. 501–510.

[8] Z. Brakerski and S. Goldwasser, "Circular and leakage resilient public-key encryption under subgroup indistinguishability, or: Quadratic residuosity strikes back," in *Advances Cryptology—CRYPTO*. Berlin, Germany: Springer, 2010, pp. 1–20.

[9] Y. Chen, B. Qin, and H. Xue, "Regularly lossy trapdoor functions and their applications," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1803–1844, 2011.

[10] D. Cash, F. H. Liu, A. O'Neill, and C. Zhang, "Reducing the leakage in practical order-revealing encryption," Cryptol. ePrint Arch., Tech. Rep. 2016/661, 2016.

[11] D. Dachman-Soled, F. H. Liu, and H. S. Zhou, "Leakage-resilient circuits revisited—Optimal number of computing components without leak-free hardware," in *Advances Cryptology-EUROCRYPT* (Lecture Notes in Computer Science), vol. 9057. Berlin, Germany: Springer, 2015, pp. 131–158.

[12] D. Dachman-Soled, S. D. Gordon, F. H. Liu, A. O'Neill, and H. S. Zhou, "Leakage resilience from program obfuscation," *J. Cryptol.*, no. 2, pp. 1–83, 2018.

[13] Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs, "Cryptography against continuous memory attacks," in *Proc. FOCS*, 2010, pp. 511–520.

[14] Y. Dodis, A. B. Lewko, B. Waters, and D. Wichs, "Storing secrets on continually leaky devices," in *Proc. FOCS*, 2011, pp. 688–697.

[15] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, 2008.

[16] E. Kiltz and K. Pietrzak, "Leakage resilient ElGamal encryption," in *Advances Cryptology—ASIACRYPT*. Springer, 2010, pp. 595–612.

[17] C. Peikert and B. B. Waters, "Lossy trapdoor functions and their applications," in *Proc. STOC*, 2008, pp. 187–196.

[18] H. Wee, "Dual projective hashing and its applications—Lossy trapdoor functions and more," in *Advances Cryptology-EUROCRYPT*. Springer, 2012, pp. 246–262.

[19] V. Koppula, O. Pandey, Y. Rouselakis, and B. Waters, "Deterministic public-key encryption under continual leakage," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 9696. Berlin, Germany: Springer, 2016, pp. 304–323.

[20] S. Li, Y. Mu, M. Zhang, and F. Zhang, "Updatable lossy trapdoor functions and its application in continuous leakage," in *Provable Security* (Lecture Notes in Computer Science), vol. 10005, Berlin, Germany: Springer, 2016, pp. 309–319.

[21] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," in *Advances Cryptology—CRYPTO*, vol. 5677. Germany: Springer, 2009, pp. 18–35.

[22] B. Qin, S. Liu, K. Chen, and M. Charlemagne, "Leakage-resilient lossy trapdoor functions and public-key encryption," in *Proc. AsiaPKC*, 2013, pp. 3–12s.

[23] M. Zhang and Y. Mu, "Token-leakage tolerant and vector obfuscated IPE and application in privacy-preserving two-party point/polynomial evaluations," *Comput. J.*, vol. 59, no. 4, pp. 493–507, 2016.

[24] M. Zhang, W. Leng, Y. Ding, and C. Tang, "Tolerating sensitive-leakage with larger plaintext-space and higher leakage-rate in privacy-aware Internet-of-Things," *IEEE Access*, vol. 6, pp. 33859–33870, 2018.

[25] M. Zhang, Y. Yao, B. Li, and C. Tang, "Accountable mobile e-commerce scheme in intelligent cloud system transactions," *J. Ambient Intell. Hum. Comput.*, to be published. [Online]. Available: https://doi.org/10.1007/s12652-017-0672-4

[26] M. Zhang, Y. Zhang, Y. Su, Q. Huang, and Y. Mu, "Attribute-based hash proof system under learning-with-errors assumption in obfuscator-free and leakage-resilient environments," *IEEE Syst. J.*, vol. 11, no. 2, pp. 1018–1026, Jun. 2017.

MINGWU ZHANG was a JSPS Fellow (Japan Society for the Promotion of Science) with the Institute of Mathematics for Industry, Kyushu University, Japan, from 2010 to 2012. From 2015 to 2016, he was a Senior Research Fellow with the Centre for Computer and Information Security, University of Wollongong, Australia. He is currently a Professor with the School of Computers, Hubei University of Technology His current research interests include cryptography technology for clouds and big data, and privacy preservation.

JIAJUN HUANG is currently pursuing the master's degree with the School of Computers, Hubei University of Technology. His current research interests include cryptography technology for decentralized networks and secure computations in clouds.

HUA SHEN received the M.S. and Ph.D. degrees from Wuhan University in 2007 and 2014, respectively. She is currently an Associate Professor with the School of Computers, Hubei University of Technology. Her research interests include the technology of privacy-preserving, information security, and secure cloud computing.

ZHE XIA received the B.Sc. degree from Wuhan University, China, in 2004, and the M.Sc. and Ph.D. degrees from the University of Surrey, U.K., in 2005 and 2009, respectively. He held a postdoctoral position at the Trustworthy Voting Systems Group, University of Surrey, from 2009 to 2013. He joined the Wuhan University of Technology as an Associate Professor in 2013. His research interests include cryptography and information security, particularly design and analysis of secure voting systems. He has published around 20 papers in this field, including some top journals and conferences, such as the IEEE Transactions of Information Forensics and Security and USENIX Security. He has served as the Program Committee Member for several international conferences. He is currently an Associate Editor of the *Journal of Information Security and Application*.

YONG DING received the Ph.D. degree in cryptography from Xidian University, China. He is a Professor and the Director of the Guangxi Key Laboratory of Cryptography and Information Security. He is also the Vice Dean of the School of Computer Science and Information Security, Guilin University of Electronic Technology. His main research interests include cloud security, cryptography, and information security.