Public-Key Encryption In The Standard Model Against Strong Leakage Adversary

Public-key Encryption in the Standard Model against Strong Leakage Adversary

Janaka Alawatugoda

Department of Computer Engineering, Faculty of Engineering
University of Peradeniya, Peradeniya 20400, Sri Lanka

Institute for Integrated and Intelligent Systems
Griffith University, Qld 4111, Australia

Abstract

Over the years, security against adaptively chosen-ciphertext attacks (CCA2) is considered as the strongest security definition for public-key encryption schemes. With the up rise of side-channel attacks, new security definitions are proposed, addressing leakage of secret keys together with the standard CCA2 definition. Among the new security definitions, security against continuous and after-the-fact leakage-resilient CCA2 can be considered as the strongest security definition, which is called as security against (continuous) adaptively chosen-ciphertext leakage attacks (continuous CCLA2). In this paper, we present a construction of a public-key encryption scheme, namely LR-PKE, which satisfies the aforementioned security definition. The security of our public-key encryption scheme is proven in the standard model, under decision BDH assumption. Thus, we emphasize that our public-key encryption scheme LR-PKE is (continuous) CCLA2-secure in the standard model. For our construction of LR-PKE, we have used a strong one-time signature scheme and a leakage-resilient refreshing protocol as underlying building blocks. The leakage bound is $0.15n \log p - 1$ bits per leakage query, for a security parameter k and a statistical security parameter n, such that $\log p \geq k$ and n is a function of k. It is possible to see that LR-PKE is efficient enough to be used for real-world usage.

1 Introduction

Traditionally, security analysis of cryptographic primitives is in a black-box model, where the adversary is given access to the primitive through well-defined interfaces (oracles). However, this does not truly reflect the real-world scenario, as the adversary may obtain lot of information about the cryptographic primitive from its actual implementation. This information paths are often called as *side-channels*. The attacks which are originated with the aid of side-channels are often called as *side-channel attacks*. Few examples of information which may leak during executions of cryptographic systems, and so allow side-channel attacks, include timing information [1, 2, 3], electromagnetic radiation [4], and power consumption [5]. During the past couple of decades side-channel attacks have become a increasingly popular method of attacking cryptographic systems.

Leakage-resilient cryptography emerged as a theoretical foundation to address the issue of attacks which are originated with the aid of side-channels. Here, it is assumed that the adversary has access to side-channel information, which is modeled by allowing the adversary to specify arbitrary leakage functions and obtain leakage from the secret key, as dictated by these functions. Note that, some restrictions must be imposed on the class of allowable leakage functions, as otherwise an adversary can simply read off the entire secret key from memory. Depending upon these restrictions, many theoretical models of leakage have emerged in the recent literature [6, 7, 8, 9].

1.1 Bounded Leakage vs Continuous Leakage

In models assuming bounded leakage there is an upper limit on the amount of leakage information for the entire period of execution. The security guarantee only holds if the leakage amount is below the prescribed limit. Motivated by "cold boot" attacks, Akavia et al. [7] constructed a general framework to model memory-leakage attacks for public-key encryption schemes. With the knowledge of the public key, the adversary can choose an efficiently computable, arbitrary leakage function, f, and send it to the leakage oracle. The leakage oracle gives $f(sk)$ to the adversary where sk is the secret key. The only restriction here is that the sum of output length of all the leakage functions that an adversary can obtain is limited by some parameter λ, which is smaller than the size of sk.

On the other hand, in models allowing continuous leakage the adversary is allowed to obtain leakage over and over for a polynomial number of iterations during the period of execution. Naturally, there is a limit on the amount of leakage that the adversary can obtain in each single execution, but the total amount of leakage that the adversary can obtain for the entire period of execution is unlimited. In the work of Micali et al. [6], a general framework was introduced to model the leakage that occurs during computation with secret parameters. This framework relies on the assumption that *only computation leaks information* and that leakage only occurs from the secret memory portions which are actively involved in a computation. The adversary is allowed to obtain leakage from polynomial number of computations, with the restriction that leakage per occurrence is limited by some leakage parameter λ. The overall leakage amount is unlimited and in particular it can be larger than the size of the secret key. Brakerski et al. [8] proposed a leakage model, in which it is not assumed that the information is only leaked from the secret memory portions involved in computations. Instead, it is assumed that leakage happens from the entire secret memory, but the amount of leakage per occurrence is limited by a leakage parameter λ. As same as to the Micali et al. [6], polynomial number of leakage occurrences are allowed continuously, and the overall leakage amount is arbitrarily large.

After-the-fact Leakage.

The above leakage models generally address the leakage which happens *before* the *challenge* is given to the adversary. In security experiments for public-key cryptosystems, the challenge is to distinguish the real plaintext corresponding to a particular ciphertext from a random plaintext. Generally, leakage which happens after the challenge is given to the adversary is considered as after-the-fact leakage. In the following section, we discuss more details about the leakage security notions which address after-the-fact leakage.

1.2 Leakage Security Notions to Public-key Encryption Schemes

Earlier security notions for public-key encryption schemes, such as chosen-plaintext security (CPA), chosen-ciphertext security (CCA1), and adaptively chosen-ciphertext security (CCA2), did not address the attacks based on secret key leakage. This motivates the development of leakage security notions for public-key encryption schemes. Naor and Segev [10] defined several security notions; chosen-plaintext key-leakage attacks, a-priori chosen-ciphertext key-leakage and a-posteriori chosen-ciphertext attacks. In their security definitions, the adversary is not allowed to obtain leakage after the challenge ciphertext is given. Putting step forward, Halevi and Lin [11] defined a security notion called after-the-fact leakage-resilient chosen-plaintext security (CPLA2). In their security definition, the adversary is allowed to obtain leakage even after the challenge ciphertext is given. Giving further more power to the adversary, Dziembowski and Faust [12] defined a security notion called adaptively chosen-ciphertext leakage security (CCLA2), in which the adversary is allowed to access the decryption oracle adaptively and obtain leakage information even after the challenge ciphertext is given. As far as our knowledge the security notion defined by Dziembowski and Faust [12]. is the strongest security notion for public-key encryption schemes.

Public-key encryption scheme	Leakage model	Leakage constrain	Complexity assumption	Proof model
Dziembowski and Faust [12]	Continuous	After-the-fact	DDH	Random oracle model
Li et al. [15]	Bounded	Before-the-fact	q-TABDHE	Standard model
Zhou and Yang [16]	Continuous	Before-the-fact	DDH	Standard model
Qin and Liu [17]	Bounded	Before-the-fact	DDH	Standard model
Sun et al. [18]	Bounded	Before-the-fact	q-TABDHE	Standard model
Our construction LR-PKE	Continuous	After-the-fact	decision-BDH	Standard model

Table 1: Comparison of (selected) leakage-resilient CCA2-type public-key encryption schemes

1.3 Our Contribution

In this paper, we construct an efficient public-key encryption scheme, namely LR-PKE. More precisely, the LR-PKE scheme possesses CCA2-type security against after-the-fact leakage, hence it satisfies CCLA2 security. Moreover, it is secure against continuous leakage of the secret key and the security is proven in the standard model.

In order to obtain this result, first, we construct a public-key encryption scheme, namely PKE, which is a CCA2-secure public-key encryption scheme. This is obtained by Boneh and Boyen's selectively secure ID-based encryption scheme [13] using the conversion techniques by Canetti et al. [14]. Note that our scheme, PKE, does not use any NIZK proofs. Then, in order to construct LR-PKE, which is CCLA2-secure, we apply the leakage-resilient storage scheme and its refreshing protocol of Dziembowski and Faust [12] to PKE. Thus, we construct after-the-fact, leakage-resilient public-key encryption scheme LR-PKE, which is secure against continuous leakage of the secret key. Our public-key encryption scheme LR-PKE can tolerate a leakage amount as same as the underlying leakage-resilient refreshing protocol. Thus, the leakage bound is $0.15n \log p - 1$ bits per leakage query, for a security parameter k and a statistical security parameter n, such that $\log p \geq k$ and n is a function of k.

The Table 1 shows a comparison between several leakage-resilient CCA2-type public-key encryption schemes. It is clear to see that LR-PKE achieves CCA2-type security against continuous, after-the-fact leakage of the secret key, in the standard model. When compared with our non-leakage-resilient version, that is PKE, in order to achieve CCLA2, we have to perform $3n$ more exponentiation and two more pairing computations, at the decryption stage. Detailed analysis of computation cost is presented at the end of section 3.2. Dziembowski and Faust [12] mentioned about a construction of a public-key encryption scheme in the standard model, by using Groth-Sahai proof system [19]. However it gives rather inefficient scheme than our LR-PKE.

2 Preliminaries

We briefly recall the definitions of various cryptographic primitives and security definition which we use throughout this paper.

2.1 Leakage Model

Throughout the paper, we frequently borrow notions for our leakage model from Dziembowski and Faust [12].

2.1.1 Leakage Game.

We assume that secret information are stored into two parts of the memory, say L and $R \in \{0,1\}^s$, and an adversary $\mathcal{A}$ wants to learn some information from L and R. For a positive integer λ, we define λ-*leakage game* between an adaptive adversary $\mathcal{A}$, called λ-limited adversary, and a leakage oracle $\Lambda(L, R)$ as follows: the adversary queries 2 adaptively chosen leakage functions (f_1, f_2) to the oracle $\Lambda(L, R)$. Then the oracle replies with $f_1(L)$ and $f_2(R)$ with restriction that the adversary cannot learn

more than λ-bits from each L and R at the end of the game. We denote $Out(\mathcal{A}, \Lambda(L, R))$ by the output of this game.

2.1.2 Leakage from Computations.

In practice, we assume that only computations involving L or R leak their information to the adversary and information from L and R are leaked independently. This can be described in the following setting.

Consider a two-party protocol Π between the parties P_L and P_R. Initially, the party P_L (respectively, P_R) is given the input L (resp. R) and at the end of the protocol each party have the results L' and R', respectively. The execution of the protocol proceeds in rounds. In each round, one party (owner) computes a message and send it to the other. The message may be computed from owner's input (eg. L or R), randomness chosen by the owner, and the received message from the earlier rounds. In the setting that only computation leaks information, the computations carried out by P_L (resp. P_R) with the initial state L (resp. R) in each round may leak information on L (resp. R) independently from R (resp. L). Precisely, the adversary $\mathcal{A}$ adaptively chooses two leakage functions (f_1^i, f_2^i) and obtains $f_1^i(L)$ and $f_2^i(R)$ from each i-th round in the execution of the protocol (even after the challenge ciphertext is issued). At this time, the adversary is allowed to play the λ-leakage game with access to the leakage oracle $\Lambda((L, \rho_L, M_L); (R, \rho_R, M_R))$, where ρ_x is the randomness used by $x \in \{L, R\}$ and M_x is the previously received message by user $x \in \{L, R\}$ in the execution of the protocol. We sometimes omit ρ_x or M_x when they are obviously null from the context. We denote the output of the adversary $\mathcal{A}$ after running such execution of the protocol Π by $\mathcal{A} \leftrightarrows (\Pi(L, R) \to (L', R'))$.

2.2 Leakage-resilient Storage and Refreshing Protocol

In Dziembowski and Faust [12], they used a leakage-resilient storage to store the secret key into two split parts and introduced its refreshing protocol to construct public-key cryptosystems secure against continuous leakage attacks.

2.2.1 Leakage-resilient Storage.

A leakage-resilient storage (LRS) $\Phi = (\mathrm{Encode}, \mathrm{Decode})$ is a scheme to encode a message in $\mathcal{M}$, where $\mathrm{Encode} : \mathcal{M} \to \mathcal{L} \times \mathcal{R}$ is a probabilistic, efficiently computable function and $\mathrm{Decode} : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$ is a deterministic, efficiently computable function satisfying $\mathrm{Decode}(\mathrm{Encode}(S)) = S$ for any $S \in \mathcal{M}$.

An LRS Φ is called (λ, ϵ)-secure if for any $S, S' \in \mathcal{M}$ and any λ-limited adversary $\mathcal{A}$, we have

$$\Delta(Out(\mathcal{A}, \Lambda(L, R)), Out(\mathcal{A}, \Lambda(L', R'))) \leq \epsilon,$$

where $(L, R) := \mathrm{Encode}(S)$ and $(L', R') := \mathrm{Encode}(S')$. Here, Δ denotes the statistical distance.

In Dziembowski and Faust [12], they use the fact that the LRS from inner product is (λ, ϵ)-secure for some λ and $\epsilon > 0$. Precisely, for a field $\mathbb{F}$, an LRS $\Phi_{\mathbb{F}}^{n,m} = (\mathrm{Encode}_{\mathbb{F}}^{n,m}, \mathrm{Decode}_{\mathbb{F}}^{n,m})$ is defined as follows:

- $\mathrm{Encode}_{\mathbb{F}}^{n,m}(S)$ chooses $L \leftarrow \mathbb{F}^n \backslash \{0^n\}$ and samples $R \leftarrow \mathbb{F}^{n \times m}$ such that $L \cdot R = S$.
- $\mathrm{Decode}_{\mathbb{F}}^{n,m}(L, R)$ computes $L \cdot R = S$.

Corollary 1 ([12]). *Suppose $|\mathbb{F}| = \Omega(n)$ and $m < n/20$, then the LRS $\Phi_{\mathbb{F}}^{n,m}$ is a $(0.3|\mathbb{F}^n|, \epsilon)$-secure LRS, where ϵ is negligible in the statistical security parameter $n \in \mathbb{N}$.*

2.2.2 Refreshing Protocol.

It is obvious that the above LRS scheme is not secure, if we allow an adversary to obtain leakage information continuously from L and R. To prevent this obstacle, Dziembowski and Faust [12] introduced a refreshing protocol Refresh of an LRS Φ. The main idea is to refresh the encoded secret

(L, R) securely to a newly encoded value (L', R') so that it does not reveal enough information to recover the secret key even when an λ-limited adversary can observe the leakage continuously from the refreshing protocol.

Precisely, a refreshing protocol $(L', R') \leftarrow \mathrm{Refresh}(L, R)$ is a two-party protocol between P_L and P_R holding L and R respectively. At the end of the protocol, each party outputs L' and R' respectively satisfying $\mathrm{Decode}(L, R) = \mathrm{Decode}(L', R')$.

Let Refresh a refreshing protocol, $\Phi = (\mathrm{Encode}, \mathrm{Decode})$ an LRS, $\mathcal{A}$ an λ-limited adversary, $\ell \in \mathbb{N}$ and $S \in \mathcal{M}$. We consider the following experiments $\mathrm{Exp}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S, \ell)$:

- For a secret S, encode it by $(L^0, R^0) \leftarrow \mathrm{Encode}(S)$.
- For $i = 1$ to ℓ, run $\mathcal{A}$ against the refreshing protocol: $\mathcal{A} \leftrightarrows (\mathrm{Refresh}(L_{i-1}, R_{i-1}) \rightarrow (L_i, R_i))$.
- Return $b \in \{0, 1\}$ output by $\mathcal{A}$.

As described in the previous paragraph, the leakage from each execution of the protocol is obtained via the leakage oracle $\Lambda((L_i, \rho_{L_i}, M_{L_i}); (R_i, \rho_{R_i}, M_{R_i}))$. Finally, the security of the refreshing protocol is defined by indistinguishability notion.

Definition 1 (A $(\ell, \lambda, \epsilon_1)$-secure refreshing protocol [12])**.** *For a LRS Φ with message space $\mathcal{M}$ and $\ell \in \mathbb{N}$ number of leakage rounds, a refreshing protocol* Refresh *is $(\ell, \lambda, \epsilon_1)$-secure, if for any λ-limited adversary $\mathcal{A}$ and any two secrets $S, S' \in \mathcal{M}$, we have*

$$\Delta(\mathrm{Exp}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S, \ell), \mathrm{Exp}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S', \ell)) \leq \epsilon_1.$$

Theorem 1 ([12])**.** *Let $m/3 \leq n$, $n \geq 16$ and $\ell \in \mathbb{N}$. Let n, m and $\mathbb{F}$ be such that $\Phi_{\mathbb{F}}^{n,m}$ is (λ, ϵ)-secure for some $\lambda, \epsilon > 0$. Then the protocol* $\mathrm{Refresh}_{\mathbb{F}}^{n,m}$ *is a $(\ell, \lambda/2 - 1, \epsilon')$-secure refreshing protocol for $\Phi_{\mathbb{F}}^{n,m}$ with $\epsilon' = 2\ell|\mathbb{F}|^m(3|\mathbb{F}|^m\epsilon + m|\mathbb{F}|^{-n-1})$.*

Definition 2 (A $(\ell, \lambda, \epsilon_1)$-secure refreshing protocol with auxiliary information [12])**.** *Let $Z \subset \mathcal{M}^m$ be an $m' < m$ dimensional affine subspace defined by W and Q. For a LRS Φ with message space $\mathcal{M}$ and $\ell \in \mathbb{N}$, a refreshing protocol* Refresh *is $(\ell, \lambda, \epsilon_1)$-secure with auxiliary information (W, Q), if for any λ-limited adversary $\mathcal{A}$ and any two secrets $S, S' \in \mathcal{M}$, we have,*

$$\Delta(\mathrm{Exp}^{\mathrm{aux}}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S, \ell, W, Q),$$
$$\mathrm{Exp}^{\mathrm{aux}}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S', \ell, W, Q)) \leq \epsilon_1.$$

Theorem 2 (Generalization of Theorem 1 [12])**.** *Let $m/4 \leq n$, $n \geq 16$ and $\ell \in \mathbb{N}$. Let (W, Q) be as above defining an $m' < m$ dimensional affine subspace Z. Let n, m' and $\mathbb{F}$ be such that $\Phi_{\mathbb{F}}^{n,m'}$ is (λ, ϵ)-secure for some $\lambda, \epsilon > 0$. Then the protocol* $\mathrm{Refresh}_{\mathbb{F}}^{n,m}$ *is a $(\ell, \lambda/2 - 1, \epsilon')$-secure refreshing protocol with auxiliary information defined by (W, Q) for $\Phi_{\mathbb{F}}^{n,m}$ with $\epsilon' = 2\ell|\mathbb{F}|^m(3|\mathbb{F}|^{2m}\epsilon + m|\mathbb{F}|^{-n-1})$.*

Corollary 2 ([12])**.** *Suppose $|\mathbb{F}| = \Omega(n)$ and $m = o(n)$, then the* $\mathrm{Refresh}_{\mathbb{F}}^{n,m}$ *is a $(\ell, 0.15n \log |\mathbb{F}|) - 1, \epsilon_1)$-secure refreshing protocol for $\Phi_{\mathbb{F}}^{n,m}$, where ℓ is a polynomial and ϵ_1 is negligible in the statistical security parameter $n \in \mathbb{N}$.*

2.3 Strong One-time Signature Scheme

Simply, the adversary should be unable to forge a valid message/signature pair, after receiving a signature on any single message m of the adversarys choice. Note that we require so-called strong security in this case, so that it should be infeasible for the adversary to generate even a different signature on the same message m.

Definition 3 (Strong one-time signature scheme [20])**.** *A signature scheme* $\mathrm{Sig} = (\mathrm{SigGen}, \mathrm{Sign}, \mathrm{Vrfy})$ *is a strong one-time* (SUF-OT) *signature scheme, if the success probability of any probabilistic polynomial time adversary $\mathcal{C}$ in the following game,* $\mathrm{Adv}_{\mathrm{Sig}}^{\text{SUF-OT}}(\mathcal{C})$*, is negligible in the security parameter k:*

- *The* (SUF-OT) *challenger runs* $(ssk, svk) \leftarrow \mathrm{SigGen}(1^k)$ *and* $\mathcal{C}$ *is given* 1^k *and* svk.
- $\mathcal{C}(1^k, svk)$ *may do one of the following:*
  - $\mathcal{C}$ *may output a pair* (m^*, σ^*) *and halt. In this case* (m, σ) *e undefined.*
  - $\mathcal{C}$ *may output a message* m*, and is then given in return* $\sigma \leftarrow \mathrm{Sign}(m)$*. Following this,* $\mathcal{C}$ *outputs* (m^*, σ^*).

We say the adversary succeeds if $\mathrm{Vrfy}(m^*, \sigma^*) = 1$ *but* $(m^*, \sigma^*) \neq (m, \sigma)$ *(assuming* (m, σ) *are defined). We stress that the adversary may succeed even if* $m^* = m$.

2.4 The Decision Bilinear Diffie-Hellman Assumption (decision BDH)

The decision bilinear Diffie-Hellman assumption (decision BDH) assumption is a variant of decision Diffie-Hellman (DDH) assumption.

Definition 4 (Decision bilinear Diffie-Hellman assumption (decision BDH) [21])**.** *Let* k *be the security parameter and* Gen *be a group generation algorithm. Let* $(\mathbb{G}, \mathbb{G}_T, p, e) \leftarrow \mathrm{Gen}(1^k)$*, where* p *is a prime number, the description of two groups* $\mathbb{G}, \mathbb{G}_T$ *of order* p*, and the description of an admissible bilinear map* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$*. Let* g *be a generator of* $\mathbb{G}$.

The decision bilinear Diffie-Hellman (decision BDH) problem in $(\mathbb{G}, \mathbb{G}_T, p, e)$ *is as follows: Consider two distributions* $(g, g^a, g^b, g^c, e(g,g)^{abc})$ *and* (g, g^a, g^b, g^c, T) *for some* $a, b, c \in \mathbb{Z}_p$*, and random* $T \in \mathbb{G}_T$*. It is said that decision BDH assumption holds in* $(\mathbb{G}, \mathbb{G}_T, p, e)$*, if for all probabilistic polynomial time algorithms* $\mathcal{B}$*, the advantage in distinguishing the two distributions is given as,*

$$\mathrm{Adv}^{\mathrm{decisionBDH}}_{\mathbb{G},\mathbb{G}_T,p,e}(\mathcal{B}) = \Big|\Pr\big[\mathcal{B}\big(g, g^a, g^b, g^c, e(g,g)^{abc}\big) = 1\big] - \Pr\big[\mathcal{B}\big(g, g^a, g^b, g^c, T\big) = 1\big]\Big|$$

is negligible for a given security parameter k.

3 Leakage-resilient CCA2-secure Public-key Encryption Scheme

We begin with several basic definitions for a public-key encryption scheme.

3.1 Definitions

For security parameter k, a public-key encryption scheme $\mathrm{PKE} = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ consists of the following algorithms.

- $(pk, sk) \leftarrow \mathrm{KeyGen}(1^k)$: It outputs a public/secret key pair.
- $c \leftarrow \mathrm{Enc}(pk, m)$: A probabilistic algorithm that outputs $c = \mathrm{Enc}(pk, m)$ with inputs a message m and the public key pk.
- $m = \mathrm{Dec}(sk, c)$: A deterministic algorithm that decrypts the ciphertext c together with the secret key sk. We always have $m = \mathrm{Dec}(sk, \mathrm{Enc}(pk, m))$.

The strongest security notion for public-key encryption scheme is security against *adaptively chosen-ciphertext attacks* (CCA2 secure). In CCA2 attack against a public-key encryption scheme, the adversary who is given pk picks two messages m_0, m_1 and guesses $b \in \{0, 1\}$ on input $c^* = \mathrm{Enc}(pk, m_b)$ while the adversary is allowed to ask for the decryption of chosen-ciphertexts prior and after to seeing c^*.

In the setting that computation leaks information, it is natural to consider leakage information from the queries to the decryption oracle. This yields the following extension of CCA2-security against leakage attacks.

Definition 5 ([12], Security against Chosen Ciphertext Leakage Attacks (CCLA2)). *Let k be the security parameter. A public-key encryption scheme* $\mathrm{PKE} = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ *is CCLA2 secure if for any λ-limited adversary $\mathcal{A}$ the probability that the experiment below outputs 1 is at most $1/2 + negl(k)$.*

1. *The challenger runs $(pk, sk) \leftarrow \mathrm{KeyGen}(1^k)$ and sends pk to $\mathcal{A}$.*
2. *The adversary $\mathcal{A}$ asks for the decryption of a ciphertext c learning at most λ-bits of the current secret sk for each query: $\mathcal{A} \leftrightarrows (\mathrm{Dec}(sk, c) \to sk')$. Set $sk := sk'$ for the next round.*
3. *$\mathcal{A}$ outputs two messages, m_0, m_1, and sends them to the challenger.*
4. *The challenger computes $c^* \leftarrow \mathrm{Enc}(pk, m_b)$ for $b \in \{0, 1\}$ and gives it to $\mathcal{A}$.*
5. *Repeat $\mathcal{A} \leftrightarrows (\mathrm{Dec}(sk, c) \to sk')$ for $c \neq c^*$ learning at most λ-bits of the current secret sk. Set $sk := sk'$ for the next round.*
6. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$. If $b = b'$, then output 1, otherwise output 0.*

3.2 An Efficient CCLA2-secure Public-key Encryption Scheme in the Standard Model

In this section, we propose an efficient CCLA2-secure public-key encryption scheme in the standard model.

3.2.1 Description of this Section.

As a building block of our construction, we require a public-key encryption scheme, namely PKE, that is CCA2-secure in the standard model. For this, we use selective-ID secure identity based encryption scheme in the standard model proposed by Boneh and Boyen [13]. This ID-based encryption scheme can be used to construct public-key encryption scheme PKE, together with techniques by Canetti et al. [14].

However, to do this, the secret key should be information theoretically hidden even the corresponding public key is given. Originally, the public-key encryption scheme obtained from the above technique does not satisfy this condition. In other words, the secret key in this case is simply $x \in \mathbb{F}_p$ and corresponding public key is of form g^x for some group element g of order p, therefore it is not information theoretically hidden at all. To fix this, we propose a slightly modified version of Boneh and Boyen's ID-based encryption scheme and prove that it is still selective-ID secure without random oracles under the BDH assumption. Then, we apply techniques by Dziembowski and Faust [12], to make this public-key encryption scheme secure against continuous leakage attacks.

Once we have such an ID-based encryption scheme, then we convert it to the public-key encryption scheme as before. The secret key of this public-key encryption scheme is of form (x_1, x_2) and it is information theoretically hidden even given the corresponding public key. Similarly as in Dziembowski and Faust [12], we encode (x_1, x_2) as $(L, R) \leftarrow \mathrm{Encode}^{n,2}_{\mathbb{Z}_p}(x_1, x_2)$ using an LRS scheme $\Phi^{n,2}_{\mathbb{Z}_p}$ and implement the decryption process as a two-party protocol between P_L and P_R. The decryption process is combined with the refreshing protocol $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}$ so that it outputs refreshed secret key (L', R') after each execution of the decryption. This makes the proposed scheme CCLA2-secure.

3.2.2 Modified Boneh and Boyen's ID-based Encryption Scheme.

We consider the following modified version of Boneh and Boyen's ID-based encryption scheme. Let IBE be the modified ID-based encryption scheme and $\mathbb{G}$ be a bilinear group of prime order p. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be the bilinear map. Pick random elements $g_1, g_2, g_3, \widetilde{h} \in \mathbb{G}$ and random elements $a_1, b_1 \in \mathbb{Z}_p$, let $mpk = (g_1, g_2, h := g_1^{a_1} g_2^{b_1}, g_3, \widetilde{h})$ be the public parameters and let $msk = (a_1, b_1)$ be the master secret key. For $ID := I \in \mathbb{Z}_p$, the secret key d_I is computed by

$$d_I = (g_3^{a_1}(h^I \widetilde{h})^{r_1}, g_3^{b_1}(h^I \widetilde{h})^{r_2}, g_1^{r_1} g_2^{r_2}),$$

where r_1, r_2 are random elements from $\mathbb{Z}_p$.

To encrypt a message $m \in \mathbb{G}_T$ with an identity $I \in \mathbb{Z}_p$, one picks a random element $s \in \mathbb{Z}_p$ and computes

$$ct = \big(e(h, g_3)^s \cdot m, {g_1}^s, {g_2}^s, (h^I\widetilde{h})^s\big).$$

To decrypt a ciphertext $ct = (u, v_1, v_2, w)$, one computes

$$m = u \cdot \frac{e(w, {g_1}^{r_1}{g_2}^{r_2})}{e\big(v_1, {g_3}^{a_1}\big(h^I \cdot \widetilde{h}\big)^{r_1}\big) \cdot e\big(v_2, {g_3}^{b_1}\big(h^I \cdot \widetilde{h}\big)^{r_2}\big)}.$$

Theorem 3. *The ID-based encryption scheme IBE is selective-ID-CPA (chosen plaintext attacks) secure without random oracles if the decision* BDH *assumption holds in* $\mathbb{G}$.

Proof. (sketch) For a given ID-based encryption attacker $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that solves the decision BDH problem. Algorithm $\mathcal{B}$ is given as input a tuple $(g_1, g_1^{a_1}, g_1^{a_2}, g_1^{a_3}, T)$ where T is either $e(g_1, g_1)^{a_1 a_2 a_3}$ or an uniform random element in $\mathbb{G}_T$. Set $A_1 = g_1^{a_1}, A_2 = g_1^{a_2}, A_3 = g_1^{a_3}$.

In the beginning of the game, $\mathcal{A}$ selects an $I^* \in \mathbb{Z}_p$ as a target identity. Then $\mathcal{B}$ picks random elements $\beta, b_1 \in \mathbb{Z}_p$ and sets $g_2 := g_1^{\beta}$ and $h := A_1 {g_2}^{b_1}$. He picks again a random element $\alpha_1 \in \mathbb{Z}_p$ and sets $\widetilde{h} = h^{-I^*} {g_1}^{\alpha_1} {g_2}^{b_1 I^*}$. Then he gives public parameters $mpk = (g_1, g_2, h, g_3 = A_2, \widetilde{h})$ to $\mathcal{A}$. Let $b_1 I^*$ be α_2.

Whenever $\mathcal{A}$ asks for the private key corresponding to the identity $I \neq I^*$, $\mathcal{B}$ (who does not know the master secret key) picks random elements $r_1, r_2 \in \mathbb{Z}_p$ and sets

$$d_0 = {g_3}^{\frac{-\alpha_1}{I - I^*}} \big(h^{I - I^*} {g_1}^{\alpha_1} {g_2}^{\alpha_2}\big)^{r_1},$$
$$d_1 = {g_3}^{b_1} \cdot A_1 \cdot {g_1}^{\frac{\alpha_1}{I - I^*}} \big(h^{I - I^*} {g_1}^{\alpha_1} {g_2}^{\alpha_2}\big)^{r_2},$$

and

$$d_2 = {g_1}^{r_1} {g_2}^{\frac{-1}{I - I^*}} {g_2}^{r_2 + \frac{1}{I - I^*}} \quad .$$

Then $\mathcal{B}$ returns (d_0, d_1, d_2) as the private key for the identity I. It is easy to verify the correctness of the private key.

When $\mathcal{A}$ sends $m_0, m_1 \in \mathbb{G}_T$ as a challenge query, then $\mathcal{B}$ responds with the ciphertext $ct = \big(m_b \cdot T \cdot e(A_2, A_3)^{\beta b_1}, A_3, {A_3}^{\beta}, {A_3}^{\alpha_1 + \beta\alpha_2}\big)$. If $T = e(g_1, g_1)^{a_1 a_2 a_3}$, then ct is a valid ciphertext, otherwise it is independent from the adversary's view. It completes the proof.

□

3.2.3 CCA2-secure Public-key Encryption Scheme from IBE.

We construct a new public-key encryption scheme based on the pre-described ID-based encryption scheme IBE. By generic conversion of Canetti et al. [14], one can construct a CCA2-secure public-key encryption scheme from any selective-ID-secure ID-based encryption scheme with strong one-time signature scheme. Let this public-key encryption scheme is PKE = (KeyGen, Enc, Dec). Let the notations be as same as the previous section. Let Sig = (SigGen, Sign, Vrfy) be a strong one-time signature scheme.

- $(pk, sk) \leftarrow \text{KeyGen}(1^k)$: Pick random elements $g_1, g_2, g_3, \widetilde{h} \in \mathbb{G}$ and $a_1, b_1 \in \mathbb{Z}_p$. Set $pk = (g_1, g_2, h = g_1^{a_1} g_2^{b_1}, g_3, \widetilde{h})$ and $sk = (a_1, b_1)$.
- $c \leftarrow \text{Enc}(pk, m)$: Run $\text{SigGen}(1^k)$ to obtain a key pair (svk, ssk), where ssk is a signing key and $svk \in \mathbb{Z}_p$ is a corresponding verification key. Encrypt a message $m \in \mathbb{G}_T$ using the above ID-based encryption scheme with the identity $I = svk$: pick a random $s \in \mathbb{Z}_p$ and compute

  $$ct = (u, v_1, v_2, w) = \big(e(h, g_3)^s \cdot m, {g_1}^s, {g_2}^s, H(svk)^s\big),$$

  where $H(svk) = h^{svk}\widetilde{h}$. Sign ct using $\sigma \leftarrow \text{Sign}(ssk, ct)$. Return $c = (svk, ct, \sigma)$.

- $m \leftarrow \mathrm{Dec}(sk, m)$: Verify whether $\mathrm{Vrfy}(svk, ct, \sigma) = 1$ or not. If failed, then output $\perp$. Otherwise, parse ct as (u, v_1, v_2, w), pick random elements $r_1, r_2 \in \mathbb{Z}_p$ and compute

$$m = u \cdot \frac{e(w, {g_1}^{r_1} {g_2}^{r_2})}{e\big(v_1, {g_3}^{a_1} H(svk)^{r_1}\big) \cdot e\big(v_2, {g_3}^{b_1} H(svk)^{r_2}\big)} \ .$$

As the underlying ID-based encryption scheme IBE is selective-ID-secure in the standard model based on decision BDH assumption, the proposed public-key encryption scheme PKE is CCA2-secure based on the same assumption without random oracles.

3.2.4 Leakage-resilient Public-key Encryption Scheme.

We construct a public-key encryption scheme, namely LR-PKE $= (\mathrm{KeyGen}', \mathrm{Enc}', \mathrm{Dec}', \mathrm{Refresh}^{n,2}_{\mathbb{Z}_p})$, that is secure against continuous leakage attacks. As in Dziembowski and Faust [12], we encode the secret key (a_1, b_1) as $L \in \mathbb{Z}_p^n$ and $R := (R_1, R_2) \in \mathbb{Z}_p^{n \times 2}$ using an LRS scheme and store them into two split parts. To prevent from continuous leakage attacks, the secret key (L, R) is refreshed using the refreshing protocol Refresh, whenever the decryption process is carried out. As described earlier, the decryption is processed by a two-party protocol between P_L and P_R where P_x is given $x \in \{L, R\}$. In the following, we use the LRS scheme $\Phi^{n,2}_{\mathbb{Z}_p} = (\mathrm{Encode}^{n,2}_{\mathbb{Z}_p}, \mathrm{Decode}^{n,2}_{\mathbb{Z}_p})$ described in Section 2.2.

Exponentiation of a group element $g \in \mathbb{G}$ (or, in $\mathbb{G}_T$) by a vector $A = (A_1, \ldots, A_n) \in \mathbb{Z}_p^n$ is defined naturally by $g^A := (g^{A_1}, \cdots, g^{A_n})$. Also, the exponentiation of a vector $V = (V_1, \ldots, V_n) \in \mathbb{G}^n$ (or, in $\mathbb{G}_T^n$) by a vector $B = (B_1, \ldots, B_n)$ is defined by pointwise exponentiations, $V^B := (V_1^{B_1}, \ldots, V_n^{B_n})$.

Key Generation $\mathrm{KeyGen}'(1^k)$: Let $g_1, g_2, g_3, \widetilde{h} \xleftarrow{\$} \mathbb{G}$, $a_1, b_1 \xleftarrow{\$} \mathbb{Z}_p$ and $(L, R) \leftarrow \mathrm{Encode}^{n,2}_{\mathbb{Z}_p}(a_1, b_1)$. Write $R = (R_1, R_2)$, where $R_i \in \mathbb{Z}_p^n$ is the i-th column of R so that $L \cdot R_1 = a_1$ and $L \cdot R_2 = b_1$. Let $sk = (L, R)$ and $pk = (p, g_1, g_2, h := g_1^{a_1} {g_2}^{b_1}, g_3, \widetilde{h})$.

Encryption $\mathrm{Enc}'(pk, m)$: It proceeds as same as the Enc in our PKE from IBE.

Decryption $\mathrm{Dec}'(sk, c)$: For $sk = (L, R)$, L is given to P_L and R is given to P_R. Both parties receive c and parse it as (svk, ct, σ). One of the parties verifies the signature by checking $\mathrm{Vrfy}(svk, ct, \sigma) = 1$. If it fails, then output $\perp$, otherwise parse ct as (u, v_1, v_2, w) and proceed as follows:

1. P_R computes $V := e(v_1, g_3)^{R_1} \circ e(v_2, g_3)^{R_2}$, where $\circ$ denotes pointwise multiplication.
2. P_L computes $U = (U_1, \ldots, U_n) := V^{-L}$, pick random elements $r_1, r_2 \in \mathbb{Z}_p$, and outputs

$$u \cdot \frac{e(w, {g_1}^{r_1} {g_2}^{r_2})}{e(v_1, H(svk)^{r_1}) \cdot e(v_2, H(svk)^{r_2})} \cdot \prod_i U_i.$$

Refreshing $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}(L, R)$: After the decryption operation, the refreshing protocol $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}$ is executed on L and R and outputs L' and R' respectively, i.e. $(L', R') \leftarrow \mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}(L, R)$. Then, (L', R') is set as the secret key for the next round.

3.2.5 Security Analysis of LR-PKE.

Security analysis of this scheme is similar to that in Dziembowski and Faust [12]. Differently, in our case, we replace the NIZK proof with one-time signatures to check the validity of the ciphertexts. This makes our construction is efficient as well as secure in the standard model. First, we show that the leakage from a single decryption operation can be simulated using a simulator $\mathcal{S}$, which has access to a leakage oracle $\Lambda(L^*, R^*)$, where $(L^*, R^*) \leftarrow \mathrm{Encode}^{n,1}_{\mathbb{Z}_p}(a_1)$, and an auxiliary information $\mathrm{aux} = R_1 + \alpha R_2$ with $\alpha = \log_{g_1}(g_2)$. The following lemma is the analogous version of Lemma 7 in Dziembowski and Faust [12].

Lemma 1. *Let* $(pk, sk = (L, R)) \leftarrow \mathrm{KeyGen}(1^k)$ *with* $(a_1, b_1) = \mathrm{Decode}^{n,2}_{\mathbb{Z}_p}(L, R)$. *Then for any (unbounded)* λ*-limited adversary* $\mathcal{A}$*, there exists a (unbounded)* λ*-limited simulator* $\mathcal{S}$ *given access to the leakage oracle* $\Lambda(L^*, R^*)$ *such that for any* $b \in \{0, 1\}$

$$\Pr[Out(\mathcal{S}(pk, \mathrm{aux}), \Lambda(L^*, R^*)) = b] = \Pr[(\mathcal{A}(pk) \leftrightarrows (\mathrm{Dec}((L, R), c) \to (L', R'))) = b],$$

where $L^* \cdot R^* = a_1$ and $\mathrm{aux} = R_1 + \alpha R_2$ with $\alpha = \log_{g_1}(g_2)$.

Proof. (sketch) The proof is similar to that in Dziembowski and Faust [12]. For the simulation to work, $\mathcal{A}$ can only ask for valid ciphertexts, i.e. $\log_{g_1}(v_1) = \log_{g_2}(v_2) = \log_{H(svk)}(w) = s$.

The simulation game is described as follows:

1. The unbounded simulator $\mathcal{S}$ is given the public key pk and auxiliary information that $R_1 + \alpha R_2$ for $\alpha = \log_{g_1}(g_2)$. He sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ issues a query $c = (svk, ct, \sigma)$ for the decryption. $\mathcal{S}$ verifies σ with respect to svk. If it fails, output $\perp$, otherwise, parse ct as (u, v_1, v_2, w).
3. $\mathcal{S}$ checks the validity of the ciphertext: pick random elements $r_1, r_2 \in \mathbb{Z}_p$ and test the equality

$$\frac{e(w, g_1^{r_1} g_2^{r_2})}{e(v_1, H(svk)^{r_1}) \cdot e(v_2, H(svk)^{r_2})} \stackrel{?}{=} 1,$$

where $H(svk) = h^{svk}\widetilde{h}$. If it fails, output $\perp$, otherwise compute $\log_{g_1}(v_1) = s$ and return $u/e(h, g_3)^s$ to $\mathcal{A}$ with simulations of the leakage from the decryption.

Note that the validity check of the ciphertext only involves with the publicly known values. The probability for an invalid ciphertext, i.e. $s_1 := \log_{g_1}(v_1)$, $s_2 := \log_{g_2}(v_2)$, and $s_3 := \log_{H(svk)}(w)$ are mutually distinct, to pass the check is as same as the probability that,

$$e(g_1, H(svk))^{s_3 - s_1} \cdot e(g_2, H(svk))^{s_3 - s_2} = e(g_1, H(svk))^{(s_3 - s_1) + \alpha(s_3 - s_2)} = 1,$$

which equals to a negligible value $1/p$.

Now we need to simulate the leakage from the decryption process. The message sent from P_R equals to $e(v_1, g_3)^{R_1} \circ e(v_2, g_3)^{R_2} = e(g_1, g_3)^{s(R_1 + \alpha R_2)}$ and P_L uses randomness r_1 and r_2. So the leakage information learnt by the unbounded adversary $\mathcal{A}$ can be described as playing the game with the leakage oracle

$$\Lambda((L, r_1, r_2, s(R_1 + \alpha R_2); R).$$

The secret key is also unknown to the simulator $\mathcal{S}$ who is required to simulate the leakage above. The unbounded simulator $\mathcal{S}$ given access to the leakage oracle $\Lambda(L^*, R^*)$ with $L^* \cdot R^* = a_1$, he sets $L = L^*$ and $R_1 = R^*$ and computes $\alpha = \log_{g_1}(g_2)$. Then the leakage of R_2 can be described from the relation $R_2 = (\mathrm{aux} - R^*)/\alpha$ and the leakage of $s(R_1 + \alpha R_2)$ is from the relations $s \cdot \mathrm{aux}$. The leakage from r_1 and r_2 is clear. This gives a perfect simulation when the ciphertext is valid.

□

Above we have shown that the leakage from a single decryption operation can be perfectly simulated. Then, we will use this observation to prove that the leakage from several decryption operations will also not help the adversary to learn about the encoded secret. For this, we will require to refresh the encoded secret key periodically. Therefore, after the decryption operation, the refreshing protocol $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}$ is executed on L and R and it outputs L' and R' respectively. Then, we set (L', R') as the secret key for the next round. We denote such decryption operation by Dec'', which is $(\mathrm{Dec}' + \mathrm{Refresh}^{n,2}_{\mathbb{Z}_p})$. Now it is possible to show that an adversary cannot learn enough information on the secret key, even he can issue arbitrary ℓ number of queries to the decryption oracle leaking information on the secret. The following is the analogous version of Lemma 8 in Dziembowski and Faust [12].

Lemma 2. *Let* $(pk, sk) \leftarrow \mathrm{KeyGen}(1^k)$ *and* $\ell, n \in \mathbb{N}$. *Suppose that* $\Phi^{n,2}_{\mathbb{Z}_p}$ *is a* (λ, ϵ)*-secure LRS and that the decryption process* $\mathrm{Dec}''(\cdot)$ *is only run on valid ciphertexts. Let* W *be the vector* $(1, \alpha)$ *with* $\alpha = \log_{g_1}(g_2)$. *Then for every* $S = (a_1, b_1), S' = (a'_1, b'_1)$ *that satisfies* $g_1^{a_1} g_2^{b_1} = g_1^{a'_1} g_2^{b'_1}$, *and any* $(\lambda/2 - 1)$*-limited adversary* $\mathcal{A}$, *we have,*

$$\Delta(\mathrm{Exp}^{\mathrm{aux}}_{\mathrm{Dec}''(\cdot)}(\mathcal{A}, S, \ell, W, pk), \mathrm{Exp}^{\mathrm{aux}}_{\mathrm{Dec}''(\cdot)}(\mathcal{A}, S', \ell, W, pk)) \leq \epsilon_1 \ ,$$

where ϵ_1 *is negligible in the statistical security parameter* n.

The proof is as same as the proof of Lemma 8 of Dziembowski and Faust [12].

Proof. (sketch) Note that (W, pk) defines a 1-dimensional subspace $Z \subset (\mathbb{Z}_p)^2$ that contains all the pairs (a_1, b_1) that correspond to the public key pk. For any $S, S' \in Z$ and any $(\lambda/2 - 1)$-limited adversary $\mathcal{A}$ we have by the Theorem 2 (Generalization of Theorem 1) of Dziembowski and Faust [12], for refreshing of $\Phi^{n,2}_{\mathbb{Z}_p}$ that,

$$\Delta(\mathrm{Exp}^{\mathrm{aux}}_{\mathrm{Refresh}}(\mathcal{A}, S, \ell, W, pk), \mathrm{Exp}^{\mathrm{aux}}_{\mathrm{Refresh}}(\mathcal{A}, S', \ell, W, pk)) \leq 2\ell p^6(3\epsilon + 2p^{-n-5}) \ . \quad (1)$$

The above experiment addresses only the leakage from the refreshing of (L_i, R_i). In order to combine this with leakage from the decryption operation we use the leakage simulation from Lemma 1. We can apply this lemma since (1)–equation 1 is shown by reduction to the (λ, ϵ)-security of $\Phi^{n,2}_{\mathbb{Z}_p}$ and (2)–aux_i is known to the leakage simulator. This completes the proof of Lemma 2, by proving that ϵ_1 is negligible in the statistical security parameter n. □

Now we prove that our construction LR-PKE is CCLA2-secure in the standard model under the decision BDH assumption.

Theorem 4. *The proposed public-key encryption scheme LR-PKE is CCLA2-secure in the standard model, if the decision BDH assumption holds, the underlying signature scheme* Sig *is* SUF-OT*-secure, and the underlying refreshing protocol* $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}$ *is* $(\ell, \lambda/2, \epsilon_1)$*-secure refreshing protocol of a* (λ, ϵ)*-secure leakage-resilient storage scheme* $\Phi^{n,2}_{\mathbb{Z}_p}$.

Let $\mathcal{A}$ *be any adversary against the CCLA2 challenger of the public-key encryption scheme LR-PKE. Then, the advantage of* $\mathcal{A}$, $\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{LR-PKE}}(\mathcal{A})$ *is:*

$$\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{LR-PKE}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{decisionBDH}}_{\mathbb{G},\mathbb{G}_T,p,e}(\mathcal{B}) + \mathrm{Adv}^{\text{SUF-OT}}_{\mathrm{Sig}}(\mathcal{C}) + \epsilon_1 \ .$$

Proof. We prove the Theorem 4 using the game hoping technique.

Game 0: This is the original experiment given in Definition 5. The initial encryption/decryption keys are generated as $(pk, sk) \leftarrow \mathrm{KeyGen}'(1^k)$. This allows us to answer the decryption and leakage queries normally. Game 0 challenger runs $\mathrm{SigGen}(1^k)$ to obtain a key pair (svk^*, ssk^*), and sets svk^* as the target identity. When $\mathcal{A}$ sends $m_0, m_1 \in \mathbb{G}_T$ as a challenge query, target ciphertext c^* is computed normally for a message $m_b \xleftarrow{\$} \{m_0, m_1\}$. More precisely, we compute $c^* = (svk^*, ct^*, \sigma^*)$ as follows: randomly pick $s \xleftarrow{\$} \mathbb{Z}_p$, compute $ct^* = \big(u^* = e(h, g_3)^s \cdot m_b, v_1^* = g_1^s, v_2^* = g_2^s, w^* = \mathrm{H}(svk^*)^s\big), \sigma^* \leftarrow \mathrm{Sign}(ssk^*, ct^*)$ and get $c^* = (svk^*, ct^*, \sigma^*)$. Since this is the original experiment given in Definition 5, we have,

$$\mathrm{Adv}_{\text{Game 0}}(\mathcal{A}) = \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{LR-PKE}}(\mathcal{A}) \ .$$

Game 1: This is as Game 0, other than we generate the target ciphertext c^* of the message m_b using the secret key $sk = (a_1, b_1)$. More precisely, we compute $c^* = (svk^*, ct^*, \sigma^*)$ as follows: randomly pick $s \xleftarrow{\$} \mathbb{Z}_p$, compute $ct^* = \big(u^* = e({g_1}^s, g_3^{a_1}) \cdot e({g_2}^s, g_3^{b_1}) \cdot m_b, v_1^* = {g_1}^s, v_2^* = {g_2}^s, w^* = \mathrm{H}(svk^*)^s\big), \sigma^* \leftarrow \mathrm{Sign}(ssk^*, ct^*)$ and get $c^* = (svk^*, ct^*, \sigma^*)$. Obviously, the distance between Game 0 and Game 1 is 0. Thus,

$$|\mathrm{Adv}_{\text{Game }0}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }1}(\mathcal{A})| = 0 \ .$$

Game 2: This is as Game 1, other than we sample ${v_1}^*$ and ${v_2}^*$ at random such that $\log_{g_1}({v_1}^*) \neq \log_{g_2}({v_2}^*)$ to generate the target ciphertext c^* of the message m_b.

The view of Game 2 can be simulated with the aid of a decision BDH challenger, and it is possible to construct an algorithm $\mathcal{B}$ against the decision BDH challenge, using the adversary $\mathcal{A}$ as a sub routine. Algorithm $\mathcal{B}$ is given as input a tuple $(g_1, g_1^{a_1}, g_1^{a_2}, g_1^{a_3}, T)$ where T is either $e(g_1, g_1)^{a_1 a_2 a_3}$ or an uniform random element in $\mathbb{G}_T$. Algorithm $\mathcal{B}$ sets $A_1 = g_1^{a_1}, A_2 = g_1^{a_2}, A_3 = g_1^{a_3}$. Then $\mathcal{B}$ picks random elements $\beta, b_1 \in \mathbb{Z}_p$ and sets $g_2 := g_1^{\beta}$ and $h := A_1 {g_2}^{b_1}$. He picks again a random element $\alpha_1 \in \mathbb{Z}_p$ and sets $\widetilde{h} = h^{-svk^*} {g_1}^{\alpha_1} {g_2}^{b_1 svk^*}$. Then he gives public parameters $mpk = (g_1, g_2, h, g_3 = A_2, \widetilde{h})$ to $\mathcal{A}$. Let $b_1 svk^*$ be α_2. When $\mathcal{A}$ sends $m_0, m_1 \in \mathbb{G}_T$ as a challenge query, algorithm $\mathcal{B}$ picks $m_b \xleftarrow{\$} \{m_0, m_1\}$ and computes $ct^* = \big(m_b \cdot T \cdot e(A_2, A_3^{\beta})^{b_1}, A_3, {A_3}^{\beta}, {A_3}^{\alpha_1 + \beta\alpha_2}\big)$. Then computes $\sigma^* \leftarrow \mathrm{Sign}(ssk^*, ct^*)$ and sends $c^* = (svk^*, ct^*, \sigma^*)$ to $\mathcal{A}$. If $T = e(g_1, g_1)^{a_1 a_2 a_3}$, then the view simulated by algorithm $\mathcal{B}$ is identical to Game 1, otherwise identical to Game 2. Thus,

$$|\mathrm{Adv}_{\text{Game }1}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }2}(\mathcal{A})| \leq \mathrm{Adv}^{\text{decisionBDH}}_{\mathbb{G},\mathbb{G}_T,p,e}(\mathcal{B}) \ .$$

Game 3: This is as Game 2, other than we reject to answer the decryption queries for invalid ciphertexts, that is $c = (svk, ct, \sigma)$ such that $\mathrm{Vrfy}(svk, ct, \sigma) \neq 1$.

The view of Game 3 can be simulated with the aid of a strong one-time signature challenger of a signature scheme Sig, and it is possible to construct an algorithm $\mathcal{C}$ against the SUF-OT challenge, using the adversary $\mathcal{A}$ as a sub routine. The SUF-OT challenger runs $\mathrm{SigGen}(1^k)$ to obtain a key pair (svk^*, ssk^*), and sends svk^* to the algorithm $\mathcal{C}$. The algorithm $\mathcal{C}$ sets svk^* as the target identity. When $\mathcal{A}$ sends $m_0, m_1 \in \mathbb{G}_T$ as a challenge query, algorithm $\mathcal{C}$ picks $m_b \xleftarrow{\$} \{m_0, m_1\}$ and computes ct^* as in Game 2. Then query the SUF-OT challenger to obtain the signature σ^* for the ciphertext ct^*, processes the target ciphertext $c^* = (svk^*, ct^*, \sigma^*)$. Note that the adversary does not have the signing key ssk^*, which corresponds to the target identity svk^*. Therefore, if the adversary produces a (valid) decryption query $(c = (svk^*, ct, \sigma)$ such that $(ct, \sigma) \neq (ct^*, \sigma^*)$, that ciphertext consists a forged signature. As long as the adversary $\mathcal{A}$ does not make a forgery the the view simulated by algorithm $\mathcal{C}$ is identical to Game 2, otherwise identical to Game 3. Thus,

$$|\mathrm{Adv}_{\text{Game }2}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }3}(\mathcal{A})| \leq \mathrm{Adv}^{\text{SUF-OT}}_{\text{Sig}}(\mathcal{C}) \ .$$

Game 4: This is as Game 3 with the following changes.

1. Sample (a_1, b_1) at random and compute pk from (a_1, b_1). Sample $(a_1', b_1') \leftarrow \{(y, z) | {g_1}^y {g_2}^z = {g_1}^{a_1} {g_2}^{b_1}\}$ at random and sets $\mathrm{Encode}^{n,2}_{\mathbb{Z}_p}(a_1', b_1')$ as sk'.
2. Answer the decryption and leakage queries with sk' until the adversary asks for the challenge (m_0, m_1).
3. Generate the target ciphertext c^* by using (a_1, b_1) as in the previous game.
4. Answer the decryption and leakage queries with sk'.

It is easy to see that the distance between Game 4 and Game 3 is negligible by the Lemma 2. This is as same as Game 5 in the proof of Theorem 4 of Dziembowski and Faust [12]. Thus,

$$|\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \epsilon_1 \ .$$

Consider the target ciphertext $c^* = (svk^*, ct^*, \sigma^*)$ where $ct^* = \big(m_b \cdot z, {g_1}^{s_1}, {g_2}^{s_2}, H(svk)^{s_1}$ (analogous to the ct^* mentioned in Game 2). In that ct^*, the value $z = e(g_1, g_3)^{s_1 a_1} \cdot e(g_2, g_3)^{s_2 b_1}$ and $s_1 \neq s_2$. We can show that z behaves as an information theoretical one-time pad. An unbounded adversary in Game 4 can compute from the public key pk the value $\log_{g_1}(h) = a_1 + \beta b_1$. Since we used $sk' \leftarrow \mathrm{Encode}^{n,2}_{\mathbb{Z}_p}(a'_1, b'_1)$ to answer the leakage queries and (a'_1, b'_1) is chosen independent to (a_1, b_1), the adversary in Game 4 learns no more than $a_1 + \beta b_1$. Thus, together with the pk and the information leaned from the target ciphertext, the adversary $\mathcal{A}$ learns,

$$\begin{bmatrix} \log_{g_1}(h) \\ \log_{e(g_1,g_3)}(z) \end{bmatrix} = \begin{bmatrix} 1 & \beta \\ s_1 & \beta s_2 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \ .$$

Since $\beta \neq 0$ and $s_1 \neq s_2$ the 2×2 matrix is non-singular. Therefore, $\log_{g_1}(h)$ and $\log_{e(g_1,g_3)}(z)$ are linearly independent. Thus,

$$\mathrm{Adv}_{\text{Game 4}}(\mathcal{A}) = 0 \ .$$

Using above equations,

$$\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{LR-PKE}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{decisionBDH}}_{\mathbb{G},\mathbb{G}_T,p,e}(\mathcal{B}) + \mathrm{Adv}^{\mathrm{SUF\text{-}OT}}_{\mathrm{Sig}}(\mathcal{C}) + \epsilon_1 \ .$$

Thus, we get the claimed result.

□

3.2.6 Practical Information.

Now we discuss about the practical information about our leakage-resilient public-key encryption scheme, LR-PKE, which will be useful for implementation stage.

Leakage Bound. Trivially, the leakage bound of the public-key encryption scheme LR-PKE is as same as the leakage bound of the underlying refreshing protocol $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}$. As shown in Corollary 1 of Dziembowski and Faust [12], the leakage bound of the LRS $\Phi^{n,2}_{\mathbb{Z}_p}$ is $0.3|\mathbb{Z}_p|^n$ bits, for a statistical security parameter n. As shown in Corollary 2 of Dziembowski and Faust [12], the leakage bound of the refreshing protocol $\mathrm{Refresh}^{n,2}_{\mathbb{Z}_p}$ is $0.15|\mathbb{Z}_p|^n - 1 = 0.15n \log p - 1$ bits. Therefore, per leakage query $0.15n \log p - 1$ bits of leakage is allowed from each of the two encodes L_i and R_i of the secret key, independently from each other. For a security parameter k, $\log p \geq k$ and n is a function of k.

Computation Cost. The computation cost of our LR-PKE can be analysed as follows;

- Key Generation:
  1. computation of $h = 2$ exponentiations

  Total computation cost = 2 exponentiations (which is as same as the key generation of PKE)

- Encryption:
  1. computation of $H(svk) = 1$ exponentiation
  2. computation of $H(svk)^s = 1$ exponentiation
  3. computation of ${g_1}^s = 1$ exponentiation

4. computation of ${g_2}^s = 1$ exponentiation
5. computation of $e(h, g_3)^s = 1$ pairing and 1 exponentiation
6. computation of $\sigma = 1$ signature generation

Total computation cost = 5 exponentiations, 1 pairing and 1 signature generation (which is as same as the encryption of PKE)

- Decryption:

1. verifying $\sigma = 1$ signature verification
2. computation of $e(w, {g_1}^{r_1} {g_2}^{r_2}) = 2$ exponentiation and 1 pairing
3. computation of $e(v_1, H(svk)^{r_1}) = 1$ exponentiation and 1 pairing
4. computation of $e(v_2, H(svk)^{r_2}) = 1$ exponentiation and 1 pairing
5. computation of $e(v_1, g_3)^{R_1} \circ e(v_2, g_3)^{R_2} = 2$ pairings and $2n$ exponentiations
6. computation of $V^{-L} = n$ exponentiations

Total computation cost = $(4+3n)$ exponentiations, 5 pairings and 1 signature verification (differently, in the decryption of PKE the total is 6 exponentiations, 3 pairings and 1 signature verification)

Note that we ignored the cost of multiplication operations since it is significantly lower than exponentiation or pairing computations.

4 Conclusion and Future Works

In this paper, we present a construction of a public-key encryption scheme, namely LR-PKE, which is (continuous) CCLA2-secure in the standard model. The security of our public-key encryption scheme is proven in the standard model, under decision BDH assumption. For our construction of LR-PKE, we have used a strong one-time signature scheme and a leakage-resilient refreshing protocol as underlying building blocks. The leakage bound is $0.15n \log p - 1$ bits per leakage query, for a security parameter k and a statistical security parameter n, such that $\log p \geq k$ and n is a function of k. Moreover, it is possible to see that LR-PKE is efficient enough to be used for real-world usage. As for future works, we can use this LR-PKE scheme to construct leakage-resilient authenticated key exchange protocols. Further, it is interesting to focus on how to avoid the additional assumption of independent leakage from each splits of the secret key.

Acknowledgements research was initiated while the author was visiting NTT Secure Platform Laboratories, Japan. The author would like to thank Taechan Kim and Tatsuaki Okamoto of NTT for their valuable inputs for this work. Moreover, the author would like to acknowledge the grant URG/2018/19/E of University of Peradeniya, Sri Lanka.

References

[1] Bernstein, D. J. (2005) Cache-timing attacks on AES. Technical report. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[2] Brumley, D. and Boneh, D. (2003) Remote timing attacks are practical. *USENIX Security Symposium*, pp. 1–14.

[3] Kocher, P. C. (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. *CRYPTO*, pp. 104–113.

[4] Hutter, M., Mangard, S., and Feldhofer, M. (2007) Power and EM attacks on passive 13.56MHz RFID devices. *CHES*, pp. 320–333.

[5] Messerges, T., Dabbish, E., and Sloan, R. (2002) Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, **51**, 541–552.

[6] Micali, S. and Reyzin, L. (2004) Physically observable cryptography (extended abstract). *Theory of Cryptology Conference*, pp. 278–296.

[7] Akavia, A., Goldwasser, S., and Vaikuntanathan, V. (2009) Simultaneous hardcore bits and cryptography against memory attacks. *Theory of Cryptology Conference*, pp. 474–495.

[8] Brakerski, Z., Kalai, Y. T., Katz, J., and Vaikuntanathan, V. (2010) Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. *IACR Cryptology ePrint Archive*, **Report 2010/278**.

[9] Dodis, Y., Kalai, Y. T., and Lovett, S. (2009) On cryptography with auxiliary input. *STOC*, pp. 621–630.

[10] Naor, M. and Segev, G. (2009) Public-key cryptosystems resilient to key leakage. *CRYPTO*, pp. 18–35.

[11] Halevi, S. and Lin, H. (2011) After-the-fact leakage in public-key encryption. *Theory of Cryptology Conference*, pp. 107–124.

[12] Dziembowski, S. and Faust, S. (2011) Leakage-resilient cryptography from the inner-product extractor. *ASIACRYPT*, pp. 702–721.

[13] Boneh, D. and Boyen, X. (2004) Efficient selective-id secure identity-based encryption without random oracles. *EUROCRYPT*, pp. 223–238.

[14] Canetti, R., Halevi, S., and Katz, J. (2004) Chosen-ciphertext security from identity-based encryption. *EUROCRYPT*, pp. 207–222.

[15] Li, J., Teng, M., Zhang, Y., and Yu, Q. (2016) A leakage-resilient cca-secure identity-based encryption scheme. *Comput. J.*, **59**, 1066–1075.

[16] Zhou, Y. and Yang, B. (2017) Continuous leakage-resilient public-key encryption scheme with CCA security. *Comput. J.*, **60**, 1161–1172.

[17] Qin, B. and Liu, S. (2013) Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. *ASIACRYPT*, pp. 381–400.

[18] Sun, S., Gu, D., and Liu, S. (2013) Efficient leakage-resilient identity-based encryption with CCA security. *Pairing-Based Cryptography*, pp. 149–167.

[19] Groth, J. and Sahai, A. (2008) Efficient non-interactive proof systems for bilinear groups. *EUROCRYPT*, pp. 415–432.

[20] Canetti, R., Halevi, S., and Katz, J. (2003) Chosen-ciphertext security from identity-based encryption. *IACR Cryptology ePrint Archive*, **Report 2003/182**.

[21] Boneh, D. and Franklin, M. K. (2003) Identity-based encryption from the weil pairing. *SIAM J. Comput.*, **32**, 586–615.