cryptography

Article

A CCA-PKE Secure-Cryptosystem Resilient to Randomness Reset and Secret-Key Leakage

Alfonso Labao * and Henry Adorna

Department of Computer Science, University of the Philippines Diliman, Quezon City 1101, Philippines; hnadorna@up.edu.ph
* Correspondence: alfonso.labao@upd.edu.ph

Abstract: In recent years, several new notions of security have begun receiving consideration for public-key cryptosystems, beyond the standard of security against adaptive chosen ciphertext attack (CCA2). Among these are security against randomness reset attacks, in which the randomness used in encryption is forcibly set to some previous value, and against constant secret-key leakage attacks, wherein the constant factor of a secret key's bits is leaked. In terms of formal security definitions, cast as attack games between a challenger and an adversary, a joint combination of these attacks means that the adversary has access to additional encryption queries under a randomness of his own choosing along with secret-key leakage queries. This implies that both the encryption and decryption processes of a cryptosystem are being tampered under this security notion. In this paper, we attempt to address this problem of a joint combination of randomness and secret-key leakage attacks through two cryptosystems that incorporate hash proof system and randomness extractor primitives. The first cryptosystem relies on the random oracle model and is secure against a class of adversaries, called non-reversing adversaries. We remove the random oracle oracle assumption and the non-reversing adversary requirement in our second cryptosystem, which is a standard model that relies on a proposed primitive called L^M lossy functions. These functions allow up to M lossy branches in the collection to substantially lose information, allowing the cryptosystem to use this loss of information for several encryption and challenge queries. For each cryptosystem, we present detailed security proofs using the game-hopping procedure. In addition, we present a concrete instantiation of L^M lossy functions in the end of the paper—which relies on the DDH assumption.

Keywords: cryptography; public-key encryption; chosen ciphertext attack; randomness attack; secret-key leakage attack

check for updates

Citation: Labao, A.; Adorna, H. A CCA-PKE Secure-Cryptosystem Resilient to Randomness Reset and Secret-Key Leakage. *Cryptography* **2022**, *6*, 2. https://doi.org/10.3390/cryptography6010002

Academic Editor: Josef Pieprzyk

Received: 6 December 2021
Accepted: 27 December 2021
Published: 4 January 2022

1. Introduction

Adaptive Chosen ciphertext attack (CCA2) secure cryptosystems. Since the invention of the Diffie–Helman key exchange and the RSA primitive, public-key cryptography has become one of the most well-studied areas in cryptography research [1]. Currently, the security notion required of any public-key cryptosystem is security against adaptive chosen ciphertext attacks [2,3], or CCA2 security. Security against adaptive chosen ciphertext attacks guarantees that ciphertexts are not malleable, which implies that ciphertexts cannot be modified in transit by some efficient adversarial algorithm. Initially, encryption schemes provided CCA2 security under the random oracle model [4]. Random oracle-based models are heuristic in approach and are randomness-recovering, i.e., they allow a scheme to recover its randomness during an encryption. However, they rely on very strong assumptions, for example, that some functions, i.e., hash functions, are indistinguishable from truly random functions. A more practical CCA2-secure public-key encryption scheme is presented in [3], which relies on the decisional Diffie–Helman (DDH) assumption. The scheme of [3] uses hash-proof systems, which involve projective hash functions that perform function delegation through auxiliary information. Without this auxiliary information, however, the function's behaviour is close to uniform and it is hard to distinguish as non-random. The design of several practical public-key encryption schemes essentially use this hash proof system by [5] or some variants of it [6,7].

Following the hash proof system of [3], several other CCA2 secure cryptosystems have been proposed. Among these is the CCA2 secure cryptosystem of [8] that relies on a lossy function primitive. The lossy function primitive is a collection of functions, such that some functions in the collection, i.e., the lossy functions, substantially lose significant information from the input. It is difficult, however, to determine if a function is lossy or not. By exploiting the loss of information in the function, along with the computational difficulty in determining the type of a function, several CCA2 secure cryptosystems can be developed under the standard model, thereby presenting an alternative to the practical CCA2 cryptosystem of [3].

Yet, while more practical and efficient CCA2 cryptosystems are being developed, recently, a number of cryptography papers have called into question the security guarantees provided by CCA2 security [9–14], due to newer types of attacks. Among the common categories of these newer attacks are (i) randomness attacks and (ii) secret-key leakage attacks. Both of these attack categories have been shown to be strong enough to trivially break CCA2 security. These attacks are briefly described as follows.

Randomness Attacks. The first category, i.e., randomness attacks, considers the case where the randomness used in cryptosystems fails to be truly random. These attacks tamper with the encryption process of a cryptosystem. Randomness failures can be due to a faulty, pseudorandom generator design and implementation [15], or due to simple attacks, such as virtual machine resets [16]. In a virtual machine reset—called randomness reset attack—a computer is forced to restart to some previous state stored in memory and random number variables are reset to some previous value. This applies, especially, to virtual machine systems that use virtual machine monitors (or hypervisors) to manage several operating systems. Given the increased use of cloud computing, such as with Amazon's servers, several systems have become reliant on virtual machines. A feature of a virtual machine monitor is the taking of snapshots [16] of the system's state, where the snapshot includes all items of the system in memory, at a certain point in time, for backup and fault tolerance. Included in this snapshot are the random numbers used by the operating system for its encryptions. A hacker, however, may force a virtual machine to be reset to a prior snapshot and re-use the randomness therefrom. For instance, a hacker may perform a denial-of-service attack against a virtual machine, whereupon the virtual machine is forced to be reset to some previous state. In particular [17] point out that snapshots of virtual machines may impair security due to the reuse of security-critical states. Ref. [18] exhibits virtual machine reset vulnerabilities in TLS clients and servers, wherein the attacker takes advantage of snapshots to expose a server's DSA signing key. Given these actual examples, ref. [16] considered the effect of virtual machine resets on existing CCA2 secure cryptosystems. The results were not positive, as [16] showed a scenario wherein an adversary can trivially break CCA2 security by exploiting such vulnerabilities.

We demonstrate such vulnerabilities on a simpler cryptosystem in Section 3, wherein we present a concrete example of a randomness reset attack in the context of the ElGamal cryptosystem, along with the effect of a randomness reset attack on the formal definition of CCA2 security as done in [16]. Briefly, CCA2 security is formally defined in terms of an attack game between a challenger and an adversary [19], where the adversary may perform challenge queries and decryption queries. The adversary's task is to correctly guess the underlying message of a challenge ciphertext, given that the challenge ciphertext cannot serve as input to a decryption query. An attack game with a randomness reset attack will incorporate additional encryption queries, in which random numbers can be set, by the adversary, to some previous value. These may present difficulties to some existing cryptosystems, since, unlike challenge ciphertexts, ciphertexts from encryption queries may validly serve as input for decryption queries. In fact, randomness reset attacks are so strong that [11,16,20] have been forced to rule-out situations wherein adversaries can perform arbitrary queries. Instead, adversaries are assumed to satisfy the equality pattern, respecting constraint. The cryptosystem of [16] provides a generic transformation to render a CCA2 public-key cryptosystem secure against randomness reset attacks. The transformation involves feeding a random number and an associated plaintext message to a pseudorandom function. If the joint entropy of the random number and the plaintext message is high enough, the security properties of the pseudorandom function can fix any faulty randomness in the cryptosystem. In Section 4, we present a list of schemes that consider various other types of randomness reset attacks.

Secret Key Leakage Attacks. The second category, i.e., secret-key leakage attacks, considers the case where an adversary learns bits of the secret key [9]. These attacks tamper with the decryption process of a cryptosystem. Leakage of secret keys may, perhaps, be due to some devious means, such as side-channel attacks. For instance [21] have reported that practical implementations of cryptosystems in software are often vulnerable to side-channel attacks. For example, the power traces of 8000 encryptions are sufficient to extract the secret key of ASIC AES, which is substantially faster than a brute-force search for the secret key. It follows that, given enough bits of the secret key, a simple exhaustive search over the set of candidate keys can break any cryptosystem's security. A formal definition of this attack is first considered in [9]. Several articles have provided cryptosystems that are provably secure against secret-key leakage attacks, on top of CCA2 security [12,13]. In particular, the scheme of [13] provides a cryptosystem that is secure against a constant factor of secret key bits leaked to the adversary, where the factor can be as high as $1/2 - o(1)$ bits of the secret key. The scheme of [13] is composed of an ensemble of various cryptographic primitives, such hash proof systems [5], lossy functions [8], and randomness extractors [22,23]. In particular, ref. [13] proposed the one-time lossy filter, which is a special type of lossy function that does not implement a trapdoor. Unlike the cryptosystem of [3], the scheme of [13] is randomness-recovering and can tolerate a higher degree of secret-key leakage. The paper of [12] showed that the cryptosystem of [13] is also secure against arbitrary functions of secret-key leakage. In Section 3, we illustrate how secret-key leakage attacks would affect the formal definition of CCA2 security (expressed as an attack game between a challenger and an adversary). Briefly, secret-key leakage attacks would provide additional leakage queries for the adversary, where he gets to learn a constant factor of bits of the secret key. In Section 4, we present a list of various other types of secret-key leakage attacks along with the primitives that they use.

Our contributions. As noted in [11], given these new types of attacks, an interesting problem is the construction of a public-key cryptosystem that is both resistant to randomness attacks and secret-key leakage attacks. Attacks that jointly involve both types effectively tamper with both the encryption and decryption processes of a cryptosystem, and the cryptosystem has to deal with attacks from both sides. On the one hand, in a randomness attack, the randomness involved in encryption is tampered to some value dictated by the adversary. On the other hand, in a secret-key leakage attack, the adversary learns information about the secret key involved in decryption. In terms of attack games between a challenger and an adversary, this implies that the adversary has access to additional encryption queries and secret-key leakage queries, aside from the usual decryption and challenge queries in a CCA2 attack game. To address these challenges, we propose two cryptosystems; the first is a random oracle model, and the second is a standard model that relies on a proposed primitive, called L^M lossy functions. A collection of L^M lossy functions provides multiple lossy branches and is simple to construct from existing ABO lossy functions [8]. Having multiple lossy branches is crucial for our cryptosystem, given that the adversary may query multiple encryption and challenge queries, and, unlike challenge ciphertexts, encrypted ciphertexts can validly serve as input for decryption. By having several lossy branches, the cryptosystem is able to exploit the loss of information given by the lossy branch even under multiple encryption and challenge queries. We can say that the L^M lossy function forms the core primitive of our second cryptosystem since, without this primitive, the hash proof systems would be insufficient for security (at least in the context of constructions). The presentation of our cryptosystems follows [2,10], which begins from random oracle models and is followed by standard models. This is because, while the random oracle model from [4] is useful for simplifying security proofs, it relies on the strong assumption that some hash functions are truly random, which may not necessarily be true, in practice [2]. For this reason, standard models usually follow initial random models, albeit with some added complexity in their schemes. Both of our proposed cryptosystems apply several primitives, such as hash proof systems, pseudorandom functions, and randomness extractors.

To put our contributions into context, the problem mentioned in [11] considers general classes of related randomness and related secret-key leakage attacks. For this paper, however, we approach the problem under a more limited class of randomness reset attacks [16], in which random numbers are reset to previous values, and under constant-bit secret-key leakage attacks [13], where a constant number of bits of the secret key are leaked. At the end of the paper, we present concrete instances of L^M collections that rely on the decisional Diffie–Helman assumption, and on ElGamal matrix encryptions. We present security proofs for our proposed cryptosystems using the well-known game-hopping proving scheme, as described in [2,19].

2. Preliminaries

2.1. Notations

Given the set of natural numbers $\mathbb{N}$, let $[a]$ denote the set $\{1, 2, \ldots, a\}$ for any $a \in \mathbb{N}$. Let $\kappa \in \mathbb{N}$ denote a security parameter following standard cryptography literature [2]. A function $f(\kappa)$ is *negligible in* κ if $f(\kappa) = o(\kappa^{-c})$ for every fixed constant c. A function $f(\kappa)$ is *superpolynomial in* κ if $1/f(\kappa)$ is negligible. Throughout the paper, the notation $x \leftarrow X$ refers to x being randomly drawn from the probability distribution of a random variable, X. Let $\mathcal{A}$ be any probabilistic polynomial-time algorithm. The *advantage* of $\mathcal{A}$ is defined to be its capacity to distinguish between the probability distributions of two collections of random variables. For instance, let $\mathcal{X} = \{X_\kappa\}_{\kappa \in \mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa \in \mathbb{N}}$ be two collections of random variables indexed by κ. The advantage of $\mathcal{A}$, in this instance, is $|\Pr(\mathcal{A}(1^\kappa, x) = 1) - \Pr(\mathcal{A}(1^\kappa, y) = 1)|$ for $x \leftarrow X_\kappa$ and $y \leftarrow Y_\kappa$. Two collections of random variables $\mathcal{X}$ and Y are *computationally indistinguishable* if the advantage of any polynomial-time algorithm is negligible in κ. The *statistical distance* between two random variables X and Y with the same domain D is denoted as $\Delta(X, Y) = (1/2) \sum_{z \in D} |\Pr(X = z) - \Pr(Y = z)|$ [22]. The *min-entropy* of X is denoted as $\Gamma_\infty(X) = -\log(\max_z \Pr(X = z))$. If X is conditioned on Y, the *average min-entropy of* X *conditioned on* Y is $\tilde{\Gamma}_\infty(X|Y) = -\log(E_{y \leftarrow Y} 2^{-\Gamma_\infty(X|Y=y)})$.

2.2. Hashing and Randomness Extractors

Given the security parameter $\kappa \in \mathbb{N}$, let $l(\kappa)$ and $l'(\kappa)$ be values of polynomials in κ. A *hash function* maps inputs of length $l(\kappa)$ to outputs of length $l'(\kappa)$, where $l'(\kappa) < l(\kappa)$. A family of hash functions $\mathcal{H} = \{h_i : X \to Y\}_{i \in \mathbb{N}}$ with domain X and range Y is *pairwise independent* if, for every distinct pair $x, x' \leftarrow X$ and $y, y' \leftarrow Y$, the probability that $h_i(x) = y$ and $h_i(x') = y'$ is equal to $1/|Y|^2$ for any $h_i \in \mathcal{H}$. On the other hand, if, for every distinct pair $x, x' \leftarrow X$, we have $\Pr_{h_i \leftarrow \mathcal{H}}(h_i(x) = h_i(x')) = 1/|Y|$ for any $h_i \in \mathcal{H}$, the family $\mathcal{H}$ is a *universal* family of hash functions, which is a strictly weaker property than pairwise independence [13]. A family of hash functions is *collision resistant* if no polynomial time algorithm can compute a distinct pair $x, x' \leftarrow X$ such that $h_i(x) = h_i(x')$ for any $h_i \in \mathcal{H}$. The following useful result regarding average min-entropy will be used in several security proofs.

Lemma 1 ([24]). *Given the random variables X, Y, and Z, suppose that Y has 2^r possible values; then, $\Gamma_\infty(X|Y, Z) \geq \Gamma_\infty(X|Z) - r$*

Definition 1. *Randomness Extractor. Let X and Z be random variables such that $X \in \{0,1\}^a$ and $Z \in \{0,1\}^b$ with $a, b \in \mathbb{N}$ and $b < a$. Let Y any random variable such that $\tilde{\Gamma}_\infty(X|Y) \geq \nu$ for some $\nu \geq 0 \in \mathbb{R}$. Let R be any random variable. An efficiently computable function, $E : X \times R \to Z$ is* *an average case* (ν, ϵ) *strong extractor if* $\Delta((Y, r, E(X, r)), (Y, r, Uz)) \leq \epsilon$*, where* $r \leftarrow R$*,* Uz *is the uniform distribution over* Z*, and* $\epsilon > 0 \in \mathbb{R}$*.*

Concrete instantiations of strong randomness extractors involve a family of universal hash functions. This leads to the following lemma.

Lemma 2 ([25]). *A universal family of hash functions* $\mathcal{H}$ *can be used as average-case* $(\tilde{\Gamma}_\infty(X|Y), \epsilon)$ *strong extractors whenever* $\tilde{\Gamma}_\infty(X|Y) \geq \log|Z| + 2\log(1/\epsilon)$*.*

2.3. Public Key Cryptosystems and CCA2 Security

A public-key cryptosystem consists of three probabilistic algorithms, $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, described as follows.

1. $\mathcal{G}(1^\kappa)$ is an initialization algorithm that outputs a public/secret key pair (pk, sk) given security parameter $\kappa \in \mathbb{N}$.
2. $\mathcal{E}(pk, m; r)$ is an encryption algorithm that outputs a ciphertext, c, given pk, a plaintext message, m, and a sampled random number, r, during computation.
3. $\mathcal{D}(sk, c)$ is a decryption algorithm that outputs m such that $m = \mathcal{D}(sk, \mathcal{E}(pk, m; r))$.

We now describe security against adaptive chosen ciphertext attacks or CCA2 security using attack games, following [4].

Security Notion of Adaptive Chosen Ciphertext Attack (CCA2 Security)

This security notion is defined in terms of an attack game between a challenger and an adversary, $\mathcal{A}$, in which both are polynomial-time algorithms. On input κ, the challenger draws $a \in \{0, 1\}$ then generates the public/secret key pair $(pk, sk) \leftarrow \mathcal{G}(1^\kappa)$. It forwards pk to $\mathcal{A}$. $\mathcal{A}$ can perform decryption queries by providing a ciphertext, c, to the challenger, and the challenger returns $\mathcal{D}(sk, c)$. $\mathcal{A}$ performs a challenge query by giving a message pair (m_0, m_1) to the challenger, and the challenger returns the *challenge ciphertext*, $c^* = \mathcal{E}(pk, m_a)$. To prevent trivial wins, any ciphertext input for decryption should not equal c^*. The game ends when $\mathcal{A}$ outputs a guess $a' \in \{0, 1\}$, who wins the game if $a = a'$. The advantage of $\mathcal{A}$ is $|\Pr(a' = a) - 1/2|$. If the advantage of any polynomial-time adversary for this game is negligible in κ, the public-key cryptosystem is CCA2 secure.

2.4. Hash Proof Systems

A hash proof system is an encapsulation system that uses a *projective hash* [5]. The domain of a projective hash consists of two disjoint sets, the *valid* set and the *invalid* set. Each projective hash function is associated with a *projection function* whose role is to provide auxiliary information. Without this auxiliary information, it is computationally difficult to evaluate the projective hash over the valid set in its domain, and its behaviour is close to uniform. In more detail, let Λ_{sk_H} denote a projective hash with ciphertext domain C. Let $V \subseteq C$ denote the set of *valid ciphertexts* and $C \setminus V$ denote the set of *invalid ciphertexts*. Let K denote a set of encapsulated ciphertexts. A hash proof system, H, consists of three polynomial time algorithms (H_g, H_{pub}, H_{priv}) that are as follows.

1. $H_g(1^\kappa)$ is a parameter generation algorithm that generates a secret key sk_H and projective hash $\Lambda_{sk_H} : C \to K$, with the associated projection function μ. It computes public key $pk_H = \mu(sk_H)$, representing auxiliary information.
2. $H_{pub}(pk_H, c_H, \omega)$ is a public evaluation algorithm that, given pk_H, ciphertext $c_H \in C$, and witness, ω, of the fact that $c_H \in V$, outputs $k \in K$.
3. $H_{priv}(sk_H, c)$ is a private evaluation algorithm that, given sk_H, ciphertext $c_H \in C$, outputs $k \in K$, without requiring witness, ω, of the fact that $c_H \in V$.

A key property required of H is the *subset membership hardness* property, whereby V is computationally difficult to distinguish from $C \setminus V$. Formally, let $\mathcal{A}$ be any polynomial-time algorithm. The advantage of $\mathcal{A}$ with respect to the subset membership problem over $\{C, V\}$

is defined as $|\Pr(\mathcal{A}(c_0) = 1|c_0 \leftarrow V) - \Pr(\mathcal{A}(c_1) = 1|c_1 \leftarrow C \setminus V)|$. If this advantage is negligible for any $\mathcal{A}$, the subset membership problem over $\{C, V\}$ is computationally hard.

Definition 2. *Given $\epsilon \geq 0$, a projective hash function, Λ_{sk_H}, with corresponding projection function, μ, is ϵ-universal if, for all pk_H, $c \in C \setminus V$, we have $\Pr(\Lambda_{sk_H}(c) = k|(pk, c)) \leq \epsilon$, where the probability is computed over all sk_H and $pk_H = \mu(sk_H)$.*

The following lemma and definition will be used in the security proofs.

Lemma 3 ([13]). *Let Λ_{sk_H} be an ϵ-universal projective hash function with the associated projection function, μ. For all pk_H and invalid ciphertexts $c_H \in C \setminus V$, we have $\Gamma_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H)) \geq \log(1/\epsilon)$, where sk_H is a randomly drawn secret key, and $pk_H = \mu(sk_H)$.*

Definition 3. *A hash proof system, H, is ϵ-universal if the underlying projective hash function is ϵ-universal and the underlying subset membership problem is computationally hard.*

2.5. Lossy Functions

2.5.1. Lossy Functions

A *lossy* function is a function that loses information from its input. A *collection of lossy functions* (lossy collection) consists of a set of injective functions, along with a set of lossy functions [8]. Let $n(\kappa)$ and $p(\kappa)$ be values of polynomials in $\kappa \in \mathbb{N}$. Given κ, the input length of any function in the collection is $n(\kappa)$, and the size of the domain is $2^{n(\kappa)}$. A lossy function in the collection has an image size of, at most, $2^{\log p(\kappa)} = p(\kappa)$, where $p(\kappa) < 2^{n(\kappa)}$. For convenience, the dependence of n and p on κ is omitted hereafter. The functions in the collection are indexed by the set S. A collection of (n, p) *lossy functions* is given by the polynomial-time algorithms (L_g, L_e):

1. $L_g(1^\kappa, i)$ for $i \in \{0, 1\}$ is a function index sampling algorithm. If $i = 1$, it outputs $s = L_g(1^\kappa, 1) \in S$, where s is the index of an injective function. If $i = 0$, its output is $s = L_g(1^\kappa, 0) \in S$, where s is the index of a lossy function.
2. $L_e(s, k)$ is an evaluation algorithm that, on input $s \in S$ and $k \in \{0, 1\}^n$, outputs an element in $\{0, 1\}^n$. If s refers to an injective function, L_e is injective. If s refers to a lossy function, the image size of L_e is, at most, p.

Definition 4. *Required properties of a collection of lossy functions: (i) the index of an injective or lossy function can be efficiently sampled, (ii) the distribution of $L_g(1^\kappa, 0)$ is computationally hard to distinguish from the distribution of $L_g(1^\kappa, 1)$.*

2.5.2. ABO Lossy Functions

A collection of all-but-one (ABO) lossy functions (ABO collection) consists of functions that are each equipped with a set of *branches* [8]. One branch corresponds to a *lossy branch*, while the rest are *injective branches*. Let $\mathcal{B} = \{B_\kappa\}_{\kappa \in \mathbb{N}}$ denote the collection of branches indexed by κ. Given κ, define $B := B_\kappa$ with lossy branch $b^\circ \in B$. The functions in the ABO collection are indexed by the set S. A (n, p) collection of ABO lossy functions is given by polynomial-time algorithms $(L_{g^{abo}}, L_{e^{abo}})$, which are as follows.

1. $L_{g^{abo}}(1^\kappa, b^\circ)$ is a function index sampling algorithm that, given lossy branch $b^\circ \in B$, outputs function index $s = L_{g^{abo}}(1^\kappa, b^\circ) \in S$.
2. $L_{e^{abo}}(s, b, k)$ is an evaluation algorithm that, on input $s \in S$, branch $b \in B$ and $k \in \{0, 1\}^n$, outputs an element in $\{0, 1\}^n$. If $b = b^\circ$, its image has size at most p. Otherwise, it is injective.

Definition 5. *Required properties of a collection of ABO lossy functions: (i) given κ, a lossy branch $b^\circ \in B$ can be efficiently sampled; (ii) $L_{g^{abo}}$ can efficiently sample s given b°; (iii) it is* *computationally difficult to distinguish the distributions of $L_{g^{abo}}(1^\kappa, b_0^\circ)$ from $L_{g^{abo}}(1^\kappa, b_1^\circ)$ for any $b_0^\circ \neq b_1^\circ$; and (iv) given s, it is computationally difficult to determine b°.*

2.5.3. L^M Lossy Functions

A collection of L^M lossy functions (or L^M collection for short) generalizes the ABO collection. Each function in the L^M collection is equipped with a set of *branches*, but there are several possible *lossy branches*. L^M collections are similar to ABN lossy functions in [26] and ABM lossy functions in [27]. However, they are simpler and can be constructed from a set of ABO collections using Cartesian products. Let $\mathcal{B} = \{B_\kappa\}_{\kappa \in \mathbb{N}}$ denote the collection of branches indexed by κ. Define $B := B_\kappa$, with lossy branch set $B^\circ \subseteq B$ of size $M = |B^\circ|$ and with elements $b^\circ \in B^\circ$. Define q to be an ordered tuple that corresponds to B°, i.e., $q = (b_1^\circ, b_2^\circ, \ldots, b_M^\circ)$ and $b_i^\circ \in B^\circ$ for $i \in [M]$. The functions in the L^M collection are indexed by the set S. A (n, p) collection of L^M lossy functions is given by the polynomial-time algorithms (L_{g^M}, L_{e^M}), which are as follows.

1. $L_{g^M}(1^\kappa, q)$ is a function index sampling algorithm that takes as input q corresponding to B° and outputs $s \in S$.
2. $L_{e^M}(s, b, k)$ is an evaluation algorithm that, on input $s \in S$, branch $b \in B$, and $k \in \{0,1\}^n$, outputs an element in $\{0,1\}^{nM}$. If $b \in B^\circ$, the image size is at most $2^{n(M-1)+\log(p)}$.

Definition 6. *Required properties of a collection of L^M lossy functions: (i) given κ, a lossy branch set $B^\circ \subseteq B$ can be efficiently sampled; (ii) L_{g^M} can efficiently sample s, given q corresponding to B°; (iii) it is computationally difficult to distinguish the distributions of $L_{g^M}(1^\kappa, q_0)$ from $L_{g^{abo}}(1^\kappa, q_1)$ for $q_0 \neq q_1$; and (iv) given s, it is computationally difficult to generate an element of B°.*

2.6. Pseudorandom Functions

A *pseudorandom function* $P : R \times M \to Y$, where R is a key space and M is an input data block, is a deterministic algorithm that behaves like a truly random function [2].

Security Notion of a Pseudorandom Function

The *security of a pseudorandom function*, P, is defined in terms of an attack game between a challenger and an adversary. Given κ, at the start of the game the challenger draws $a \in \{0,1\}$ and selects a random function, f, from M to Y. The adversary submits a sequence of queries to the challenger, where each query consists of an element $m \in M$. If $a = 0$, the challenger draws $r \leftarrow R$ and submits $P(r, m)$ to the adversary. If $a = 1$, the challenger submits $f(m)$ to the adversary. The game ends once the adversary submits a guess $a' \in \{0,1\}$ who wins if $a' = b$. The advantage of the adversary in this game is defined as $|\Pr(a' = a) - 1/2|$. The pseudorandom function, P, is a *secure PRF* if the advantage of any polynomial time adversary in this game is negligible in κ.

2.7. Strongly Unforgeable One-Time Signatures

A strongly unforgeable one-time signature scheme has the *strong one-time unforgeability* property [8] and is given by the algorithms below.

Key Generation. $F_g(1^\kappa)$. On input κ, F_g outputs the verification/signing key pair (vk_σ, sk_σ).

Signing. $F_s(sk_\sigma, x)$: given sk_σ and a plaintext x, outputs a signature σ.

Verification. $F_v(vk_\sigma, x, \sigma)$: given vk_σ, x, and σ, it outputs 0 if $\sigma \neq F_s(sk_\sigma, x)$ and 1 otherwise.

Security Notion of a Strongly Unforgeable One-Time Signature Scheme

The security of a strongly unforgeable one-time signature scheme is defined in terms of an attack game consisting of a challenger and an adversary, $\mathcal{A}$. Given κ, at the start of the game, the challenger generates (vk_σ, sk_σ) and gives vk_σ to adversary $\mathcal{A}$. $\mathcal{A}$ queries a plaintext message, x, to the challenger and the challenger returns $\sigma \leftarrow F_s(sk_\sigma, x)$. $\mathcal{A}$ wins the game if it outputs a distinct message signature pair (x', σ') such that $F_v(vk_\sigma, x', \sigma') = 1$. A signature scheme is *strongly unforgeable one-time secure* if no probabilistic polynomial time adversary can win the attack game described with non-negligible probability.

3. Security Notions

For illustration, we first describe randomness reset attacks and secret-key leakage attacks in the context of the ElGamal public-key cryptosystem. Recall that given a group, $\mathbb{G}$, of prime order p, with generator g, the ElGamal cryptosystem draws a secret key, $sk = z \leftarrow \mathbb{Z}_p$, and defines the public key as $pk = h = g^z$. A message, $m \in \mathbb{Z}_p$, is encrypted as $c = (g^r, h^r g^m)$ for a randomly chosen $r \leftarrow \mathbb{Z}_p$.

Randomness Reset Attack Example. In a randomness reset attack [16], an adversary can force the cryptosystem to re-use a previous random number. In terms of the ElGamal cryptosystem above, suppose that Alice draws a secret key, $sk = z$, and gives the public key, $pk = h = g^z$, to Bob. Bob now encrypts a message, m, by drawing a random number, r_0, and sends the ciphertext $c_0 = (g^{r_0}, h^{r_0} g^m)$ to Alice. In a normal setting, without randomness reset attacks, suppose that Bob wants to send another message, $m_a \in \{m_0, m_1\}$, for $m \neq m_0 \neq m_1$, to Alice. To do this in the normal setting Bob draws a fresh random number, r_1, and sends the new ciphertext $c_1' = (g^{r_1}, h^{r_1} g^{m_a})$ to Alice. Given that c_1' and c_0 involve different random numbers, they are computationally indistinguishable for any efficient adversary. In a setting with randomness reset attacks, however, an adversary forces Bob to re-use r_0 in encrypting m, i.e., $c_1 = (g^{r_0}, h^{r_0} g^m)$. This arbitrarily breaks the security of the ElGamal cryptosystem. To see this, suppose that some adversary obtained c_0 and c_1, and let the adversary know, as well, m, m_0, and m_1. The adversary can compute $c_0/c_1 = h^{r_0} g^m / h^{r_0} g^{m_1} = g^{m-m_a}$. It follows that if $m_a = m_0$, we have $c_0/c_1 = g^{m-m_0}$, but if $m_a = m_1$, we have $c_0/c_1 = g^{m-m_1}$. The adversary can compute g^{m-m_0} and g^{m-m_1} on its own, given that it knows g, m_0, and m_1. It follows that, with randomness reset attacks, the ElGamal cryptosystem is not even semantically secure. Relating this scenario to a CCA2 attack game between a challenger and an adversary, a randomness reset attack allows the adversary to perform encryption queries apart from challenge queries. In the example above, the adversary can ask the challenger to encrypt m (encryption query) followed by the encryption of $m_a \in \{m_0, m_1\}$. The adversary can also force the challenger to re-use a previous random number in both encryption and challenge queries. For more details on the power of randomness attacks, in [16], it has been shown that randomness reset attacks may break arbitrary CCA2 cryptosystems that are more complicated than the ElGamal crypstosystem if no additional primitives are applied to secure the randomness.

Constant secret-key leakage attack example. In a secret-key leakage attack, the adversary can obtain bits of the secret key. The secret-key leakage attack in [13] considers the leakage of a constant factor of bits of the secret key, where it should be a percentage of the secret key's length. The percentage for any public-key cryptosystem should obviously not equal one. Otherwise, the entire secret key is leaked. In terms of the ElGamal cryptosystem, this implies that $sk = z$ is leaked to the adversary, which arbitrarily breaks the cryptosystem. However, in some cryptosystems, such as the Cramer and Shoup CCA2 cryptosystem [3], the allowable amount leakage is even lower—by a factor of $1/2 - o(1)$. This is because the security of the Cramer and Shoup cryptosystem involves jointly using two secret keys, and, if either is leaked, the entire cryptosystem is insecure. Relating this scenario to a CCA2 attack game between a challenger and an adversary, a constant secret-key leakage attack involves leaking some constant number of bits to the adversary through a leakage query.

We now present the attack game corresponding to the security notion of a public-key cryptosystem that is secure against (i) adapative chosen ciphertext attacks, (ii) randomness reset attacks, and (iii) constant secret-key leakage attacks. Table 1 presents the attack game.

The attack game is initialized by the challenger through **Initialize**, where he generates the public/secret key pair (pk, sk), and forwards pk to the adversary. The adversary has access to (i) decryption queries **Dec**, (ii) secret-key leakage queries **Leak**, (iii) challenge query **Challenge**, and (iv) encryption queries **Enc**, which are all described in Table 1. In Table 1, the adversary has access to a set of indices that are mapped to prior random numbers generated during encryption or challenge queries. In any subsequent encryption query or challenge query, the adversary can use any of these indices at will, representing a randomness reset attack. The adversary can ask the challenger in an encryption query to use any public key—this follows [16]. In a leakage query, **Leak**, the adversary may request up to $\lambda(\kappa)$ bits of the secret key sk. The game ends once the adversary outputs a bit, a', through **Finalize**, who wins if $a' = a$. The advantage of the adversary, in this case, is defined as $|\Pr(a' = a) - 1/2|$, and a cryptosystem is secure with respect to the attack game of Table 1 if no polynomial-time adversary has non-negligible advantage.

Table 1. Attack game with adaptive chosen ciphertext attack, randomness reset, and constant secret-key leakage, where $r(\kappa)$ is the value of a polynomial in κ that represents the combined length of the random numbers used during encryption. The adversary in this game is allowed to perform multiple encryption and challenge queries under different randomness indices. The notation `coins`$[j]$ and `ciphers`$[j]$, for $j \geq 1$, refers to the jth element of `coins` and `ciphers`, respectively.

```
proc.Initialize($\kappa$)
    $a \leftarrow \{0,1\}$
    $(pk, sk) \leftarrow \mathcal{G}(1^\kappa)$
    coins $\leftarrow \emptyset$
    ciphers $\leftarrow \emptyset$
    return $pk$

proc.Challenge($j, m_0, m_1$)
    if $|m_0| \neq |m_1|$ return $\perp$
    if coins$[j] = \perp$ then coins $\leftarrow \{0,1\}^{r(\kappa)}$
    $r_j \leftarrow$ coins$[j]$
    $c \leftarrow \mathcal{E}(pk, m, r_j)$
    ciphers $\leftarrow$ ciphers $\cup\, c$
    return $c$

proc.Dec($c$)
    if $c \in$ ciphers then return $\perp$
    else return $\mathcal{D}(sk, c)$

proc.Enc($pk', j, m$)
    if coins$[j] = \perp$ then coins $\leftarrow \{0,1\}^{r(\kappa)}$
    $r_j \leftarrow$ coins$[j]$
    $c \leftarrow \mathcal{E}(pk', m, r_j)$  return $c$

proc.Leak($\lambda(\kappa)$)
    return random $\lambda(\kappa)$ bits of $sk$

proc.Finalize($a'$)
    return $(a = a')$
```

3.1. Attack Game with Random Oracles

In certain situations, the random oracle heuristic [4] is convenient for simplifying security proofs. A *random oracle* Φ captures a truly random function and can be incorporated in cryptosystems. Suppose that a random oracle $\Phi : \mathcal{X} \to \mathcal{Y}$ is part of a cryptosystem. The corresponding attack game incorporates additional *random oracle queries*, whereby, on input $x \in \mathcal{X}$ from the adversary, the challenger returns $y \in \mathcal{Y}$ such that $\Phi(x) = y$.

3.2. Adversary Constraints

As stated in [11,16], if the adversary can perform a randomness reset attack and has no constraints on encryption/challenge queries, it can trivially win. To prevent this, the adversary is assumed to be *equality-pattern-respecting*. We provide the definition of an equality-pattern-respecting adversary below, along with the corresponding notion of a non-reversing adversary that will be used in cryptosystem 1.

Definition 7. *Let $\mathcal{A}$ be an adversary in Table 1's attack game. Let I represent the set of randomness indices mapped to random numbers generated by the challenger during $\mathcal{A}$'s challenge and encryption queries. Let $\mathcal{A}$ perform $\mathcal{Q}_e$ encryption queries. Let $\mathcal{A}$ perform $\mathcal{Q}_{c,i}$ challenge queries using index $i \in I$. Let MI represent the set of input messages, m, in the encryption queries done by $\mathcal{A}$ using the public* *key, pk^*, and randomness index, $i \in I$, i.e., **Enc**(pk^*, i, m). Let $(m_0^{i,1}, m_1^{i,1}) \ldots (m_0^{i,\mathcal{Q}_{c,i}}, m_1^{i,\mathcal{Q}_{c,i}})$ represent the set of message pairs given in challenge queries using randomness index $i \in I$. For Table 1's attack-game, we say that $\mathcal{A}$ is equality-pattern-respecting: (1) if we have, for all $i \in I$ and for all $j \neq k \in [\mathcal{Q}_{c,i}]$, $m_0^{i,j} = m_0^{i,k}$ if $m_1^{i,j} = m_1^{i,k}$, and (2) if, for all $i \in [\mathcal{Q}_e]$ and $j \in [\mathcal{Q}_{c,i}]$, we have $m_0^{i,j} \notin MI$ and $m_1^{i,j} \notin MI$.*

Definition 8. *Let $\mathcal{A}$ be an adversary in Table 1 attack game that performs $\mathcal{Q}_e$ encryption queries and $\mathcal{Q}_d$ decryption queries. Let $\{c_i\}_{i\in[\mathcal{Q}_e]}$ denote the set of ciphertext outputs c_i received by $\mathcal{A}$ from the challenger in its encryption queries. Let $\{c_j\}_{j\in[\mathcal{Q}_d]}$ denote the set of ciphertexts c_j submitted by $\mathcal{A}$ for its decryption queries. We say that $\mathcal{A}$ is non-reversing if, for any $c_i' \in \{c_i\}_{i\in[\mathcal{Q}_e]}$, we have $c_i' \notin \{c_j\}_{j\in[\mathcal{Q}_d]}$ at any point in the game.*

4. Comparison of Cryptosystems/Lossy Functions

Given the security notion presented in the previous section, for context, we present a list of various cryptosystems in Table 2 that deal with the related notions of randomness attacks and secret-key leakage attacks. From Table 2, the types of randomness attacks considered in the literature involve linear and polynomial functions of random numbers involved in the encryption process. The same holds for secret-key leakage attacks, where leakage may consist of affine or polynomial functions of bits of the secret key, along with bounded degrees of secret-key tampering. Our proposed cryptosystems are listed in the last two lines of Table 2 and consider joint attacks involving randomness reset and constant secret-key leakage. Similar to the other constructions in Table 2, we propose both a random oracle and standard model of our cryptosystems.

In Table 3, we list several lossy function constructions from the literature. From Table 3, the first lossy function collection is from [8] which uses the DDH assumption. The next lossy function constructions present improvements in terms of the number of lossy function branches or tags that can be sampled efficiently while retaining their amounts of lossiness. In particular, the construction of [27] provides an efficient lossy function that can sample a superpolynomially large number of lossy tags. The construction of [27], however, is quite complicated, since it involves Waters signatures along with chameleon hashing. For the purposes of our cryptosystems, our proposed lossy function collection (the last line of Table 3) is able to sample up to M lossy branches per function index but is compromised by a rather high amount of lossiness, i.e., $2^{n(M-1)+\log(p)}$. Yet, despite this amount of lossiness, the security proof is still held intact, given that the factor $2^{n(M-1)+\log(p)}/2^n$ is still superpolynomial in κ. In addition, our proposed lossy function collection is simpler to construct—involving only the DDH assumption, along with the Cartesian product operation. In terms of size complexity, we show, in the FIS and LBS columns, the size of the function index and the lossy branch index, respectively, where size is measured in terms of matrix representations. For instance, the function index size of the lossy functions in [8] is n^2, which means that the index is a square matrix consisting of n rows and n columns. Our proposed L^M collection has a larger function index size, of n^2M, and a larger branch size, of nM. This is because it relies on the M Cartesian product operation. A concrete instantation of an L^M collection is presented in Section 6.

Table 2. List of CCA2 secure PKE schemes that incorporate either randomness attack or secret-key leakage attack along with their primitives. Scheme models are classified according to random oracle or standard, where standard refers to schemes that do not use random oracles. Our proposed schemes are in the last two lines of the table and incorporate joint randomness reset attack and constant-bit secret-key leakage attack.

Reference	Randomness Attack	Secret-Key Leakage Attack	Model	Primitives/Assumptions
Canetti and Goldwasser [28]	–	–	random oracle	random oracle assumption
Cramer and Shoup [3]	–	–	standard	hash proof system/DDH
Yilek [16]	randomness reset	–	standard	pseudorandom function
Peikert and Waters [8]	–	–	standard	lossy functions/DDH/DCR
Wee [29]	–	linear leakage	standard	BDDH/LWE
Qin and Liu [13]	–	constant leakage	standard	hash proof system + lossy filter/DDH
Bellare et al. [10]	chosen distribution attack	–	random oracle	random oracle assumption
Bellare et al. [10]	chosen distribution attack	–	standard	lossy functions
Paterson et al. [11]	linear/polynomial	–	random oracle	random oracle assumption
Paterson et al. [11]	linear functions	–	standard	pseudorandom function
Paterson et al. [11]	polynomial functions	–	standard	CIS hash functions
Paterson et al. [30]	vector of functions	–	standard	Goldreich-Levin extractor
Boneh et al. [31]	–	affine leakage	random oracle	random oracle assumption
Boneh et al. [31]	–	polynomial leakage	standard	EDBDH
Faonio and Venturi [12]	–	leakage + bounded tampering	standard	hash proof system + lossy filter/RSI
ours	randomness reset	constant leakage	random oracle	random oracle assumption
ours	randomness reset	constant leakage	standard	hash proof system + L^M lossy functions

Table 3. List of lossy function constructions found in the literature. Our proposed L^M construction is shown in the last line of the table, where up to M lossy branches are given for each function index. While the lossiness of our construction is higher than the other schemes, it is simpler to construct and uses only the DDH assumption. ABO: all-but-one lossy functions; ABN: all-but-N lossy functions; ABM: all-but-many lossy functions; LF: lossy function; LB: lossy branch; LT: lossy tag; DS: domain size, LS: lossiness size; FIS: function index size (in terms of matrix representation, where n is the number of rows/columns); LBS: lossy branch index size; DDH: decisional Diffie–Helman; DCR: decisional composite residuousity; QR: quadratic residuousity; CH: chameleon hash.

Reference	Primitive	No. of LF/LB/LT	DS	LS	FIS	LBS	Assumptions
Peikert and Waters [8]	lossy functions	several	2^n	$2^{\log p}$	n^2	n	DDH/DCR (lattice)
Peikert and Waters [8]	ABO lossy functions	1 LB/function	2^n	$2^{\log p}$	n^2	n	DDH
Hemenway et al. [26]	ABN lossy functions	N LB/function	2^n	$2^{\log p}$	-	-	DDH/DCR/QR
Hofheinz [27]	ABM lossy functions	superpolynomial LT	2^n	$2^{\log p}$	-	-	Waters sig./CH
Qin and Liu [13]	one-time lossy filter	superpolynomial LT	2^n	$2^{\log p}$	n^2	n	DDH/CH
ours	L^M lossy functions	M LB/function	2^{nM}	$2^{n(M-1)+\log(p)}$	$n^2 M$	nM	DDH

5. Proposed Cryptosystems

5.1. Cryptosystem 1

In this section, we present our first cryptosystem that is secure against the attack game of Table 1. It uses several primitives, such as hash proof systems, randomness extractors, and pseudorandom functions, along with a random oracle Φ. Using a random oracle assumption simplifies the security proof and usually serves as the starting point in cryptosystem design, as done in [10]. However, as mentioned, the random oracle assumption is rather strong. In addition, this cryptosystem is limited to facing non-reversing adversaries who cannot submit prior-encrypted ciphertexts for decryption. We overcome the non-reversing limitation in the next cryptosystem—which also does away with the random oracle requirement.

5.2. Cryptosystem 1 Requirements

Let $\mathcal{Q}_e$, $\mathcal{Q}_d$, and $\mathcal{Q}_c$ denote the bounds in the number of encryption, decryption and challenge queries respectively. The requirements of cryptosystem 1 are as follows.

1. An ϵ_1-universal hash proof system H for some $\epsilon_1 > 0$ given by (H_g, H_{pub}, H_{priv}). The ciphertext domain of H is C, with $V \subseteq C$ as its valid subset. SK_H and PK_H denote the secret-key space and public-key space of H, with elements $sk_H \in SK_H$ and $pk_H \in PK_H$. The space W of the witnesses of H is set to R. The encapsulated key space of H is K, with elements $k \in K$. H_{pub} is set as $H_{pub} : PK_H \times C \times R \to K$, and $H_{priv} : SK_H \times C \to K$. The projective function is $\Lambda_{sk_H} : C \to K$, with associated projection function $\mu : SK_H \to PK_H$.
2. A (n, p) ABO collection given by $(L_{g^{abo}}, L_{e^{abo}})$. The set of branches is B with elements $b \in B$, and lossy branch $b^\circ \in B$. Functions are indexed by S, and $L_{e^{abo}} : S \times B \times K \to \{0,1\}^n$
3. E is a $((\nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l))/\mathcal{Q}_c, \epsilon_2)$ average-case strong-randomness extractor
4. A secure pseudorandom function $P : R \times M \to R$
5. A strongly unforgeable one-time signature scheme given by (F_g, F_s, F_v). SK_σ and VK_σ denote the spaces of signature and verification keys, with elements $sk_\sigma \in SK_\sigma$ and $vk_\sigma \in VK_\sigma$, and where the domain of F_s is $C \times \{0,1\}^l$, and the domain of VK_σ is equal to B.
6. Elements of $K, R, C, SK_H, PK_H, S, B$ have length n. Elements of M have length l.
7. The values of ν, l, and λ are such that $\lambda \leq \nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l) - \alpha(\log \kappa)$ for some $\alpha \geq 0$.
8. n and p satisfy $p < 2^{\sqrt{n}}$.
9. The polynomial in κ whose value is n is superpolynomial with respect to $\mathcal{Q}_e$, $\mathcal{Q}_d$, and $\mathcal{Q}_c$.

5.3. Cryptosystem 1

Key Generation. $\mathcal{G}(1^\kappa)$ first runs $H_g(1^\kappa)$ to obtain $sk_H \in SK_H$ and Λ_{sk_H} with μ. It computes $pk_H = \mu(sk_H) \in PK_H$. The output is a public/secret key pair (pk, sk), where $pk = pk_H$ and $sk = sk_H$.

Encryption. $\mathcal{E}(pk, m)$: on input $pk = pk_H$ and message $m \in M$, let $\Phi : K \to \{0,1\}^n$ denote a random oracle. It performs the following:

1. It samples $r_1' \leftarrow R$ then computes $r_1 = P(r_1', m)$. It sets $\omega = r_1$. Using ω, it chooses $c_H \in V$
2. It samples $r_2' \leftarrow R$, then computes $r_2'' = r_2' \oplus pk_H$, followed by $r_2 = P(r_2'', m) \in R$.
3. Using r_2, it computes $k = H_{pub}(pk_H, c_H, \omega)$ and $\Psi = E(k, r_2) \oplus m$
4. It samples $(vk_\sigma, sk_\sigma) \leftarrow F_g(1^\kappa)$ and computes $\sigma = F_s(sk_\sigma, (c_H, r_2, \Psi))$
5. It computes $\Pi = \Phi(k)$
6. It returns ciphertext $c = (\sigma, c_H, r_2, vk_\sigma, \Psi, \Pi)$

Decryption. $\mathcal{D}(sk, c)$: on input $sk = sk_H$ and $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$, performs the following:

1. The algorithm checks if $F_v(vk_\sigma, (c_H, r_2, \Psi), \sigma) = 1$. If not, it outputs $\perp$.
2. It computes $k' = H_{priv}(sk_H, c_h)$
3. It computes $\Pi' = \Phi(k')$
4. It checks if $\Pi = \Pi'$. If not, it outputs $\perp$
5. It returns the plaintext message $m = \Psi \oplus E(k', r)$.

For cryptosystem 1, we note that the role of P is to generate fresh random numbers r_1 and r_2 using the joint entropy of the message m and old random numbers r'_1 and r'_2 (due to reset attacks). It follows that r_1 and r_2 serve as the actual randomness input to H_{pub} and E, respectively. To show correctness of the cryptosystem, given m and pk, $\mathcal{E}$ first computes r_1 and r_2 followed by $k \leftarrow H_{pub}(pk_H, c_H, r_1 = \omega)$, $\Psi = E(k, r_2) \oplus m$, and $\Pi = \Phi(k)$. Let $c = (\sigma, C_H, r_2, vk_\sigma, \Psi, \Pi)$ be the corresponding ciphertext where vk_σ is jointly sampled with sk_σ under F_g, and σ is derived using F_s as shown above. If this c were given to $\mathcal{D}$, it follows that $F_v(vk_\sigma, (c_H, r_2, \Psi), \sigma) = 1$ given that $\sigma = F_s(sk_\sigma, (c_H, r_2, \Psi))$ and the signature scheme (F_g, F_s, F_v) satisfies the correctness property. Having passed this first check, $\mathcal{D}$ computes $k' = H_{priv}(sk_H, c_H)$ and we have $k' = k$, given that H_{priv} uses the same c_H from encryption and sk_H is paired with pk_H using H_g. Given that (H_g, H_{pub}, H_{priv}) satisfies the correctness property, it follows that $k' = k$ as claimed. Given that $k' = k$, we have $\Pi' = \Pi$ under Φ given that Φ is a function, thereby passing the second check. Finally, with $k' = k$, we have $m = \Psi \oplus E(k', r_2) = \Psi \oplus E(k, r_2)$, and the original message m is recovered.

5.4. Security Results for Cryptosystem 1

Theorem 1. *Let $\Phi : K \to \{0,1\}^n$ be a random oracle and let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ denote cryptosystem 1. For any non-reversing, equality-respecting, polynomial-time adversary that makes (a) at most $\mathcal{Q}_c$ challenge queries under multiple randomness indices, (b) at most $\mathcal{Q}_e$ encryption queries under multiple randomness indices, and (c) at most $\mathcal{Q}_d$ decryption queries, and following the attack game of Table 1, then cryptosystem 1 is secure against (i) adaptive chosen ciphertext attack, (ii) λ bits of secret-key leakage, and (iii) randomness reset attacks.*

Game 0. This game implements the original cryptosystem with no modifications. The following attack game incorporates random oracle Φ queries in the attack game of Table 1. Φ is modelled using an associative array, `map`, which follows the faithful/forgetful gnome method [2]. The notation `map`$[j]$ for $j \geq 1$ refers to the jth element of `map`.

proc.Initialize(κ)

```
1. $(pk, sk) \leftarrow \mathcal{G}(1^\kappa)$
2. initialize empty associative array map $: K \to \{0,1\}^n$
3. initialize empty arrays coins and ciphers
4. send $pk$ to adversary
```

proc.Enc(pk', j, m)

```
1. if coins$[j] = \perp$ then randomly sample $(r'_1, r'_2, vk_\sigma, sk_\sigma)$ and set coins$[j] = (r'_1, r'_2, vk_\sigma, sk_\sigma)$
2. compute $c \leftarrow \mathcal{E}(pk', m)$ with line 5 modified as:
   - if map$[k] = \perp$ then $\zeta \leftarrow \{0,1\}^n$ and set map$[k] = \zeta$. Set $\Pi =$ map$[k]$
3. return $c$
```

proc.Challenge(j, m_0, m_1)

```
1. if $|m_0| \neq |m_1|$ return $\perp$
2. if coins$[j] = \perp$ then randomly sample $(r'_1, r'_2, vk_\sigma, sk_\sigma)$ and set coins$[j] = (r'_1, r'_2, vk_\sigma, sk_\sigma)$
3. compute $c^* \leftarrow \mathcal{E}(pk, m)$ with the line 5 modified as:
   - if map$[k] = \perp$ then $\zeta \leftarrow \{0,1\}^n$ and set map$[k] = \zeta$. Set $\Pi =$ map$[k]$
```

```
    4. ciphers ← ciphers ∪ $c^*$
    5. return $c^*$
proc.Dec($c$)
    1. if $c \in$ ciphers, return $\perp$
    2. compute $m = \mathcal{D}(sk, c)$ with line 3 modified as:
        - if map$(k') = \perp$, then $\zeta \leftarrow \{0,1\}^n$ and set map$[k'] = \zeta$. Set $\Pi' =$ map$[k']$
    3. return $m$
proc.Leak($\lambda(\kappa)$)
    1. return $\lambda(\kappa)$ bits of $sk$
proc.Oracle($k$)
    1. if map$(k) = \perp$, then $\zeta \leftarrow \{0,1\}^n$ and set map$[k] = \zeta$
    2. return map$[k]$
```

The adversary can perform any number of encryption queries under different randomness indices in `coins`. Prior to making any challenge query, the adversary can request λ bits of the secret key. At any point in the game, the adversary can perform a decryption query under the non-reversing condition.

Game 1. This game is similar to Game 0, except that (r_1, r_2) are drawn randomly, instead of being computed using the pseudorandom function P in $\mathcal{E}$.

Game 2. This game is similar to Game 1, except that once the adversary submits a ciphertext $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$ for decryption such that $vk_\sigma \in c'$ for some $c' \in$ `ciphers`, the challenger returns $\perp$.

Game 3. This game is similar to Game 2, except that Π is sampled randomly in $\mathcal{E}$ instead of being queried using Φ.

Game 4. This game is similar to Game 3, except that the challenger computes $k = H_{priv}(sk_H, c_H)$ instead of H_{pub} in $\mathcal{E}$.

Game 5. This game is similar to Game 4, except that c_H is sampled from $C \setminus V$ instead of V in $\mathcal{E}$.

Game 6. This game is similar to Game 5, except that if the adversary submits a ciphertext $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$ for decryption such that $c_H \in C \setminus V$, the challenger returns $\perp$.

Game 7. This game is similar to Game 6, except that the challenger draws Ψ uniformly at random from $\{0,1\}^l$ instead of computing $\Psi = E(k, r) \oplus m$ in $\mathcal{E}$

Proposition 1. *Game 0 and Game 1 are computationally indistinguishable, given the security of the pseudorandom function P.*

Proof. To prove this claim, we define hybrid experiments, H_0, H_1, H_2, where H_0 emulates the challenger in Game 0, H_1 samples r_1 randomly, and H_2 samples both r_1 and r_2 randomly. We have to show show that for any $i \in \{0,1\}$, experiments H_i and H_{i+1} are computationally indistinguishable.

Suppose that some adversary can efficiently distinguish between H_i and H_{i+1} for $i \in \{0,1\}$. Using this adversary, we construct a simulator that breaks the security of P. The simulator has access to an oracle that, on input m, returns a number, y, where we either have $y = P(r, m)$ for some random number r, or y is sampled using a truly random function. For $i \in \{0,1,2\}$, the simulator emulates the challenger in Game 0 perfectly in the initialization phase and draws $a \in \{0,1\}$. If $i = 0$, the simulator does not modify anything from the challenger in Game 0. For H_0, H_1, H_2, the simulator knows sk and it can answer decryption and secret-key leakage queries. In encryption queries with input

(pk', m), if $i = 1$, the simulator sends m to its oracle and receives y, where we either have $y = P(r'_1, m)$ or y is sampled randomly. It sets $r_1 = y$ and proceeds with the rest, as before. If $i = 2$, it samples r_1 randomly and sends m to its oracle. It receives y where we either have $y = P(r'_2 \oplus pk'_H, m)$ or y is sample randomly. It sets $r_2 = y$.

In challenge queries, the input is a message pair, (m_1, m_2). If $i = 1$, the simulator sends (m_1, m_2) to its oracle and receives y, where either have $y = P(r'_1, m_a)$ or y is sampled randomly. If $i = 2$, the simulator samples r_1 randomly and sends (m_1, m_2) to its oracle. It receives y, where we either have $y = P(r'_2 \oplus pk_H, m_a)$ or y is sample randomly. When the adversary submits a guess $a' \in \{0, 1\}$, the simulator outputs 1 if $a = a'$ and 0 otherwise. The probability that the oracle computes y using P and the simulator outputs 1 is equal to the advantage of the simulator in distinguishing between outputs of P and randomly sampled numbers. In turn, the simulator's advantage is equal to the probability that the adversary outputs a' such that $a' = a$ less $1/2$. However, due to the pseudo-randomness of P, no efficient adversary is able to output a' such that $a = a'$ with non-negligible probability. Hence, the simulator's advantage is likewise negligible. By construction, experiment H_0 perfectly simulates Game 0, while experiment H_2 perfectly simulates Game 1. The proposition thus follows. □

Proposition 2. *Game 1 and Game 2 are computationally indistinguishable, given the strong one-time existential unforgeability of the signature scheme.*

Proof. Games 1 and 2 behave the same, except when the adversary submits a ciphertext query $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$, such that $F_v(vk_\sigma, (c_H, \Psi), \sigma) = 1$ and $vk_\sigma \in c'$ for some $c' \in$ `ciphers` but $c \neq c'$. We construct a simulator that attacks the security of the signature scheme as follows. The simulator emulates Game 2 against an adversary. The simulator has access to an oracle that provides it with a verification key vk_σ upon request. Since the simulator does not know sk_σ, it can query the oracle for a signature, where on input (c_H, r, Ψ), the oracle returns σ computed using the hidden sk_σ associated with the latest vk_σ provided to the simulator. The simulator emulates the challenger in every aspect of the initialization phase of Game 1, but requests for a preliminary vk^0_σ. Since the simulator knows sk, it can answer any decryption query and secret leakage query. Moreover, prior to any challenge or encryption query, for any decryption query with input $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$, the simulator checks if $vk_\sigma = vk^0_\sigma$ and if $F_v(vk_\sigma, (c_H, r, \Psi), \sigma) = 1$. If this condition is met, the simulator outputs $((c_H, r, \Psi), \sigma)$ as a forgery and terminates the simulation. If this event does not occur and the simulator encounters the first challenge or encryption query, the simulator uses vk^0_σ as the verification key and queries the oracle for σ. Subsequent challenge or encryption queries require the simulator to ask the oracle for a fresh vk_σ as well as for σ. Once it receives another decryption query with input $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$, it checks if $vk_\sigma = vk'_\sigma$ with $vk'_\sigma \in c'$ for some $c' \in$ `ciphers`. If this is true for some c', it checks if $\sigma \notin c'$ and $(c_H, r, \Psi) \notin c'$ and $F_v(vk_\sigma, (c_H, r, \Psi), \sigma) = 1$. If this is true as well, the simulator outputs $((c_H, r, \Psi), \sigma)$ and terminates the simulation. By construction, the simulator emulates Game 2 perfectly against the adversary, and the advantage of the simulator in coming up with a forgery is equal to the probability that the adversary queries a ciphertext that meets the conditions mentioned. However, because the signature scheme is strongly one-time unforgeable, no efficient adversary can query such a ciphertext with non-negligible probability. It follows that the probability whereby the simulator outputs a valid forgery is negligible. Thus, with large probability, the ciphertexts that involve $vk_\sigma = vk'_\sigma$ with $vk'_\sigma \in c'$ for some $c' \in$ `ciphers`, and which meet the check requirements for decryption are not forgeries and are equal to some prior challenge ciphertext. However, no challenge ciphertexts can be valid for decryption as of Game 0. □

Proposition 3. *Game 2 and Game 3 are computationally indistinguishable, given that Φ is a random oracle.*

Proof. We note that due to the non-reversing nature of the adversary, no ciphertext outputs of prior encryption or challenge queries can be submitted for decryption. It follows that in Game 3, the adversary cannot use decryption to check if Π is randomly drawn or not. Thus, the only event where Games 2 and 3 differ is when the adversary performs an oracle query in Game 3 on some input k, where k is computed under line 3 of $\mathcal{E}$ in some prior encryption or challenge query, and is associated with some randomly drawn Π' such that $\Phi(k) = \mathtt{map}[k] \neq \Pi'$ in Game 2. The probability that this event occurs is $(\mathcal{Q}_c + \mathcal{Q}_e)/|K|$. Since $|K| = 2^n$, this probability is negligible. □

Proposition 4. *Game 3 and Game 4 are perfectly equivalent.*

Proof. The claim readily follows, since the change from computing k using H_{pub} to computing k using H_{priv} and is merely conceptual. □

Proposition 5. *Game 4 and Game 5 are computationally indistinguishable, given the hardness of the underlying subset membership problem in the hash proof system.*

Proof. To prove this claim, we define two experiments, H_0 and H_1. H_0 and H_1 behave the same, except that H_0 samples c_H from V, while H_1 samples c_H from $C \setminus V$. Suppose that some adversary can distinguish H_0 and H_1 with non-negligible probability. Using this adversary, we construct a simulator that can break the hardness of the underlying subset membership problem of hash proof system H. The simulator has access to an oracle that, on input pk_H, provides it with c_H, where we either have $c_H \in V$, or $c_H \in C \setminus V$. At initialization, the simulator emulates the challenger of Game 4 in all aspects. In encryption queries with input (pk', m) the simulator forwards $pk'_H \in pk$ to its oracle and receives c_H. In challenge queries, the simulator forwards pk_H to its oracle and receives c^*_H. In both challenge and encryption queries, it does not compute for c_H using H_{pub}. Since the simulator knows sk, the simulator can answer any decryption or secret-key leakage queries. Once the adversary submits a guess $a' \in \{0,1\}$, the simulator outputs 1 if $a' = a$. It follows that the advantage of the simulator in distinguishing $c_H \in V$ from $c_H \in C \setminus V$ is equal to the probability that the adversary outputs a' such that $a' = a$. However, given the underlying subset membership problem of hash proof system H is hard, the probability that $a' = a$ is negligible. It follows that the simulator's advantage is likewise negligible. Experiment H_0 emulates Game 4, while H_1 emulates Game 5, thereby proving the proposition. □

Proposition 6. *Game 5 and Game 6 are computationally indistinguishable, given the ϵ_1-universal hash proof system.*

Proof. Define Z to be the event that some ciphertext $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$ with $c_H \in C \setminus V$, is accepted for a decryption query in Game 5, but is rejected in Game 6. Games 5 and 6 proceed identically until Z occurs. We claim the following.

$$\Pr(Z) \leq \frac{\mathcal{Q}_d 2^{\lambda + (\mathcal{Q}_c + \mathcal{Q}_e)(l)}}{2^\nu - \mathcal{Q}_d} \tag{1}$$

Let $c = (\sigma, c_H, r, vk_\sigma, \Psi, \Pi)$ be a decryption query input that triggers Z. In encryption, c_H serves as input to Λ_{sk_H}. From the adversary's point of view, Λ_{sk_H} is dependent on the set of challenge ciphertexts $\{c^*_j\}_{j \in [\mathcal{Q}_c]}$ where $c^*_j = (\sigma^*_j, c^*_{H,j}, r^*_j, vk^*_{\sigma,j}, \Psi^*_j, \Pi^*_j)$ and on the set of encryption query outputs $\{c_i\}_{i \in [\mathcal{Q}_e]}$, where $c_i = (\sigma_i, c_{H,i}, r_i, vk_{\sigma,i}, \Psi_i, \Pi_i)$. For $j \in [\mathcal{Q}_c]$, σ^*_j do not provide additional information on Λ_{sk_H} since it is a function of $(c^*_j, \Psi^*_j, vk^*_{\sigma,j})$. The same applies to σ_i for $i \in [\mathcal{Q}_e]$. The sets of verification keys $\{vk^*_{\sigma,j}\}_{j \in [\mathcal{Q}_c]}$ and $\{vk^*_i\}_{i \in [\mathcal{Q}_e]}$ do not provide additional information on Λ_{sk_H} since they are sampled independently. Likewise, $\{\Pi^*_j\}_{j \in [\mathcal{Q}_c]}$ and $\{\Pi_i\}_{i \in [\mathcal{Q}_e]}$ do not provide additional information since they are randomly sampled as of Game 5. It follows that only $\{c^*_{H,j}, \Psi^*_j\}_{j \in [\mathcal{Q}_c]}$ and $\{c_{H,i}, \Psi_i\}_{i \in [\mathcal{Q}_e]}$

provide information on Λ_{sk_H}. Given that for all $j \in [\mathcal{Q}_c]$, Ψ_j^* has 2^l possible values and for all $i \in [\mathcal{Q}_e]$, Ψ_i has 2^l possible values, we have the following using Lemma 1.

$$\begin{aligned}&\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H, \lambda, \{c_j^*\}_{j\in[\mathcal{Q}_c]}, \{c_i\}_{i\in[\mathcal{Q}_e]}))\\ &\quad \geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H, \lambda)) - \mathcal{Q}_e(l) - \mathcal{Q}_c(l)\end{aligned}$$

Applying Lemma 1 to the secret-key leakage λ, the above reduces to:

$$\begin{aligned}&\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H, \lambda, \{c_j^*\}_{j\in[\mathcal{Q}_c]}, \{c_i\}_{i\in[\mathcal{Q}_e]}))\\ &\quad \geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H)) - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l)\end{aligned}$$

Using the fact that H is an ϵ_1-universal hash proof system we have:

$$\Gamma_\infty(\Lambda_{sk_H}(c_H)|pk_H, c_H) \geq \nu = \log(1/\epsilon_1)$$

In addition, we can assume that $k = \Lambda_{sk_H}(c_H)$ has not been queried to the oracle, since the event that k is queried is taken into consideration in Proposition 3. It follows that, from the adversary's point of view, the mapping of Π to k is injective. Since injective mappings preserve average min-entropies, we have:

$$\tilde{\Gamma}_\infty(\Pi|(pk_H, c_H, \lambda, \{c_j^*\}_{j\in[\mathcal{Q}_c]}, \{c_i\}_{i\in[\mathcal{Q}_e]})) \geq \nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l)$$

Taking the logarithm of both sides and multiplying by -1, the probability that Z occurs is at most $\mathcal{Q}_d 2^{\lambda+(\mathcal{Q}_c+\mathcal{Q}_e)(l)}/2^\nu$. Assuming that up to $\mathcal{Q}_d$ decryption queries are not rejected, the adversary can rule out up to $\mathcal{Q}_d$ values of $k \in K$ (i.e., outputs of Λ_{sk_H}). Combining these, we have Equation (1) which represents an upper bound on the probability of Z. This probability is negligible given that $\lambda \leq \nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l) - \alpha(\log\kappa)$ under the assumptions. This proves the proposition. □

Proposition 7. *Game 6 and Game 7 are computationally indistinguishable, given that E is a* $((\nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l))/\mathcal{Q}_c, \epsilon_2)$ *average case strong randomness extractor.*

Proof. By Game 6, all ciphertexts with an invalid c_H component are explicitly rejected for decryption. Given any challenge ciphertext $c_j^* = (\sigma_j^*, c_{H,j}^*, r_j^*, vk_{\sigma,j}^*, \Psi_j^*, \Pi_j^*)$ for $j \in \mathcal{Q}_c$, the adversary cannot learn any additional information on $\Lambda_{sk_H}(c_{H,j}^*)$ aside from those provided by pk, Ψ_j^*, the secret-key leakage λ, and outputs of up to $\mathcal{Q}_e$ encryption queries: $\{c_i\}_{i\in[\mathcal{Q}_e]}$ and $\mathcal{Q}_c$ challenge queries: $\{c_j^*\}_{j\in[\mathcal{Q}_c]}$. Let $\Psi_i \in c_i$ and $\Psi_j^* \in c_j^*$. Both Ψ_i and Ψ_j^* have 2^l possible values for $i \in [\mathcal{Q}_e]$ and $j \in [\mathcal{Q}_c]$. Denote by $v_\mathcal{A}$ the information from the point of view of the adversary, i.e., $v_\mathcal{A} = (pk_H, \{\Psi_j^*\}_{j\in[\mathcal{Q}_c]}, \{\Psi_i\}_{i\in[\mathcal{Q}_e]}, \lambda)$. Under the assumptions of the cryptosystem, for all pk_H and $c_{H,j}^* \in C \setminus V$, we have $\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|(pk_H, c_{H,j}^*)) \geq \nu$. Combining these, we apply Lemma 1, and have the following result for each $j \in \mathcal{Q}_c$.

$$\begin{aligned}\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|v_\mathcal{A}) &\geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|(pk_H, c_{H,j}^*)) - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l)\\ &\geq \nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l)\end{aligned}$$

Given that extractor E is a $((\nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(l))/\mathcal{Q}_c, \epsilon_2)$ average case strong randomness extractor, the value of $E(\Lambda_{sk_H}(c_{H,j}^*))$ is ϵ_2 close to uniform from the point of view of the adversary. The claim thus follows. Combining all these claims prove the stated theorem as well. □

5.5. Cryptosystem 2

The rationale for constructing cryptosystem 2 over cryptosystem 1 is to do away with the non-reversing property of the adversary along with the need for random oracles. Cryptosystem 2 uses an L^M collection of lossy functions. The idea behind an L^M collection is that the system can sample up to M lossy branches. Having several lossy branches is useful given that, in a cryptosystem that uses lossy function primitives, the branch is published in the ciphertext.

5.6. Cryptosystem 2 Requirements

Let $\mathcal{Q}_e$, $\mathcal{Q}_d$, and $\mathcal{Q}_c$ denote the bounds in the number of encryption, decryption, and challenge queries, respectively. Define $\theta := n(M-1) + \log(p)$. Define $M := \mathcal{Q}_e + \mathcal{Q}_c$. All requirements of cryptosystem 2 are the same as cryptosystem 1, except for 2, 3, 6, and 7 which are, now, as follows.

2. An (n, p) L^M lossy function collection given by (L_{g^M}, L_{e^M}) with function index set S, branch set B, set of lossy branches $B^\circ \subseteq B$, and lossy branch $b^\circ \in B^\circ$, and where $L_{e^M} : S \times B \times K \to \{0,1\}^{nM}$.
3. E is a $((\nu - \lambda - ((\mathcal{Q}_c + \mathcal{Q}_e)(\theta + l))/\mathcal{Q}_c), \epsilon_2)$ average-case strong-randomness extractor
6. Elements of K, R, C, SK_H, PK_H have length n. Elements of M have length l. Elements of S have length n^2M.
7. ν, l, λ, and p are such that $\lambda \leq \nu - \lambda - ((\mathcal{Q}_c + \mathcal{Q}_e)(\theta + l)) - \alpha(\log(\kappa))$ for some constant $\alpha \geq 0$.

5.7. Cryptosystem 2

Key Generation. $\mathcal{G}(1^\kappa)$ first runs $H_g(1^\kappa)$ to obtain $sk_H \in SK_H$, and Λ_{sk_H} with μ. It computes $pk_H = \mu(sk_H) \in PK_H$. It defines $q := (0^n_1, 0^n_2, .., 0^n_M)$ and generates $s \leftarrow L_{g^{abo}}(1^\kappa, q)$. The output is a public/secret key pair (pk, sk), where $pk = (pk_H, s)$ and $sk = sk_H$ along with q.

Encryption. $\mathcal{E}(pk, m)$: on input $pk = (pk_H, s)$ and message $m \in M$, performs the following:

1. It samples $r'_1 \leftarrow R$, then computes $r_1 = P(r'_1, m)$ and sets $\omega = r_1$. Using pk_H and ω, it chooses $c_H \in V$.
2. It samples $r'_2 \leftarrow R$, then computes $r''_2 = r'_2 \oplus pk_H$, followed by $r_2 = P(r''_2, m) \in R$. Using r_2, it computes $k = H_{pub}(pk_H, c_H, \omega)$ and $\Psi = E(k, r_2) \oplus m$.
3. It generates $(vk_\sigma, sk_\sigma) \leftarrow F_g(1^\kappa)$. It defines $b = vk_\sigma$, and computes $\sigma = F_s(sk_\sigma, (c_H, r_2, \Psi))$.
4. It computes $\Pi = L_{e^M}(s, b, k)$.
5. It returns ciphertext $c = (\sigma, c_H, r_2, b, \Psi, \Pi)$

Decryption. $\mathcal{D}(sk, c)$: on input $sk = sk_H$ and $c = (\sigma, c_H, r, b, \Psi, \Pi)$, performs the following:

1. The algorithm checks if $F_v(b, (c_H, r_2, \Psi), \sigma) = 1$. If not, it outputs $\perp$
2. It computes $k' = H_{priv}(sk_H, c_h)$ and $\Pi' = L_{e^M}(s, b, k')$
3. It checks if $\Pi = \Pi'$. If not, it outputs $\perp$
4. It returns the plaintext message $m = \Psi \oplus E(k', r)$

For cryptosystem 2, the role of P is to generate fresh random numbers, r_1 and r_2, using the joint entropy of the message m and old random numbers r'_1 and r'_2 (due to reset attacks). Similar to cryptosystem 1, r_1 and r_2 serve as the actual randomness input to H_{pub} and E, respectively. To show correctness of the cryptosystem, given m and pk, $\mathcal{E}$ first computes r_1 and r_2 followed by $k \leftarrow H_{pub}(pk_H, c_H, r_1 = \omega)$, $\Psi = E(k, r_2) \oplus m$, and $\Pi = L_{e^M}(s, b, k)$, where s is part of pk and b is equal to vk_σ. Let $c = (\sigma, C_H, r_2, b, \Psi, \Pi)$ be the corresponding ciphertext where $b = vk_\sigma$ is jointly sampled with sk_σ under F_g and σ is derived using F_s as shown above. If this c were given to $\mathcal{D}$, it follows that $F_v(b, (c_H, r_2, \Psi), \sigma) = 1$ given that $b = vk_\sigma$, $\sigma = F_s(sk_\sigma, (c_H, r_2, \Psi))$ and the signature scheme (F_g, F_s, F_v) satisfies the correctness property. Having passed this first check, $\mathcal{D}$ computes $k' = H_{priv}(sk_H, c_H)$ and we have $k' = k$ given that H_{priv} uses the same c_H from encryption and sk_H is paired with pk_H using H_g. Given that (H_g, H_{pub}, H_{priv}) satisfies the correctness property of a hash proof system, we have $k' = k$ as claimed. Given that $k' = k$, we have $\Pi' = \Pi$ under L_{e^M}, given that s and b are the same as those is used in $\mathcal{E}$ and L_{e^M} is a function—thereby passing the second check. Finally, with $k' = k$, we have $m = \Psi \oplus E(k', r_2) = \Psi \oplus E(k, r_2)$, and the original message m is recovered.

5.8. Security Results for Cryptosystem 2 Scheme

Theorem 2. *Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ denote cryptosystem 2. For any equality-respecting, polynomial-time adversary that makes (a) at most $\mathcal{Q}_c$ challenge queries under multiple randomness indices, (b) at most $\mathcal{Q}_e$ encryption queries under multiple randomness indices, and (c) at most $\mathcal{Q}_d$ decryption queries, and following the attack game of Table 1, then cryptosystem 2 is secure against (i) adaptive chosen ciphertext attack, (ii) λ bits of secret-key leakage, and (iii) randomness reset attacks.*

Denote the challenge ciphertext as $c_j^* = (\sigma_j^*, c_{j,H}^*, r_j^*, b_j^*, \Psi_j^*, \Pi_j^*)$ for $j \in [\mathcal{Q}_c]$.

Game 0. This game implements the original cryptosystem with no modifications.

```
proc.Initialize($\kappa$)
    1. $(pk, sk, q) \leftarrow \mathcal{G}(1^\kappa)$
    2. initialize empty arrays coins and ciphers
    3. initialize empty associative arrays $\texttt{keys}_e$ and $\texttt{keys}_c$
    4. for $i \in [\mathcal{Q}_e]$ sample $(vk_{\sigma,i}, sk_{\sigma,i}) \leftarrow F_g(1^\kappa, q)$ and set $\texttt{keys}_e = \texttt{keys}_e \cup (vk_{\sigma,i}, sk_{\sigma,i})$
    5. for $j \in [\mathcal{Q}_c]$ sample $(vk_{\sigma,j}^*, sk_{\sigma,j}^*) \leftarrow F_g(1^\kappa, q)$ and set $\texttt{keys}_c = \texttt{keys}_c \cup (vk_{\sigma,j}^*, sk_{\sigma,j}^*)$
    6. send $pk$ to adversary
proc.Enc($pk', j, m$)
    1. if coins[$j$] $=\perp$ then randomly sample $(r_1', r_2', vk_\sigma, sk_\sigma)$ and set coins[$j$] $= (r_1', r_2', vk_\sigma, sk_\sigma)$
    2. compute $c \leftarrow \mathcal{E}(pk', m)$
    3. return $c$
proc.Challenge($j, m_0, m_1$)
    1. if $|m_0| \neq |m_1|$ return $\perp$
    2. if coins[$j$] $=\perp$ then randomly sample $(r_1', r_2', vk_\sigma, sk_\sigma)$ and set coins[$j$] $= (r_1', r_2', vk_\sigma, sk_\sigma)$
    3. compute $c^* \leftarrow \mathcal{E}(pk, m)$
    4. ciphers $\leftarrow$ ciphers $\cup\, c^*$
    5. return $c^*$
proc.Dec($c$)
    1. if $c \in$ ciphers, return $\perp$
    2. compute $m = \mathcal{D}(sk, c)$
    3. return $m$
proc.Leak($\lambda(\kappa)$)
    1. return $\lambda(\kappa)$ bits of $sk$
```

Game 1. This game is similar to Game 0, except that in $\mathcal{E}$, the challenger samples r_1, r_2 randomly.

Game 2. This game is similar to Game 1, except that once the adversary submits a ciphertext $c = (\sigma, c_H, r, b, \Psi, \Pi)$ for decryption such that $b = vk_\sigma^*$ and $vk_\sigma^* \in c'$ for some $c' \in$ `ciphers`, the challenger automatically returns $\perp$.

Game 3. This game is similar to Game 2, except in encryption query i for $i \in [\mathcal{Q}_e]$, on input (pk', j, m) from the adversary such that `coins`$[j] =\perp$, instead of sampling a fresh verification/signing key pair $(vk_\sigma, sk_\sigma) \leftarrow F_g(1^\kappa, q)$, it sets the verification/signing key pair as $(vk_{\sigma,i}, sk_{\sigma,i}) \in \texttt{keys}_e$ from the initialization phase.

Game 4. This game is similar to Game 3, except in challenge query j for all $j \in [\mathcal{Q}_c]$, on input (j', m_0, m_1) from the adversary such that $\texttt{coins}[j'] = \perp$, instead of sampling a fresh verification/signing key pair $(vk^*_\sigma, sk^*_\sigma) \leftarrow F_g(1^\kappa, q)$, it sets the verification/signing key pair as $(vk^*_{\sigma,j}, sk^*_{\sigma,j}) \in \texttt{keys}_e$ from the initialization phase.

Game 5. This game is similar to Game 4, except that during initialization, it defines $q := (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\sigma,\mathcal{Q}_e}, vk^*_{\sigma,1}, vk^*_{\sigma,2}, \ldots, vk^*_{\sigma,\mathcal{Q}_c})$ instead of $q := (0^n_1, 0^n_2, \ldots, 0^n_M)$.

Game 6. This game is similar to Game 5, except that in $\mathcal{E}$, the challenger computes $k = H_{priv}(sk_H, c_H)$ instead of using H_{pub}.

Game 7. This game is similar to Game 6, except that in encryption queries or challenge queries, c_H is sampled from $C \setminus V$ instead of V.

Game 8. This game is similar to Game 7, except that if the adversary submits a ciphertext $c = (\sigma, c_H, r, b, \Psi, \Pi)$ for decryption such that $c_H \in C \setminus V$, the challenger returns $\perp$.

Game 9. This game is similar to Game 8, except that in $\mathcal{E}$, the challenger draws Ψ uniformly at random from $\{0,1\}^l$ instead of computing $\Psi = E(k, r) \oplus m$.

Proposition 8. *Game 0 and Game 1 are computationally indistinguishable, given the security of the pseudorandom function P.*

Proof. The proof for this proposition is similar to the proof for Proposition 1. □

Proposition 9. *Game 1 and Game 2 are computationally indistinguishable, given the strong one-time existential unforgeability of the signature scheme.*

Proof. The proof for this proposition is similar to the proof for Proposition 2 since $b = vk_\sigma$ in $\mathcal{E}$. □

Proposition 10. *Games 2 and 3 are perfectly equivalent.*

Proof. We note that the only difference in Games 2 and 3 is that, in Game 3, the verification/signature keys used in encryption queries are drawn from the initialization phase instead of being sampled on the fly. This does not affect any other part of the computation. □

Proposition 11. *Games 3 and 4 are perfectly equivalent.*

Proof. We note that the only difference in Games 3 and 4 is that, in Game 4, the verification/signature keys used in challenge queries are drawn from the initialization phase instead of being sampled on the fly. This does not affect any other part of the computation. □

Proposition 12. *Game 4 and 5 are indistinguishable given that two candidate lossy branch sets of the L^M collection are computationally indistinguishable.*

Proof. To prove this proposition, we two experiments A and B. Experiment A is a sequence of sub-experiments $H_1, \ldots, H_{\mathcal{Q}_e+1}$. Experiment B is a sequence of sub-experiments $H'_1, \ldots, H_{\mathcal{Q}_c+1}$. For all sub-experiments in A and B, assume a fixed $\texttt{keys}_e$ and $\texttt{keys}_c$.

For experiment A, sub-experiment H_1 emulates the challenger of Game 4 perfectly. Given $i' \in [\mathcal{Q}_e + 1]$, sub-experiment $H_{i'}$ defines q as $q := (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\sigma,i'-1}, 0^n_{i'}, \ldots, 0^n_M)$ in the initialization phase and computes $s \leftarrow L_g(1^\kappa, q)$,where for $i \in [\mathcal{Q}_e]$, we have $vk_{\sigma,i} \in (vk_{\sigma,i}, sk_{\sigma,i})$ such that $(vk_{\sigma,i}, sk_{\sigma,i}) \in \texttt{keys}_e$. Suppose that there exists an efficient adversary that can distinguish between $H_{i'}$ and $H_{i'+1}$. Using this adversary, we construct a simulator that can distinguish between two candidate lossy branch sets of the L^M collection. The simulator has access to an oracle that, on input (q_0, q_1), provides it with the function index s, where s can either be $s \leftarrow L_g(1^\kappa, q_0)$ or $s \leftarrow L_g(1^\kappa, q_1)$. At initialization, given $i' \in [\mathcal{Q}_e]$, the simulator constructs $q_0 = (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\sigma,i'-1}, 0^n_{i'}, \ldots, 0^n_M)$

and $q_1 = (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\sigma,i'-1}, vk_{\sigma,i'}, 0^n_{i'+1} \ldots, 0^n_M)$. It forwards (q_0, q_1) to its oracle and receives function index s. Using s, it constructs pk and sk, draws $a \in \{0,1\}$, then forwards pk to the adversary. Since the simulator knows pk, it can answer encryption and challenge queries. Since it knows sk, it can answer decryption and secret-key leakage queries. Once the adversary outputs a guess a', the simulator outputs 1 if $a' = a$. The advantage of the simulator in distinguishing between two candidate lossy branch sets of the L^M collection is equivalent to the probability that the adversary outputs a' such that $a' = a$ less $1/2$. However, given that it is computationally difficult to distinguish between two lossy branch sets in an L^M collection, no efficient adversary can output a' such that $a' = a$ with non-negligible probability. It follows that the advantage of the simulator is likewise negligible. By construction, if the oracle computes $s \leftarrow L_g(1^\kappa, q_0)$, the simulator is performing sub-experiment $H_{i'}$. If the oracle computes $s \leftarrow L_g(1^\kappa, q_1)$, the simulator is performing sub-experiment $H_{i'+1}$ for any $i' \in [\mathcal{Q}_e]$.

For experiment B, sub-experiment H'_1 emulates sub-experiment $H_{\mathcal{Q}_e+1}$ of experiment A perfectly. Given $j' \in [\mathcal{Q}_c]$, sub-experiment $H_{j'}$ defines q as:

$$q = (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\mathcal{Q}_e}, vk^*_{\sigma,1}, vk^*_{\sigma,2}, \ldots, vk^*_{\sigma,i'+j'-1}, 0^n_{i'+j'}, \ldots, 0^n_M)$$

in the initialization phase and computes $s \leftarrow L_g(1^\kappa, q)$,where for $j \in [\mathcal{Q}_c]$, we have $vk^*_{\sigma,j} \in (vk^*_{\sigma,j}, sk^*_{\sigma,j})$ such that $(vk^*_{\sigma,j}, sk^*_{\sigma,j}) \in \mathtt{keys}_c$, and where for $i \in [\mathcal{Q}_e]$, we have $vk_{\sigma,i} \in (vk_{\sigma,i}, sk_{\sigma,i})$ such that $(vk_{\sigma,i}, sk_{\sigma,i}) \in \mathtt{keys}_e$. Suppose that there exists an efficient adversary that can distinguish between $H_{j'}$ and $H_{j'+1}$. Using this adversary, we construct a simulator that can distinguish between two candidate lossy branch sets of the L^M collection. The simulator has access to an oracle that, on input (q_0, q_1), provides it with function index s, where s can either be $s \leftarrow L_g(1^\kappa, q_0)$ or $s \leftarrow L_g(1^\kappa, q_0)$. At initialization, given $j' \in [\mathcal{Q}_c]$, the simulator constructs (q_0, q_1) as follows

$$q_0 = (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\mathcal{Q}_e}, vk^*_{\sigma,1}, vk^*_{\sigma,2}, \ldots, vk^*_{\sigma,\mathcal{Q}_e+j'-1}, 0^n_{\mathcal{Q}_e+j'}, \ldots, 0^n_M)$$
$$q_1 = (vk_{\sigma,1}, vk_{\sigma,2}, \ldots, vk_{\mathcal{Q}_e}, vk^*_{\sigma,1}, vk^*_{\sigma,2}, \ldots, vk^*_{\sigma,\mathcal{Q}_e+j'-1}, vk^*_{\sigma,\mathcal{Q}_e+j'}, 0^n_{\mathcal{Q}_e+j'+1} \ldots, 0^n_M)$$

It forwards (q_0, q_1) to its oracle and receives function index s. Using s, it constructs pk and sk, draws $a \in \{0,1\}$, then forwards pk to the adversary. Since the simulator knows pk, it can answer encryption and challenge queries. Since it knows sk, it can answer decryption and secret-key leakage queries. Once the adversary outputs a guess a', the simulator outputs 1 if $a' = a$. The advantage of the simulator in distinguishing between two candidate lossy branch sets of the L^M collection is equivalent to the probability that the adversary outputs a' such that $a' = a$ less $1/2$. However, given that it is computationally difficult to distinguish between two lossy branch sets in an L^M collection, no efficient adversary can output a' such that $a' = a$ with non-negligible probability. It follows that the advantage of the simulator is likewise negligible. By construction, if the oracle computes $s \leftarrow L_g(1^\kappa, q_0)$, the simulator is performing sub-experiment $H_{j'}$. If the oracle computes $s \leftarrow L_g(1^\kappa, q_1)$, the simulator is performing sub-experiment $H_{j'+1}$ for any $j' \in [\mathcal{Q}_c]$.

Combining the above, we have that sub-experiment H_1 of experiment A perfectly simulates Game 4. Sub-experiment H'_1 of experiment B perfectly simulates sub-experiment $H_{\mathcal{Q}_e+1}$ of experiment A. Sub-experiment $H'_{\mathcal{Q}_c+1}$ of experiment B perfectly simulates Game 5. Since all sub-experiments in A and B are pairwise indistinguishable, the proposition thus follows. □

Proposition 13. *Games 5 and 6 are perfectly equivalent.*

Proof. The proof for this proposition is similar to the proof for Proposition 4. □

Proposition 14. *Games 6 and 7 are indistinguishable given the hardness of the underlying subset membership problem in H.*

Proof. The proof for this proposition is similar to the proof for Proposition 5. □

Proposition 15. *Game 7 and Game 8 are computationally indistinguishable, given (i) the lossy property of the L^M collection, (ii) the computational hardness of determining lossy branches for the L^M collection, and (iii) the ϵ_1-universal property of H.*

Proof. Define Z to be the event that some ciphertext $c = (\sigma, c_H, r, b, \Psi, \Pi)$ with $c_H \in C \setminus V$, is accepted for a decryption query in Game 7, but is rejected in Game 8. It follows that Game 7 and Game 8 proceed identically until Z occurs. Given at most $\mathcal{Q}_d$ decryption queries, $\mathcal{Q}_e$ encryption queries, and $\mathcal{Q}_c$ challenge queries, we claim the following.

$$\Pr(Z) \leq \Pr(Z_1) + \Pr(Z_2) \leq \mathcal{Q}_d\Theta + \mathcal{Q}_d\gamma \tag{2}$$

where γ represents the probability of some adversary generating a lossy branch for the L^M collection and $\Theta = (2^{\lambda+\mathcal{Q}_c(\theta+l)+\mathcal{Q}_e(\theta+l)})/(2^\nu - \mathcal{Q}_d)$, with $\theta := n(M-1) + \log(p)$ and $\nu = \log(1/\epsilon_1)$. We first prove the equation for $\mathcal{Q}_d\Theta$. $\mathcal{Q}_d\Theta$ represents the event Z_1 that c is accepted for decryption, and where $b \in c$ is an injective branch with $c_h \in C \setminus V$. Let $c = (\sigma, c_H, r, b, \Psi, \Pi)$ be a decryption query input that triggers Z_1. In encryption, c_H serves as input to Λ_{sk_H}. From the adversary's point of view, Λ_{sk_H} is dependent on pk_H, c_H, λ, the set $\{c_i\}_{i\in[\mathcal{Q}_e]}$ of encryption query outputs and the set $\{c_j^*\}_{j\in[\mathcal{Q}_c]}$ of challenge ciphertexts. For $i \in [\mathcal{Q}_e]$ and $j \in [\mathcal{Q}_c]$, the signatures $\sigma_i \in c_i$ and $\sigma_j^* \in c_j^*$ do not provide additional information on Λ_{sk_H}, since they are functions of $(c_{H,i}, \Psi_i, b_i)$ and $(c_{H,j}^*, \Psi_j^*, b_j^*)$, respectively. For $i \in [\mathcal{Q}_e]$ and $j \in [\mathcal{Q}_c]$, the branches b_i and b_j^* likewise do not provide additional information as they are independently sampled. Using results of Lemma 1, we prove the following:

$$\begin{aligned}
&\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H, \Lambda, \{c_i\}_{i\in[\mathcal{Q}_e]}, \{c_j^*\}_{j\in[\mathcal{Q}_c]}))\\
&\quad\geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk, c_H, \{c_i\}_{i\in[\mathcal{Q}_e]}, \{c_j^*\}_{j\in[\mathcal{Q}_c]})) - \lambda\\
&\quad\geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk, c_H, \{c_i\}_{i\in[\mathcal{Q}_e]})) - \lambda - \mathcal{Q}_c\theta - \mathcal{Q}_c l\\
&\quad\geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_H)|(pk, c_H)) - \lambda - \mathcal{Q}_c\theta - \mathcal{Q}_c l - \mathcal{Q}_e\theta - \mathcal{Q}_e l\\
&\quad\geq \nu - \lambda - \mathcal{Q}_c(\theta + l) - \mathcal{Q}_e(\theta + l)
\end{aligned}$$

The first inequality applies Lemma 1. For the second inequality, given challenge ciphertext $c_j^* = (\sigma_j^*, c_{H,j}^*, r_j^*, b_j^*, \Psi_j^*, \Pi_j^*)$ for $j \in [\mathcal{Q}_c]$, information on sk_H is provided by $c_{H,j}^*$, Ψ_j^* and Π_j^* as shown above. Given that Ψ_j^* has 2^l possible values, and Π_j^* has $2^{\log(p)} = p$ possible values, the second inequality follows by applying Lemma 1. For the third inequality, given any encryption query output $c_i = (\sigma_i, c_{H,i}, r_i, b_i, \Psi_i, \Pi_i)$ for $i \in [\mathcal{Q}_e]$, information on sk_H is covered by $c_{H,i}$, Ψ_i and Π_i as shown above, where Ψ_i has 2^l possible values, while Π_i has $2^{\log(p)} = p$ possible values. Given that the total number of encryption queries is $\mathcal{Q}_e$, the third inequality follows from Lemma 1. For the last inequality, for all pk_H and $c_H \in C \setminus V$, we have $\Gamma_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H)) \geq \nu = \log(1/\epsilon_1)$, under the assumption that H is an ϵ_1-universal hash proof system.

Using the above set of inequalities, we note that $\Pi = L_{e^M}(s, b, k)$ with $k = \Lambda_{sk_H}(c_H)$. Starting with Game 2, all ciphertexts for decryption involve injective branches. Injective functions preserve the min-entropy of its input, and we have $\tilde{\Gamma}_\infty(L_{e^{abo}}(s, b, \Lambda_{sk_H}(c_H))|v_{\mathcal{A}}) \geq \nu$, where $v_{\mathcal{A}} = (pk_H, c_H, \Lambda, \{c_i\}_{i\in[\mathcal{Q}_e]}, \{c_j^*\}_{j\in[\mathcal{Q}_c]})$. Using the fact that H is an ϵ_1-universal hash proof system we have $\Gamma_\infty(\Lambda_{sk_H}(c_H)|(pk_H, c_H)) \geq \nu = \log(1/\epsilon_1)$. We then have:

$$\tilde{\Gamma}_\infty(L_{e^M}(s, b, \Lambda_{sk_H}(c_H))|v_{\mathcal{A}}) \geq \nu - \lambda - \mathcal{Q}_c(\theta + l) - \mathcal{Q}_e(\theta + l)$$

Taking the logarithm of both sides and multiplying by -1, the probability that Z_1 occurs is at most $\mathcal{Q}_d 2^{\lambda+(\mathcal{Q}_e+\mathcal{Q}_c)(\theta+l)}/2^\nu$. Assuming that up to $\mathcal{Q}_d$ decryption queries are not rejected, the adversary can rule out up to $\mathcal{Q}_d$ values of $k \in K$ (i.e., outputs of Λ_{sk_H}). Combining these, we have:

$$\Pr(Z_1) \leq \mathcal{Q}_d \frac{2^{\lambda+(\mathcal{Q}_e+\mathcal{Q}_c)(\theta+l)}}{2^{\nu} - \mathcal{Q}_d} = \mathcal{Q}_d \Theta$$

$\Pr(Z_1)$ is negligible given that $\lambda \leq \nu - \lambda - (\mathcal{Q}_e + \mathcal{Q}_c)(\theta + l) - \alpha(\log \kappa)$ by assumption. We now state the proof for the second element $\mathcal{Q}_d\gamma$ at the right hand side of Equation (2). $\mathcal{Q}_d\gamma$ accounts for the event Z_2 that an adversary submits c for decryption such that $b \in c$ is a new lossy branch, i.e., $b \notin c_i$ for any prior encryption query output c_i and $b \notin c_j^*$ for any $c_j^* \in \texttt{ciphers}$. Suppose that Z_2 occurs under some efficient adversary. Using this adversary, we construct a simulator that can efficiently generate a lossy branch for L^M. The simulator has access to an oracle that provides it with a function index s, but the oracle does not disclose the corresponding set of lossy branches. In encryption query i for $i \in [\mathcal{Q}_e]$, the oracle provides the simulator with a signing/verification key pair $(sk_{i,\sigma}, vk_{i,\sigma})$ such that $vk_{i,\sigma}$ is equal to a lossy branch in L^M. The same is done by the oracle for challenge queries. At initialization, the simulator completely emulates the challenger in Game 10, but requests its oracle for s. Since the simulator knows pk_H, s, $\texttt{coins}$, and can request its oracle for signing and verification keys, it can answer any encryption or challenge query. Since the simulator knows sk, it can answer secret-key leakage queries and decryption queries. The simulator keeps a list of all the ciphertext inputs it received for a decryption query. At the end of the game, the simulator randomly picks $i \in [\mathcal{Q}_d]$ and outputs the branch b_i of the ith decryption query ciphertext input. Suppose that with probability γ, the adversary is able to query a ciphertext for decryption such that its branch is lossy. The probability that the simulator outputs a lossy branch for L^M is then $\gamma/\mathcal{Q}_d$. However, given that it is computationally difficult to generate a lossy branch for L^M, γ is negligible. It follows that $\mathcal{Q}_d\gamma$ in Equation (2) or $\Pr(Z_2)$ is also negligible. Combining all these facts the proposition follows. □

Proposition 16. *Game 8 and Game 9 are computationally indistinguishable, given that E is a* $((\nu - \lambda - ((\mathcal{Q}_c + \mathcal{Q}_e)(\theta + l))/\mathcal{Q}_c), \epsilon_2)$ *average case strong randomness extractor.*

Proof. By Game 8, all ciphertexts with an invalid c_H component are explicitly rejected for decryption. Denote challenge ciphertext c_j^* as $c_j^* = (\sigma_j^*, c_{H,}^*, r_j^*, b_j^*, \Psi_j^*, \Pi_j^*)$ for $j \in \mathcal{Q}_c$. Denote an encryption query output as c_i as $c_i = (\sigma_i, c^{H,i}, r_i, b_i, \Psi_i, \Pi_i)$ for $i \in \mathcal{Q}_e$. For each c_j^*, the adversary cannot learn any information on $\Lambda_{sk_H}(c_{H,j}^*)$ aside from those provided by pk_H, Π_j^* and Ψ_j^*, λ, encryption query outputs $\{c_i\}_{i\in[\mathcal{Q}_e]}$ and challenge ciphertexts $\{c_j^*\}_{j\in[\mathcal{Q}_j]}$. Both Ψ_j^* and Ψ_i have 2^l possible values for $i \in [\mathcal{Q}_e]$ and $j \in [\mathcal{Q}_c]$. Π_j^* has $2^{\log(p)} = p$ possible values, and Π_i has $2^{\log(p)} = p$ possible values for $i \in [\mathcal{Q}_e]$ and $j \in [\mathcal{Q}_c]$. Under the assumptions of the cryptosystem, for all pk_H and $c_{H,j}^* \in \mathcal{C} \setminus \mathcal{V}$, we have $\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|(pk_H, c_{J,j}^*)) \geq \nu$. Combining these, we apply Lemma 1, and have the following result for each $j \in [\mathcal{Q}_c]$.

$$\begin{aligned}
&\tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|(pk_H, c_j^*)) \\
&\quad \geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|(pk_H, c_{H,j}^*, \lambda, \{\Psi_j^*\}_{j\in[\mathcal{Q}_c]}, \{\Psi_i\}_{i\in[\mathcal{Q}_e]}, \{\Pi_j^*\}_{j\in[\mathcal{Q}_c]}, \{\Pi_i\}_{i\in[\mathcal{Q}_e]})) \\
&\quad \geq \tilde{\Gamma}_\infty(\Lambda_{sk_H}(c_{H,j}^*)|(pk_H, c_H^*)) - \lambda - \mathcal{Q}_c(\log(p) + l) - \mathcal{Q}_e(\log(p) + l) \\
&\quad \geq \nu - \lambda - (\mathcal{Q}_c + \mathcal{Q}_e)(\log(p) + l)
\end{aligned}$$

Applying the above for $j \in [\mathcal{Q}_c]$, and given that extractor E is a $((\nu - \lambda - ((\mathcal{Q}_c + \mathcal{Q}_e)(\theta + l))/\mathcal{Q}_c), \epsilon_2)$ average case strong randomness extractor, the value of $E(\Lambda_{sk_H}(c_H^*))$ is ϵ_2 close to uniform from the point of view of the adversary. The claim thus follows. Combining all these claims prove the stated theorem as well. □

6. Concrete Instantiations

Let $\mathcal{A}_{ddh}$ be a group sampling algorithm that takes as input a security parameter, κ, and which outputs a tuple, $(p, \mathbb{G}, g)$, where $\mathbb{G}$ is a cyclic group of order p for some prime p, and g is the generator of the group. Let $a, b, c \leftarrow \mathbb{Z}_p$. The DDH assumption states that the ensemble of tuples $\{(\mathbb{G}, g^a, g^b, g^{ab})\}_{\kappa\in\mathbb{N}}$ is computationally difficult to distinguish from the ensemble of tuples $\{(\mathbb{G}, g^a, g^b, g^c)\}_{\kappa\in\mathbb{N}}$

6.1. Hash Proof System Based on DDH

This concrete example of a hash proof system follows [2].

Parameter Generation. $H_g(1^\kappa)$ first runs $\mathcal{A}_{ddh}$ to obtain $(p, \mathbb{G}, g) \leftarrow \mathcal{A}_{ddh}(1^\kappa)$. It randomly samples $u \leftarrow \mathbb{G}$ followed by $\sigma, \tau \leftarrow \mathbb{A}_p$. It defines the secret key as $sk_H = (\sigma, \tau)$. It defines the projective hash function $\Lambda_{sk_H} : \mathbb{G}^2 \to \mathbb{G}$ as $\Lambda_{sk_H}(v, w) = v^\sigma, w^\tau$. In this case, the projection function μ associated with Λ_{sk_H} is the same, i.e., $\mu := \Lambda_{sk_H}$. Using μ, it constructs the public key $pk_H = (h, u)$, where $h := \mu(g, u) = g^\sigma u^\tau$. In this case, the set of ciphertexts C consists of $\mathbb{G}^2$, i.e., $C := \mathbb{G} \times \mathbb{G}$. The set of valid ciphertexts $V \subseteq C$ consists of all element (u, v) of $\mathbb{G} \times \mathbb{G}$ for which (u, v, w) is a DH-triple, i.e., $u^\omega v^\omega = w^\omega$.

Public Evaluation. $H_{pub}(pk_H, c_H, \omega)$ takes as input public key $pk_H = (h, u)$, $c_H \in \mathbb{G} \times \mathbb{G}$ and witness $\omega \in \mathbb{Z}_p$ that $c_H \in V$. Let $c_H = (v, w)$. Given u, H_{pub} first checks if $(v, w) = g^\omega u^\omega$. If true, H_{pub} outputs h^ω, which equals to $\Lambda_{sk_H}(v, w)$ given that:

$$\Lambda_{sk_H}(v, w) = v^\sigma w^\sigma = (g^\omega)^\sigma (u^\omega)^\tau = (g^\sigma u^\tau)^\omega = h^\omega$$

We note here that the value of $\Lambda_{sk_H}(v, w)$ is evaluated using only the auxiliary information h, without knowing the secret key (σ, τ).

Private Evaluation. $H_{priv}(sk_H, c_H)$ takes as input private key $sk_H = (\sigma, \tau)$, and $c_H \in \mathbb{G} \times \mathbb{G}$. Let $c_H = (v, w)$. H_{priv} evaluates Λ_{sk_H} on any element of C without requiring a witness. In this case, H_{priv} outputs $v^\sigma u^\tau$.

As shown in [2] if Λ_{sk_H} is evaluated in (v, w) without using auxiliary information h, the probability that $\Lambda_{sk_H}(v, w)$ equals some number is $1/p$, which is close to uniform.

6.2. ABO Function Collection Based on DDH

Given κ, let $(p, \mathbb{G}, g) \leftarrow \mathcal{A}_{ddh}(1^\kappa)$. The ElGamal cryptosystem is initiated as follows: (i) a secret key is drawn randomly $z \leftarrow \mathbb{Z}_p$ and (ii) the public key is computed as $h = g^z$ [8]. Given public key h and secret key z, let E_h denote the encryption algorithm of ElGamal, while D_z is the decryption algorithm. Encryption of a message $m \in \mathbb{Z}_p$ is done by first drawing random number $z \leftarrow \mathbb{Z}_p$, and the ciphertext c is the tuple $E_h(m, r) = c = (g^r, h^r g^m)$. Decryption on the other hand is computed as $D_z(c) = \log_g(h^r g^m / g^{rz})$. The correctness of the ElGamal cryptosystem follows from $\log_g(h^r g^m / g^{rz}) = \log_g(g^{rz} g^m / g^{rz}) = \log_g(g^m) = m$. From [8], this cryptosystem is semantically secure under the DDH assumption and is *additively homomorphic* in the sense that, given two ciphertexts $c_a = E_h(m_1, r_1)$ and $c_b = E_H(m_2, r_1)$ for any pair of messages $m_1, m_2 \in \mathbb{Z}_p$ and random elements $r_1, r_2 \in \mathbb{Z}_p$, they can be 'added', in the sense that $c_a \boxtimes c_b = E_h(m_1 + m_2, r_1 + r_2)$, where $\boxtimes$ is the coordinate-wise multiplication of ciphertexts. This can, likewise, be applied to the exponentiation operation, i.e., $E_h(m, r)^x = E_h(mx, rx)$. We also define the operation $\boxplus$; given ciphertext $c = (c_1, c_2)$, we can add a scalar $v \in \mathbb{Z}_p$ to the underlying plaintext, i.e., $c \boxplus v := (c_1, c_2 g^v) = E_h(m + v, r)$, using the r used in computing c.

6.2.1. Matrix Encryption

Let $n > 0$ be some integer and p a prime number. Given a matrix $\boldsymbol{M} = (m_{i,j}) \in \mathbb{Z}_p^{n\times n}$, $\boldsymbol{M}$ can be encrypted based on ElGamal. Encryption produces indistinguishable ciphertexts under the DDH assumption (Lemma 4 below). The first step is to sample $z_j \in \mathbb{Z}_p$ for $j \in [n]$, followed by computing $h_j = g^{z_j}$ for $j \in [n]$. Encrypting $\boldsymbol{M}$ is done by drawing n random numbers $r_i \in \mathbb{Z}_p$, and then, for rows $i \in [n]$ and columns $j \in [n]$, we compute the ciphertext $c_{ij} = E_{h_j}(m_{ij}, r_i)$. This scheme outputs a ciphertext tuple $(\boldsymbol{C}_1, \boldsymbol{C}_2)$ that serves as the encryption of $\boldsymbol{M}$, where the vector $\boldsymbol{C}_1$ and the matrix $\boldsymbol{C}_2$ are as follows.

$$\boldsymbol{C}_1 = \begin{pmatrix} g^{r_1} \\ \vdots \\ g^{r_n} \end{pmatrix} \qquad \boldsymbol{C}_2 = \begin{pmatrix} h_1^{r_1} g^{m_{1,1}} & h_2^{r_1} g^{m_{1,2}} & \cdots & h_n^{r_1} g^{m_{1,n}} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{r_n} g^{m_{n,1}} & h_2^{r_n} g^{m_{n,2}} & \cdots & h_n^{r_n} g^{m_{nn}} \end{pmatrix}$$

Lemma 4 ([8])**.** *The ElGamal matrix encryption scheme produces indistinguishable ciphertexts.*

6.2.2. DDH-Based ABO Function Collection

Let a (n, p) collection of ABO functions be given by $L_{g_{abo}}, L_{e_{abo}}$ ([8] also includes a trapdoor algorithm but we do not include it here). Given κ, let $B := B_\kappa$ denote the set of branches with lossy branch $b^\circ \in B$. Each element of B belongs to $\mathbb{Z}_p^{1\times n}$. The algorithms are described as follows:

Function sampling algorithm. $L_{g_{abo}}(1^\kappa, b^\circ)$ takes as input the security parameter κ, and the lossy branch $b^\circ \in B$. Denote $b^\circ := (b_1^\circ, b_2^\circ, \ldots, b_n^\circ)$. The algorithm first runs $(p, \mathbb{G}, q) \leftarrow \mathcal{A}_{ddh}(1^\kappa)$. Let $\boldsymbol{I}$ denote the identity matrix over $\mathbb{Z}_p^{n\times n}$. $L_{g_{abo}}$ samples a vector of secret keys $\boldsymbol{z} = \{z_j\}_{i\in[n]}$ and forms the vector $\boldsymbol{h} = g^{\boldsymbol{z}} \in \mathbb{Z}_p^{1\times n}$. It also samples a random vector, $\boldsymbol{r} \in \mathbb{Z}_p^{1\times n}$. The output $\boldsymbol{s}$ of $L_{g_{abo}}$ consists of the function index $\boldsymbol{s} \in \mathbb{Z}_p^{n\times n}$, which is the second element (i.e., the matrix $\boldsymbol{C}_2$) of the ElGamal matrix encryption of $-(b^\circ \boldsymbol{I})$ (using $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{z}$), along with the $\boldsymbol{t}$ consisting of the vector of secret keys $\boldsymbol{z} = \{z_j\}_{i\in[n]} \in \mathbb{Z}_p^{1\times n}$. The vectors $\boldsymbol{r}$ and $\boldsymbol{h}$ do not need to be published publicly. In detail, we have $\boldsymbol{s}$ as follows.

$$\boldsymbol{s} = \begin{pmatrix} h_1^{r_1} g^{-b_1^\circ} & h_2^{r_1} & \cdots & h_n^{r_1} \\ h_1^{r_2} & h_2^{r_2} g^{-b_2^\circ} & \cdots & h_n^{r_2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{r_n} & h_2^{r_n} & \cdots & h_n^{r_n} g^{-b_n^\circ} \end{pmatrix}$$

Evaluation algorithm. $L_{e_{abo}}(\boldsymbol{s}, b, \boldsymbol{m})$ takes as input the function index consisting of the matrix $\boldsymbol{s} = -(b^\circ \boldsymbol{I})$ sampled by $L_{g_{abo}}$, a desired branch b from B, and a vector $\boldsymbol{m} \in \mathbb{Z}_p^{1\times n}$. The output consists of a vector $\boldsymbol{\Pi} = \boldsymbol{m}(\boldsymbol{s} \boxplus b\boldsymbol{I}) \in \mathbb{Z}_p^{1\times n}$. In summary:

$$\boldsymbol{\Pi} = \boldsymbol{m}(\boldsymbol{s} \boxplus b\boldsymbol{I}) = \begin{pmatrix} g^{m_1} \\ g^{m_2} \\ \vdots \\ g^{m_n} \end{pmatrix}^T \begin{pmatrix} h_1^{r_1} g^{(b_1 - b_1^\circ)} & h_2^{r_1} & \cdots & h_n^{r_1} \\ h_1^{r_2} & h_2^{r_2} g^{(b_2 - b_2^\circ)} & \cdots & h_n^{r_2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{r_n} & h_2^{r_n} & \cdots & h_n^{r_n} g^{(b_n - b_n^\circ)} \end{pmatrix}$$
$$= \begin{pmatrix} h^{\langle \boldsymbol{m}, \boldsymbol{r} \rangle} g^{m_1(b_1 - b_1^\circ)} \\ h^{\langle \boldsymbol{m}, \boldsymbol{r} \rangle} g^{m_2(b_2 - b_2^\circ)} \\ \vdots \\ h^{\langle \boldsymbol{m}, \boldsymbol{r} \rangle} g^{m_n(b_n - b_n^\circ)} \end{pmatrix} = \begin{pmatrix} g^{z_1(m_1 r_1 + m_2 r_2 + \ldots m_n r_n) + m_1(b_1 - b_1^\circ)} \\ g^{z_2(m_1 r_1 + m_2 r_2 + \ldots m_n r_n) + m_2(b_2 - b_2^\circ)} \\ \vdots \\ g^{z_n(m_1 r_1 + m_2 r_2 + \ldots m_n r_n) + m_n(b_n - b_n^\circ)} \end{pmatrix}$$

Let y_j denote the jth coordinate of $\boldsymbol{\Pi}$ and let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denote the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$. Let m_j denote the jth coordinate of $\boldsymbol{m}$, and let $E_{h_j}^{c_2}$ denote the function E_{h_j} that discards the first element c_1 in its output and returns only c_2. Using the homomorphic properties of the system we restate y_j as: $y_j := E_{h_j}^{c_2}((b - b^\circ)m_j, r := \langle \boldsymbol{m}, \boldsymbol{r} \rangle)$. This is computed without knowing $\boldsymbol{r}$ and $\boldsymbol{h}$ from the sampling stage; oOnly $\boldsymbol{s}$ is needed to compute $\boldsymbol{\Pi}$.

Lemma 5 ([8]). *The algorithms $L_{g^{abo}}, L_{e^{abo}}$, described above, give a collection of (n, p) ABO lossy functions under the DDH assumption for $\mathcal{A}_{ddh}$.*

6.3. L^M Lossy Function Collection

Let n, p be values of polynomials in κ, and let $M \in \mathbb{N}$. Let S denote the set of function indices of a L^M collection and let $\mathcal{B} = \{B\}_{\kappa \in \mathbb{N}}$ denote the collection of sets of branches indexed by κ. A concrete instantiation of a (n, p) L^M collection is given by algorithms L_{g^M}, L_{e^M}, which are as follows.

Concrete Instantiation of the L^M Collection

Given κ, define $B := B$ as the branch set corresponding to function index $s \in S$. Define $B^\circ \subseteq B$ as a lossy branch set of size M with elements $b_i^\circ \in \mathbb{Z}_p^{1\times n}$ for $i \in [M]$. Using B°, define q as the ordered tuple $q := (b_1^\circ, b_2^\circ, \ldots, b_M^\circ)$ for $b_i^\circ \in B^\circ$, $i \in [M]$. Suppose that $(L_{g^{abo}}, L_{e^{abo}})$ are algorithms that give a (n, p) ABO collection satisfying the required properties. Using $(L_{g^{abo}}, L_{e^{abo}})$, we implement (L_{g^M}, L_{e^M}) as follows.

Function sampling algorithm. $L_{g^M}(1^\kappa, q)$ takes as input κ and q. The algorithm computes $(p, \mathbb{G}, q) \leftarrow \mathcal{A}_{ddh}(1^\kappa)$. Given q and p, L_{g^M} outputs s, where $s \in S$. Let $s \in \mathbb{Z}_{p,1}^{n\times n} \times \mathbb{Z}_{p,2}^{n\times n} \times ... \times \mathbb{Z}_{p,M}^{n\times n}$. Denote by s_i the ith coordinate of s, denote by q_i the ith coordinate of q. Using the algorithm $L_{g^{abo}}$, we compute $s_i \leftarrow L_{g^{abo}}(1^\kappa, q_i)$ for $i \in [M]$ so that $s = (s_1, s_2, ..., s_M)$.

Evaluation algorithm. $L_{e^M}(s, b, m)$: on input s, branch $b \in B$ and message $m \in \mathbb{Z}_p^{1\times n}$, the output is a vector $\boldsymbol{\pi} \in \mathbb{Z}_{p,1}^{1\times n} \times \mathbb{Z}_{p,2}^{1\times n} \times ... \times \mathbb{Z}_{p,M}^{1\times n}$. Denote by s_i the ith coordinate of s and denote by $\boldsymbol{\pi}_i$ the ith coordinate of $\boldsymbol{\pi}$. Using the algorithm $L_{e^{abo}}$, $\boldsymbol{\pi}_i$ is computed as $\boldsymbol{\pi}_i = L_{e^{abo}}(s_i, b, m)$ for $i \in [M]$, so that $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, .., \boldsymbol{\pi}_M)$.

Lemma 6. *Assume that $(L_{g^{abo}}, L_{e^{abo}})$ are algorithms that give a (n, p) ABO collection of lossy functions satisfying the required properties for an ABO collection. Using $(L_{g^{abo}}, L_{e^{abo}})$, the algorithms (L_{g^M}, L_{e^M}), presented above, give a L^M collection of lossy functions that satisfies the required properties of a L^M collection under the DDH assumption.*

Proof. Property 1 is satisfied since $q \in \mathbb{Z}_{p,1}^{1\times n} \times \mathbb{Z}_{p,2}^{1\times n} \times ... \times \mathbb{Z}_{p,M}^{1\times n}$, which can be constructed in polynomial time by drawing M elements of $\mathbb{Z}_p^{1\times n}$. For property 3, each coordinate s_i of the function index s is computed as $s_i = L_{g^{abo}}(1^\kappa, q_i)$ in polynomial time. Since this is done M times, the total time of L_{g^M} is polynomial. For property 4, to generate an element of B°, one has to sample elements of $\mathbb{Z}_p^{1\times n}$, which cannot be done in polynomial time. To prove property 3, we construct experiments $H_1, \ldots, H_{M+1}$. Let $q_a, q_b \in \mathbb{Z}_p^{(1\times n)M}$ with $q_a \neq q_b$ represent two ordered tuples. Denote $q_a = (q_{1,1}, q_{1,2}, \ldots, q_{1,M})$ and $q_b = (q_{2,1}, q_{2,2}, \ldots, q_{2,M})$. Experiment H_0 computes $s \leftarrow L_{g^M}(1^\kappa, q_a)$. For $i' \in [M+1]$, experiment $H_{i'}$ constructs q as follows:

$$q = (q_{2,1}, q_{2,2}, q_{2,3}, \ldots, q_{2,i'-1}, q_{1,i'}, q_{1,i'+1}, \ldots, q_{1,M})$$

Suppose that some polynomial time adversary can distinguish between $H_{i'}$ and $H_{i'+1}$ for any $i' \in [M-1]$. Using this adversary, we construct a simulator that breaks property 3 of the ABO collection given by $(L_{g^{abo}}, L_{e^{abo}})$. The simulator has access to an oracle that, on input (b_0°, b_1°), provides it with either $s \leftarrow L_{g^{abo}}(1^\kappa, b_0^\circ)$ or $s \leftarrow L_{g^{abo}}(1^\kappa, b_1^\circ)$. For $i' \in [M-1]$, let q_a, q_b be fixed with $q_a \neq q_b$ and where $q_a = (q_{1,1}, q_{1,2}, \ldots, q_{1,M})$ and

$q_b = (q_{2,1}, q_{2,2}, \ldots, q_{2,M})$. Given $i' \in [M-1]$, the simulator constructs q^0 and q^1 as follows during initialization:

$$q^0 = (q_{2,1}, q_{2,2}, q_{2,3}, \ldots, q_{2,i'-1}, q_{1,i'}, q_{1,i'+1}, \ldots, q_{1,M})$$
$$q^1 = (q_{2,1}, q_{2,2}, q_{2,3}, \ldots, q_{2,i'-1}, q_{2,i'}, q_{1,i'+1}, \ldots, q_{1,M})$$

For $i \in [M] \setminus i'$, the simulator computes $s_i \leftarrow L_{g^{abo}}(1^\kappa, q^0)$, since q^0 and q^1 are equal for all coordinates not equal to i'. For $s_{i'}$, the simulator forwards $(q_{1,i'}, q_{2,i'})$ to its oracle and it receives $s_{i'}$. The simulator then forms $s = (s_1, s_2, \ldots, s_M)$ and forwards it to its adversary. Once the adversary outputs a guess $a' \in \{0,1\}$, the simulator outputs a'. The advantage of the simulator is thus equal to the probability that the adversary outputs $a' = 0$ and the oracle computes $s_i \leftarrow L_{g^{abo}}(1^\kappa, q^0)$. However, given that the ABO collection given by $(L_{g^{abo}}, L_{e^{abo}})$ satisfies the required properties, no efficient adversary would be able to output a' correctly with non-negligible probability. It follows that the advantage of the simulator is likewise negligible. By construction, if the oracle computes $s_{i'} \leftarrow L_{g^{abo}}(1^\kappa, q_{1,i'})$, the simulator is performing experiment $H_{i'}$, while if the oracle computes $s_{i'} \leftarrow L_{g^{abo}}(1^\kappa, q_{2,i'})$, the simulator is performing experiment $H_{i'+1}$ for $i \in [M-1]$. Experiment H_0 computes $s \leftarrow L_{g^M}(1^\kappa, q_a)$, while H_{M+1} computes $s \leftarrow L_{g^M}(1^\kappa, q_b)$. This proves property 3.

We now show that (L_{g^M}, L_{e^M}) satisfy the lossiness property. Let $s \leftarrow L_{g^M}(1^\kappa, q)$ for some q. We note that to efficiently sample a lossy branch, simply pick any ith coordinate q_i of q. Let $\boldsymbol{\pi}_j$ denote the jth coordinate of $\boldsymbol{\pi}$, where $\boldsymbol{\pi}$ is the output of $L_{e^M}(s, q_i, k)$ for some $k \in \mathbb{Z}_p^{1\times n}$. For $j \in [M]$ with $i \neq j$, we have $\boldsymbol{\pi}_j = L_{e^{abo}}(s_j, q_i, k)$, as injective since $q_i \neq q_j$. For $j = i$, we have $\boldsymbol{\pi}_j = L_{e^{abo}}(s_i, q_i, k)$, which is lossy given that $s_i \leftarrow L_{g^{abo}}(1^\kappa, q_i)$, i.e., s_i is computed with q_i as the lossy branch. It follows that $\boldsymbol{\pi}_i$ has, at most, p possible values. Transferring from the $\mathbb{Z}_p$ domain to the binary domain, let $\boldsymbol{\pi}_j$, with $i \neq j$ have $2^{n'}$ possible values for some $n' \in \mathbb{N}$. It follows that $\boldsymbol{\pi}$ can have, at most, $2^{n'(M-1)+\log(p)}$ possible values. This proves the lossiness property. $\square$

7. Conclusions and Future Work

In this paper, we proposed cryptosystems that seek to address the problem of constructing CCA2 secure public-key cryptosystems that are resilient against related randomness attacks and related key attacks, albeit under the more limited classes of randomness reset attacks and constant-bit secret-key leakage attacks. Under this security notion, attacks from both the encryption and decryption processes of a cryptosystem are considered. We formally define this security notion in terms of an attack game between a challenger and an adversary, where the adversary has access to encryption queries with randomness reset and secret-key leakage queries aside from the standard challenge queries and decryption queries in CCA2 security. Under this attack game, we have presented two cryptosystems that are provably secure, where the first cryptosystem relies on the random oracle assumption, while the second cryptosystem is a standard model that relies on a proposed primitive called L^M lossy functions, which can provide up to M lossy branches. In particular, the second cryptosystem uses the loss of information provided by multiple lossy branches to render it secure under a bounded number of encryption and challenge queries. While the L^M collection exhibits a higher degree of information in the lossy branch and requires more memory than other lossy functions, it is easy to construct, as it uses Cartesian product operations over ABO lossy function primitives.

For future work, stronger public-key cryptosystems could, perhaps, be developed such that they are secure against more general types of related randomness attacks or related key attacks. For instance, instead of a randomness reset, a randomness attack may involve linear or polynomial functions of the randomness. The same can be said of secret-key leakage attacks, wherein, instead of the leakage of a constant number of bits, the leakage could be arbitrary functions of the secret key. In addition, further analysis can be done on the efficiency and experimental performance of the algorithm. In particular, the following can be explored: (1) to experimentally assess the efficiency/speed of cryptosystem 1 using an appropriate hash function as a simulated random oracle, and to compare this with cryptosystem 2 using the LM hash function and with other algorithms; (2) to experimentally assess the memory complexity of the LM lossy function relative to the other hash functions in the literature; (3) to provide a discussion on the time and space complexity of our algorithm relative to other algorithms; and (4) to perform an experimental simulation of a randomness reset attack and constant secret-key leakage attack, and demonstrate the algorithm's response and resiliency to these attacks

Author Contributions: Conceptualization, A.L.; formal analysis, A.L. and H.A.; writing—review and editing, H.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Engineering Research and Development Technology (ERDT) program of the Department of Science and Technology (DOST), Philippines.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Koç, Ç.K.; Özdemir, F.; Ödemiş Özger, Z. Rivest-Shamir-Adleman Algorithm. In *Partially Homomorphic Encryption*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 37–41.
2. Boneh, D.; Shoup, V. A graduate Course in Applied Cryptography. 2020. Available online: https://toc.cryptobook.us/book.pdf (accessed on 23 December 2021).
3. Cramer, R.; Shoup, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 13–25.
4. Bellare, M.; Rogaway, P. Random oracles are practical: A paradigm for designing efficient protocols. In Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, VA, USA, 3–5 November 1993; pp. 62–73.
5. Cramer, R.; Shoup, V. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, 28 April–2 May 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 45–64.
6. Lucks, S. A variant of the Cramer-Shoup cryptosystem for groups of unknown order. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, 1–5 December 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 27–45.
7. Lafourcade, P.; Robert, L.; Sow, D. Fast Cramer-Shoup Cryptosystem. In Proceedings of the 18th International Conference on Security and Cryptography SECRYPT 2021, Lieusant, Paris, 6–8 July 2021.
8. Peikert, C.; Waters, B. Lossy trapdoor functions and their applications. *SIAM J. Comput.* **2011**, *40*, 1803–1844. [CrossRef]
9. Naor, M.; Segev, G. Public-key cryptosystems resilient to key leakage. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 18–35.
10. Bellare, M.; Brakerski, Z.; Naor, M.; Ristenpart, T.; Segev, G.; Shacham, H.; Yilek, S. Hedged public-key encryption: How to protect against bad randomness. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, 6–10 December 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 232–249.
11. Paterson, K.G.; Schuldt, J.C.; Sibborn, D.L. Related randomness attacks for public key encryption. In Proceedings of the International Workshop on Public Key Cryptography, Buenos Aires, Argentina, 26–28 March 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 465–482.
12. Faonio, A.; Venturi, D. Efficient public-key cryptography with bounded leakage and tamper resilience. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, 4–8 December 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 877–907.
13. Qin, B.; Liu, S. Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, 1–5 December 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 381–400.
14. Marton, K.; Suciu, A.; Ignat, I. Randomness in digital cryptography: A survey. *Rom. J. Inf. Sci. Technol.* **2010**, *13*, 219–240.
15. Yuen, T.H.; Zhang, C.; Chow, S.S.; Yiu, S.M. Related randomness attacks for public key cryptosystems. In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, Singapore, 14–17 April 2015; pp. 215–223.
16. Yilek, S. Resettable public-key encryption: How to encrypt on a virtual machine. In Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 1–5 March 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 41–56.

17. Garfinkel, T.; Rosenblum, M. When Virtual Is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments. In *HotOS*; USENIX: Berkeley, CA, USA, 2005.
18. Ristenpart, T.; Yilek, S. When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography. In *NDSS*; Internet Society: Reston, VA, USA, 2010.
19. Bellare, M.; Rogaway, P. Code-Based Game-Playing Proofs and the Security of Triple Encryption. *IACR Cryptol. ePrint Arch.* **2004**, *2004*, 331.
20. Austrin, P.; Chung, K.M.; Mahmoody, M.; Pass, R.; Seth, K. On the impossibility of cryptography with tamperable randomness. *Algorithmica* **2017**, *79*, 1052–1101. [CrossRef]
21. Tiri, K. Side-channel attack pitfalls. In Proceedings of the 2007 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 15–20.
22. Nisan, N.; Ta-Shma, A. Extracting randomness: A survey and new constructions. *J. Comput. Syst. Sci.* **1999**, *58*, 148–173. [CrossRef]
23. Dodis, Y.; Vaikuntanathan, V.; Wichs, D. Extracting Randomness from Extractor-Dependent Sources. *Advances in Cryptology–EUROCRYPT 2020.* 2020; pp. 313–342. Available online: https://par.nsf.gov/servlets/purl/10165162 (accessed on 23 December 2021).
24. Dodis, Y.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 523–540.
25. Håstad, J.; Impagliazzo, R.; Levin, L.A.; Luby, M. A pseudorandom generator from any one-way function. *SIAM J. Comput.* **1999**, *28*, 1364–1396. [CrossRef]
26. Hemenway, B.; Libert, B.; Ostrovsky, R.; Vergnaud, D. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Seoul, Korea, 4–8 December 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 70–88.
27. Hofheinz, D. All-but-many lossy trapdoor functions. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, 15–19 April 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 209–227.
28. Canetti, R.; Goldwasser, S. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 90–106.
29. Wee, H. Public key encryption against related key attacks. In Proceedings of the International Workshop on Public Key Cryptography, Darmstadt, Germany, 21–23 May 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 262–279.
30. Paterson, K.G.; Schuldt, J.C.; Sibborn, D.L.; Wee, H. Security against related randomness attacks via reconstructive extractors. In Proceedings of the IMA International Conference on Cryptography and Coding, Oxford, UK, 15–17 December 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 23–40.
31. Boneh, D.; Boyen, X.; Shacham, H. Short group signatures. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 41–55.