IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 16, 2018, accepted May 24, 2018, date of publication June 11, 2018, date of current version July 6, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2842713*

# Tolerating Sensitive-Leakage With Larger Plaintext-Space and Higher Leakage-Rate in Privacy-Aware Internet-of-Things

**MINGWU ZHANG[1,2], WENTAO LENG[1], YONG DING[3], AND CHUNMING TANG[4], (Member, IEEE)**

[1]School of Computer Sciences, 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
[4]School of Mathematics and Information Science, Guangzhou University, Guangzhou 510000, China

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

This work was supported in part by the National Natural Science Foundation of China under Grants 61672010 and 61702168, in part by the Fund of the Hubei Key Laboratory of Transportation Internet of Things under Grant WHUTIOT-2017B001, in part by the Key Laboratory of Mathematics and Interdisciplinary Sciences, Guangdong Higher Education Institutes, Guangzhou University, and in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201717.

**ABSTRACT** When executing a program or storing data in a medical Internet of Things (mIoT) system, physical side-channels analysis, such as recent-timing, cold-reboot, and virtual-machine attacks, might obtain partial information about internal sensitive medical data/states in memory that the attacker can gain partial privacy information. Leakage-resilient cryptography has led to better implementation of many cryptographic primitives that can be proven secure against attackers who can obtain limited sensitive information about *private keys*, *randomness*, and other *internal states*, and therefore prevents from breaking the security. In this paper, to tolerate the sensitive information leakage in mIoT, we first present a leakage-resilient public-key encryption mechanism that is semantically secure against adaptively chosen-ciphertext attacks in the presence of key leakage under standard decisional Diffie–Hellman assumption. Our construction employs a special universal hashing in multiplicative group to provide an efficient strong extractor, and a key derivation function to derive one or more symmetric keys from a single value. Also, the plaintext space of the scheme is extended to the full domain field of group so as to provide a larger space for the message. We emphasis that our scheme can be deployed in mIoT since the limited power and energy budgets, the communication and computation cost, and the leakage attack are taken into account. Using the first scheme as a building block, we also give a protocol construction to achieve the security resilient to randomness leakage and key leakage. Our schemes feature with a shorter key size and a larger plaintext space. Concretely, the private-key contains only four elements in the finite field, and the allowable key-leakage rate is 25%, which provides a higher leakage rate than Naor Segev (leakage rate is 16.7%) and its variants. It is worth highlighting of the construction resilient to both key leakage and randomness leakage, simultaneously, and is flexible to deploy in easy-to-attack outdoor nodes such as in medical IoT and smart grids, since in these nodes the private keys and randomness are either stored or generated in outdoor privacy-aware environments.

**INDEX TERMS** Sensitive information leakage, key entropy, randomness leakage, leakage rate, medical Internet of Things.

## I. INTRODUCTION

In the practical applications in medical Internet of Things, node data is highly sensitive and physical computational devices in such system could leak side-channel information such as private-key, randomness, or secret internal state [12], [32]. Taking an executable computer program or internal storage as an example, recent timing, cold-reboot and virtual-machine attacks show that physical side-channels might leak partial information about internal states from CPU, data/instruct register and RAM (Random Access Memory) when the program is running or has loaded into the memory. As the nodes are exposed at outside and the data in nodes are highly sensitive in medical Internet of Things systems, they can much more vulnerable to hacking. As yet another example, we may need to use a cryptosystem within the context of a more complex protocol (for example,

secure multi-party evaluation and outsourcing computation) that might leak some sensitive information about private-key or data. However, modern cryptosystem is mainly based on the assumptions that users have secrets generated uniformly and "well-protected" from a perfect randomness (with high entropy), and the private-key is perfectly hidden from possible attackers [7], [33]. Even when the private-key is not exposed, the randomness used in the encryption can be leaked? The security reduction will be loose if the randomness is *not fully randomized* (*i.e.*, pseudo-random generator (PRG) is controlled by virus) or the key loaded into the memory is *partially leaked* (*i.e.*, RAM or ROM is monitored by worms), especially in medical IoTs.

To solve this problem, leakage-resilience cryptosystems [1], [2], [9], [25], [30] were proposed to provide a powerful tool and allow us to easily analyze the security of cryptographic constructions in the presence of possible leakage sources such as *side-channels, memories and even performed procedures*. Leakage-resilient cryptographic primitives, whose schemes are proven secure against the attackers who can obtain additional sensitive information about *private keys, randomness and internal states*, can be used to design secure protocols for practical applications such as *securely electronic voting, outsourced computing* and *sensitive data analyzing in bigdata, etc*.

### A. MOTIVATION

The traditional security definitions for encryption techniques are mainly concerned with data privacy in that a ciphertext does not reveal the plaintext information. In the leakage situation, for example in medical IoTs, we should consider a stronger attack ability by *observing behavior of the protocol executions in the presence of potential leakage* since the data is very sensitive. In order to simulate a large class of potential leakage in side-channel attacks from the view of an attacker, we specify an efficiently *computable leakage function* $f$ and allow the attacker to learn the output of $f$ applied to the private-key and possibly other states in the security game. Obviously, *the stronger attack ability, the more complex of the encryption system*. Constructing efficient, secure and practical public-key encryption (PKE) is an emergent requirement in the leakage-resilient cryptosystems.

*Leakage rate* $\rho$ describes the allowable leakage amount related to total key or internal states, i.e. $\rho = \ell/|sk|$, where $\ell$ is the allowable leakage amount and $|sk|$ denotes the length of a secret key $sk$. Leakage rate is a crucial and important performance for a leakage-resilient system. How to improve the leakage rate in a PKE system is emergent in designing leakage-resilient secure applications. Furthermore, it is more efficient when providing a *larger plaintext space* in a PKE system, since the larger plaintext space the more possible data in use.

### B. OUR RESULT AND APPROACH

In this work, we construct a more efficient and practical public-key encryption for medical IoTs in the presence of sensitive key/randomness leakage, and provide the same security guarantee with a leak-free model. In particular, based on the known leakage-resilient PKEs in [22] and [25], we improve the performance in private-key size and plaintext space. We also extend our scheme to achieve the security resilient to both key leakage and randomness leakage. The private-key of our scheme is only 4 elements in $\mathbb{F}_q$ that has the same length with [22] and shorter than Naor-Segev scheme with 6 elements [25], and the allowable leakage rate is at most 25%, which has a higher leakage rate than Naor-Segev's scheme (with 16.7% leakage rate) and its variants [22].

There are several security models of leakage-resilience, differing in what sensitive information can be available to the attacker. Our scheme is called the *bounded-leakage* (a.k.a *memory leakage*) model, in which the attacker can learn *arbitrary information about the private-key/randomness*, as long as the total number of bits learned is limited by a bound $\ell$. We formalize this security notion by giving the attacker access to a *leakage oracle* that it can repeatedly and adaptively query this oracle to get the sensitive information. Also, each query to the oracle is launched by a *leakage function* $f$ and the oracle responds with $f(sk)$.

Like in [22], [25], and [29], we use a randomness extractor to remedy the private-key leakage (reproduce a uniformly distributed key) and give the security proof by virtue of hash proof system technique (HPS) that has systematically stated by Cramer and Shoup in CS-PKE scheme [8]. However, CS-PKE is an *"almost"* leakage-resilient encryption which needs a randomness extractor to remove so-called *almost*, however this is a striking result as CCA security without random oracle and could be achieved by only adding a few more exponentiations to ElGamal encryption scheme. In CS-PKE, a private-key has the form $sk = (a_1, a_2, b_1, b_2)$ and the public key corresponds to components $(c, d)$, where $c = g_1^{a_1} g_2^{a_2}$ and $d = g_1^{b_1} g_2^{b_2}$, along with a target collision resistant function $\mathsf{TCR}$. A ciphertext has the form $(g_1^r, g_2^r, \mathsf{SE}_{k_0}(m), \mathsf{MAC}_{k_1}(\cdot))$, where $k_0$ and $k_1$ are random, which guarantees that, in the ciphertext $ct$, the plaintext component $\mathsf{SE}_{k_0}(m)$ and authentication component $\mathsf{MAC}_{k_1}(\cdot)$ are randomized to against the Chosen-Ciphertext Attack (IND-CCA).

Actually, if we employ this model in the leakage case, it can not ensure the true randomness of authentication component since $\mathsf{MAC}_{k_1}(\cdot) = u_1^{a_1+b_1\alpha} u_2^{a_2+b_2\alpha}$ can be considered as a function with input $sk$, and it may leak some information by querying the leakage oracle (the information entropy of the output of $\mathsf{MAC}$ is reduced). In our scheme, we employ two universal hash functions as the strong extractor and provide the proof framework in the hash proof system, in which one is a 1-universal hash ($v = c^r d^{r\alpha}$) and the other is a 2-universal hash ($e = m \cdot c^{k_0} d^{k_0 s}$). Actually, universal hash functions constitute good aveon-rage-case extractors retaining nearly the same parameters as in the original leftover hash lemma. Differing from introducing new randomness $(z_1, z_1)$ in [22] and [25], we divide the random component $v = c^r d^{r\alpha}$ into two parties $k_0|k_1$ by introducing a key

derivation function (KDF): $k_0$ is for encrypting the plaintext and $k_1$ is for authenticating the ciphertext. These two parties have high entropy when the component $v$ has high entropy, and thus the authentication will reject all invalid ciphertexts. Actually, in our construction, $v$ has high entropy since it is computed by a strong extractor from universal hashing, *i.e.*, $H_{s_1,s_2,\cdots,s_n}(m, h_1, h_2, \cdots, h_n) = m \cdot h_1^{s_1} h_2^{s_2} \cdots h_n^{s_n}$.

In order to gain a larger plaintext space (independent to the leakage parameter), like in [22], we employ a universal hash function $H_{s_1,s_2,\cdots,s_n}(m, h_1, h_2, \cdots, h_n)$ as an extractor, which has been analyzed and used to hash an identity string into a group element. The plaintext space is in group $\mathbb{G}$ (*i.e.*, the size of a plaintext is $\log q$) and the leakage is bounded by $\ell \leq \log q - \omega(\log \lambda)$ which is independent to the plaintext length, whereas in Naor-Segev's construction $\ell \leq \log q - m - \omega(\log \lambda)$ where $m$ is the plaintext length. That is, our scheme provides a higher leakage rate (25%) and larger plaintext space (each element is $\log q$), simultaneously.

### C. RELATED WORK

Building cryptographic schemes secure even if the secrets such as keys and randomness are partially leaked is a state-of-the-art trend in secure systems, motivating partially from side channel attacks. In 2009, inspired by the cold boot attacks, Akavia *et al.* [2] first presented a so-called bounded-leakage model and constructed a leakage-resilient Chosen-Plaintext Attacks (IND-lrCPA) secure scheme under the Learning-With-Error assumption, in which the allowable leakage of a private-key is bounded by $|sk|/\mathsf{polylog}(sk)$. Naor and Segev [25] presented two leakage-resilient constructions based on decisional Diffie-Hellman assumption to against IND-lrCPA attacks and IND-lrCCA attacks respectively, which employ 1-universal hash proof system [8]. They also indicated that, in general, a PKE scheme can achieve the security against IND-lrCPA with the help of randomness extractors and hash proof mechanism. For an IND-lrCCA-secure scheme, they used the Naor-Yung paradigm in [24] to prompt better leakage tolerance, but the Naor-Yung paradigm is quite inefficient since it needs the low efficient Non-Interactive Zero-Knowledge (NIZK). To improve the efficiency, Naor and Segev also considered the Cramer-Shoup scheme under (standard) DDH assumption [8] in which the leakage rate is $1/4 - o(1)$ in non-adaptive IND-lrCCA1 security and $1/6 - o(1)$ in adaptive IND-lrCCA2 security. Baek *et al.* [3] proposed a variant KEM of Cramer-Shoup scheme in [8] to provide a shorter ciphertext size with the security of CCA2. Nguyen *et al.* [26] presented a LR-PKE from 4-wise independent hash functions to achieve constrainted CCA security (IND-lrCCCA), whose model is a weak security than IND-lrCCA. Qin and Liu [29] presented a leakage-resilient CCA-secure PKE with one-time lossy filter in hash proof system framework. Nielsen *et al.* [27] gave the connection between leakage tolerance and adaptive security and showed that leakage tolerance is equivalent to a weaker semi-adaptively security. Recently, Faonio and Venturi [14] presented a public-Key cryptography resilient to bounded leakage and tamper resilience simultaneously.

In Asiacrypt 2010, Dodis *et al.* [9] designed a mechanism to support better leakage tolerance, but with a big trade-off in efficiency since it relies on the $k$-Decisional Linear assumption (k-DLIN) in pairing-based operator (*i.e.*, finite cyclic group with bilinear operation). Liu *et al.* [22] improved the efficiency of Naor-Segev's scheme in that the leakage bound is independent of the plaintext space whereas the performance is almost as efficient as the original Naor-Segev scheme [25]. Zhang and Mu [34] proposed a leakage-resilient inner-product encryption that can tolerate at most 1/2 key leakage, which is based on the somewhat inefficient bilinear map in composite order. Kiltz and Pietrzak [18] constructed an ElGamal encryption resilient to continual key leakage, which uses two insulated sub-states (independent memory devices) to store a secret key. Brakerski and Goldwasser [6] proposed a leakage-resilient PKE under the subgroup indistinguishability, which is a theoretical performance evaluation in improving the leakage rate.

Bellare *et al.* [5] considered the case of leaking the randomness that is sampled from a non-uniformly random distribution. Namiki *et al.* [23] stated that a PKE against a priori randomness-leakage attack can be constructed from any secure PKE but it is impossible for a posteriori randomness-leakage attack. Recently, Hazay *et al.* [16] proposed leakage-resilient secure transformations so as to construct a LR-PKE scheme from any standard public-key encryption, and a LR-PRF (Leakage-resilient ) scheme or LR-MAC scheme from any one-way function etc, which impose on minimum assumption to achieve the leakage resilience. These show the independent significance of our work.

## II. TECHNICAL PRELIMINARIES

### A. TERMINOLOGY AND NOTATION

The set of all binary strings of length $n$ bits is denoted as $\{0, 1\}^n$. We let $\lambda$ denote the *security parameter* and $\ell$ denote the *leakage parameter* (*i.e.*, *leakage bound*). A probabilistic polynomial-time algorithm, denoted by PPT, is an algorithm whose running time is bounded by some polynomial in $\lambda$ on all inputs. For a random variable $X$, we use notation $s \leftarrow S$ or $x \in X$ to denote sampling $x$ uniformly at random from $X$. We say that $X$ is *samplable*, if there exists a PPT algorithm with output distribution identical (statistically close) to the uniform distribution on $X$.

For two distributions $X$ and $Y$, we let $\Delta(X, Y)$ denote their *statistical distance*, *i.e.*, $\Delta(X, Y) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{a \in \mathcal{X}} |\Pr[X = a] - \Pr[Y = a]|$. We denote $X \approx_s Y$ as $X$ and $Y$ are *statistically close* if their statistical distance is negligible. A function is *negligible*(denoted $\varepsilon(\lambda)$) if it is smaller than the inverse of any polynomial, for all large enough value of $\lambda$. That is, for every positive polynomial $p(\cdot)$, for all sufficiently large $m \in N$, $\varepsilon(\cdot) < 1/p(n)$. Obviously, $1 - \varepsilon(\cdot)$ is overwhelming, if $\varepsilon(\cdot)$ is negligible.

We use $\mathbb{F}_q$ to denote the ring of integers modulo $q$. Let $\mathbb{G}$ denote a group of prime order $q$, such that computing

discrete logarithms in this group is infeasible. Let $g_1$ and $g_2$ denote independently chosen generators of $\mathbb{G}$, hence no party can gain the discrete logarithm of $g_1$ with respect to $g_2$. We can achieve selecting $g_1$ and $g_2$ by first choosing $g_1 \in \mathbb{G}$, $r \in \mathbb{F}_q$ randomly and then computing $g_2 = g_1^r$ and erasing the randomness $r$.

### B. INFORMATION ENTROPY AND STATISTICAL DISTANCE

*Definition 1 (k-Source Random Variable):* A k-source $X$ is said to be a random variable that taking values from $\{0,1\}^n$ for an integer $n$ that satisfies $\Pr[X = x] \leq 2^{-k}$.

*Lemma 1 (Difference Lemma, see [22], [25]):* Let $X_1, X_2$, $F$ be events defined over some probability distributions $\mathcal{X}$, and $X_1 \wedge \neg F = X_2 \wedge \neg F$. Then $\Pr[X_1] - \Pr[X_2] \leq \Pr[F]$.

*Definition 2 (Min-Entropy):* The min-entropy of a random variable $X$ is

$$\mathcal{H}_\infty(X) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x]) \tag{1}$$

The min-entropy represents the probability that an attacker has in guessing the outcome of $X$ given no additional information. Obviously, $\max_x \Pr[X = x] = 2^{-\mathcal{H}_\infty(X)}$.

A distribution $X$ is called $k$-source if it has min-entropy $\mathcal{H}_\infty(X) \geq k$. For a secret key $sk$, $sk$ is fully randomness (*i.e.*, fully hidden) if $k = |sk|$, and $sk$ is fully revealed if $k = 0$ which means that the key is totally leaked.

The average conditional min-entropy $\tilde{\mathcal{H}}_\infty(X|Z)$ of $X$ given $Z$ is represented the probability that an attacker has in guessing the outcome of $X$ given the outcome of a possibly correlated $Z$. That is,

*Definition 3 (Average Conditional Min-Entropy):* The average conditional min-entropy is defined by

$$\begin{aligned}\tilde{\mathcal{H}}_\infty(X|Z) &\stackrel{\text{def}}{=} -\log\left(\mathbb{E}_{z\leftarrow Z}\left[\max_x \Pr[X = x|Z = z]\right]\right) \\ &= -\log(\mathbb{E}_{z\leftarrow Z}[2^{-\mathcal{H}_\infty(X|Z=z)}])\end{aligned} \tag{2}$$

*Lemma 2:* Let $X, Y, Z$ be random variables and $Y$ takes $2^r$ possible values, then

$$\begin{aligned}\tilde{\mathcal{H}}_\infty(X|(Y,Z)) &\geq \tilde{H}_\infty((X,Y)|Z) \\ &\geq \tilde{H}_\infty(X|Z) - r \\ &\geq \mathcal{H}_\infty(X,Z) - r\end{aligned} \tag{3}$$

*Lemma 3:* Let $X, Y$ be random variables, then

$$\tilde{\mathcal{H}}_\infty(X|Y) \geq \mathcal{H}_\infty(X,Y) - \log|Y| \tag{4}$$

*Corollary 1:* For any map $f$ over $X$, $f : \{0,1\}^* \to \{0,1\}^\ell$ s.t. $\ell \leq \log X$, then

$$\tilde{\mathcal{H}}_\infty(X|f(X)) \geq \mathcal{H}_\infty(X) - \ell \tag{5}$$

*Corollary 2:* The probability distribution $\mathcal{X}$ has $(k,\epsilon)$-computational min-entropy $k$ if there exists a distribution $\mathcal{Y}$ with min-entropy $k$ such that $\mathcal{X}$ and $\mathcal{Y}$ are $(k,\epsilon)$-computationally indistinguishable.

### C. UNIVERSAL HASHING AND EXTRACTOR

An extractor $\mathsf{EXT}$ is a function to be applied to output from a weakly random entropy source. Informally, an extractor is defined as a function that takes as input a $k$-source variable and a randomness (called seed), and outputs a nearly-uniform bit string (the statistical distance between the extractor's output and a random string is very small). An extractor is strong if concatenating the seed with the extractor's output yields a distribution that is still close to uniform.

*Definition 4 (Strong Extractor):* A function $\mathsf{EXT} : \{0,1\}^n \times \mathsf{SEED} \to \{0,1\}^m$ is called an average-case $(k,\epsilon)$-randomness strong extractor if for all pairs random variables $(X,Z)$ such that $X$ is an $n$-bit string s.t. $\tilde{\mathcal{H}}_\infty(X|Z) \geq k$,

$$\Delta((\mathsf{EXT}(X,s),s,Z);\ (U,s,Z)) \leq \epsilon \tag{6}$$

where $s$ is at random picked in seed space $\mathsf{SEED}$ and $U$ is at random chosen from $\{0,1\}^m$.

Note that $\mathsf{EXT}(X,s)$ is nearly random given $s$ and $Z$ when $\epsilon$ is small enough. Intuitively, an extractor takes a weakly random $n$-bit but $k$-entropic input and a short and uniformly random seed, and produces an $m$-bit output that looks uniformly random, even to someone who sees part (but not all) of the source.

Dodis *et al.* [10], [11] proved that any extractor is in fact an average-case strong extractor, for an appropriate setting of the parameters.

*Lemma 4 (see [11]):* For any $\delta > 0$, if $\mathsf{EXT}$ is a (worst-case) $(k - \log(1/\delta), \epsilon)$- extractor, then $\mathsf{EXT}$ is also an average-case $(k, \epsilon + \delta)$-strong randomness extractor.

*Definition 5 (Universal Hashing):* A family of functions $\{H : \mathcal{X} \to \mathcal{Y}\}_{k\in\mathcal{K}}$ is called universal hash if $\Pr_{k\leftarrow\mathcal{K}}[H_k(x_1) = H_k(x_2)] \leq \frac{1}{|\mathcal{Y}|}$ for all distinct $x_1, x_2 \in \mathcal{X}$.

*Theorem 1:* Let $X$ be a random variable defined on $\mathcal{X} = \{0,1\}^n$ with min-entropy $\mathcal{H}_\infty(X) \geq k$, and $\mathcal{H} \stackrel{\text{def}}{=} \{\{0,1\}^n \to \{0,1\}^{k-2\epsilon}\}$ be a universal$_2$ class of hash function family. Let $X \leftarrow \mathcal{X}$ be randomly chosen from $\mathcal{X}$ and $h$ be randomly and uniformly chosen from $\mathcal{H}$. Then the distribution of $(h; h(X))$ is $2^{-\epsilon}$ close to the uniform distribution in the trace distance, i.e. application of a function randomly chosen from $\mathcal{H}$ is a $(k, 2^{-\epsilon})$-strong extractor.

*Lemma 5 (Leftover Hash Lemma, see [4], [17]):* Let $\mathcal{H} : \{h : \mathcal{X} \to \mathcal{Y}\}$ be a universal hash family and $U$ be a uniform distribution over field $\mathcal{Y}$. Let $f : \mathcal{X} \to \mathcal{Z}$ be any function. For any random variables $X \in \mathcal{X}$ and $Z \in \mathcal{Z}$, the statistical distance

$$\Delta((h, h(X), f(X)); (h, U, f(X))) \leq \tfrac{1}{2}\sqrt{\gamma(X) \cdot |Y| \cdot |Z|} \tag{7}$$

where $\gamma(X) \stackrel{\text{def}}{=} \max_x \Pr[X = x]$.

The above lemma means that if the right-hand side in Eq. (7) is negligible, and thus $h(X)$ is almost random even given $h$ and the leakage $f(X)$.

Actually, the leftover hash lemma indicates that a family of universal hash functions $\{\mathcal{H}_k\}$ can play the role of an average-case $(k,\epsilon)$-extractor $\mathsf{EXT} : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ with $\log|\mathcal{Y}| \leq l - 2\log(1/\epsilon) + 2$, in which $\mathcal{K}$ plays the seed family.

*Definition 6 (Decisional Diffie-Hellman Assumption (DDH)):* Let $\lambda$ be the security parameter and $\mathbb{G}$ be a finite

group of order $q$ determined by $\lambda$. Define two distributions $\mathcal{D} = (g_1, g_2, g_1^r, g_2^r)$ and $\mathcal{R} = (g_1, g_2, g_1^r, g_2^{r'})$ where $g_1, g_2 \in \mathbb{G}$, $r, r' \in \mathbb{F}_q$ and $r \neq r'$. The Diffie-Hellman assumption means that the advantage of any PPT attacker $\mathscr{A}$ (or algorithm) is negligible in distinguishing these two distributions, *i.e.*,

$$\Big|\Pr\left[\mathscr{A}((g_1, g_2, g_1^r, g_2^r) \leftarrow \mathcal{D}) = 1\right] - \Pr\left[\mathscr{A}((g_1, g_2, g_1^r, g_2^{r'}) \leftarrow \mathcal{R}) = 1\right]\Big| \leq \varepsilon(\lambda)$$

where $\varepsilon(\cdot)$ is a negligible function in parameter $\lambda$.

Actually, in the above definition, the elements in $\mathcal{D}$ form a valid Diffie-Hellman tuple, and the elements in $\mathcal{R}$ do not. Form the view of DDH assumption, it is computationally infeasible to distinguish between a valid tuple and an invalid one.

In cryptographic primitives, a key derivation function (namely KDF) derives one or more private keys from a secret value using a pseudo-random function, which prevents an attacker who obtains a derived key from learning useful information about either the input secret value or any of the other derived keys [19]. The KDF is specified by an international standard filed in ISO-18033-2.

### D. HASH PROOF SYSTEM

*Definition 7:* Let $\mathcal{S}$ and $\mathcal{L} \subset \mathcal{S}$ be sets, where $\mathcal{S}$ is the set of all ciphertexts and $\mathcal{L}$ is the set of all *valid* ciphertexts. Let $\mathcal{K}$ be the set of private keys. A hash proof system (namely HPS) $\mathsf{HPS} \stackrel{\text{def}}{=} (\mathsf{Key}, \mathcal{P}, \mathcal{V})$ is comprised of three algorithms:

- **Key generation algorithm** Key: On input a security parameter $\lambda$, this algorithm outputs a public-key and private-key pair $(pk, sk)$.
- **Encapsulation key algorithm** $\mathcal{P}$: On input a public key $pk$, a valid ciphertext $x \in \mathcal{L}$, and a witness $w$ of the fact that $x \in \mathcal{L}$, this algorithm outputs an encapsulated key $k \in \mathcal{K}$.
- **Decapsulation key algorithm** $\mathcal{V}$: On input a private-key $sk$ and a valid ciphertext $x \in \mathcal{L}$, this algorithm outputs a decapsulated key $k$.

The reader is referred to [8] for the detail about the hash proof system. Let the probability space defined by selecting $sk$ from the set of keys, we give the definitions of 1-universal HPS and 2-universal HPS as follows:

*Definition 8 (1-Universal HPS):* A hash proof system is 1-universal if $\forall x \in \mathcal{S}\backslash\mathcal{L}$ and $k \in \mathcal{K}$, $\Pr[\mathcal{P}(sk, x) = k] = \frac{1}{|\mathcal{K}|}$.

*Definition 9 (2-Universal HPS):* A hash proof system is 2-universal if $\forall x_1, x_2 \in \mathcal{S}\backslash\mathcal{L}$ and $k_1, k_2 \in \mathcal{K}$ such that $x_1 \notin \mathcal{L} \cup \{x_2\}$, $\Pr[\mathcal{P}(sk, x_1) = k_1 | \mathcal{P}(sk, x_2) = k_2] = \frac{1}{|\mathcal{K}|}$.

*Definition 10 (k-Entropic HPS):* A hash proof system is $k$-entropic if $\forall x \in \mathcal{S}\backslash\mathcal{L} : \tilde{\mathcal{H}}_\infty(\mathcal{P}(sk, x)|pk, x) \geq k$.

## III. SYNTAX OF LEAKAGE-RESILIENT PKE IN THE PRESENCE OF KEY LEAKAGE

We start by defining the syntax of PKE and the leakage-resilient security requirement in the presence of key leakage.

*Definition 11:* A public-key encryption scheme $\pi$ is comprised of three regular PPT algorithms:

- **Key generation algorithm** Key: On input a security parameter $\lambda$, the algorithm produces a pair $(pk, sk)$ of matching public and private keys.
- **Encryption algorithm** Enc: On input a message $m$ and a public key $pk$, this algorithm produces a ciphertext $ct$ of $m$. Note that this algorithm may be probabilistic (involving random coins $r \in \mathbb{F}_p$, and then denoted $\mathsf{Enc}_{pk}(m; r)$)
- **Decryption algorithm** Dec: On input a ciphertext $ct$ and the private-key $sk$, this algorithm outputs the plaintext $m$ if succeeds and output $\perp$ otherwise.

*Definition 12 (Perfect Correctness):* Let $ct \in \mathcal{C}$ be a set of valid ciphertexts of plaintext $m \in \mathcal{M}$ under a given public key $pk$. For all correctly generated ciphertext: for all $ct \leftarrow \mathsf{Enc}(pk; m)$ with $m \in \mathcal{M}$, then $\Pr[m \leftarrow \mathsf{Dec}(sk; ct)] = 1$ for every $ct \in \mathcal{C}$.

We model the key leakage by providing the attacker with access to a leakage oracle: the attacker submits any polynomial-time computable function $f$ (modeled as leakage function) and receives $f(sk)$, with only the constraint that the sum of all the leakage is bounded by a predetermined bound $\ell$. Also, we allow the attacker to gain the leakage w.r.t the previous knowledge such as leakage, public-key and the other obtainable and available states etc. More concretely, the attacker can select different leakage function $f_j$ for the $j$-query adaptively, with the restriction that $\sum_j f_j(sk) \leq \ell$.

*Definition 13 (Key Leakage Oracle $\mathcal{O}_{\mathsf{Leak}}$):* A key leakage oracle $\mathcal{O}_{\mathsf{Leak}}$ is parameterized by a leakage parameter $\ell$ and a private-key $sk$. A query to the leakage oracle $\mathcal{O}_{\mathsf{Leak}}$ is taken a function $f_j$ and a key $sk$ as inputs, and the oracle computes $f_j(sk)$. The oracle $\mathcal{O}_{\mathsf{Leak}}$ returns at most $\ell$-bit for one key $sk$, and ignores all queries afterwards.

In the simulation of security proof, we can use a queue to record all the leakage of queries that the attacker makes, and sum the total leakage for each queried key to specify that the leakage oracle $\mathcal{O}_{\mathsf{Leak}}$ is correctly executed (the leakage of each key is no more than $\ell$ bits) like in Definition 13.

*Definition 14 (Key-Leakage Attack Security Prior to Chosen-Ciphertext):* A public-key encryption $\pi = (\mathsf{Key}, \mathsf{Enc}, \mathsf{Dec})$ is said to be IND-lrCCA-secure in the presence of $\ell$-bit key-leakage if for any probabilistic polynomial-time algorithm $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ holds that

$$Adv_{\mathscr{A}}^{\mathsf{IND\text{-}lrCCA}}(\lambda) \stackrel{\text{def}}{=} \Big|\Pr[Exp_{\mathscr{A}}^{\mathsf{IND\text{-}lrCCA}}(\lambda) = 0] - \Pr[Exp_{\mathscr{A}}^{\mathsf{IND\text{-}lrCCA}}(\lambda) = 1]\Big|$$

is negligible in security parameter $\lambda$, *i.e.*, $Adv_{\mathscr{A}}^{\mathsf{IND\text{-}lrCCA}}(\lambda) \leq \varepsilon(\lambda)$, where the code-based presentation of the game $Exp_{\mathscr{A}}^{\mathsf{IND\text{-}lrCCA}}$ is formally given as follows:

1. $(pk, sk) \leftarrow \mathsf{Key}(\lambda)$
2. $(m_0, m_1, aux) \leftarrow \mathscr{A}_1^{\mathcal{O}_{\mathsf{Dec}}(\cdot), \mathcal{O}_{\mathsf{Leak}}(\cdot)}(pk)$
   $s.t. |m_0| = |m_1| and |\mathcal{O}_{\mathsf{Leak}}| \leq \ell$
3. $b^* \leftarrow \{0, 1\}$

4. $ct^* \leftarrow \mathsf{Enc}_{pk}(m_{b^*})$
5. $b \leftarrow \mathscr{A}_2^{\mathcal{O}_{\mathsf{Dec}(\cdot)}}(ct^*, aux)$
6. $return(b = b^*)$

Actually, the experiment as above is an algorithm that, when run on $1^\lambda$, given the adversary $\mathscr{A}$ that can have access to the oracles $\mathcal{O}_{\mathsf{Dec}}(\cdot)$ and $\mathcal{O}_{\mathsf{Leak}}(\cdot)$ to get some (helpful) auxiliary information, eventually outputs either a zero or one.

## IV. CONCRETE CONSTRUCTION OF KEY-LEAKAGE RESILIENT PKE

Let $\lambda$ be the security parameter, $\mathbb{G}$ be a cyclic group of order $q$, and an injective key derivation function $\mathsf{KDF}$: $\mathbb{G} \to \{0,1\}^{2\lambda}$. Dodis *et al.* [13] provided an efficient design of key-derivation function that taking an imperfect source as input and outputs a uniformly distributed string with less entropy waste, which can fit our scheme. Let $(\mathsf{SYE}(k; m), \mathsf{SYD}(k; c))$ be a CCA secure private-key encryption scheme under symmetric key $k$, in which $m$ is a plaintext and $c$ is a decryptable ciphertext. The construction of key-leakage resilient PKE is described as follows:

- $\mathsf{Key}(\lambda)$ The key generation algorithm creates the public-key and private-key as follows:

K1. Taking the security parameter $\lambda$ as input, generate a cyclic group description $\mathcal{T} = (\mathbb{G}, g, q)$, where $g$ is a generator of $\mathbb{G}$ and $q$ (with length $\log q = \lambda$) is the order of $\mathbb{G}$;
K2. At random select two generators $g_1, g_2 \leftarrow \mathbb{G}$;
K3. At random select $a_1, a_2, b_1, b_2 \leftarrow \mathbb{F}_q$;
K4. Calculate $c = g_1^{a_1} g_2^{a_2}$ and $d = g_1^{b_1} g_2^{b_2}$;
K5. Choose a target collision resistant function $\mathsf{TCR} : \mathbb{G}^2 \times \mathbb{F}_q \to \mathbb{F}_q$, and an injective key derivation function $\mathsf{KDF} : \mathbb{G} \to \mathbb{F}_q^2$;
K6. Choose a private-key encryption $\mathcal{E} = (\mathsf{SYE}(\cdot), \mathsf{SYD}(\cdot))$;
K7. Set the private-key as $sk = (a_1, a_2, b_1, b_2) \in \mathbb{F}_q^4$;
K8. Publish the public-key $pk = (\mathcal{T}, g_1, g_2, c, d, \mathsf{TCR}, \mathsf{KDF}, \mathcal{E})$;

- $\mathsf{Enc}_{pk}(m)$ Let $m \in \mathbb{G}$ be the plaintext. The encryption algorithm proceeds the procedure as follows:

E1. Select $r, s \leftarrow \mathbb{F}_q$ randomly;
E2. Calculate $u_1 = g_1^r$ and $u_2 = g_2^r$;
E3. Calculate $\alpha = \mathsf{TCR}(u_1, u_2, s)$ and $(k_0|k_1) \leftarrow \mathsf{KDF}(c^r d^{r\alpha})$;
E4. Calculate $e = m \cdot (cd^s)^{k_0}$;
E5. Calculate $w = \mathsf{SYE}(k_1; e)$;
E6. Output $ct = (u_1, u_2, e, w, s) \in \mathbb{G}^4 \times \mathbb{F}_q$.

- $\mathsf{Dec}_{sk}(ct)$ On input a ciphertext $ct = (u_1, u_2, e, w, s)$ and a private-key $sk = (a_1, a_2, b_1, b_2)$, the decryption algorithm does the follows:

D1. Calculate $\alpha = \mathsf{TCR}(u_1, u_2, s)$;
D2. Calculate $v = u_1^{a_1+b_1\alpha} u_2^{a_2+b_2\alpha}$;
D3. Calculate $(k_0|k_1) \leftarrow \mathsf{KDF}(v)$;
D4. Verify $w \stackrel{?}{=} \mathsf{SYD}(k_1; e)$, return $\perp$ if the test fails;
D5. Output $m = \frac{e}{c^{k_0} d^{k_0 s}}$.

Correctness. Suppose that $sk = (a_1, a_2, b_1, b_2)$ is a valid decryption key for a ciphertext $ct = (u_1, u_2, e, w, s)$, then

$$\begin{aligned} v &= u_1^{a_1+b_1\alpha} u_2^{a_2+b_2\alpha} = g_1^{ra_1+rb_1\alpha} g_2^{ra_2+rb_2\alpha} \\ &= (g_1^{a_1} g_2^{a_2})^r (g_1^{b_1} g_2^{b_2})^{r\alpha} = c^r d^{r\alpha} \end{aligned} \tag{8}$$

Compared with [20] and [25], in our scheme, the plaintext space is the finite group $\mathbb{G}$ instead of arbitrary binary string $\{0,1\}^*$, which is independent to the leakage parameter $\ell$.

## V. SECURITY

### A. PROOF IDEA

The security guarantees that the attacker must not guess the plaintext $m_b$ from the ciphertext $ct$. Suppose that $ct^* = (u_1^*, u_2^*, e^*, w^*, s^*)$ be the challenge ciphertext in the $\mathsf{IND\text{-}IrCCA2}$ game, *i.e.*, $ct^* = \mathsf{Enc}_{pk}(m_b)$ for randomly picked $m_b$ ($b \in \{0,1\}$) from the attacker's challenge pair $(m_0, m_1)$.

Note that a ciphertext $ct$ is called a *valid* ciphertext if $(g_1, g_2, u_1, u_2)$ is a valid Diffie-Hellman tuple. Otherwise, it is called *invalid*, that is, $u_1 = g_1^{r_1}$, $u_2 = g_2^{r_2}$ and $r_1 \neq r_2$. Also, $ct$ is called *consistent* if it succeeds in verification equation in decryption algorithm, and is *inconsistent* if fails. If a key can decrypt a ciphertext, the ciphertext should be valid and consistent w.r.t the key.

We will use a series of games to prove the security, in which each game is a slight modification of the previous one. In the first game, the ciphertext is the actual challenge ciphertext $ct^* = \mathsf{Enc}_{pk}(m_b)$. Obviously, in this case, $r_1 = r_2 = r$ and the ciphertext is valid and consistent for the challenge ciphertext. Then, we convert the challenge ciphertext $ct^*$ into an invalid but consistent ciphertext by selecting $r_1 \neq r_2$. The attacker cannot detect this change because of the Diffie-Hellman problem in Definition 6. More concretely, $(g_1, g_2, g_1^{r_1}, g_2^{r_1}) \approx_c (g_1, g_2, g_1^{r_1}, g_2^{r_2})$ for $r_1 \neq r_2$.

In the view of attacker, the useful information about the private-key $sk$ come from fourfold:

1. the public key $pk$,
2. the challenge ciphertext $ct^*$,
3. the leakage $f(sk)$ from leakage oracle $\mathcal{O}_{\mathsf{Leak}}$ and,
4. possible decryption oracle $\mathcal{O}_{\mathsf{Dec}}$.

Actually, in case $(iv)$ the attacker gains no more private-key information during querying a decryption oracle of a valid ciphertext but for the message. In the next game, we convert the challenge ciphertext into an inconsistent one, and prove that, given $(c, d, ct^*)$ and at most $\ell$-bit of a valid private-key w.r.t the challenge ciphertext $ct^*$, the probability of an invalid ciphertext passing through the verification is negligible, which is derived from the average min-entropy of the private-key by the strong extractor and the collision resilience of $\mathsf{TCR}$.

In the challenge ciphertext, the component $e^*$ is for ephemeral key extraction to mask the plaintext, $w^*$ is for the consistency verification. In the above game, we show that the attacker cannot decrypt the challenge ciphertext after giving some leakage (cannot pass through the consistency check).

**TABLE 1. Indistinguishable games for security proofs.**

| *Game* | *Functionality* | *Hard ass.* | *Remarks* |
|---|---|---|---|
| $\text{Game}_0$ | IND-lrCCA in $Exp_{\mathscr{A}}^{\text{IND-lrCCA}}$ | – | Actual scheme and security model (*consistent* and *valid* ciphertext) |
| $\text{Game}_1$ | revise $ct^*$ with DDH key ($r_1 = r_2$) | trivial | |
| $\text{Game}_2$ | $ct^*$ to inconsistent ($r_1 \neq r_2$) | DDH | *invalid* ciphertext |
| $\text{Game}_3$ | revise $ct^*$ with unchanged $\alpha$ | TCR | $ct^* = (u_1, u_2, e^*, w^*, s)$ but $\mathsf{TCR}(u_1, u_2, s) = \alpha^*$ |
| $\text{Game}_4$ | revised with new $\alpha^*$ ($\alpha^* \neq \alpha$) | HPS/KDF | Universal hash as extractor, *inconsistent* ciphertext |
| $\text{Game}_5$ | replace $e^*$ with a randomness in $\mathbb{G}$ | HPS | *Inconsistent* and *invalid* ciphertext (randomized value to $\mathscr{A}$) |

At the final game, we consider whether the plaintext component $e^*$ reveals the plaintext or not. Actually, $e^*/m_b = c^{k_0} d^{k_0 s}$, which is also an output of strong extractor, *i.e.*, $\mathsf{EXT}((c^{k_0}, d^{k_0}); s)$. $k_0$ and $k_1$ can model as randomness if $v$ has high entropy. $k_0|k_1$ is generated by a key derivation function and has high entropy, since $v = c^r d^{r\alpha}$ is computed by a strong extractor (universal hash).

### B. SECURITY PROOF

We define a series of games ant their functionalities in Table 1. The first game $\text{Game}_0$ is our actual construction, which is well-formed to one that has private-key $sk$. In the last game $\text{Game}_6$, all components of the challenge ciphertext $ct^*$ are formed an invalid and inconsistent ciphertext with respect to the private-key $sk$. That is, in the view of the attacker, this ciphertext is randomized. Obviously, if these games are computationally indistinguishable, we can prove that the attacker has negligible probability advantage in attacking our scheme. We have the following theorem:

*Theorem 2:* Let $\lambda$ be the security parameter for a finite group $\mathbb{G}$ of order $q$, *i.e.*, $q = 2^\lambda$, and $\ell$ be the allowable leakage for the private-key. Let $\mathsf{TCR}$ be a target collision-resistant hash function. If the decisional Diffie-Hellman problem in $\mathbb{G}$ is hard and $\mathcal{E}$ is semantic secure, our scheme is $(\lambda, \ell, \epsilon)$-IND-lrCCA secure public key encryption, such that

$$\begin{cases} \ell \leq \log q - \omega(\log \lambda) \leq \dfrac{1}{4}|sk| \\ \epsilon \leq Adv_{\mathscr{A}_1}^{\mathsf{DDH}}(\lambda) + Adv_{\mathscr{A}_2}^{\mathsf{TCR}}(\lambda) + 2^{\ell-\lambda} + 2^{\frac{\ell-\lambda}{2}-1} \\ \qquad\qquad + \dfrac{Q \cdot 2^{\ell-\lambda}}{1 - Q/2^\lambda} \end{cases} \tag{9}$$

where $Q$ is the number of decryption queries that the attacker makes, $Adv_{\mathscr{A}_1}^{\mathsf{DDH}}(\cdot)$ and $Adv_{\mathscr{A}_2}^{\mathsf{TCR}}(\cdot)$ denote the advantage that the attacker breaks the DDH assumption and target collision resistant function, respectively.

*Proof:* Let $X_i$ denote the event of Game $i$. We deploy the adversary $\mathscr{A}$ as a sub-algorithm to solve DDH problem if $\mathscr{A}$ can successfully break our scheme with non-negligible advantage $\epsilon$.

The $\text{Game}_0$ models the IND-lrCCA experiment in Definition 14. We let the components in challenge ciphertext to label with character $*$. That is, $ct^* = (u_1^*, u_2^*, e^*, w^*, s^*) = \mathsf{Enc}_{pk}(m_b)$, in which the intermediate values are specified by $r^*, \alpha^*, v^*, k_0^*$ and $k_1^*$. In the view of the attacker, these intermediate variables are fully hidden.

In $\text{Game}_1$, knowing the private-key $sk = (a_1, a_2, b_1, b_2)$, the challenger replaces the component $v^*$ by

$$(u_1^*)^{a_1 + b_1\alpha^*} (u_2^*)^{a_2 + b_2\alpha^*}$$

where $u_1^* = g_1^{r^*}$, $u_2^* = g_2^{r^*}$ and $\alpha^* = \mathsf{TCR}(u_1^*, u_2^*, s^*)$ for randomly selected $r^*, s^* \in \mathbb{F}_q$. As this game does not change the value of $v^*$ (note that $v^* = (cd^{\alpha^*})^{r^*} = (u_1^*)^{a_1 + b_1\alpha^*} (u_2^*)^{a_2 + b_2\alpha^*}$), then $\Pr[X_0] = \Pr[X_1]$.

The $\text{Game}_2$ is the same as $\text{Game}_1$ except replacing $(u_1^*, u_2^*)$ with $(g_1^{r_1^*}, g_2^{r_2^*})$ for randomly selected distinct $r_1^*, r_2^* \in \mathbb{F}_q$. We show that $\Pr[X_2] - \Pr[X_1]$ is negligible under the decisional Diffie-Hellman assumption, since $(g_1, g_2, g_1^{r^*}, g_2^{r^*}) \approx_c (g_1, g_2, g_1^{r_1^*}, g_2^{r_2^*})$. That is, $|\Pr[X_2] - \Pr[X_1]| \leq Adv_{\mathscr{A}}^{\mathsf{DDH}}(\lambda)$.

In $\text{Game}_3$, the response of decryption oracle will reject all queries such that (*i*) queried decryption ciphertext $ct = (u_1, u_2, e^*, w^*, s) \neq ct^*$; (*ii*) $\mathsf{TCR}(u_1, u_2, s) = \alpha^*$. The probability of $|\Pr[X_3] - \Pr[X_2]|$ is close to the hash collision occurring of $\mathsf{TCR}$. Let $X_D$ be the event that $\text{Game}_3$ rejects a decryption query. According to Difference Lemma 1, $|\Pr[X_3] - \Pr[X_2] \leq \Pr[X_D]| \leq Adv_{\mathscr{A}}^{\mathsf{TCR}}(\lambda)$.

In $\text{Game}_4$, the response of decryption oracle will reject the query if $ct \neq ct^*$ and $\alpha = \mathsf{TCR}(u_1, u_2, s) \neq \alpha^*$. We show that $\Pr[X_4] - \Pr[X_3]$ is negligible by proving this type of ciphertext is rejected with overwhelming probability like in $\text{Game}_3$. For the view of the attacker, the possible information about the system status is derived from the following aspects:

1. public key $(q, g_1, g_2, c, d)$;
2. challenge plaintext $(m_0, m_1)$;
3. challenge ciphertext $ct^* = (u_1^*, u_2^*, e^*, w^*, s^*)$;
4. $k_0$ and $k_1$ from the challenge ciphertext $ct^*$;
5. queried $\ell$-bit leakage from leakage oracle $\mathcal{O}_{\mathsf{Leak}}$;
6. information from decryption oracle $\mathcal{O}_{\mathsf{Dec}}$.

We now discuss the components that contain the sensitive information of private-key $sk = (a_1, a_2, b_1, b_2)$. In the public key $pk$, $g_1$ and $g_2$ are not involved in any secret value in private key $sk$ (their distribution are independent), and the elements $c = g_1^{a_1} g_2^{a_2}$ and $d = g_1^{b_1} g_2^{b_2}$ implicitly conceal the private-key values. For the same reason, we can see that only components $e^*$ and $w^*$ in challenge ciphertext $ct^*$ implicitly contain the information of secret key $sk$, and the access to leakage oracle explicitly reveals the sensitive $\ell$-bit private-key. Note that in $\text{Game}_4$ $ct^*$ is changed as $ct^* = (u_1, u_2, e^*, w^*, s)$ where $u_1 \neq u_1^*, u_2 \neq u_2^*$ and $s \neq s^*$.

$$
\begin{aligned}
&\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)\mid c,d,ct^*,m_b,m_{1-b},\ell\text{- bit leakage}\big)\\
&=\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)\mid c,d,ct^*,m_b,\ell\text{-leakage}\big) && \boxed{m_{1-b}\text{ leaks no information about the key}}\\
&=\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)|c,d,u_1^*,u_2^*,w^*,s^*,e^*/m_b,\ell\text{-bit leakage}\big)\\
&=\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)|c,d,w^*,e^*/m_b,\ell\text{-bit leakage}\big) && \boxed{u_1^*,u_2^*\text{ and }s^*\text{ are independent to the key}}\\
&\geq\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)|c,d,w^*,e^*/m_b\big)-\ell && \boxed{\text{Lemma 2, Corollary 1}}\\
&\geq\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)|c,d,e^*/m_b\big)-\ell && \boxed{\textbf{SYE/SYD}\text{ decided by key }k_0}\\
&\geq\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)|e^*/m_b\big)-2\log q-\ell && \boxed{c\text{ and }d\text{ lose of entropy at most }2\log q}\\
&\geq 3\log q-2\log q-\ell && \boxed{e^*/m\text{ loses of entropy at most }\log q}\\
&=\log q-\ell
\end{aligned}
\tag{11}
$$

However, replaced elements $u_1, u_2$ and $s$ do not involved in the private key $sk$, then at this point the query to decryption oracle does not gain any useful information for the private-key $sk$.

Let $X_E$ be the event that the decryption oracle outputs reject, then $\Pr[X_4 \wedge \neg X_E] = \Pr[X_3 \wedge \neg X_E]$. By Lemma 1, $|\Pr[X_4] - \Pr[X_3]| \leq \Pr[X_E]$. We now give the probability that $X_E$ occurs.

Assume that $\mathscr{A}$ obtains $\ell$-bit leakage about the private-key from $f(sk)$. Given $f(sk)$ and the tuple $(c, d)$, the entropy of $v^*$ is at least $\log q - \ell$ (*i.e.*, $v^*$ is the output of universal hash and acts as an output of strong extractor), and thus the entropy of $k_0$ and $k_1$ are also $\log q - \ell$ because of the property of **KDF** (*i.e.*, $(k_0|k_1) \leftarrow \mathbf{KDF}(v^*)$, and **KDF** is an injective function). The entropy of $k_0$ and $k_1$ are $\log q - \ell$ initially, then the probability of finding a tuple $(K_{0,j}, K_{1,j}) \in \mathbb{F}_q^2$ satisfying $K_{0,j} = k_0$ and $K_{1,j} = k_1$ for $j$-th search is $\Pr[K_{0,j} = k_0, K_{1,j} = k_1] \leq 2^{-(\log\,(q+1-j)-\ell)} = \frac{2^\ell}{q+1-j}$. That is, seeing $\ell$-bit private-key and the test of **KDF**, the probability of finding correct symmetric key in $\mathcal{E}$ (*i.e.*, produce a consistent challenge ciphertext and pass the verification in decryption algorithm) is

$$Adv_{\mathscr{A}}^{\mathbf{KDF}} = \Pr[K_{0,j} = k_0, K_{1,j} = k_1] \leq \frac{2^\ell}{q+1-j} \tag{10}$$

Without leakage, the entropy of $sk$ is at least $2\log q$ (the entropy of $sk$ is $4\log q$ since $sk$ has four elements in $\mathbb{F}_q$ but the public-key $c$ and $d$ leak less than $2\log q$ according to Lemma 3). At this point, the reminder entropy of private-key $(a_1, a_2, b_1, b_2)$ conditioned by public-key, challenge ciphertext and the leakage is analyzed in Eq. 11, as shown at the top if this page.

That is, gaining the knowledge from the public key $pk$, the challenge ciphertext $ct^*$ and the $\ell$-bit private-key, the probability of an attacker guessing the private-key $sk$ is at most $2^\ell/q$.

We now continue to consider the decryption query for an invalid ciphertext (*i.e.*, $(g_1, g_2, u_1, u_2)$ not a valid DDH tuple) for $ct \neq ct^*$ and $\alpha \neq \alpha^*$. Suppose that $\mathscr{A}$ queries an invalid ciphertext $ct = (u_1, u_2, e, w, s)$ with $u_1 = g_1^{r_1}$ and $u_2 = g_2^{r_2}$. Assume that $g_2 = g_1^\gamma$ and let $v = u_1^{a_1+b_1\alpha} u_2^{a_2+b_2\alpha}$, then

$$\begin{bmatrix}\log_{g_1} c\\ \log_{g_1} d\\ \log_{g_1} v^*\\ \log_{g_1} v\end{bmatrix} = \underbrace{\begin{bmatrix}1 & 0 & \gamma & 0\\ 0 & 1 & 0 & \gamma\\ r_1^* & r_1^*\alpha^* & r_2^*\gamma & r_2^*\gamma\alpha^*\\ r_1 & r_1\alpha & r_2\gamma & r_2\gamma\alpha\end{bmatrix}}_{M} \times \begin{bmatrix}a_1\\ b_1\\ a_2\\ b_2\end{bmatrix} \tag{12}$$

In Eq.12, $\det(M) = \gamma^2(r_2^* - r_1^*)(r_2 - r_1)(\alpha^* - \alpha) \neq 0$ since $r_2^* \neq r_1^*$, $r_2 \neq r_1$ and $\alpha^* \neq \alpha$. This means that $v$ is random from the attacker's point of view. As for a further comment, the attacker cannot produce a new $v$ with the linear combination in exponent of $c, d$ and $v^*$. Let $Q$ be the number of decryption queries that the attacker makes, according to Eq.10, the $j$-th invalid ciphertext is accepted by the distinguisher with probability at most $\frac{2^\ell}{q-j+1}$. The bound of probability of $\Pr[X_E]$ with $Q$ decryption queries is $\frac{2^\ell}{q/Q-1}$. Thus,

$$|\Pr[X_4] - \Pr[X_3]| \leq Adv_{\mathscr{A}}^{\mathbf{KDF}} + \frac{2^\ell}{q/Q-1} = \frac{2^\ell}{q} + \frac{2^\ell}{q/Q-1}$$

In $\text{Game}_5$, all queried invalid ciphertexts are rejected by the decryption oracle (like the end of $\text{Game}_4$), but the encrypted plaintext component $e^*$ is replaced by a random element in $\mathbb{G}$. Note that after this change all the components in the challenge ciphertext are random except the seed $s$. However, the seed is public and does not influence the security. Like in $\text{Game}_4$, the decryption oracle cannot help the attacker gain more information about the private-key. We only consider the public key $pk$, the leakage and the related components $(e^*, w^*)$ in the challenge ciphertext $ct^*$.

We now show that $e^*/m_b$ provides a $(2\log q - \ell, \delta)$-extractor with $\mu = (u_1^*)^{a_1}(u_2^*)^{a_2}$ and $\nu = (u_1^*)^{b_1}(u_2^*)^{b_2}$ as inputs in universal hashing defined in 5 and sampled in Section II-C. Given $(c, d, w^*, \ell\text{-leakage})$, the conditional min-entropy $\tilde{\mathcal{H}}_\infty(a_1, a_2, b_1, b_2)$ is calculated as follows:

$$
\begin{aligned}
&\tilde{\mathcal{H}}_\infty\big(\mu,\nu|c,d,w^*,\ell\text{-bit leakage}\big)\\
&=\tilde{\mathcal{H}}_\infty\big((u_1^*)^{a_1}(u_2^*)^{a_2},(u_1^*)^{b_1}(u_2^*)^{b_2}|c,d,w^*,\ell\text{-bit leakage}\big)\\
&=\tilde{\mathcal{H}}_\infty\big((a_1,a_2,b_1,b_2)|c,d,w^*,\ell\text{-bit leakage}\big)\\
&\qquad\boxed{\text{injection from }(a_1,a_2,b_1,b_2)\text{ to }c\text{ and }d}
\end{aligned}
$$

$$\geq \tilde{\mathcal{H}}_{\infty}\big((a_1, a_2, b_1, b_2)|c, d, w^*\big) - \ell$$

Lemma 2, Corollary 1

$$\geq 2\log q - \ell \tag{13}$$

Define a universal hash $H_s(\mu, \nu) = \mu \cdot \nu^s$ as a $(2\log q, \delta)$-extractor, we have

$$\begin{aligned}\frac{w^*}{m_b} &= H_s(\mu, \nu) = H_s((u_1^*)^{a_1}(u_2^*)^{a_2}, (u_1^*)^{b_1}(u_2^*)^{b_2}) \\ &= (u_1^*)^{a_1}(u_2^*)^{a_2}((u_1^*)^{b_1}(u_2^*)^{b_2})^s \\ &= (u_1^*)^{a_1+b_1 s}(u_2^*)^{a_2+b_2 s}\end{aligned} \tag{14}$$

By the leftover hash lemma in Lemma 5, the statistical distance between $w^*$ and a uniformly distributed randomness $U \in \mathbb{G}$ is

$$\Delta(w^*, U) \leq \frac{1}{2}\sqrt{q \cdot \frac{2^\ell}{q^2}} = \frac{\sqrt{2^\ell/q}}{2} = \sqrt{\frac{2^{\ell-2}}{q}} \tag{15}$$

As $U$ is randomly selected from $\mathbb{G}$, and thus $b$ is information-theoretically hidden from $w^*/m_b$. That is, $\Pr[X_5] = 1/2$ to output $b = 0$ or $b = 1$. At the same time, $\Pr[X_5] - \Pr[X_4] \leq \delta = \sqrt{2^{\ell-2}/q}$. Note that $\lambda$ is the system security parameter. When setting $q = 2^\lambda$ is large enough and thus $\lambda \gg \ell$, *i.e.*,

$$\delta = \sqrt{2^{\ell-2}/q} = 2^{\frac{\ell-\lambda-2}{2}} \approx 0 \tag{16}$$

Taken from $\text{Game}_0$ to $\text{Game}_5$ into account, we obtain that the scheme is leakage-resilient secure under the bound $\ell \leq \log q - \omega(\log\lambda)$ with negligible advantage

$$\begin{aligned}\epsilon &\leq Adv^{\mathsf{DDH}}_{\mathscr{A}_1}(\lambda) + Adv^{\mathsf{TCR}}_{\mathscr{A}_2}(\lambda) + \frac{2^\ell}{q} + \frac{2^\ell}{q/Q - 1} + \sqrt{2^{\ell-2}/q} \\ &= Adv^{\mathsf{DDH}}_{\mathscr{A}_1}(\lambda) + Adv^{\mathsf{TCR}}_{\mathscr{A}_2}(\lambda) + 2^{\ell-\lambda} + \frac{Q \cdot 2^\ell}{2^\lambda - Q} + 2^{\frac{\ell-\lambda-2}{2}} \\ &= Adv^{\mathsf{DDH}}_{\mathscr{A}_1}(\lambda) + Adv^{\mathsf{TCR}}_{\mathscr{A}_2}(\lambda) + 2^{\ell-\lambda} + 2^{\frac{\ell-\lambda-2}{2}} + \frac{Q \cdot 2^{\ell-\lambda}}{1 - Q/2^\lambda}\end{aligned} \tag{17}$$

in security parameter $\lambda \in \mathbb{F}^+$ and order $q = 2^\lambda$.

## VI. EXTENSION, PERFORMANCE AND DISCUSSION

### A. ACHIEVING RANDOMNESS-LEAKAGE RESILIENCE

In the encryption, the randomness is possibly sampled from a non-uniformly random distribution [31] or the randomness is partially leaked. Yu *et al.* [31] considered the leakage-resilient PRG assuming a non-adaptive leakage function and a small public memory. Bellare *et al.* [5] discussed a condition that the random string in the encryption is leaked but an entropically guaranteed distribution for the randomness. In this section, we construct a public-key encryption scheme even if the randomness used in the encryption algorithm is leaked, as long as the key leakage. The leakage function $g$ of randomness is arbitrary with the restriction that the output length is bounded by a predetermined parameter.

In the encryption algorithm in Section IV, the used randomness $r$ is selected in $\mathbb{F}_q$ and $s$ is chosen from the seed field of $\mathsf{SEED}$. However, the randomness $s$ acts as the seed in the strong extractor and is also public in the ciphertext, thus the leakage of $s$ will not degrade the security. We only consider the leakage of randomness $r$.

We provide a construction of public-key encryption resilient to both key leakage and randomness leakage. Differing to the key leakage prior to the chosen-ciphertext, we consider the randomness leakage occurs that is prior to the public-key generation, which is a weak randomness leakage. The weak randomness leakage is possible in practical applications such as smart grids and wireless sensor networks. For example, in the statefull encryption in smart grids, the nodes store lots of early generated randomness, and a randomness is randomly selected from the randomness list to create the key and perform the encryption. As the randomness are stored in the nodes, they are facing lots of threats such as being monitored and leaked.

Let $\pi = (\mathsf{Key}, \mathsf{Enc}, \mathsf{Dec})$ be the key-leakage resilient PKE proposed in Section IV, and $\widehat{\mathsf{EXT}} : \{0,1\}^{\log q} \times \mathsf{SEED} \to \{0,1\}^{\log q}$ be an average-case randomness extractor. Actually, this extractor can be constructed by a universal hash defined in Section IV. We give the public-key encryption construction $\widehat{\pi} = (\widehat{\mathsf{Key}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Dec}})$ resilient to both key leakage and randomness leakage, which uses $\pi$ as a building block. Note that it is straightforward to transform any key-leakage resilient PKE into the scheme resilient to both weak-randomness leakage and key leakage.

- $\widehat{\mathsf{Key}}$: At first call $\mathsf{Key}(1^\lambda)$ to generate $(pk, sk)$, and at random select $t \in \mathbb{F}_q$. Output the public key $\widehat{pk} = (pk, t)$ and the private-key $\widehat{sk} = sk$.
- $\widehat{\mathsf{Enc}}_{pk}(m)$: At random select $r \in \mathbb{F}_q$, and output the ciphertext $\widehat{ct} = \mathsf{Enc}_{pk}(m; \widehat{\mathsf{EXT}}(r; t))$.
- $\widehat{\mathsf{Dec}}_{sk}(ct)$: Output $\mathsf{Dec}_{sk}(\widehat{ct})$.

*Theorem 3:* Let $\pi = (\mathsf{Key}, \mathsf{Enc}, \mathsf{Dec})$ be a $(\lambda, \ell_1, \epsilon_1)$-$\mathsf{IND\text{-}lrCCA}$ secure PKE, and $\widehat{\mathsf{EXT}}$ be an average-case $(\log q - \ell_2, \epsilon_2)$-strong extractor. Then the public-key encryption $\widehat{\pi} = (\widehat{\mathsf{Key}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Dec}})$ is $(\lambda, \ell_1, \ell_2, \epsilon_1 + \epsilon_2)$-$\mathsf{IND\text{-}lrCCA}$ secure resilient to $\ell_1$-bit key-leakage and $\ell_2$-bit weak randomness-leakage.

### B. PERFORMANCE

In this section, we give the performance analysis compared with CS-PKE in [8], NS-PKE in [25], DHL-PKE in [9], LWZ-PKE in [22], LWZ-PKE [22], KNP-PKE in [20], which is listed in Table 2 and Table 3.

The leakage rate is defined as the ratio of allowable leakage amount to the length of private-key or randomness, *i.e.*, $\rho = \frac{\ell}{|sk|}$. In our construction $\pi$, the leakage bound is

$$\ell = \log q - \omega(\log\lambda) \tag{18}$$

and the private key has 4 elements in $\mathbb{F}_q$, then the leakage rate of private-key is

$$\rho_\pi = \frac{\log q - \omega(\log\lambda)}{4\log q} = 1/4 - o(1) \tag{19}$$

**TABLE 2. Leakage-resilient security.**

| Scheme | Security | Key leakage | Randomness leakage | Allowable message length ($m$) | Security ass. |
|---|---|---|---|---|---|
| [8] | IND-CCA2 | no | no | $\log q$ | DDH |
| [25] | IND-lrCCA2 | yes | no | $\log q - \ell - \omega(\lambda)$ | DDH |
| [21] | IND-lrCCA1 | yes | no | $\log q - \ell - \omega(\lambda)$ | DDH |
| [20] | IND-lrCCA2 | yes | no | $\log q - \ell - \omega(\lambda)$ | DCR |
| [22] | IND-lrCCA2 | yes | no | $\log q$ | DDH |
| [9] | IND-lrCCA2 | yes | no | $\log q/3 - \ell - \omega(\lambda)$ | d-DLIN |
| our $\pi$ | IND-lrCCA2 | yes | no | $\log q$ | DDH |
| our $\widehat{\pi}^{\dagger}$ | IND-lrCCA2 | yes | yes | $\log q$ | DDH |

($i$) The extractor is a $(k, \epsilon)$-strong extractor, where$k$ is the entropy of private-key and $\epsilon$ is the statistical distance between the extractor output and a uniformly distributed randomness; $\ell$ : leakage bound; $|\mathbb{G}|$: length of an element of group $\mathbb{G}$.
($ii$) For all message length with $\log q$, the message space is in total $\mathbb{G}$, which is independent to the parameter of extractor. They also provide a larger message space than the schemes that depend on the output of extractors.
($iii$) IND-CCA2: indistinguishable against chosen-ciphertext attacks; IND-lrCCA1: indistinguishable against leakage-resilient chosen-ciphertext attacks such that the decryption query occurs before the challenge; IND-lrCCA2: indistinguishable against chosen-ciphertext attacks such that the decryption query occurs before and after the challenge; DDH: decisional Diffi-Hellman assumption; DCR: decisional composite residuosity assumption; k-LIN: decisional Linear assumption.

**TABLE 3. Performance of leakage resilience.**

($i$) $\lambda$: security parameter; $m$: bits of message; $t$: length of seed; $d$: parameter of d-DLIN; $q$: order of group $\mathbb{G}$, *i.e.*, $q = 2^{\lambda}$.
($ii$) In [25], there have two constructions: the first scheme achieves $25\%$ leakage rate with IND-lrCCA1 security level; the second scheme gains $16.7\%$ leakage rate with IND-lrCCA2 security level.
($iii$) $^{\dagger}$: scheme $\widehat{\pi}$ is secure resilient both key leakage and randomness leakage. We can use the universal hash that is same in $\pi$ to obtain $(\log q - \omega(\log \lambda))$-bit randomness leakage resilience and then achieve $1 - o(1)$-leakage rate of weak randomness leakage.

| Scheme | Leakage rate ($\rho$) | Leakage bound ($\ell$) | # of $sk$ | # of $ct$ |
|---|---|---|---|---|
| [8] | 0 | 0 | $4\|\mathbb{F}_q\|$ | $4\|\mathbb{G}\|$ |
| [25][1] | 25% | $\log q - m - \omega(\log \lambda)$ | $4\|\mathbb{F}_q\|$ | $3\|\mathbb{G}\| + t + m$ |
| [25][2] | 16.7% | $\log q - m - \omega(\log \lambda)$ | $6\|\mathbb{F}_q\|$ | $3\|\mathbb{G}\| + t + m$ |
| [21] | 25% | $\log q - m - \omega(\log \lambda)$ | $4\|\mathbb{F}_q\|$ | $3\|\mathbb{G}\| + t + m$ |
| [20] | 8.3% | $\log q - m - \omega(\log \lambda)$ | $4\|\mathbb{F}_q\|$ | $3\|\mathbb{G}\| + t + m$ |
| [22] | 16.7% | $\log q - \omega(\log \lambda)$ | $6\|\mathbb{F}_q\|$ | $4\|\mathbb{G}\| + \|\mathbb{F}_q\|$ |
| [9] | $1 - o(1)$ | $\|sk\| - d$ | $(m+2)(d+1)\|\mathbb{F}_q\|$ | $(d+m+1)\|\mathbb{G}\|$ |
| our $\pi$ | 25% | $\log q - \omega(\log \lambda)$ | $4\|\mathbb{F}_q\|$ | $4\|\mathbb{G}\| + \|\mathbb{F}_q\|$ |
| our $\widehat{\pi}^{\dagger}$ | 25% (key)<br>1-o(1) (rand) | $\log q - \omega(\log \lambda)$-key +<br>$\log q - \omega(\log \lambda)$-rand | $4\|\mathbb{F}_q\|$ | $4\|\mathbb{G}\| + \|\mathbb{F}_q\|$ |

The leakage rate of private-key is also $\rho_{\widehat{\pi},key} = 1/4 - o(1)$. The leakage rate of randomness is determined by the performance of extractor $\widehat{\mathsf{EXT}}$. If we use an average-case $(\log q - \ell_2, \epsilon_2)$-strong extractor, then the leakage rate of randomness is $\rho_{\widehat{\pi},rand} = \ell_2/\log q$. We can obtain $1 - o(1)$ randomness leakage rate by choosing a universal hash like in $\pi$ in Section IV.

CS-PKE is an efficient CCA2-secure encryption using the hash proof system, but it does not support the key leakage. Naor and Segev [25] improved the CS-PKE scheme to tolerate at most $1/4$ size of the key under IND-lrCCA1 and $1/6$ size of the key under IND-lrCCA2 be leaked respectively, however, the schemes are somewhat inefficient since the plaintext length $m$ is dependent to the leakage bound, *i.e.*, $\ell + m \leq \log q - \omega(\log \lambda)$. LWZ-PKE is an improved version of CS-PKE such that the plaintext space is independent of the amount of the leakage bound, in which the plaintext space is defined in group $\mathbb{G}$. Our $\pi$ and $\widehat{\pi}$ schemes have the same plaintext space with LWZ-PKE, which use a universal hashing technique as a strong randomness extractor, which hashes multiple group elements into a single element in $\mathbb{G}$. DHL-PKE [9] can achieve $1 - o(1)$ leakage rate, which provides a better possible leakage tolerance. However, the key/ciphertext sizes and computation efficiency are the worst than the other schemes and the security assumption is based on the k-DLIN in bilinear pairing. The performance comparison is listed in Table 3.

Regarding the key size, our schemes and KNP-PKE are more efficient than the other schemes, because they need only 4 elements so as to reduce about 1/3 bit-length of key than NS-PKE and LWZ-PKE. It is noticed that our $\widehat{\pi}$ is secure against chosen-ciphertext attacks under both key leakage and randomness leakage, simultaneously.

### C. DISCUSSION

Our scheme $\pi$ considers that the leakage occurs before the challenge ciphertext is generated, that is, the attacker could only ask for leakage on the private-key before it sees the challenge ciphertext. Otherwise, given the challenge ciphertext, the attacker can device a leakage function that encodes the ciphertext and then leaks to its exactly the one bit that we try to hide during the encryption procedure. To solve this prob-

lem, we can use the entropy-bounded leakage [15] to describe the key-leakage function $f$ and randomness-leakage function $g$, and leave the plaintext still has some min-entropy even the attacker obtains the leakage after seeing the challenge ciphertext.

In the entropy-bounded leakage, the attacker can learn a noisy version of all of the (bounded) memory, in which we restrict the amount of entropy-loss caused by the leakage rather than restricting its length. This is a more general scenario than the bounded leakage. The leakage is not of bounded length, but it is guaranteed that the private-key/randomness are still unpredictable given the leakage, which models a realistic leakage model in the applications. That is, the attacker can learn any random variable $\Omega$ (the leakage) for which the conditional information of private-key/randomness and $\Omega$ given the public-key is at most $\ell$. More concretely, the attacker can learn any random variable $\Omega$ for which $\tilde{\mathcal{H}}_\infty(sk|pk) - \tilde{\mathcal{H}}_\infty(sk|pk, \Omega) \leq \ell$. Otherwise, we consider that the private-key/randomness are fully revealed to the attacker.

## VII. CONCLUSION

In this work, we proposed two efficient public-key encryption schemes resilient to bounded key and/or randomness leakage, in which the private keys are shorter than several known constructions and the plaintext spaces are independent of neither the leakage bound nor the imposed (strong) extractor. Our first scheme was proven to be secure against chosen-ciphertext attacks resilient to key leakage, in which the private-key is involved in both ciphertext consistent verification and the randomness distillation. We employed a special kind of universal hash to play the 1-universal/2-universal extractors, and prove the security from hash proof system technique. We note that the proposed balance the computation cost, communication overhead and the security and it can be deployed in medical IoTs.

The second scheme was constructed using the first scheme as building blocks, and it achieves the security resilient to both key leakage and randomness leakage, simultaneously. Our schemes enjoy a higher relative leakage rate (*i.e.*, $25\% - o(1)$) and a larger plaintext space (*i.e.*, encode the message with $\log q$-bit in $\mathbb{G}$) in the presence of sensitive key/randomness leakage.

## REFERENCES

[1] J. Alwen, Y. Dodis, and D. Wichs, "Leakage-resilient public-key cryptography in the bounded-retrieval model," in *Proc. Crypto*, in Lecture Notes in Computer Science, vol. 5677, 2009, pp. 36–54.

[2] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, "Simultaneous hardcore bits and cryptography against memory attacks," in *Proc. TCC*, in Lecture Notes in Computer Science, vol. 5444, 2009, pp. 474–495.

[3] J. Baek, W. Susilo, J. K. Liu, and J. Zhou, "A new variant of the Cramer-Shoup KEM secure against chosen ciphertext attack," in *Applied Cryptography and Network Security—ACNS* (Lecture Notes in Computer Science), vol. 5336. Berlin, Germany: Springer Verlag, 2009, pp. 143–155.

[4] B. Barak *et al.*, "Leftover hash lemma, revisited," in *Proc. Crypto*, in Lecture Notes in Computer Science, vol. 6841, 2011, pp. 1–20.

[5] M. Bellare *et al.*, "Hedged public-key encryption: How to protect against bad randomness," in *Proc. Asiacrypt*, in Lecture Notes in Computer Science, vol. 5912, 2009, pp. 232–249.

[6] Z. Brakerski and S. Goldwasser, "Circular and leakage resilient public-key encryption under subgroup indistinguishability," in *Proc. Crypto*, in Lecture Notes in Computer Science, vol. 6223, 2010, pp. 1–20.

[7] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM J. Comput.*, vol. 17, no. 2, pp. 230–261, 1988.

[8] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Proc. Eurocrypt*, in Lecture Notes in Computer Science, vol. 2332, 2002, pp. 45–64.

[9] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs, "Efficient public-key cryptography in the presence of key leakage," in *Proc. Asiacrypt*, in Lecture Notes in Computer Science, Singapore, vol. 6477, Dec. 2010, pp. 613–631.

[10] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith, "Robust fuzzy extractors and authenticated key agreement from close secrets," *IEEE Internet Things J.*, vol. 58, no. 9, pp. 6207–6222, Sep. 2012.

[11] V. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–119, 2008.

[12] Y. Dodis and K. Pietrzak, "Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks," in *Proc. Crypto*, in Lecture Notes in Computer Science, vol. 6223, 2010, pp. 21–40.

[13] D. Dodis, K. Pietrzak, and D. Wichs, "Key derivation without entropy waste," in *Proc. Eurocrypt*, in Lecture Notes in Computer Science, vol. 8441, 2014, pp. 93–110.

[14] A. Faonio and D. Venturi, "Efficient public-key cryptography with bounded leakage and tamper resilience," in *Proc. Asiacrypt*, in Lecture Notes in Computer Science, vol. 10031, 2016, pp. 877–907.

[15] S. Halevi and H. Lin, "After-the-fact leakage in public-key encryption," in *Proc. TCC*, in Lecture Notes in Computer Science, vol. 6597, 2011, pp. 107–124.

[16] C. Hazay, A. López-Alt, H. Wee, and D. Wichs, "Leakage-resilient cryptography from minimal assumptions," *J. Cryptol.*, vol. 29, no. 3, pp. 514–551, 2015.

[17] J. Katz and V. Vaikuntanathan, "Smooth projective hashing and password-based authenticated key exchange from lattices," in *Proc. Asiacrypt*, in Lecture Notes in Computer Science, vol. 5912, 2009, pp. 636–652.

[18] E. Kiltz and K. Pietrzak, "Leakage resilient ElGamal encryption," in *Proc. Asaicrypt*, in Lecture Notes in Computer Science, vol. 6377, 2010, pp. 595–612.

[19] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *Proc. Crypto*, in Lecture Notes in Computer Science, vol. 6223, 2010, pp. 631–648.

[20] K. Kurosawa, R. Nojima, and L. T. Phong, "New leakage-resilient CCA-secure public key encryption," *J. Math. Cryptol.*, vol. 7, no. 4, pp. 297–312, 2013.

[21] S. Li, F. Zhang, Y. Sun, and L. Shen, "Efficient leakage-resilient public key encryption from DDH assumption," *Cluster Comput.*, vol. 16, no. 4, pp. 797–806, 2013.

[22] S. Liu, J. Weng, and Y. Zhao, "Efficient public key cryptosystem resilient to key leakage chosen ciphertext attacks," in *Proc. CT-RSA*, in Lecture Notes in Computer Science, vol. 7779, 2013, pp. 84–100.

[23] H. Namiki, K. Tanaka, and K. Yasunaga, "Randomness leakage in the KEM/DEM framework," in *Proc. ProvSec*, in Lecture Notes in Computer Science, vol. 6980, 2011, pp. 309–323.

[24] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proc. STOC*, 1990, pp. 427–437.

[25] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," in *Proc. Crypto*, in Lecture Notes in Computer Science, vol. 5677, 2009, pp. 18–35.

[26] M. H. Nguyen, K. Yasunaga, and K. Tanaka, "Leakage-resilient CCA2 public-key encryption from 4-wise independent hash functions," in *Proc. ATC*, Aug. 2011, pp. 14–17 .

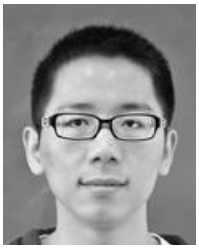
[27] J. B. Nielsen, D. Venturi, and A. Zottarel, "On the connection between leakage tolerance and adaptive security," in *Proc. PKC*, in Lecture Notes in Computer Science, vol. 7778, 2013, pp. 497–515.

[28] S. Park, K. Lee, and D. H. Lee, "New constructions of revocable identity-based encryption from multilinear maps," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1564–1577, Aug. 2015.

[29] B. Qin and S. Liu, "Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter," in *Proc. Asiacrypt*, 2013, pp. 381–400.

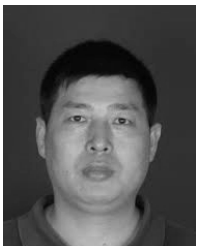
[30] B. Qin, S. Liu, and K. Chen, "Efficient chosen-ciphertext secure public-key encryption scheme with high leakage-resilience," *IET Inf. Secur.*, vol. 9, no. 1, pp. 32–42, 2015.
[31] Y. Yu, F. X. Standaert, O. Pereira, and M. Yung, "Practical leakage-resilient pseudorandom generators," in *Proc. ACM-CCS*, Chicago, IL, USA, Oct. 2010, pp. 141–151.
[32] M. Zhang, B. Yang, and T. Takagi, "Bounded leakage-resilient functional encryption with hidden vector predicate," *Comput. J.*, vol. 56, no. 4, pp. 464–477, 2013.
[33] M. Zhang, C. Wang, T. Takagi, and Y. Mu, "Functional encryption resilient to hard-to-invert leakage," *Comput. J.*, vol. 58, no. 4, pp. 735–749, 2015.
[34] 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.
[35] Z. Zhang, S. S. M. Chow, and Z. Cao, "Post-challenge leakage in public-key encryption," *Theor. Comput. Sci.*, vol. 572, pp. 25–49, 2015.
[36] Y. Zhou and B. Yang, "Continuous leakage-resilient public-key encryption scheme with CCA security," *Comput. J.*, vol. 60, no. 8, pp. 1161–1172, 2017.

**MINGWU ZHANG** was a Japan Society for the Promotion of Science Fellow 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 networks, secure computation, and privacy preservation.



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

**YONG DING** received the Ph.D. degree in cryptography from Xidian University, China. He is currently 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.



**CHUNMING TANG** is currently a Full Professor with the School of Mathematics and Information Science, Guangzhou University, China, where he is also the Vice Director of the Key Laboratory of Information Security and the Vice Dean of the School of Mathematics and Information Science. His current research interests include information security and the foundation of cryptography.