A Generic Chosen-Ciphertext Key-Leakage Secure Public Key Encryption Scheme from Hash Proof System

Rupeng Yang [*1], Qiuliang Xu [†1], Yongbin Zhou [‡2], Chengyu Hu [§1], and Zuoxia Yu [¶1]

[1]School of Computer Science and Technology, Shandong University, Jinan, 250101, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

November 28, 2013

Abstract

We present a new generic construction of public key encryption (PKE) scheme that is *secure against a-posteriori chosen-ciphertext λ-key-leakage attacks* (LR-CCA-2 secure) from any universal *hash proof system* (HPS). Our construction relies only on the existence of universal hash proof systems, which makes our scheme simple, clean and efficient. Furthermore, our construction is a potential way to construct LR-CCA-2 secure PKE scheme from minimal assumption.

1 Introduction

Cryptography has achieved great success in the standard model, which assumes every party has its own secret and the adversary can only access such secret in some restricted way. However when a cryptographic system is implemented in the real world, some unintended information will be leaked. And we refer to attacks that obtain such information as *side channel attacks* (e.g. [13, 9]). To protect against side channel attacks, we can extend the standard model to capture such attacks and design cryptographic systems that are secure under these new models.

Micali and Reyzin put forward such a model in their pioneering work [14]. However, their model relies on the assumption that *only computation leaks information* and thus can not capture attacks that can obtain secret information that is not being used in computation such as cold boot attacks introduced by [9].

Inspired by the cold boot attacks, Akavia, Goldwasser and Vaikuntanathan [1] formalized a new model that can leak any efficiently computable functions of the secret key adaptively as long as the total amount of leakage is bounded. In [15], Naor and Segev extended the model in [1] to the setting of chosen-ciphertext security. In this work, we follow the model of [15] and present a new generic construction of a secure PKE scheme.

[*]orbbyrp@gmail.com
[†]Corresponding author: xql@sdu.edu.cn
[‡]zhouyongbin@iie.ac.cn
[§]hcy@sdu.edu.cn
[¶]yuzuoxia1990@gmail.com

1.1 Our Results

We propose a new generic construction of public key encryption scheme that is LR-CCA-2 secure from any universal hash proof system. More precisely, our scheme uses a *universal* HPS to mask the plaintexts and a $universal_2$ HPS to verify the ciphertexts. Since hash proof systems use some private information that should be put in the secret key, our scheme is more involved to prove and its leakage rate is lower compared with other generic constructions with LR-CCA-2 security (e.g. [15, 16]). However, our scheme is quite efficient as HPS can be implemented efficiently. And in fact the total leakage amount it can tolerate is not lower in the same setting compared to other constructions. We have a lower leakage rate just because we have a bigger secret key. In addition, our construction relies only on the existence of universal HPS, which makes our scheme simple and clean. Furthermore, Hazay et al. [11] showed that a weak HPS can be constructed from any standard PKE scheme. And if their construction can be adapted to construct a standard HPS, our scheme will be an LR-CCA-2 secure PKE from *minimal assumption*.

1.2 Related Work

Key-leakage attacks The model considered in this paper was first formalized by Akavia, Goldwasser and Vaikuntanathan in [1], and then it was extended to the setting of chosen-ciphertext security by Naor and Segev in [15].

There are two other generic constructions of PKE schemes [15, 16] that are LR-CCA-2 secure to our best knowledge. Both of them use a universal HPS to mask the plaintexts just as we do. The difference is the way they verify the ciphertexts. The scheme in [15] uses NIZK to verify the ciphertexts and thus their construction is very inefficient. The scheme in [16] uses a lossy filter to verify the ciphertexts and their efficiency is comparable to ours but the construction of lossy filter is more complicated.

There are various cryptographic primitives based on this model, e.g. the signature schemes [12, 3], the identity based encryption schemes [2] and so on.

Extended key-leakage attacks There are several ways to extend the model in this paper to capture more scenarios.

One approach is to relax the restriction of the amount of leakage information. Naor and Segev also present a model in [15] that the total amount of leakage information is not bounded and requires only that the secret key still has a certain amount of min-entropy given the leakage. And the model in [7] requires only that the secret key can not be efficiently recovered given the leakage.

Another approach is to relax the time that the adversary can obtain side-channel information. The work in [10] consider after-the-fact model in encryption scenarios and it allows the adversary to access the leakage oracle after the challenge phase. However, they also modify the notion of security into reserving the pseudo-entropy of encrypted message. The models in [4, 6] do not bound the overall amount of leakage and only restrict the leakage amount in and between each period after dividing the total lifetime of the system into periods.

2 Preliminaries

2.1 Basic Notions

We first recall some basic notions and terminologies here.

A function f mapping non-negative integers to non-negative reals is *negligible* if for every polynomial p(·), there exists $n_0 \geq 0$ such that for all integers $n > n_0$, $f(n) < \frac{1}{p(n)}$.

We denote by $x \xleftarrow{U} X$ the operation of picking x uniformly at random from a set X. We denote by U_m a random variable with uniform distribution over $\{0,1\}^m$. We denote by $\mathbb{E}_{x \leftarrow X}[f(x)]$ the expected value of random variable f(X). We denote by $[n]$ the set $\{1, 2, 3, \ldots, n\}$. All logarithms in this paper are with base 2.

Definition 2.1. *Let X and Y be two random variables with range U. The statistical distance between X and Y is defined as*

$$\Delta(X,\ Y) = \frac{1}{2} \sum_{u \in U} |Pr[X = u] - Pr[Y = u]|$$

Lemma 2.2. *Let X, Y and Z be three random variables.*

1. $\Delta(X,\ Z) \leq \Delta(X,\ Y) + \Delta(Y,\ Z)$.
2. *For every randomized function f,* $\Delta(f(X),\ f(Y)) \leq \Delta(X,\ Y)$. *We remark that, the randomness used by f is independent of X and Y.*

2.2 Leakage-Resilient Public-Key Encryption

We present the notion of *public key encryption scheme* and its security definition in key-leakage attacks here. Our approaching of security is identical to the definition for a-posteriori chosen-ciphertext key-leakage attacks in [5].

We consider a PKE scheme as a tuple Π = (KeyGen, Enc, Dec). Here *KeyGen* is a probabilistic polynomial-time key generation algorithm that takes the security parameter 1^n as input and outputs a public-key/private-key pair (PK, SK) from the key space $\mathcal{PK} \times \mathcal{SK}$. *Enc* is a probabilistic polynomial-time encryption algorithm that takes the public key PK and a message M from the message space $\mathcal{M}$ as input and outputs the ciphertext C. *Dec* is a deterministic polynomial-time decryption algorithm that takes the secret key SK and a ciphertext C as input and outputs either a message $M \in \mathcal{M}$ or a reject symbol $\perp$.

A PKE scheme should satisfy a "correctness" property. That is to say, the decryption of an encrypted message should be identical to the original message. More formally, a PKE scheme is correct if for all M from the message space $\mathcal{M}$ it holds that

$$Pr[Dec(SK,\ Enc(PK,\ M)) \neq M \ where\ (PK, SK) \leftarrow KeyGen(1^n)]$$

is negligible. Here the probability is taken over the internal coin tosses of the relevant algorithms.

Informally, a PKE scheme is secure under chosen-ciphertext key-leakage attacks if the adversary can not break the scheme even when he can obtain the decryptions of ciphertexts of his choosing and some partial information on the secret key of the encryption scheme.

We model the ability of such adversary by providing him with access to two kinds of oracles. One is the decryption oracle, denoted $\mathcal{D}$(SK, ·), takes a string C as input and outputs a decryption using

the secret key SK. We also denote by $\mathcal{D}_{\neq C}(SK,\ \cdot)$ a decryption oracle that decrypts any ciphertext other than C. Another oracle is the leakage oracle, denoted $\mathcal{O}^{\lambda,n}(SK,\ \cdot)$, takes the description of any efficiently computable function f as input and outputs f(SK). Here λ is a function mapping non-negative integers to non-negative integers and we require that the total number of bits leaked by the leakage oracle is at most $\lambda(n)$ for the security parameter n.

Now we consider the game $Expt^{LeakageCCA2}_{\Pi,\ \mathcal{A}}(n)$ played by an adversary and a benign simulator. For simplicity, we model the adversary $\mathcal{A}$ as two oracle machines $\mathcal{A}_1$ and $\mathcal{A}_2$ and then the game can be viewed as a simulator running by invoking these oracle machines.

1. $(PK,\ SK) \leftarrow KeyGen(1^n)$.
2. $(M_0,\ M_1,\ state) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\lambda,n}(SK,\ \cdot),\ \mathcal{D}(SK,\ \cdot)}(PK)$ *such that* $|M_0| = |M_1|$.
3. $b \overset{U}{\leftarrow} \{0,1\}$.
4. $C \leftarrow Enc(PK,\ M_b)$.
5. $b' \leftarrow \mathcal{A}_2^{\mathcal{D}_{\neq C}(SK,\ \cdot)}(C,\ state)$.
6. Output 1 if $b = b'$ and 0 otherwise.

Definition 2.3. *A public key encryption scheme* Π*=(KeyGen, Enc, Dec) is secure against a-posteriori chosen-ciphertext* $\lambda(n)$*-key-leakage attacks if for any probabilistic polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *it holds that:*

$$Adv^{LeakageCCA2}_{\Pi,\ \mathcal{A}}(n) \overset{def}{=} |Pr[Expt^{LeakageCCA2}_{\Pi,\ \mathcal{A}}(n) = 1] - \frac{1}{2}|$$

is negligible in n. Here the probability is taken over the internal coin tosses of the relevant algorithms and the adversary.

2.3 Randomness Extraction

We recall some basic notions relating to randomness extractors here.

Definition 2.4 ([8])**.** *Let X be a random variable. Then the min-entropy of X,denoted* $H_\infty(X)$*, is defined as*

$$H_\infty(X) = -\log(\underset{x \leftarrow X}{max} Pr[X = x])$$

Definition 2.5 ([8])**.** *Let X and Y be two random variables. Then the average min-entropy of X conditioned on Y, denoted* $\tilde{H}_\infty(X|Y)$*, is defined as*

$$\begin{aligned}\tilde{H}_\infty(X|Y) &= -\log(\mathbb{E}_{y \leftarrow Y}[\underset{x \leftarrow X}{max} Pr[X = x|Y = y]]) \\ &= -\log(\mathbb{E}_{y \leftarrow Y}[2^{-H_\infty(X|Y=y)}])\end{aligned}$$

Lemma 2.6 ([8])**.** *Let X, Y and Z be random variables. If Y has at most* 2^λ *possible values, then* $\tilde{H}_\infty(X|(Y,\ Z)) \geq \tilde{H}_\infty((X,\ Y)|Z) - \lambda \geq \tilde{H}_\infty(X|Z) - \lambda$.

The proof of this lemma can be found in [8]. Now we generalize this lemma for our security proof.

Lemma 2.7. *Let X, Y and Z be random variables. The range of X, Y and Z are $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$. In addition, $\mathcal{Y}$ has at most N possible values. Let $\mathcal{F}$ be a family of functions from $\mathcal{X}$ to Π and let π be an element of Π Then we have:*

$$\sum_{y\in\mathcal{Y}}\sum_{z\in\mathcal{Z}} \underset{f\in\mathcal{F}}{max} Pr[f(X)=\pi \wedge Y=y \wedge Z=z] \leq N \cdot \sum_{z\in\mathcal{Z}} \underset{f\in\mathcal{F}}{max} Pr[f(X)=\pi \wedge Z=z]$$

Proof.

$$\begin{aligned}
&\sum_{y\in\mathcal{Y}}\sum_{z\in\mathcal{Z}} \underset{f\in\mathcal{F}}{max} Pr[f(X)=\pi \wedge Y=y \wedge Z=z] \\
&\leq \sum_{y\in\mathcal{Y}}\sum_{z\in\mathcal{Z}} \underset{f\in\mathcal{F},y'\in\mathcal{Y}}{max} Pr[f(X)=\pi \wedge Y=y' \wedge Z=z] \\
&\leq N \cdot \sum_{z\in\mathcal{Z}} \underset{f\in\mathcal{F},y'\in\mathcal{Y}}{max} Pr[f(X)=\pi \wedge Y=y' \wedge Z=z] \\
&\leq N \cdot \sum_{z\in\mathcal{Z}} \underset{f\in\mathcal{F}}{max} Pr[f(X)=\pi \wedge Z=z]
\end{aligned}$$

□

Definition 2.8 ([8])**.** *Let U be a set. A function Ext : $U \times \{0,1\}^t \to \{0,1\}^m$ is an average-case (k, ϵ)-strong extractor if for all pairs of random variables (X, I) such that the range of X is U and $\tilde{H}_\infty(X|I) \geq k$, it holds that*

$$\Delta((Ext(X,\ R),\ R,\ I), (U_m,\ R,\ I)) \leq \epsilon$$

where R is uniform on $\{0,1\}^t$.

We remark that as we can easily encode elements in U into binary strings in our settings, we just write Ext as a function from $U \times \{0,1\}^t$ to $\{0,1\}^m$ for the given U for simplicity.

For the existence of average-case randomness extractors, Dodis et al. proved that from any family of universal hash functions we can get an average-case strong extractor. More precisely, we have:

Lemma 2.9. *Fix an output length m, for any $k \geq 0$ and $\epsilon \geq 0$ we have an average-case $(k,\ \epsilon)$-strong extractor from $U \times \{0,1\}^t$ to $\{0,1\}^m$ as long as $m \leq k - 2\log(\frac{1}{\epsilon}) + 2$.*

2.4 Hash Proof System

Now we present a brief description of *hash proof systems* introduced by Cramer and Shoup [5], and refer the reader to [5] for more details. For simplicity, we ignore negligible errors that mentioned in [5]. Lemma 2.2 shows that they have negligible effects on the results. As we will use $universal_2$ HPS to help verify the validity of the ciphertexts in our PKE scheme, we do not simply view HPS as a key encapsulation mechanism here.

2.4.1 Projective hashing

Let $\mathcal{X},\ \mathcal{Y},\ \mathcal{K},\ \mathcal{S}$ be finite, non-empty sets. Let $\mathcal{L}$ be a non-empty, proper subset of $\mathcal{X}$. Let $\mathcal{H} = (\mathcal{H}_k)_{k\in\mathcal{K}}$ be a collection of functions indexed by $\mathcal{K}$. And for every $k \in \mathcal{K}$, $\mathcal{H}_k$ is a function from $\mathcal{X}$ into $\mathcal{Y}$. Let α be a function from $\mathcal{K}$ into $\mathcal{S}$. Set $\mathbf{H} = (\mathcal{H},\ \mathcal{K},\ \mathcal{X},\ \mathcal{L},\ \mathcal{Y},\ \mathcal{S},\ \alpha)$.

Definition 2.10 ([5]). *$\boldsymbol{H} = (\mathcal{H},\ \mathcal{K},\ \mathcal{X},\ \mathcal{L},\ \mathcal{Y},\ \mathcal{S},\ \alpha)$, defined as above, is called a projective hash family (for $(\mathcal{X},\ \mathcal{L})$) if for all $k \in \mathcal{K}$, the action of $\mathcal{H}_k$ on $\mathcal{L}$ is determined by $\alpha(k)$.*

Definition 2.11 ([5]). *Let $\boldsymbol{H}$ defined as above be a projective hash family, and let $\epsilon \geq 0$ be a real number.*

$\boldsymbol{H}$ is ϵ-universal if for all $s \in \mathcal{S}$, $x \in \mathcal{X}\backslash\mathcal{L}$, and $y \in \mathcal{Y}$, it holds that

$$Pr[\mathcal{H}_k(x) = y\ \wedge\ \alpha(k) = s] \leq \epsilon Pr[\alpha(k) = s]$$

Here the probability is taken over the choosing of $k \in \mathcal{K}$ at random.

$\boldsymbol{H}$ is ϵ-universal$_2$ if for all $s \in \mathcal{S}$, $x,\ x^* \in \mathcal{X}$, and $y,\ y^* \in \mathcal{Y}$, with $x \notin \mathcal{L} \cup \{x^*\}$ it holds that

$$Pr[\mathcal{H}_k(x) = y\ \wedge\ \alpha(k) = s\ \wedge\ \mathcal{H}_k(x^*) = y^*] \leq \epsilon Pr[\alpha(k) = s\ \wedge\ \mathcal{H}_k(x^*) = y^*]$$

Here the probability is taken over the choosing of $k \in \mathcal{K}$ at random.

It is not hard to see that the concept of ϵ-universal describes the concept of min-entropy in some sense.

Lemma 2.12. *Let $\boldsymbol{H}$ defined as above be an ϵ-universal projective hash family. Let $\mathcal{S}'$ be the image of α on $\mathcal{K}$. Consider any conditional probability space where particular values of $x \in \mathcal{X}\backslash\mathcal{L}$ and $s \in \mathcal{S}'$ are fixed and k is chosen from $\mathcal{K}$ at random. We denote by Y the random variable $\mathcal{H}_k(x)$ on the above conditional probability space, Then we have*

$$H_\infty(Y) \geq \log(1/\epsilon)$$

Proof. As $\mathbf{H}$ is ϵ-*universal*, we have

$$\forall y \in \mathcal{Y}\ Pr[\mathcal{H}_k(x) = y\ \wedge\ \alpha(k) = s] \leq \epsilon Pr[\alpha(k) = s]$$

And as $s \in \mathcal{S}'$, we have $Pr[\alpha(k) = s] \neq 0$. So we have

$$\forall y \in \mathcal{Y}\ Pr[\mathcal{H}_k(x) = y\ |\alpha(k) = s] \leq \epsilon$$

And then we have

$$\max_{y\in\mathcal{Y}} Pr[\mathcal{H}_k(x) = y\ |\alpha(k) = s] \leq \epsilon$$

So we have

$$\max_{y\in\mathcal{Y}} Pr[Y = y] \leq \epsilon$$

And by the definition of min-entropy, we have

$$H_\infty(Y) \geq \log(1/\epsilon)$$

□

Note that the notion of "random variable" here is a little different from the notion in probability theory. However, This abuse seems harmless and is very common in cryptography, so we just ignore it in this paper.

Lemma 2.13. *An ϵ-universal$_2$ projective hash family is also an ϵ-universal projective hash family.*

Proof. Let **H** defined as above be an ϵ-*universal*$_2$ projective hash family. For all $s \in \mathcal{S}$, $x \in \mathcal{X}\backslash\mathcal{L}$ and $y \in \mathcal{Y}$, we fix an $x^* \in \mathcal{X}\backslash\{x\}$. Then we have:

$$\begin{aligned}
&Pr[H_k(x) = y \wedge \alpha(k) = s] \\
&= \sum_{y^* \in \mathcal{Y}} Pr[H_k(x) = y \wedge \alpha(k) = s \wedge H_k(x^*) = y^*] \\
&\leq \sum_{y^* \in \mathcal{Y}} \epsilon \cdot Pr[\alpha(k) = s \wedge H_k(x^*) = y^*] \\
&= \epsilon \cdot Pr[\alpha(k) = s]
\end{aligned}$$

□

2.4.2 Subset membership problem

Let **SM** be a *subset membership problem*. We consider an instance $\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}]$ of **SM**. Here, $\mathcal{X}$ and $\mathcal{W}$ are finite, non-empty sets. $\mathcal{L}$ is a non-empty proper subset of $\mathcal{X}$. $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ is a binary relation. For $x \in \mathcal{X}$ and $w \in \mathcal{W}$, w is a witness for x if $(x,\ w) \in \mathcal{R}$.

The *Subset membership problem* is to decide whether an element of $\mathcal{X}$ is chosen randomly from $\mathcal{L}$ or $\mathcal{X}\backslash\mathcal{L}$ given an instance Λ.

We also need some algorithms of **SM**.

1. *Param* is a probabilistic polynomial-time algorithm that takes the security parameter 1^n as input and outputs an instance Λ of **SM**.

2. *Sample* is a probabilistic polynomial-time algorithm that takes an instance Λ of **SM** as input and outputs a random $x \in \mathcal{L}$ together with a witness $w \in \mathcal{W}$ for x.

Definition 2.14 ([5])**.** *Let **SM** be a subset membership problem, and Param be its instance sampling algorithm. We say that **SM** is hard if for any probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that*

$$\begin{aligned}
Adv_{\mathcal{A}}^{\boldsymbol{SM}}(n) = |&Pr[\mathcal{A}(1^n,\ \Lambda,\ x) \text{ where } \Lambda \leftarrow Param(1^n) \text{ and } x \overset{U}{\leftarrow} \mathcal{L}] \\
&- Pr[\mathcal{A}(1^n,\ \Lambda,\ x) \text{ where } \Lambda \leftarrow Param(1^n) \text{ and } x \overset{U}{\leftarrow} \mathcal{X}\backslash\mathcal{L}]|
\end{aligned}$$

is negligible in n. Here the probability is taken over the sample of Λ, the uniformly chosen of x and the internal coin tosses of $\mathcal{A}$.

2.4.3 Hash proof system

A *hash proof system* **P** is based on a subset membership problem **SM** and associates with each instance Λ of **SM** a projective hash family **H**. More precisely, for each instance $\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}]$, the projective hash family $\mathbf{H} = (\mathcal{H},\ \mathcal{K},\ \mathcal{X},\ \mathcal{L},\ \mathcal{Y},\ \mathcal{S},\ \alpha)$ is based on the same $\mathcal{X}$ and $\mathcal{L}$.

Additionally, **P** provides following algorithms to carry out basic operations. We also remark that for a specific HPS **P**, given an instance of the underlying subset membership problem, the corresponding projective hash family is fixed and quite trivial to get. So we do not discriminate them here.

1. *Gen* is a probabilistic polynomial-time algorithm that takes an instance Λ of **SM** as input and outputs a pair (k,s). Here k is uniform on $\mathcal{K}$ and $s = \alpha(k)$.

2. *Pub* is a deterministic polynomial-time algorithm that takes an instance Λ of **SM**, $s \in \mathcal{S}$ such that s=$\alpha(k)$ for some $k \in \mathcal{K}$, and $x \in \mathcal{L}$ together with a witness $w \in \mathcal{W}$ for x as input and outputs $y \in \mathcal{Y}$ such that $\mathcal{H}_k(x) = y$.

3. *Priv* is a deterministic polynomial-time algorithm that takes an instance Λ of **SM**, $k \in \mathcal{K}$ and $x \in \mathcal{L}$ as input and outputs $y \in \mathcal{Y}$ such that $\mathcal{H}_k(x) = y$.

It is easy to see that Pub and Priv are in fact evaluating the same function. More precisely, $Pub(\Lambda,\ s,\ x,\ w) = Priv(\Lambda,\ k,\ x)$ as long as $s = \alpha(k)$ and w is a witness for x. We call it the "correctness" of HPS.

We also need some properties of HPS, and they come from the underlying projective hash families.

Definition 2.15 ([5])**.** *Let ϵ be a function mapping non-negative integers to non-negative reals. Let **SM** be a subset membership problem. Let **P** be an HPS for **SM**.*

*We say that **P** is ϵ-universal if for all $n \geq 0$ and for all $\Lambda \leftarrow Param(1^n)$, the projective hash family **H** that **P** associates with Λ is $\epsilon(n)$-universal.*

*We say that **P** is ϵ-universal$_2$ if for all $n \geq 0$ and for all $\Lambda \leftarrow Param(1^n)$, the projective hash family **H** that **P** associates with Λ is $\epsilon(n)$-universal$_2$.*

Now we present the notion of *extended hash proof system* which is very useful in our construction. An *extended HPS* **P** is also based on a subset membership problem **SM**. But it associates with each instance Λ of **SM** a projective hash family **H** as well as a finite set $\mathcal{E}$, and for each instance $\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}]$, the projective hash family $\mathbf{H} = (\mathcal{H},\ \mathcal{K},\ \mathcal{X} \times \mathcal{E},\ \mathcal{L} \times \mathcal{E},\ \mathcal{Y},\ \mathcal{S},\ \alpha)$ is based on $\mathcal{X} \times \mathcal{E}$ and $\mathcal{L} \times \mathcal{E}$. Also, the algorithms Pub and Priv are different from the ordinary HPS, as they additionally take an extension $e \in \mathcal{E}$ as input. All other things stay the same as those of ordinary HPS.

2.5 Computational Assumptions

Let *GroupGen* be a probabilistic polynomial-time algorithm that takes 1^n as input and outputs a tuple $(G,\ q,\ g)$, where q is a prime whose length is a polynomial of n, G is a cyclic group of order q and g is generator of G.

Definition 2.16. *The Decisional Diffie-Hellman problem is hard relative to GroupGen if for any probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that*

$$|Pr[\mathcal{A}(G,\ g,\ g^{r_1},\ g^{r_2},\ g^{r_1 r_2})] - Pr[\mathcal{A}(G,\ g,\ g^{r_1},\ g^{r_2},\ g^{r_3})]|$$

is negligible in n where $(q,\ G,\ g) \leftarrow GroupGen(1^n)$ and r_1, r_2, r_3 are chosen uniformly at random from Z_q. Here the probability is taken over the chosen of r_1, r_2 and r_3 as well as the internal coin tosses of GroupGen and $\mathcal{A}$.

The DDH assumption is the assumption that the DDH problem is hard for some algorithm GroupGen.

We call the tuple $(G,\ g,\ g^{r_1},\ g^{r_2},\ g^{r_3})$ a DH tuple if $r_3 = r_1 \cdot r_2$ and non-DH tuple otherwise. Note that, since the proportion of DH tuple is negligible, it is also hard to distinguish between a DH tuple and a non-DH tuple if the DDH assumption holds.

3 Generic Construction

In this section, we present a general construction of secure public-key encryption scheme that is LR-CCA-2 secure using appropriate hash proof systems for a hard subset membership problem and appropriate randomness extractors.

Let **SM** be a subset membership problem. Let $\mathbf{P}$ be an ϵ_1-universal HPS for **SM**. Let $\hat{\mathbf{P}}$ be an ϵ_2-*universal*$_2$ extended HPS for **SM**. We use the symbol $\hat{}$ to distinguish algorithms from the two hash proof systems above. Let λ, t and m be functions mapping non-negative integers to non-negative integers. We denote by $\mathcal{E}$ the set $\{0,1\}^t \times \{0,1\}^m$. Let ϵ_3 be a function mapping non-negative integers to non-negative reals. We require that ϵ_1, ϵ_2 and ϵ_3 are negligible functions. Let n be the security parameter, and we will omit it for simplicity in some cases. For example, we will write λ instead of $\lambda(n)$ if n is obvious from context.

To simplify the notation, we consider the scheme with a fixed security parameter n as well as a fixed instance $\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}]$ of **SM**, which is an output of the algorithm Param on input 1^n, although it should be generated in the key generation algorithm. With Λ fixed, let $\mathbf{H} = (\mathcal{H},\ \mathcal{K},\ \mathcal{X},\ \mathcal{L},\ \mathcal{Y},\ \mathcal{S},\ \alpha)$ be the projective hash family that $\mathbf{P}$ associates with Λ, and let $\hat{\mathbf{H}} = (\hat{\mathcal{H}},\ \hat{\mathcal{K}},\ \mathcal{X}\times\mathcal{E},\ \mathcal{L}\times\mathcal{E},\ \hat{\mathcal{Y}},\ \hat{\mathcal{S}},\ \hat{\alpha})$ be the projective hash family that $\hat{\mathbf{P}}$ associates with Λ. Let Ext : $\mathcal{Y} \times \{0,1\}^t \to \{0,1\}^m$ be an average-case $(\log(\frac{1}{\epsilon_1}) - \lambda - 1,\ \epsilon_3)$-strong extractor. From Lemma 2.9, we know that we can construct such an extractor as long as $m \le \log(\frac{1}{\epsilon_1}) - \lambda - 2\log(\frac{1}{\epsilon_3}) + 1$. The tuple ($\Lambda$, $\mathbf{H}$, $\hat{\mathbf{H}}$, Ext) are used as the public parameters of the encryption scheme and is public automatically.

Now we describe our PKE scheme with plaintext space $\{0,1\}^m$.

Key Generation

Run algorithm Gen of $\mathbf{P}$ and $\hat{\mathbf{P}}$ to generates the key pairs.

$$(k,\ s) \leftarrow Gen(\Lambda)$$

$$(\hat{k},\ \hat{s}) \leftarrow \hat{Gen}(\Lambda)$$

The public key PK is (s, $\hat{s}$).

The private key SK is (k, $\hat{k}$).

Encryption

To encrypt a message $M \in \{0,1\}^m$ under the public key PK=(s, $\hat{s}$), one does the following.

Sample a random $x \in \mathcal{L}$, together with a corresponding witness $w \in \mathcal{W}$ using algorithm Sample of **SM**.

Compute e=$Ext(Pub(\Lambda,\ s,\ x,\ w),\ r) \oplus M$ with a random seed $r \xleftarrow{U} \{0,1\}^t$.

Compute $\hat{y} = \hat{Pub}(\Lambda,\ \hat{s},\ x,\ w,\ r,\ e)$.

The ciphertext C is (x, r, e, $\hat{y}$).

Decryption

To decrypt a ciphertext C=(x, r, e, $\hat{y}$) under the secret key SK=(k, $\hat{k}$), one does the following.

Compute $\hat{y}' = \hat{Priv}(\Lambda,\ \hat{k},\ x,\ r,\ e)$.

Check whether $\hat{y} = \hat{y}'$ and if they are not equal, output the reject symbol $\perp$ and halt.

Compute M'=$Ext(Priv(\Lambda,\ k,\ x),\ r) \oplus e$.

Output the message M'.

The correctness of the scheme holds as the correctness of both hash proof systems in the scheme holds.

Our construction is very similar to the one presented in [5], and is just a generalization of the efficient LR-CCA-2 secure scheme in [15]. In fact, the scheme in [5] is itself an LR-CCA-2 secure PKE scheme when the underlying hash proof systems are constructed appropriately.

To make sure the scheme is LR-CCA-2 secure we should bound the leakage amount λ. We find that the scheme is secure as long as for any security parameter n we have

$$\lambda \leq min(\log(\frac{1}{\epsilon_1}) - m - 2\log(\frac{1}{\epsilon_3}) + 1,\ \log(\frac{1}{\epsilon_2}) - m - \omega(\log(n)))$$

The security of our scheme follows from the following theorem.

Theorem 3.1. *The above scheme is secure against a-posteriori chosen-ciphertext λ-key-leakage attacks, assuming **SM** is a hard subset membership problem, **P** is an ϵ_1-universal HPS for **SM** with negligible ϵ_1, $\hat{\boldsymbol{P}}$ is an ϵ_2-universal$_2$ extended HPS for **SM** with negligible ϵ_2, there exists appropriate randomness extractors, and $\lambda \leq min(\log(\frac{1}{\epsilon_1}) - m - 2\log(\frac{1}{\epsilon_3}) + 1,\ \log(\frac{1}{\epsilon_2}) - m - \omega(\log(n)))$ with negligible ϵ_3.*

Proof. We prove our main theorem using the game sequence technique formalized by Shoup in [17]. We first define a sequence of games, each played between an adversary and a benign simulator. We remark that in fact every game should take a security parameter as input, but for simplicity, we just consider games as they have already taken an input "1^n" in this paper. Then for each game, we can model it as a probability space and define a target event on it. Notice that, as each game is modeled as a probability space, we do not require the processes in the games to be efficient.

We denote by Π our PKE scheme. Let n be the security parameter. For any probabilistic polynomial-time adversary $\mathcal{A}$, we define following games and denote by S_i the event that $Game_i$ outputs 1.

Game$_0$

This is almost the original game $Expt^{LeakageCCA2}_{\Pi,\ \mathcal{A}}(n)$. More precisely:

1. The public parameter ($\Lambda(\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R})$, $\mathbf{H}$, $\hat{\mathbf{H}}$, Ext), the public key PK=(s, $\hat{s}$) and the secret key SK=(k, $\hat{k}$) are generated by invoking the Key generation algorithm of the PKE scheme.
2. $(M_0,\ M_1,\ state) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\lambda,n}(SK,\ \cdot),\ \mathcal{D}(SK,\ \cdot)}(PK)$ *such that* $|M_0| = |M_1|$.

3. $b \overset{U}{\leftarrow} \{0,1\}$.
4. $x^* \overset{U}{\leftarrow} \mathcal{L}, r^* \overset{U}{\leftarrow} \{0,1\}^t, u^* = Ext(Priv(\Lambda,\ k,\ x^*), r^*), e^* = u^* \oplus M_b, \hat{y}^* = \hat{Priv}(\Lambda,\ \hat{k},\ x^*,\ r^*,\ e^*)$, $C^* = (x^*,\ r^*,\ e^*,\ \hat{y}^*)$.
5. $b' \leftarrow \mathcal{A}_2^{\mathcal{D}_{\neq C^*}(SK,\ \cdot)}(C^*,\ state)$.
6. Output 1 if $b = b'$ and 0 otherwise.

Game$_1$

In this game, we make a small change in generating the challenge ciphertext C^*. The simulator chooses x^* uniformly at random from $\mathcal{X}\backslash\mathcal{L}$ instead of $\mathcal{L}$ this time. All other operations remain identical to $Game_0$.

Game$_2$

In this game, we make a small change in responding the two decryption oracles. The simulator additionally reject a ciphertext $C = (x,\ r,\ e,\ \hat{y})$ as long as x is not in $\mathcal{L}$ even if it can pass the verification. All other operations remain identical to $Game_1$.

Game$_3$

In this game, we make a small change in generating the challenge ciphertext C^* again. The simulator use a random string from $\{0,1\}^m$ to mask the plaintext this time. For more details, $e^* = u^* \oplus M_b$ for $u^* \overset{U}{\leftarrow} \{0,1\}^m$ while the generation of x^*, r^* and $\hat{y}^*$ as well as all other operations remain identical to $Game_2$.

The validity of our main theorem follows from the following claims.

Claim 3.2. $Pr[S_0] = Pr[Expt_{\Pi,\ \mathcal{A}}^{LeakageCCA2}(n) = 1]$.

Proof. It is obvious that $Pr[S_0] = Pr[Expt_{\Pi,\ \mathcal{A}}^{LeakageCCA2}(n) = 1]$ for the correctness of hash proof systems and the fact that the simulator can simulate these oracles perfectly. □

Claim 3.3. $|Pr[S_0] - Pr[S_1]|$ *is negligible in* n.

Proof. Assume $|Pr[S_0] - Pr[S_1]| = \epsilon$ for some non-negligible function ϵ. Then we can construct a probabilistic polynomial-time distinguishing algorithm that invokes $\mathcal{A}_1$ and $\mathcal{A}_2$ to break the subset membership problem **SM**.

On input $(\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}],\ x^*)$, where Λ is an instance of **SM** that is generated using the security parameter 1^n and $x^* \in \mathcal{X}$ is chosen uniformly at random from either $\mathcal{L}$ or $\mathcal{X}\backslash\mathcal{L}$, let $\mathbf{H}$, $\hat{\mathbf{H}}$ and Ext defined as in the description above of the algorithm KeyGen of the PKE scheme, then the distinguishing algorithm $\mathcal{A}'$ works as follows:

1. $(k,\ s) \leftarrow Gen(\Lambda)$, $(\hat{k},\ \hat{s}) \leftarrow \hat{Gen}(\Lambda)$. SK=(k, $\hat{k}$), PK=(s, $\hat{s}$).
2. $(M_0,\ M_1,\ state) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\lambda,n}(SK,\ \cdot),\ \mathcal{D}(SK,\ \cdot)}(PK)$ *such that* $|M_0| = |M_1|$.
3. $b \overset{U}{\leftarrow} \{0,1\}$.
4. $r^* \overset{U}{\leftarrow} \{0,1\}^t$, $u^* = Ext(Priv(\Lambda,\ k,\ x^*), r^*)$, $e^* = u^* \oplus M_b$, $\hat{y}^* = \hat{Priv}(\Lambda,\ \hat{k},\ x^*,\ r^*,\ e^*)$, $C^* = (x^*,\ r^*,\ e^*,\ \hat{y}^*)$.

5. $b' \leftarrow \mathcal{A}_2^{\mathcal{D}_{\neq C^*}(SK,\ \cdot)}(C^*,\ state)$.

6. Output 1 if $b = b'$ and 0 otherwise.

All the oracles above are simulated by $\mathcal{A}'$ using SK. In addition, it will respond these queries correctly unless the decryption query is C^* after it has been generated or the total amount of leakage bits will exceed λ after answering.

It is easy to see that the algorithms invoked by $\mathcal{A}'$ are all efficient. In addition, $\mathcal{A}'$ can simulate the oracles efficiently because decrypting using SK is efficient and all functions submitted to the leakage oracle are efficiently computable. Therefore, our algorithm $\mathcal{A}'$ is efficient.

Further more, it is evident that when x^* is chosen uniformly at random from $\mathcal{L}$, the computation in the above algorithm proceeds just as in $Game_0$ and when x^* is chosen uniformly at random from $\mathcal{X}\backslash\mathcal{L}$, the computation in the above algorithm proceeds just as in $Game_1$. So we have:

$$Pr[\mathcal{A}'(\Lambda, x) = 1 \ where \ x \xleftarrow{U} \mathcal{L}] = Pr[S_0]$$

$$Pr[\mathcal{A}'(\Lambda, x) = 1 \ where \ x \xleftarrow{U} \mathcal{X}\backslash\mathcal{L}] = Pr[S_1]$$

As $|Pr[S_0] - Pr[S_1]| = \epsilon$ for non-negligible function ϵ, we have:

$$|Pr[\mathcal{A}'(\Lambda, x) = 1 \ where \ x \xleftarrow{U} \mathcal{L}] - Pr[\mathcal{A}'(\Lambda, x) = 1 \ where \ x \xleftarrow{U} \mathcal{X}\backslash\mathcal{L}]| = \epsilon$$

for non-negligible function ϵ.

Therefore, we have an efficient distinguishing algorithm $\mathcal{A}'$ that breaks the hardness of the subset membership problem **SM** which is a contradiction. So we have $|Pr[S_0] - Pr[S_1]|$ is negligible in n. □

Claim 3.4. $|Pr[S_1] - Pr[S_2]|$ *is negligible in n.*

Proof. It is easy to see that $Game_1$ and $Game_2$ are defined on the same probability space as they use the same randomness. Let F_2 be the event that some ciphertexts $C = (x,\ r,\ e,\ \hat{y})$ where $x \notin \mathcal{L}$ but $\hat{y} = \hat{\mathcal{H}}_{\hat{k}}(x,\ r,\ e)$ is submitted to the decryption oracle in $Game_2$. We denote such query as "invalid query". Note that $Game_1$ and $Game_2$ are identical as long as F_2 does not occur. Then we have $Pr[S_0 \wedge \neg F_2] = Pr[S_1 \wedge \neg F_2]$. Therefore:

$$\begin{aligned} |Pr[S_0] - Pr[S_1]| &= |Pr[S_0 \wedge F_2] + Pr[S_0 \wedge \neg F_2] - Pr[S_1 \wedge F_2] - Pr[S_1 \wedge \neg F_2]| \\ &= |Pr[S_0 \wedge F_2] - Pr[S_1 \wedge F_2]| \\ &\leq Pr[F_2] \end{aligned}$$

Now, what we need is to show that $Pr[F_2]$ is negligible.

Assume $\mathcal{A}$ will make at most Q(n) queries to the decryption oracles for a polynomial Q since $\mathcal{A}$ is a probabilistic polynomial-time adversary. We denote by $\mathcal{Z} = \bigcup_{i=0}^{\lambda}\{0,1\}^i$ the range of leakage information. We denote by $\mathcal{EX}$ the set $\mathcal{X} \times \{0,1\}^t \times \{0,1\}^m$ and by $\mathcal{EL}$ the set $\mathcal{L} \times \{0,1\}^t \times \{0,1\}^m$. We define $\hat{\mathcal{G}}_{\hat{k}}(a) \stackrel{def}{=} \hat{\mathcal{H}}_{\hat{k}}(x,\ r,\ e)$ for $a = (x,\ r,\ e)$ for simplicity.

We first fix the values of $\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}]$, the adversary's coins, b, x^*, r^* and k in the probability space that $Game_1$ and $Game_2$ are defined on and denote by Ω the conditional space. We denote by F_2' the event in Ω that is defined identical to F_2. Now we consider arguments in Ω whose probability

is taken over the chosen of $\hat{k}$. Note that the distribution of $\hat{k}$ in Ω is identical to the distribution of $\hat{k}$ generated by algorithm $\hat{Gen}$. Let $\hat{K}$ be an random variable over the distribution of $\hat{k}$.

We denote by "Phase I" the phase before the challenge ciphertext generates, and by "Phase II" the phase after the challenge ciphertext generates. We denote by $F_2^{(1)\prime}$ the event that invalid query occurs in Phase I and by $F_2^{(2)\prime}$ the event that invalid query occurs in Phase II. Then we have

$$\begin{aligned} Pr[F_2'] &= Pr[F_2^{(1)\prime}] + Pr[F_2^{(2)\prime} \wedge \neg F_2^{(1)\prime}] \\ &\leq Pr[F_2^{(1)\prime}] + Pr[F_2^{(2)\prime}] \end{aligned}$$

And now we need only to bound $Pr[F_2^{(1)\prime}]$ and $Pr[F_2^{(2)\prime}]$.

We further fix the value of $\alpha(\hat{K})$ to be $\hat{s}$. Then the public parameter and public key are fixed now. Further more, the response of a decryption query is also fixed given a ciphextext C=(x, r, e, $\hat{y}$). To see this, we consider following two cases:

1. $x \in \mathcal{L}$: In this case, the decryption is in fact determined by the public key although it can not be evaluated limited by the computational ability.
2. $x \notin \mathcal{L}$: In this case, the decryption oracle rejects the ciphertext directly and so the response is also fixed.

Recall that the randomness of $\mathcal{A}$ is also fixed in Ω. Therefore, the view and the behavior of $\mathcal{A}$ are fixed step by step until he makes a query to the leakage oracle. Also, the first query submitted to the leakage oracle is fixed. So we can further fix z as the total output of the leakage oracle step by step again. We then let Z be an random variable over the distribution of z. This time, the view and the behavior of $\mathcal{A}$ are fixed until the ciphertext is generated. Now we bound the probability of $F_2^{(1)\prime}$. Since all the ciphertexts submitted to the decryption oracle in Phase I are fixed in the beginning conditioned on $\hat{s}$ and z, we can bound the probability of each query C=(a, $\hat{y}$) in Phase I being an invalid query by

$$\max_{a \in \mathcal{EX} \setminus \mathcal{EL},\ \hat{y} \in \hat{\mathcal{Y}}} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} | \alpha(\hat{K}) = \hat{s} \wedge Z = z]$$

Thus we can bound the probability of $F_2^{(1)\prime}$ conditioned on any consistent $\hat{s}$ and z:

$$Pr[F_2^{(1)\prime} | \alpha(\hat{K}) = \hat{s} \wedge Z = z] \leq Q \cdot \max_{a,\ \hat{y}} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} | \alpha(\hat{K}) = \hat{s} \wedge Z = z]$$

Here, the range of a is $\mathcal{EX} \setminus \mathcal{EL}$ and the range of $\hat{y}$ is $\hat{\mathcal{Y}}$.

Therefore, we have:

$$\begin{aligned} Pr[F_2^{(1)\prime}] &= \sum_{\hat{s} \in \hat{\mathcal{S}}} \sum_{z \in \mathcal{Z}} Pr[F_2^{(1)\prime} \wedge \alpha(\hat{K}) = \hat{s} \wedge Z = z] \\ &\leq Q \cdot \sum_{\hat{s} \in \hat{\mathcal{S}}} \sum_{z \in \mathcal{Z}} \max_{a,\ \hat{y}} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} \wedge \alpha(\hat{K}) = \hat{s} \wedge Z = z] \\ &\leq Q \cdot 2^{\lambda+1} \cdot \sum_{\hat{s} \in \hat{\mathcal{S}}} \max_{a,\ \hat{y}} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} \wedge \alpha(\hat{K}) = \hat{s}] \\ &\leq Q \cdot 2^{\lambda+1} \cdot \sum_{\hat{s} \in \hat{\mathcal{S}}} \epsilon_2 \cdot Pr[\alpha(\hat{K}) = \hat{s}] \\ &\leq Q \cdot 2^{\lambda+1} \cdot \epsilon_2 \end{aligned}$$

Here, the range of a is $\mathcal{EX}\backslash\mathcal{EL}$ and the range of $\hat{y}$ is $\hat{\mathcal{Y}}$.

To see that the second inequality in the above derivation holds, we first define an auxiliary function "eql", that is from $\hat{\mathcal{Y}} \times \hat{\mathcal{Y}}$ to $\{0,1\}$ and $eql(\hat{y_1},\ \hat{y_2}) = 1$ iff $\hat{y_1} = \hat{y_2}$ for $\hat{y_1} \in \hat{\mathcal{Y}}$ and $\hat{y_2} \in \hat{\mathcal{Y}}$. And now we can rewrite $\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y}$ into $\mathcal{F}_{a,\hat{y}}(\hat{K}) = 1$, where $\mathcal{F}$ is a family of function indexed by element form $\mathcal{EX} \times \hat{\mathcal{Y}}$ and defined as $\mathcal{F}_{a,\ \hat{y}}(\hat{k}) = eql(\hat{\mathcal{G}}_{\hat{k}}(a),\ \hat{y})$. Thus, by Lemma 2.7, the second inequality holds.

The third inequality in the above derivation holds for Lemma 2.13 and the property of universal projective hash family.

Now we continue to consider the game in Ω. The view and the behavior of $\mathcal{A}$ are fixed until the challenge ciphertext is generated when $\hat{s}$ and z are fixed. Also, the challenge plaintexts are fixed. For convenience, we let A be an random variable over the distribution of the tuple $(x^*,\ r^*,\ e^*)$ although it is determined by $\hat{s}$ and z now and we denote this value as a^*. We additionally fix the value of $\mathcal{G}_{\hat{K}}(A)$ to be $\hat{y}^*$. Then the challenge ciphertext $C^* = (a^*,\ \hat{y}^*)$ is fixed. As a result, the view and the behavior of $\mathcal{A}$ are completely fixed. And now, we bound the probability of each query C=(a, $\hat{y}$) in Phase II being an invalid query. We consider the following two cases:

1. $a = a^*$: Then we have $\hat{y} \neq \hat{y}^*$ since the decryption oracle requires $C \neq C^*$ and that means this query is not an invalid query with certainty.

2. $a \neq a^*$: Since all the ciphertexts submitted to the decryption oracle in Phase II are also fixed in the beginning conditioned on $\hat{s}$, z, a^* and $\hat{y}^*$, we can bound the probability by

$$\max_{a \in \mathcal{EX}\backslash(\mathcal{EL}\cup\{a^*\}),\ \hat{y}\in\mathcal{Y}} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} | \alpha(\hat{K}) = \hat{s} \wedge Z = z \wedge \mathcal{G}_{\hat{K}}(a^*) = \hat{y}^*]$$

Thus we can bound the probability of $F_2^{(2)\prime}$ conditioned on any consistent $\hat{s}$, z, a^* and $\hat{y}^*$:

$$\begin{aligned} & Pr[F_2^{(2)\prime} | \alpha(\hat{K}) = \hat{s} \wedge Z = z \wedge \mathcal{G}_{\hat{K}}(a^*) = \hat{y}^*] \\ & \leq Q \cdot \max_{a,\ \hat{y}} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} | \alpha(\hat{K}) = \hat{s} \wedge Z = z \wedge \mathcal{G}_{\hat{K}}(a^*) = \hat{y}^*] \end{aligned}$$

Here, the range of a is $\mathcal{EX}\backslash(\mathcal{EL} \cup \{a^*\})$ and the range of $\hat{y}$ is $\hat{\mathcal{Y}}$.

Therefore, we have:

$$
\begin{aligned}
Pr[F_2^{(2)\prime}] &= \sum_{\hat{s}\in\hat{\mathcal{S}}}\sum_{z\in\mathcal{Z}}\sum_{a^*\in\mathcal{EX}}\sum_{\hat{y}^*\in\hat{\mathcal{Y}}} Pr[F_2^{(2)\prime} \wedge \alpha(\hat{K}) = \hat{s} \wedge Z = z \wedge A = a^* \wedge \mathcal{G}_{\hat{K}}(A) = \hat{y}^*] \\
&\leq Q \cdot \sum_{\hat{s}\in\hat{\mathcal{S}}}\sum_{z\in\mathcal{Z}}\sum_{a^*\in\mathcal{EX}}\sum_{\hat{y}^*\in\hat{\mathcal{Y}}} \underset{a,\ \hat{y}}{max} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} \wedge \alpha(\hat{K}) = \hat{s} \wedge Z = z \wedge A = a^* \wedge \mathcal{G}_{\hat{K}}(A) = \hat{y}^*] \\
&\leq Q \cdot 2^{\lambda+1} \cdot \sum_{\hat{s}\in\hat{\mathcal{S}}}\sum_{a^*\in\mathcal{EX}}\sum_{\hat{y}^*\in\hat{\mathcal{Y}}} \underset{a,\ \hat{y}}{max} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} \wedge \alpha(\hat{K}) = \hat{s} \wedge A = a^* \wedge \mathcal{G}_{\hat{K}}(A) = \hat{y}^*] \\
&\leq Q \cdot 2^{\lambda+m+1} \cdot \sum_{\hat{s}\in\hat{\mathcal{S}}}\sum_{\hat{y}^*\in\hat{\mathcal{Y}}} \underset{a,\ \hat{y}}{max} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} \wedge \alpha(\hat{K}) = \hat{s} \wedge \mathcal{G}_{\hat{K}}(A) = \hat{y}^*] \\
&\leq Q \cdot 2^{\lambda+m+1} \cdot \underset{a^*}{max} \sum_{\hat{s}\in\hat{\mathcal{S}}}\sum_{\hat{y}^*\in\hat{\mathcal{Y}}} \underset{a,\ \hat{y}}{max} Pr[\hat{\mathcal{G}}_{\hat{K}}(a) = \hat{y} \wedge \alpha(\hat{K}) = \hat{s} \wedge \mathcal{G}_{\hat{K}}(a^*) = \hat{y}^*] \\
&\leq Q \cdot 2^{\lambda+m+1} \cdot \underset{a^*}{max} \sum_{\hat{s}\in\hat{\mathcal{S}}}\sum_{\hat{y}^*\in\hat{\mathcal{Y}}} \epsilon_2 \cdot Pr[\alpha(\hat{K}) = \hat{s} \wedge \mathcal{G}_{\hat{K}}(a^*) = \hat{y}^*] \\
&\leq Q \cdot 2^{\lambda+m+1} \cdot \epsilon_2
\end{aligned}
$$

Here, the range of a is $\mathcal{EX}\backslash(\mathcal{EL} \cup \{a^*\})$ and the range of $\hat{y}$ is $\hat{\mathcal{Y}}$.
The third inequality holds because x^* and r^* are fixed in Ω and e^* has only 2^m possible values.
Now we can bound the probability of $Pr[F_2']$:

$$
\begin{aligned}
Pr[F_2'] &\leq Pr[F_2^{(1)\prime}] + Pr[F_2^{(2)\prime}] \\
&\leq Q \cdot 2^{\lambda+1} \cdot \epsilon_2 + Q \cdot 2^{\lambda+m+1} \cdot \epsilon_2 \\
&\leq Q \cdot 2^{\lambda+m+2} \cdot \epsilon_2 \\
&\leq \frac{Q}{2^{\omega \log(n)}}
\end{aligned}
$$

that is negligible. The last inequality in the above derivation holds because $\lambda \leq \log(\frac{1}{\epsilon_2}) - m - \omega(\log(n))$.

And now since we have $Pr[F_2']$ is negligible in any conditional probability space Ω, we can conclude that $Pr[F_2]$ is negligible in the original probability space and that completes the proof. □

Claim 3.5. *$|Pr[S_2] - Pr[S_3]|$ is negligible in n.*

Proof. *$Game_2$* and *$Game_3$* are identical except the way they mask the plaintext, or in other words, the distribution of u^*.

We first fix the values of $\Lambda[\mathcal{X},\ \mathcal{L},\ \mathcal{W},\ \mathcal{R}]$, the adversary's coins, b, x^*, $\hat{k}$ and s in both games and denote the new games as *$Game_2'$* and *$Game_3'$*. We also denote by S_i' the corresponding target event in *$Game_i'$*. Let O_i' be an random variable over the distribution of the output of *$Game_i'$*, thus $O_i' = 1$ iff S_i' occurs. Here we have $i \in \{2, 3\}$ above. Note that, these fixed values are chosen from identical distributions in *$Game_2$* and *$Game_3$*.

The distribution of k is identical in *$Game_2'$* and *$Game_3'$*. In fact, they are identical to the distribution of k in Lemma 2.12. Let K be an random variable over the distribution of k above.

Further, we denote by z the value of the total output of the leakage oracle and denote by $aux(K)$ an random variable over the distribution of z since when a specific k is fixed z is also fixed. Also, aux is fixed and identical in both games. Let R be an random variable over the uniform distribution on $\{0,1\}^t$.

Now if we further fix the value of z, the value of r^* and the value of u^*, both games are fixed and identical. We denote by o the output of each game and we have $o = f(z,\ r^*,\ u^*)$ for a deterministic function f. Therefore, we have $O'_2 = f(aux(K),\ R,\ Ext(\mathcal{H}_k(x^*),\ R))$ and $O'_3 = f(aux(K),\ R,\ U_m)$.

From Lemma 2.12 and 2.6, we have

$$\tilde{H}_\infty(\mathcal{H}_K(x^*)|aux(K)) \geq H_\infty(\mathcal{H}_K(x^*)) - (\lambda+1) \geq \log\frac{1}{\epsilon_1} - (\lambda+1)$$

Then we have

$$\Delta((Ext(\mathcal{H}_K(x^*),\ R),\ R,\ aux(K)),\ (U_m,\ R,\ aux(K))) \leq \epsilon_3$$

from the definition of randomness extractors. And by Lemma 2.2, we have

$$\Delta(O'_2,\ O'_3) \leq \epsilon_3$$

In conclusive, we have

$$\begin{aligned}
|Pr[S'_2] - Pr[S'_3]| &= |Pr[O'_2 = 1] - Pr[O'_3 = 1]| \\
&= \frac{1}{2}\cdot(|Pr[O'_2 = 1] - Pr[O'_3 = 1]| + |Pr[O'_2 = 0] - Pr[O'_3 = 0]|) \\
&= \Delta(O'_2,\ O'_3) \\
&\leq \epsilon_3
\end{aligned}$$

that is negligible.

Now since $Pr[S'_2] - Pr[S'_3]$ is negligible in any conditional probability space, we have $Pr[S_2] - Pr[S_3]$ is negligible.

□

Claim 3.6. $Pr[S_3] = \frac{1}{2}$.

Proof. Since M_b is masked by a truly random string, it is clear that b is independent of the view of the adversary, and thus b is independent of b'. So we have $Pr[S_3] = \frac{1}{2}$. □

By Claim(3.2) $\sim$ Claim(3.6) and the triangle inequality, we have

$$\begin{aligned}
Adv^{LeakageCCA2}_{\Pi,\ \mathcal{A}}(n) &= |Pr[Expt^{LeakageCCA2}_{\Pi,\ \mathcal{A}}(n) = 1] - \frac{1}{2}| \\
&= |Pr[S_0] - Pr[S_3]| \\
&\leq |Pr[S_0] - Pr[S_1]| + |Pr[S_1] - Pr[S_2]| + |Pr[S_2] - Pr[S_3]|
\end{aligned}$$

which is negligible in n. And that completes the proof of our main theorem.

□

4 Instantiation

In this section, we present a secure PKE scheme based on the Decisional Diffie-Hellman (DDH) assumption from our generic encryption scheme construction above.

Assume the DDH problem is hard relative to an algorithm GroupGen. Let $(G,\ q,\ g_0)$ be an instance generated by invoking GroupGen on 1^n for the security parameter n. Let $g_1 = g_0^r$ for $r \leftarrow \mathbb{Z}_q^*$. Let $c,\ h,\ t,\ m$ and λ be functions mapping the security parameter n to non-negative integers and let ϵ be an function mapping the security parameter n to non-negative reals. We will omit n for simplicity in some cases. Let Ext be an $(h \log q - \lambda - 1,\ \epsilon)$-strong extractor from $G^h \times \{0,1\}^t$ to $\{0,1\}^m$. Let Γ be an injective function from $\mathbb{Z}_q^2 \times \{0,1\}^t \times \{0,1\}^m$ to $\mathbb{Z}_q^c$. We regard the parameters described above as public parameters and they should be generated in the key generation phase.

Now we describe the PKE scheme with fixed public parameters as above.

Key Generation

Choose $k_{i,j} \in \mathbb{Z}_q$ at random for $i \in \{0,1\}$ and $j \in [h]$ and let $s_j = g_0^{k_{0,j}} \cdot g_1^{k_{1,j}} \in G$ for $j \in [h]$.

Choose $\hat{k}_{i,j} \in \mathbb{Z}_q$ at random for $i \in \{0,1\}$ and $j \in [c+2h-1]$ and let $\hat{s}_j = g_0^{\hat{k}_{0,j}} \cdot g_1^{\hat{k}_{1,j}} \in G$ for $j \in [c+2h-1]$.

The private key is $((k_{i,j})_{i\in\{0,1\},j\in[h]},\ (\hat{k}_{i,j})_{i\in\{0,1\},j\in[c+2h-1]})$

The public key $((s_j)_{j\in[h]},\ (\hat{s}_j)_{j\in[c+2h-1]})$.

Encryption

To encrypt a message $M \in \{0,1\}^m$, one does the following.

Choose $w \in Z_q$, $r \in \{0,1\}^t$ at random.

Compute $x_0 = g_0^w$ and $x_1 = g_1^w$.

Compute $y = (s_j^w)_{j\in[h]}$.

Compute $e = Ext(y,\ r) \oplus M$.

Compute $(\gamma_j)_{j\in[c]} = \Gamma(x_0,\ x_1,\ r,\ e)$.

Compute $\hat{y} = (\hat{s}_j^w \cdot \prod_{i=1}^c \hat{s}_{h+i+j-1}^{\gamma_i w})_{j\in[h]}$.

The ciphertext is $(x_0,\ x_1,\ r,\ e,\ \hat{y})$.

Decryption

To decrypt a ciphertext $(x_0,\ x_1,\ r,\ e,\ \hat{y})$, one does the following.

Compute $(\gamma'_j)_{j\in[c]} = \Gamma(x_0,\ x_1,\ r,\ e)$.

Compute $\hat{y}' = (x_0^{\hat{k}_{0,j}} \cdot x_1^{\hat{k}_{1,j}} \cdot \prod_{i=1}^c (x_0^{\gamma'_i \hat{k}_{0,h+i+j-1}} \cdot x_1^{\gamma'_i \hat{k}_{1,h+i+j-1}}))_{j\in[h]}$.

Check whether $\hat{y} = \hat{y}'$ and if they are not equal, output the reject symbol $\perp$ and halt.

Compute $y' = (x_0^{k_{0,j}} \cdot x_1^{k_{1,j}})_{j\in[h]}$.

Compute $M' = Ext(y',\ r) \oplus e$

Output the message M'.

It is evident from [5] that the PKE scheme above is an instantiation of our generic construction with a $\frac{1}{q^h}$-*universal* HPS and a $\frac{1}{q^h}$-$universal_2$ HPS based on a hard subset membership problem. Thus, it is LR-CCA-2 secure as long as ϵ is negligible and $\lambda \leq h \log q - m - \omega(\log n)$. In addition, the leakage rate ranges from about $\frac{1}{10}$ to about $\frac{1}{6}$ by adjusting relevant parameters such as h.

We can also instantiate our generic PKE scheme with other computational assumptions such as the DCR assumption and the QR assumption. An interesting problem is how to construct a universal HPS with minimal assumptions, e.g. there exists one way trapdoor functions, and thus we can construct PKE schemes that is LR-CCA-2 secure with one way trapdoor functions directly.

5 Conclusion

We present a new generic construction of public key encryption scheme that is LR-CCA-2 secure from any universal HPS and we give an efficient instantiation of it based on DDH assumption. However, the leakage rate of our generic construction is low as it has a big secret key inherently. It is an interesting problem to improve the leakage rate of our scheme using more sophisticated techniques. Also our scheme will be an LR-CCA-2 secure PKE from minimal assumption if HPS can be constructed from any CPA secure PKE. And we leave it as an open problem.

References

[1] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography*, pages 474–495. Springer, 2009.

[2] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *Advances in Cryptology–EUROCRYPT 2010*, pages 113–134. Springer, 2010.

[3] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *Advances in Cryptology–EUROCRYPT 2011*, pages 89–108. Springer, 2011.

[4] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 501–510. IEEE, 2010.

[5] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in CryptologyEUROCRYPT 2002*, pages 45–64. Springer, 2002.

[6] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lopez-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 511–520. IEEE, 2010.

[7] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 621–630. ACM, 2009.

[8] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in cryptology-Eurocrypt 2004*, pages 523–540. Springer, 2004.

[9] J Alex Halderman, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. Lest we remember: cold-boot attacks on encryption keys. *Communications of the ACM*, 52(5):91–98, 2009.

[10] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *Theory of Cryptography*, pages 107–124. Springer, 2011.

[11] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In *Advances in Cryptology–EUROCRYPT 2013*, pages 160–176. Springer, 2013.

[12] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology–ASIACRYPT 2009*, pages 703–720. Springer, 2009.

[13] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in CryptologyCRYPTO96*, pages 104–113. Springer, 1996.

[14] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography*, pages 278–296. Springer, 2004.

[15] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *SIAM Journal on Computing*, 41(4):772–814, 2012.

[16] Baodong Qin and Shengli Liu. Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. 2013.

[17] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.