Publicly Evaluable Pseudorandom Functions and Their Applications

Yu Chen[*]
yuchen.prc@gmail.com

Zongyang Zhang[†]
zongyang.zhang@gmail.com

April 22, 2014

Abstract

We put forth the notion of *publicly evaluable* pseudorandom functions (PEPRFs), which can be viewed as a non-trivial extension of the standard pseudorandom functions (PRFs). Briefly, PEPRFs are defined over domain X where there exists an average-case hard NP language L, and each secret key sk is associated with a public key pk. For any $x \in L$, in addition to evaluate $\mathsf{F}_{sk}(x)$ using sk as in the standard PRFs, one is also able to evaluate $\mathsf{F}_{sk}(x)$ with pk, x and a witness w for $x \in L$. We consider two security notions for PEPRFs. The basic one is weak pseudorandomness which stipulates PEPRF can not be distinguished from a uniform random function only at randomly chosen inputs. The strengthened one is adaptive weak pseudorandomness which requires PEPRF remains weak pseudorandom even when the adversary is given adaptive access to an evaluation oracle. We conduct a formal study of PEPRFs, focusing on applications, constructions, and extensions.

- We show how to construct chosen-plaintext secure (CPA) and chosen-ciphertext secure (CCA) public-key encryption (PKE) from (adaptive) PEPRFs. The construction is simple, black-box, and admits a direct proof of security. We provide evidence that (adaptive) PEPRFs exist by showing the constructions from both hash proof system and extractable hash proof system.
- We introduce the notion of publicly samplable PRFs (PSPRFs), which is a relaxation of PEPRFs, but nonetheless imply PKE. We show (adaptive) PSPRFs are implied by (adaptive) trapdoor relations, yet the latter are further implied by (adaptive) trapdoor functions. This helps us to unify and clarify many PKE schemes from different paradigms and general assumptions under the notion of PSPRFs. We also view adaptive PSPRFs as a candidate of the weakest general assumption for CCA-secure PKE.
- We explore similar extension on recently emerging predicate PRFs, putting forth the notion of publicly evaluable predicate PRFs, which, as an immediate application, imply predicate encryption.
- We propose a variant of PEPRFs, which we call publicly evaluable and verifiable functions (PEVFs). Compared to PEPRFs, PEVFs have an addition promising property named public verifiability at the cost of the best possible security inherently degrades to hard to compute on average. We justify the applicability of PEVFs by presenting a simple construction of "hash-and-sign" signatures, both in the random oracle model and standard model.

[*]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China.

[†]Research Institute for Secure Systems, National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan.

1 Introduction

Pseudorandom functions (PRFs) [GGM86] are a fundamental concept in modern cryptography. Loosely speaking, PRFs are a family of keyed functions $F = \{\mathsf{F}_{sk} : X \to Y\}_{sk \in SK}$ such that: (1) it is easy to sample the functions and compute their values, i.e., given the secret key (or seed) sk, one can efficiently evaluate $\mathsf{F}_{sk}(x)$ at all points $x \in X$; (2) given only black-box access to the function no probabilistic polynomial-time (PPT) algorithm can distinguish F_{sk} for a randomly chosen sk from a real random function, or equivalently, without sk no PPT algorithm can distinguish $\mathsf{F}_{sk}(x)$ from random at all points $x \in X$.

In this work, we extend the standard PRFs to what we call *publicly evaluable PRFs*, which partially fill the gap between the evaluation power with and without secret key. In a publicly evaluable PRF, there exists a hard on average NP language $L \subseteq X$ and each secret key sk is associated with a public key pk. In addition, for any $x \in L$, except via private evaluation with sk, one can also efficiently compute the value of $\mathsf{F}_{sk}(x)$ via public evaluation with the associated public key pk and a witness w for $x \in L$. Considering the security requirement for PEPRFs, we expect weak pseudorandomness which ensures that no PPT adversary can distinguish F_{sk} from a real random function on uniformly distributed challenge points (this differs from the standard pseudorandomness for PRFs in which the challenge points are adversarially chosen by the adversary). We defer the precise definitions in Section 3.

While PEPRFs are a conceptually simple extension of the standard PRFs, they have surprisingly powerful applications beyond what is possible with standard PRFs. Most notably, as we will see shortly, they admit a simple and black-box construction of PKE.

1.1 Motivation

PRFs have a wide range of applications in cryptography. Perhaps the most typical application is a simple and elegant construction of private-key encryption as follows: the secret key sk of PRF serves as the private key, to encrypt a message m, the sender first chooses a random $x \in X$, and then outputs ciphertext $(x, m \oplus \mathsf{F}_{sk}(x))$. It is tempting to think if PRF might also yield public-key encryption in the same way. However, such construction fails in the public-key setting when F is a standard PRF. This is because without sk no PPT algorithm can evaluate $\mathsf{F}_{sk}(x)$ (otherwise this will go against the pseudorandomness of PRF) and thus encrypting publicly is impossible. Moreover, since PRFs and one-way functions (OWFs) imply each other, the implications of PRFs are inherently confined in Minicrypt [IR89]. This result rules out the possibilities of constructing PKE from PRFs in a black-box manner.

Meanwhile, most existing PKE schemes based on various concrete hardness assumptions can be casted into several existing paradigms or general assumptions in the literature. In details, hash proof systems [CS02] encompass the PKE schemes [CS98, CS03, KD04, KPSY09], extractable hash proof systems [Wee10] encompass the PKE schemes [BMW05, Kil06, CKS09, HK09, HJKS10], one-way trapdoor permutations/functions encompass the PKE schemes [RSA78, Rab81, PW08, RS10].[1] However, the celebrated ElGamal encryption [ElG85] does not fit into any known paradigms or general assumptions. Motivated by the above discussion, we find the following intriguing question:

> *What kind of extension on PRFs can also make the above construction in private-key setting also work in public-key setting? Can it be used to explain unclassified PKE schemes? Can it yield CCA-secure PKE schemes?*

[1] The references [RSA78, Rab81] actually refer to the padded version of RSA encryption and Rabin encryption.

1.2 Our Contributions

We give positive answers to the above questions. Our main results (summarized in Figure 1) are as follows:

- In Section 3, we introduce the notion of publicly evaluable PRFs (PEPRFs), which contains several conceptually extensions of standard PRFs. In PEPRF, there is a hard-on-average NP language L over domain X and each secret key sk is associated with a public key pk. Moreover, for any $x \in L$, except via private evaluation with sk, one can efficiently evaluate $\mathsf{F}_{sk}(x)$ using pk and a witness w for $x \in L$. We also formalize security notions for PEPRFs, namely weak pseudorandomness and adaptive weak pseudorandomness.
- In Section 4, we demonstrate the power of PEPRFs by showing that they enable the construction of private-key encryption to work in the public-key setting, following the KEM-DEM methodology. In sketch, the public/secret key for PEPRF serves as the public/secret key for PKE. To encrypt a message m, the sender first samples a random $x \in L$ with witness w, then publicly evaluates $\mathsf{F}_{sk}(x)$ from pk, x and w, and outputs ciphertext $(x, m \oplus \mathsf{F}_{sk}(x))$. To decrypt, the receiver simply uses sk to compute $\mathsf{F}_{sk}(x)$ privately, then recovers m. Such construction is simple, black-box, and admits a direct proof of security.[2] In particular, in Example 3.1 we show that the celebrated ElGamal PKE can be explained neatly by a weak pseudorandom PEPRF based on the Diffie-Hellman assumption. Interestingly, the above KEM construction from PEPRFs is somewhat dual to that from trapdoor functions (TDFs). In the construction from TDFs, the sender first produces the DEM key by picking $x \xleftarrow{\text{R}} X$[3], then generates the associated ciphertext $\mathsf{TDF}_{ek}(x)$; while in the construction from PEPRFs, the sender first generates the ciphertext by picking $x \xleftarrow{\text{R}} X$, then produces the associated DEM key $\mathsf{F}_{sk}(x)$.
- In Section 5 and Section 6, as our main result, we show that both smooth hash proof system (HPS) and extractable hash proof system (EHPS) yield weak pseudorandom PEPRF, while both smooth + weakly universal$_2$ HPSs and all-but-one EHPS yield adaptively weak pseudorandom PEPRF, respectively. This means that the works on HPS and EHPS were implicitly constructing PEPRFs, and the latter abstracts out a common aspect of the former two paradigms not formalized before. The existing constructions of HPS and EHPS imply that PEPRFs are achievable under a variety of number-theoretic assumptions.
- In Section 7 we introduce the notion of publicly samplable PRFs (PSPRFs), which is a relaxation of PEPRFs, but nonetheless imply PKE. Of independent interest, we redefine the notion of trapdoor relations (TDRs). We show that injective trapdoor functions (TDFs) imply "one-to-one" TDRs, while the latter further imply PSPRFs. Compared to PEPRFs, PSPRFs help us to unify and clarify more PKE schemes based on different paradigms and general assumptions from a conceptual standpoint, and also suggests adaptive PSPRFs as a candidate of the weakest general assumption for CCA-secure PKE.
- In Section A, we put forward an extension of PSPRFs, named publicly evaluable predicate PRFs. An immediate application of publicly evaluable predicate PRFs is predicate encryption. We present a concrete construction based on recent attribute-based encryption from multilinear maps [GGH+13].
- In Section B, we propose a variant of PEPRFs named publicly evaluable and verifiable

[2] For simplicity, we treat PKE schemes as key encapsulation mechanisms (KEM) in this work. It is well known that one can generically obtain a fully fledged CCA-secure PKE by combining a CCA-secure KEM (the requirement on KEM could be weaker [HK07]) and a data encapsulation mechanism (DEM) with appropriate security properties [CS03, KD04, KV08].

[3] To obtain semantic security, one should use $\mathsf{hc}(x)$ instead of x as the DEM key, where hc is a hardcore predicate for the TDF.

functions (PEVFs). We prove the usefulness of PEVFs by presenting a simple construction of "hash-and-sign" signatures, both in the random oracle model and the standard model.

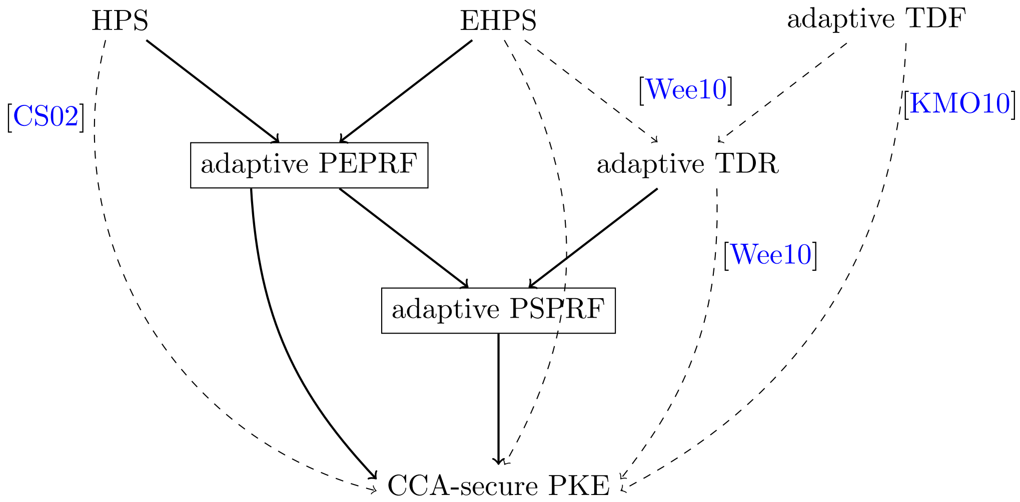

Figure 1: Summary of CCA-secure PKEs from paradigms and general assumptions. Here, HPS refers to smooth + universal$_2$ HPSs, EHPS refers to its all-but-one variant. $\rightarrow$ is an implication while $\not\rightarrow$ is a black-box separation. The bold lines and rectangles denote our contributions, while the dashed lines denote those that are straight-forward or from previous work. All of the constructions from general assumptions are black-box.

1.3 Related Work

General assumption vs. Paradigm. We first try to explain the terms of "general assumption" and "paradigm" in the context of cryptography. Roughly speaking, general assumption (also referred to generic cryptographic assumption or primitive) is usually given by one set of algorithms, which is used to fulfill some functionality and is expected to satisfy some desired security. Examples of general assumption include (trapdoor) one-way functions/permutations and pseudorandom generators/functions, etc. Different from general assumption, paradigm is usually given by two sets of algorithms, the first set of algorithms is used to fulfill some functionality, while the second set of algorithms is used to argue the first one satisfies some desired security. Examples of paradigm include hash proof system, extractable hash proof system, and lossy trapdoor hash functions, etc. Both general assumption and paradigm are highly abstracted objects since they are not tied to any specific "hard" problem. The distinguished feature between them is that the former does not specify how to attain the desired security, while the latter explicitly provide a template to establish the desired security. In light of this difference, general assumption is more abstract than paradigm.

CCA-secure PKE from general assumption or paradigm. Except the effort on constructing CCA-secure PKE from specific assumptions [HK08, MH14] or encryption schemes satisfying some weak security notions [NY90, DDN00, BCHK07, CHK10, HLW12, DS13], it is of particular theoretical interest of building CCA-secure PKE from general assumption and paradigm. Cramer and Shoup [CS02] generalized their CCA-secure PKE construction [CS98] to hash proof system (HPS) and used it as a paradigm to construct CCA-secure PKE from various decisional assumptions. Peikert and Waters [PW08] proposed lossy trapdoor functions (LTDFs) and showed a black-box construction of CCA-secure PKE from it. Rosen and Segev [RS10] put forwarded correlated-product secure trapdoor functions (CP-TDFs) and also showed a construction of CCA-secure PKE from it. Moreover, they showed that CP-TDF is strictly weaker than LTDF by giving a black-box separation between them. Kiltz et al. [KMO10] introduced (injective) adaptive trapdoor functions (ATDFs) which is strictly weaker than both LTDFs and CP-TDFs but suffices to imply CCA-secure PKE. Wee [Wee10] introduced extractable hash proof systems (EHPS) and used it as a paradigm to construct CCA-secure PKE from various search assumptions. Wee also showed that both EHPS and ATDF imply (injective) adaptive trapdoor relations (ATDRs), where is sufficient to imply CCA-secure PKE. To the best of our knowledge, so far ATDR is the weakest general assumption that implies CCA-secure PKE. Very recently, Sahai and Waters [SW13] successfully translated the private-key encryption from PRFs to PKE (with delicate modifications) by using punctured program technique in conjunction with indistinguishability obfuscation ($i\mathcal{O}$). Because the use of obfuscation, their construction is inherently non-black-box.

Predicate PRFs. Very recently, predicate PRFs are studied in three concurrent and independent works, by Kiayias et al. [KPTZ13] under the name of *delegatable PRFs*, by Boneh and Waters [BW13] under the name of *constrained PRFs*, and by Boyle, Goldwasser, and Ivan [BGI13] under the name of *functional PRFs*. In predicate PRFs, one can derive a secret key sk_p for a predicate p from the master secret key msk. A secret key sk_p enables one to evaluate $\mathsf{F}_{msk}(x)$ at points x such that $p(x) = 1$. This natural extension turns out to be useful since it has powerful applications out of the scope of standard PRFs, such as identity-based key exchange, optimal private broadcast encryption.

2 Preliminaries and Definitions

Notations. For a distribution or random variable X, we write $x \xleftarrow{\text{R}} X$ to denote the operation of sampling a random x according to X. For a set X, we use $x \xleftarrow{\text{R}} X$ to denote the operation of sampling x uniformly at random from X, and use $|X|$ to denote its size. We write κ to denote the security parameter through this paper, and all algorithms (including the adversary) are implicitly given κ as input. We write $\mathsf{poly}(\kappa)$ to denote an arbitrary polynomial function in κ. We write $\mathsf{negl}(\kappa)$ to denote an arbitrary negligible function in κ, one that vanishes faster than the inverse of any polynomial. We say a probability is overwhelming if it is $1 - \mathsf{negl}(\kappa)$, and said to be noticeable if it is $1/\mathsf{poly}(\kappa)$. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm that runs in time $\mathsf{poly}(\kappa)$. If $\mathcal{A}$ is a randomized algorithm, we write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n; r)$ to indicate that $\mathcal{A}$ outputs z on inputs $(x_1, \ldots, x_n)$ and random coins r. We will omit r and write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n)$. For distributions X, Y, we write $X \approx_s Y$ to mean that they are statistically indistinguishable.

2.1 Pseudorandom Functions

Definition 2.1 (PRFs [GGM86])**.** Pseudorandom functions consists of three polynomial-time algorithms as follows:

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, output public parameters pp which includes finite sets SK, X, Y (these sets may be parameterized by κ), and a PRF family $F = \{\mathsf{F}_{sk} : X \to Y\}_{sk \in SK}$.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key $sk \in SK$.
- $\mathsf{PrivEval}(sk, x)$: on input secret key sk and an element $x \in X$, output $y \in Y$. This algorithm is usually deterministic.

Correctness: For any $pp \leftarrow \mathsf{Setup}(\kappa)$, any $sk \leftarrow \mathsf{KeyGen}(pp)$, and any $x \in X$, it holds that:

$$\mathsf{F}_{sk}(x) = \mathsf{PrivEval}(sk, x).$$

Security: The standard security requirement for PRF is pseudorandomness. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against PRF and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ sk \leftarrow \mathsf{KeyGen}(pp); \\ state \leftarrow \mathcal{A}_1(pp); \\ b \leftarrow \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{ror}}(b,\cdot)}(state); \end{array} \right] - \frac{1}{2},$$

where $\mathcal{O}_{\mathsf{ror}}(0, x) = \mathsf{F}_{sk}(x)$, $\mathcal{O}_{\mathsf{ror}}(1, x) = \mathsf{H}(x)$, and H chosen uniformly at random from all the functions from X to Y.[4] Note that $\mathcal{A}$ can access oracle $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ adaptively polynomial times. We say that PRFs are pseudorandom if for any PPT adversary its advantage function $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ. We refer to such security as full PRF security.

Sometimes the full PRF security is not needed and it is sufficient if the function cannot be distinguished from a uniform random one when challenged on random inputs. The formalization of such relaxed requirement is weak pseudorandomness (weak PRF security), which is defined as the full PRF security except that the inputs of oracle $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ is uniformly chosen from X by the challenger instead of adversarilly chosen by $\mathcal{A}$. PRFs that satisfy weak pseudorandomness are referred to as weak PRFs. In the remainder of this paper, we will focus on weak PRFs.

2.2 Secret-Coin vs. Public-Coin PRFs

To obtain more refined security notions for weak PRFs, it is convenient make the input sampling algorithm explicit. Hereafter, let Sample be an algorithm that takes as input a random coins r and outputs an element $x \in X$. Without lose of generality, we assume the distribution $x \leftarrow \mathsf{Sample}(r)$ conditioned on $r \xleftarrow{\text{R}} R$ is statistically close to $x \xleftarrow{\text{R}} X$. Hence, in the weak PRF security experiment the challenger can sample elements uniformly at random from X by running Sample with random coins $r \xleftarrow{\text{R}} R$. Depending on if the random coins r can be made public, weak PRFs can be further divided into two categories [PS08], namely secret-coin and public-coin PRFs. Secret-coin PRFs require weak pseudorandomness holds if and only if the random coins are kept secret.[5] Opposed to secret-coin PRFs, public-coin PRFs requires the weak pseudorandomness holds even the random coins is made public. Clearly, secret-coin PRFs and public-coin PRFs constitute a partition of weak PRFs.

3 Publicly Evaluable PRFs

We now give a precise definition of PEPRFs. We begin with the syntax and then define the security.

Definition 3.1 (Publicly Evaluable PRFs)**.** PEPRFs consist of five polynomial-time algorithms as below:

[4] To efficiently simulate access to a uniformly random function H from X to Y, one may think of a process in which the adversary's queries to $\mathcal{O}_{\mathsf{ror}}(1, \cdot)$ are "lazily" answered with independently and randomly chosen elements in Y, while keeping track of the answers so that queries made more than once are answered consistently.

[5] In [PS08], secret-coin PRFs are the PRFs that satisfy weak pseudorandomness if the random coins are kept secret, which is exactly the standard weak PRFs. In this paper, we reserve the term of secret-coin PRFs for the PRFs that satisfy weak pseudorandomness if and only if the random coins is kept secret.

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp which includes finite sets SK, PK, X, Y, a language $L \subseteq X$ (these sets may be parameterized by κ) and a witness set W, as well as a PEPRF family $F = \{\mathsf{F}_{sk} : X \to Y \cup \perp\}_{sk \in SK}$.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key sk and an associated public key pk.[6]
- $\mathsf{Sample}(r)$: on input random coins r, output a random $x \in L$ along with a witness $w \in W$ for x.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, output $y \in Y$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \in Y \cup \perp$.

Correctness: For any $pp \leftarrow \mathsf{Setup}(\kappa)$ and any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, it holds that:

$$\begin{aligned} \forall x \in X : \quad & \mathsf{F}_{sk}(x) = \mathsf{PrivEval}(sk, x) \\ \forall x \in L \text{ with witness } w : \quad & \mathsf{F}_{sk}(x) = \mathsf{PubEval}(pk, x, w) \end{aligned}$$

Security: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against PEPRFs and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{eval}}(\cdot)}(pp, pk); \\ \{(x_i^*, w_i^*) \leftarrow \mathsf{Sample}(r_i^*), r_i^* \xleftarrow{\text{R}} R\}_{i=1}^{p(\kappa)}; \\ b \leftarrow \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{eval}}(\cdot)}(state, \{x_i^*, \mathcal{O}_{\mathsf{ror}}(b, x_i^*)\}_{i=1}^{p(\kappa)}); \end{array} \right] - \frac{1}{2},$$

where $p(\kappa)$ is any polynomial, $\mathcal{O}_{\mathsf{ror}}(0, x) = \mathsf{F}_{sk}(x)$, $\mathcal{O}_{\mathsf{ror}}(1, x) = \mathsf{H}(x)$ (here H is chosen uniformly at random from all functions from X to Y, $\mathcal{O}_{\mathsf{eval}}(x) = \mathsf{F}_{sk}(x)$), and $\mathcal{A}_2$ is not allowed to query $\mathcal{O}_{\mathsf{eval}}(\cdot)$ with any x_i^*. We say that PEPRFs are adaptively weak pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

The adaptively weak pseudorandomness captures the security against active adversaries, who are given adaptive access to oracle $\mathcal{O}_{\mathsf{eval}}(\cdot)$. We may also consider weak pseudorandomness which captures the security against static adversaries, who is not given access to oracle $\mathcal{O}_{\mathsf{eval}}(\cdot)$.

Remark 3.1. Different from the standard PRFs, PEPRFs require the existence of an NP language $L \subseteq X$ and a public evaluation algorithm. Due to this strengthening on functionality, we can not hope to achieve full PRF security, and hence settling for weak PRF security is a natural choice.[7] Note that the weak PRF security implicitly requires the language L must be hard on average.

Remark 3.2. In some scenarios, it will be more convenient to work with a definition that slightly restricts the adversary's power, but is equivalent to Definition 3.1. That is, $p(\kappa)$ is fixed to 1. Thanks to the existence of oracle $\mathcal{O}_{\mathsf{eval}}(\cdot)$, a standard hybrid argument can show that PEPRFs secure under this restricted definition are also secure under Definition 3.1. In the remainder of this paper, we will work with this restricted definition.

[6] In the standard PRFs, public key is not made implicit. However, it is harmless to explicitly introduce public key for PRFs, which includes the information related to secret key that can be made public. For example, in the Naor-Reingold PRF [NR04] based on the DDH assumption: $\mathsf{F}_{\vec{a}}(x) = (g^{a_0})^{\prod_{x_i=1} a_i}$, $\vec{a} = (a_0, a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ is the secret key, while g^{a_i} for $1 \le i \le n$ can be safely published as the public key. One can always think of $pk = \{\perp\}$ if there is no information can be made public.

[7] In the full PRF security experiment the inputs of $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ are chosen by the adversary, thus it may know the corresponding random coins and then evaluate $\mathsf{F}_{sk}(x^*)$ publicly.

Example 3.1. As a warm up, we present an illustrative construction of PEPRF. Let $\mathbb{G}$ be a cyclic group of prime order p with canonical generator g, then define $\mathsf{F}_{sk} : \mathbb{G} \to \mathbb{G}$ as x^{sk}, where the secret key $sk \in \mathbb{Z}_p$ and the public key $pk = g^{sk} \in \mathbb{G}$. A natural language L defined over $\mathbb{G}$ is $\{x = g^w : w \in \mathbb{Z}_p\}$, where the exponent w serves as a witness for x. For any $x \in L$, one can publicly evaluate $\mathsf{F}_{sk}(x)$ via computing pk^w. It is easy to verify that this PEPRF is weak pseudorandom assuming the DDH assumption holds in $\mathbb{G}$. Looking ahead, when applying the construction shown in Section 4 to this PEPRF, yields exactly the ElGamal PKE.

A needed relaxation. To be completely precise, we do not necessarily require the distribution of $x \leftarrow \mathsf{Sample}(r)$ conditioned on $r \xleftarrow{R} R$ is identical or statistically close to uniform. Instead, it could be some other prescribed distribution χ. In this case, weak pseudorandomness extends naturally to χ-weak pseudorandomness.

A useful generalization. In some scenarios, it is more convenient to work with a more generalized notion in which we consider a collection of languages $\{L_{pk}\}_{pk \in PK}$ indexed by the public key rather than a fixed language L. Correspondingly, the sampling algorithm takes pk as an extra input to sample a random element from L_{pk}. We refer to such generalized PEPRFs as *PEPRFs for public-key dependent languages*, and we will work with it when constructing adaptive PEPRF from hash proof system.

3.1 Relation to Secret-Coin and Public-Coin PRFs

As aforementioned, weak PRFs can be divided into secret-coin and public-coin PRFs. Again, PEPRFs can be viewed as a special case of secret-coin PRFs, in which the adversary can totally learn the PRF value at point x if the corresponding random coins are revealed. We depict their relations in Figure 2. Note that Pietrzak and Sjödin [PS08] demonstrated that secret-coin PRFs must be very artificial in Minicrypt by showing that the their existence implies two pass key-agreement and thus two pass public-key encryption. As we will see shortly, PEPRFs admit a black-box construction of PKE. This fact indicates PEPRFs, as a special case of secret-coin PRFs, are strictly stronger than PRFs (in a black-box sense), which is in line with Pietrzak and Sjödin's result.

Interestingly, secret-coin PRFs can be intuitively constructed from PEPRFs and public-coin PRFs. Suppose $\{\mathsf{G}_{sk_1} : X_1 \to Y_1\}_{sk_1 \in SK_1}$ is a public-coin PRF family and $\{\mathsf{H}_{sk_2} : X_2 \to Y_2\}_{sk_2 \in SK_2}$ is a PEPRF family, then $\{\mathsf{F}_{sk_1, sk_2}(x_1, x_2) := (\mathsf{G}_{sk_1}(x_1), \mathsf{H}_{sk_2}(x_2))\}$ constitutes a secret-coin PRF family from $X_1 \times X_2$ to $Y_1 \times Y_2$ indexed by $SK_1 \times SK_2$.

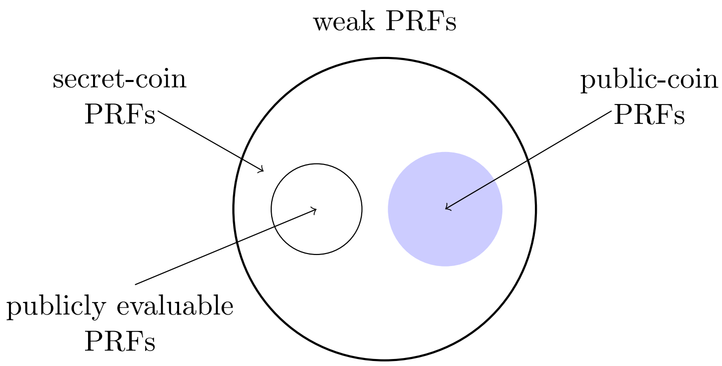

Figure 2: Relations among secret-coin, public-coin, and publicly evaluable PRFs.

4 KEM from Publicly Evaluable PRFs

In this section, we present a simple and black-box construction of KEM from PEPRFs. For compactness we refer the reader to Appendix D.2 for the definition and security notion of KEM.

- $\mathsf{Setup}(\kappa)$: run PRF.$\mathsf{Setup}(\kappa)$ to generate pp, construct mpk from pp.
- $\mathsf{KeyGen}(mpk)$: run PRF.$\mathsf{KeyGen}(pp)$ to generate (pk, sk).
- $\mathsf{Encap}(pk; r)$: run PRF.$\mathsf{Sample}(r)$ to generate a random $x \in L$ with a witness $w \in W$ for x, set x as the ciphertext c and compute PRF.$\mathsf{PubEval}(pk, x, w)$ as the DEM key k, output (c, k).
- $\mathsf{Decap}(sk, c)$: output PRF.$\mathsf{PrivEval}(sk, c)$.

Correctness of the KEM follows directly from that of PEPRFs. For security, we have the following results:

Theorem 4.1. *The KEM is CPA-secure if the underlying PEPRFs are weak pseudorandom.*

Proof. The proof is rather straightforward, which transforms an adversary $\mathcal{A}$ against the IND-CPA security of KEM to a distinguisher $\mathcal{B}$ against the weak pseudorandomness of PEPRFs. We proceed via a single game.

Game CPA: $\mathcal{B}$ receives a challenge instance (pp, pk) of PEPRFs, then simulates $\mathcal{A}$'s challenger in the IND-CPA experiment as follows:
Setup: $\mathcal{B}$ generates mpk from pp, sends (mpk, pk) to $\mathcal{A}$.
Challenge: $\mathcal{B}$ receives (x^*, y^*) from its own challenger, where $(x^*, w^*) \leftarrow$ PRF.$\mathsf{Sample}(r^*)$ and y^* is either $\mathsf{F}_{sk}(x^*)$ or randomly picked from Y. $\mathcal{B}$ sends (x^*, y^*) to $\mathcal{A}$ as the challenge.
Guess: $\mathcal{A}$ outputs its guess b' for b and $\mathcal{B}$ forwards b' to its own challenger.

Clearly, $\mathcal{A}$'s view in the above game is identical to that in the real IND-CPA experiment. Thus $\mathcal{B}$ can break the weak pseudorandomness with advantage at least $\mathsf{Adv}_{\mathcal{A}}^{\text{CPA}}(\kappa)$. This concludes the proof. □

Theorem 4.2. *The KEM is CCA-secure if the PEPRFs are adaptively weak pseudorandom.*

Proof. The proof is also straightforward, which transforms an adversary $\mathcal{A}$ against the IND-CCA security of KEM to a distinguisher $\mathcal{B}$ against the adaptively weak pseudorandomness of PEPRFs. We proceed via a single game.

Game CCA: $\mathcal{B}$ receives a challenge instance (pp, pk) of PEPRFs, then simulates $\mathcal{A}$'s challenger in the IND-CCA experiment as follows:
Setup: $\mathcal{B}$ generates mpk from pp, sends (mpk, pk) to $\mathcal{A}$.
Phase 1 - Decapsulation queries: on decapsulation query $\langle x \rangle$, $\mathcal{B}$ submits evaluation query on point x to its own challenger and forwards the reply to $\mathcal{A}$.
Challenge: $\mathcal{B}$ receives (x^*, y^*) from its own challenger, where $(x^*, w^*) \leftarrow$ PRF.$\mathsf{Sample}(r^*)$ and y^* is either $\mathsf{F}_{sk}(x^*)$ or randomly picked from Y. $\mathcal{B}$ sends (x^*, y^*) to $\mathcal{A}$ as the challenge.
Phase 2 - Decapsulation queries: same as in Phase 1 except that the decapsulation query $\langle x^* \rangle$ is not allowed.
Guess: $\mathcal{A}$ outputs its guess b' for b and $\mathcal{B}$ forwards b' to its own challenger.

Clearly, $\mathcal{A}$'s view in the above game is identical to that in the real IND-CCA experiment. Thus $\mathcal{B}$ can break the adaptively weak pseudorandomness with advantage at least $\mathsf{Adv}_{\mathcal{A}}^{\text{CCA}}(\kappa)$. This concludes the proof. □

The above results also hold if the underlying PEPRF is (adaptively) χ-weak pseudorandom.

5 Connection to Hash Proof System

Hash proof system (HPS) was introduced by Cramer and Shoup [CS02] as a paradigm of constructing PKE from a category of decisional problems, named subset membership problems. As a warm up, we first recall the notion of HPS and then show how to construct PEPRFs from it.

Hash Proof System. HPS consists of the following algorithms:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp which includes an HPS instance description $(H, SK, PK, X, L, W, \Pi, \alpha)$, where $H = \{\mathsf{H}_{sk} : X \to \Pi\}_{sk \in SK}$ is a hash family indexed by SK, L is a language defined over X, W is the associated witness set, and α is a projection from SK to PK.
- $\mathsf{KeyGen}(pp)$: on input pp, pick $sk \xleftarrow{R} SK$, compute $pk \leftarrow \alpha(sk)$, output (pk, sk).
- $\mathsf{Sample}(r)$: on input random coins r, output a random $x \in L$ together with a witness w.
- $\mathsf{Sample}'(r)$: on input random coins r, output a random $x \in X \backslash L$.
- $\mathsf{Priv}(sk, x)$: on input sk and x, output π such that $\pi = \mathsf{H}_{sk}(x)$.
- $\mathsf{Pub}(pk, x, w)$: on input pk and $x \in L$ together with a witness w for x, output π such that $\pi = \mathsf{H}_{sk}(x)$.

HPS satisfies the following basic property:

Definition 5.1 (sampling indistinguishable)**.** The two distributions induced by Sample and Sample' are computationally indistinguishable based on the hardness of the underlying subset membership problem.

The following notions capture a rich set of properties for H on input $x \in X \backslash L$.

Definition 5.2 (smooth)**.** $(pk, \mathsf{H}_{sk}(x)) \approx_s (pk, \pi)$, where $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, $x \leftarrow \mathsf{Sample}'(r)$, and $\pi \xleftarrow{R} \Pi$.

Definition 5.3 (universal$_1$)**.** For any $x \in X \backslash L$, and $\pi \in \Pi$, it holds that:

$$\Pr[\mathsf{H}_{sk}(x) = \pi \mid \alpha(sk) = pk] \leq \epsilon$$

Definition 5.4 (universal$_2$)**.** For any $x^* \in X \backslash L$, any $x \notin L \cup x^*$, and any $\pi \in \Pi$, it holds that:

$$\Pr[\mathsf{H}_{sk}(x) = \pi \mid \mathsf{H}_{sk}(x^*) = \pi^* \wedge \alpha(sk) = pk] \leq \epsilon$$

[CS02] indicated that universal$_2$ property implies universal$_1$ property, while [ADN+10] further showed that universal$_1$ property combining strong randomness extractor implies smoothness. The universal$_2$ property is much stronger than smoothness, since the former is defined for all points $x \in X \backslash L$ non-equal to x^*, where the latter is defined with respect to $x \xleftarrow{R} X \backslash L$. In the designing of CCA-secure PKE, the universal$_2$ property is necessary since the input x^* of universal$_2$ hash might be dependent on the target message choice of adversary. While in the designing of KEM, x^* totally comes from the challenge instance of external decisional problem, therefore, it is possible to weaken the universal$_2$ property. Of independent interest, we formalize weak universal$_2$ property as follows:

Definition 5.5 (weak universal$_2$)**.** For $x^* \leftarrow \mathsf{Sample}'(r)$, any $x \notin L \cup x^*$, and any $\pi \in \Pi$, it holds that:

$$\Pr[\mathsf{H}_{sk}(x) = \pi \mid \mathsf{H}_{sk}(x^*) = \pi^* \wedge \alpha(sk) = pk] \leq \epsilon$$

5.1 Construction from Smooth HPS

From smooth HPS, we construct weak pseudorandom PEPRF as follows:

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, run HPS.$\mathsf{Setup}(\kappa)$ to generate a HPS instance $(H, PK, SK, X, L, \Pi, \alpha)$, then produces public parameters $pp = (F, PK, SK, X, L, Y)$ for PEPRFs from it, where $F = H$, $Y = \Pi$. We assume the public parameters pp of HPS and PEPRFs contain essentially the same information.
- $\mathsf{KeyGen}(pp)$: on input pp, output $(pk, sk) \leftarrow$ HPS.$\mathsf{KeyGen}(pp)$.
- $\mathsf{Sample}(r)$: on input r, output $(x, w) \leftarrow$ HPS.$\mathsf{Sample}(r)$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, output $y \leftarrow$ HPS.$\mathsf{Pub}(pk, x, w)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \leftarrow$ HPS.$\mathsf{Priv}(sk, x)$.

The algorithm HPS.Sample' is not used in the construction, but it is crucial to establish the security. We have the following theorem of the above construction.

Theorem 5.1. *If the underlying subset membership problem is hard, then the PEPRF from smooth HPS is weak pseudorandom.*

We defer the proof of Theorem 5.1 in Appendix C.1.

5.2 Construction from Smooth and Weak Universal$_2$ HPS

From a smooth HPS and an associated weak universal$_2$ HPS, we construct adaptively weak pseudorandom PEPRF as follows:

- $\mathsf{Setup}(\kappa)$: on input κ, run HPS$_1$.$\mathsf{Setup}(\kappa)$ to generate a smooth HPS instance ($\tilde{H}$, PK_1, SK_1, $\tilde{X}$, $\tilde{L}$, W, Π_1, α_1), run HPS$_2$.$\mathsf{Setup}(\kappa)$ to generate a weak universal$_2$ HPS instance ($\hat{H}$, PK_2, SK_2, $\tilde{X}$, $\tilde{L}$, W, Π_2, α_2), then build $pp = (F, PK, SK, X, L, W, Y)$ for PEPRFs from them, where $X = \tilde{X} \times \Pi_2$, $Y = \Pi_1 \cup \perp$, $PK = PK_1 \times PK_2$, $SK = SK_1 \times SK_2$, and L will be defined later. We assume the two HPSs share the common sampling algorithm.
- $\mathsf{KeyGen}(pp)$: on input pp, run HPS$_1$.$\mathsf{KeyGen}(pp)$ and HPS$_2$.$\mathsf{KeyGen}(pp)$ to get (pk_1, sk_1) and (pk_2, sk_2) respectively, output $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$. We assume the public parameters pp of HPS and PEPRFs contain the same information.
- $\mathsf{Sample}(pk; r)$: on input $pk = (pk_1, pk_2)$ and random coins r, pick $(\tilde{x}, w) \leftarrow$ HPS$_1$.$\mathsf{Sample}(r)$, compute $\pi_2 \leftarrow$ HPS$_2$.$\mathsf{Pub}(pk_2, \tilde{x}, w)$. This sampling algorithm defines a collection of language $L = \{L_{pk}\}_{pk \in PK}$ over $X = \tilde{X} \times \Pi_2$ where each $L_{pk} = \{(\tilde{x}, \pi_2) : \tilde{x} \in \tilde{L} \wedge \pi_2 = \text{HPS}_2(pk_2, \tilde{x}, w)\}$. Note that a witness w for $\tilde{x} \in \tilde{L}$ is also a witness for $x = (\tilde{x}, \pi_2) \in L_{pk}$.
- $\mathsf{PubEval}(pk, x, w)$: on input $pk = (pk_1, pk_2)$ and an element $x = (\tilde{x}, \pi_2) \in L_{pk}$ together with a witness w, output $y \leftarrow$ HPS$_1$.$\mathsf{Pub}(pk_1, \tilde{x}, w)$.
- $\mathsf{PrivEval}(sk, x)$: on input $sk = (sk_1, sk_2)$ and $x = (\tilde{x}, \pi_2)$, output $y \leftarrow$ HPS$_1$.$\mathsf{Priv}(sk_1, \tilde{x})$ if $\pi_2 =$ HPS$_2$.$\mathsf{Priv}(sk_2, \tilde{x})$ and $\perp$ otherwise.

We have the following theorem of the above construction.

Theorem 5.2. *If the underlying subset membership problem is hard, then the PEPRF from smooth and weak universal$_2$ HPSs is adaptively weak pseudorandom.*

We defer the proof of Theorem 5.2 in Appendix C.2.

6 Connection to Extractable Hash Proof System

Extractable hash proof system (EHPS) was introduced by Wee [Wee10] as a paradigm of constructing PKE from search problems. In the following, we recall the notion of EHPS and then show how to construct PEPRF from it.

EXTRACTABLE HASH PROOF SYSTEM. EHPS consists of a tuple of algorithms (Setup, KeyGen, KeyGen', Pub, Priv, Ext) as below:

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, output public parameter pp which includes an EHPS instance description (H, PK, SK, S, U, Π), where H is a hash family mapping U to Π indexed by PK. Let $\mathsf{hc}(\cdot) : S \to \{0,1\}^l$ be a hardcore function for one-way binary relation R over $S \times U$.
- $\mathsf{KeyGen}(pp)$: on input public parameter pp, output a key pair (pk, sk).
- $\mathsf{KeyGen}'(pp)$: on input public parameter pp, output a key pair (pk, sk').
- $\mathsf{Sample}(r)$: on input random coins r, output a random tuple $(s, u) \in \mathsf{R}$, where s can be viewed as pre-image of u. For our purpose, we further decompose algorithm Sample to $\mathsf{SampLeft}$ and $\mathsf{SampRight}$. The former on input random coins r outputs $s \in S$, while the latter on input random coins r outputs $u \in U$. For all $r \in R$, we require that $(\mathsf{SampLeft}(r), \mathsf{SampRight}(r)) \in \mathsf{R}$.
- $\mathsf{Pub}(pk, r)$: on input pk and r, output $\pi = \mathsf{H}_{pk}(u)$ where $u = \mathsf{SampRight}(r)$.
- $\mathsf{Priv}(sk', u)$: on input sk' and $u \in U$, output $\pi = \mathsf{H}_{pk}(u)$.
- $\mathsf{Ext}(sk, u, \pi)$: on input sk, $u \in U$, and $\pi \in \Pi$, output $s \in S$ such that $(s, u) \in \mathsf{R}$ if and only if $\pi = \mathsf{H}_{pk}(u)$.

In EHPS, KeyGen' and Priv work in the hashing mode, which are only used to establish security. EHPS satisfies the following property:

Definition 6.1 (Indistinguishable)**.** The first outputs (namely pk) of KeyGen and KeyGen' are statistically indistinguishable.

ALL-BUT-ONE EXTRACTABLE HASH PROOF SYSTEM. All-but-one (ABO) EHPS is a richer abstraction of EHPS, besides algorithms (Setup, KeyGen, KeyGen', Pub, Priv, Ext), it has an additional algorithm Ext'.

- $\mathsf{KeyGen}'(pp, u^*)$: on input public parameter pp and an arbitrary $u^* \in U$, output a key pair (pk, sk').
- $\mathsf{Ext}'(sk', u, \pi)$: on input sk', $u \in U$ such that $u \neq u^*$, and $\pi \in \Pi$, output $s \in S$ such that $(s, u) \in \mathsf{R}$ if and only if $\pi = \mathsf{H}_{pk}(u)$.

In ABO EHPS, KeyGen', Priv, and Ext' work in the ABO hashing mode, which are only used to establish security. All-but-one EHPS satisfies the following property:

Definition 6.2 (Indistinguishable)**.** For any $u^* \in U$, the first output (namely pk) of KeyGen and KeyGen' are statistically indistinguishable.

6.1 Construction from (All-But-One) EHPS

From (ABO) EHPS, we construct PEPRF as follows:

- $\mathsf{Setup}(\kappa)$: on input κ, run EHPS.$\mathsf{Setup}(\kappa)$ to generate an EHPS instance (H, PK, SK, S, U, Π), and build public parameters $pp = (F, PK, SK, X, L, W, Y)$ for PEPRF from it, where $X = U \times \Pi$, $Y = \{0,1\}^l$, F, L, and W will be defined later. We assume the publicly parameters pp of PEPRF and EHPS essentially contain same information.

- $\mathsf{KeyGen}(pp)$: on input pp, output $(pk, sk) \leftarrow \text{EHPS}.\mathsf{KeyGen}(pp)$.
- $\mathsf{Sample}(r)$: on input r, compute $u \leftarrow \text{EHPS}.\mathsf{SampRight}(r)$, and $\pi \leftarrow \text{EHPS}.\mathsf{Pub}(pk, r)$, output $x = (u, \pi)$ and $w = r$. This algorithm defines a language $L = \{(u, \pi) : u \in U \wedge \pi = \mathsf{H}_{pk}(u)\}$ over X, where the random coins r used to sample u serves as a witness for $x = (u, \pi) \in L$. Note the witness set W is exactly the randomness space R used by EHPS.Sample.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, compute $s \leftarrow \text{EHPS}.\mathsf{SampLeft}(w)$, output $y \leftarrow \mathsf{hc}(s)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, parse x as (u, π), compute $s \leftarrow \text{EHPS}.\mathsf{Ext}(sk, u, \pi)$, output $y \leftarrow \mathsf{hc}(s)$. This algorithm defines $\mathsf{F}_{sk}(x)$ as $\mathsf{hc}(\mathsf{Ext}(sk, x))$.

We have the following two theorems about the above construction.

Theorem 6.1. *If the underlying binary relation* R *is one-way, then the PEPRF from EHPS is weak pseudorandom.*

We defer the proof of Theorem 6.1 in Appendix C.3.

Theorem 6.2. *If the underlying binary relation* R *is one-way, then the PEPRF from all-but-one EHPS is adaptively weak pseudorandom.*

We defer the proof of Theorem 6.2 in Appendix C.4.

7 Publicly Samplable PRFs

In this section, we consider a relaxation of the functionality for PEPRFs, that is, instead of requiring the existence of NP language L over X and the publicly evaluable property of $\mathsf{F}_{sk}(x)$, we only require that the distribution $(x, \mathsf{F}_{sk}(x))$ is efficiently samplable with pk. More precisely, algorithms $\mathsf{Sample}(r)$ and $\mathsf{PubEval}(pk, x, w)$ are replaced by algorithm $\mathsf{PubSamp}(pk; r)$, which on input pk and random coins r outputs a random tuple $(x, y) \in X \times Y$ such that $y = \mathsf{F}_{sk}(x)$. We refer to this relaxed notion as publicly samplable PRFs (PSPRFs). The (adaptively) weak pseudorandomness for PSPRFs can be defined analogously. It is easy to verify that PSPRF and KEM imply each other by viewing PSPRF.$\mathsf{PubSamp}$ (resp. PSPRF.$\mathsf{PrivEval}$) as KEM.Encap (resp. KEM.Decap[8]). In light of this observation, we view PSPRFs as a high level interpretation of KEM, which allows significantly simpler and modular proof of security. In what follows, we revisit the notion of trapdoor one-way relations, and explore its relation to PSPRFs.

7.1 Trapdoor Relations

Before revisiting the notion of trapdoor relations, we first recall a closely related notion, namely trapdoor functions (TDFs) (c.f. Appendix D.1). Briefly, TDFs are a family of functions that are easy to compute, invert with trapdoor but hard to invert given the image of a uniformly chosen input without trapdoor. Most attention in the literature has focus on injective (i.e. one-to-one) TDFs. It is well known that injective TDFs suffice for PKE [Yao82, GM84]. Bellare et al. [BHSV98] made a careful distinction for TDFs based on "the amount of non-injectivity", measured by pre-image size. A (trapdoor, one-way) function is said to have pre-image size $Q(\kappa)$ (where κ is the security parameter) if the number of pre-images of any range points is at most $Q(\kappa)$. They demonstrated that $Q(\kappa)$ is a crucial parameter with regarding to building PKE

[8]Without loss of generality, we assume KEM.Decap is deterministic. We note that there do exist probabilistic decapsulation algorithms, e.g. those that implement "implicit rejection" strategy [KV08]. In this case, we can view KEM.Decap as randomized PSPRFs.

out of TDFs by showing two facts: (i) OWFs imply TDFs with super-polynomial pre-image size; (ii) TDFs with polynomial pre-image size is sufficient to imply PKE. Kiltz et al. [KMO10] strengthened TDFs to adaptive TDFs (ATDFs), which requires TDFs remain one-way even the adversary can adaptively access an inversion oracle. They used injective ATDFs as a general assumption to construct CCA-secure PKE. Wee [Wee10] introduced the notion of trapdoor relations (TDRs) as a functionality relaxation of injective TDFs, in which the "easy to compute" property is weakened to "easy to sample". Wee also showed how to construct such TDRs from EHPS. We note that the notion of TDRs defined in [Wee10] is inherently to be "one-to-one", while the TDRs yielded from EHPS is potentially to be "one-to-many". Towards utmost generality, we redefine the notion of TDRs in a generalized way as follows:

Definition 7.1 (Trapdoor Relations)**.** A family of trapdoor relations consists of four polynomial-time algorithms as below.

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, output public parameters pp which includes finite sets EK, TD, S, U (these sets are parameterized by κ), and a binary relation family $\mathsf{R} : S \times U$ indexed by EK, which will be defined by $\mathsf{PubSamp}$ as below.
- $\mathsf{TrapGen}(pp)$: on input pp, output $(ek, td) \in EK \times TD$.
- $\mathsf{PubSamp}(ek; r)$: on input ek and random coins r, output a tuple $(s, u) \in S \times U$. Implicitly, this gives us the relation $\mathsf{R}_{ek} = \{(s, u) : \exists r \text{ s.t. } (s, u) = \mathsf{PubSamp}(ek; r)\}$. We extend the distinction of non-injectivity for functions to the setting of binary relations. Hereafter, for every element $u \in U$ we define $S_u = \{s : (s, u) \in \mathsf{R}_{ek}\}$; for every element $s \in S$ we define $U_s = \{u : (s, u) \in \mathsf{R}_{ek}\}$. Let $Q(\kappa) = \max(|S_u|_{u \in U})$ and $P(\kappa) = \max(|U_s|_{s \in S})$. Notationally, we say a binary relation $\mathsf{R} : S \times U$ is "many-to-one" if $Q(\kappa) > 1$ and $P(\kappa) = 1$; say it is "one-to-many" if $P(\kappa) > 1$ and $Q(\kappa) = 1$; say it is "many-to-many" if $Q(\kappa) > 1$ and $P(\kappa) > 1$; say it is "one-to-one" if $Q(\kappa) = P(\kappa) = 1$.
- $\mathsf{TdInv}(td, u)$: on input td and $u \in U$, output $s \in S$ or a distinguished symbol $\perp$ indicating u is not well-defined with respect to td.[9]

Correctness: We require that for any $pp \leftarrow \mathsf{Setup}(\kappa)$ any $(ek, td) \leftarrow \mathsf{KeyGen}(pp)$, and any $(s, u) \in \mathsf{R}_{ek}$, it holds that:

$$\Pr[(\mathsf{TdInv}(td, u), u) \in \mathsf{R}_{ek}] = 1$$

(Adaptive) One-wayness: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an inverter against TDRs and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[(s, u^*) \in \mathsf{R}_{ek} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ (ek, td) \leftarrow \mathsf{TrapGen}(pp); \\ state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{inv}}(\cdot)}(pp, ek); \\ (s^*, u^*) \leftarrow \mathsf{PubSamp}(ek); \\ s \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{inv}}(\cdot)}(state, u^*) \end{array}\right],$$

where $\mathcal{O}_{\mathsf{inv}}(y) = \mathsf{TdInv}(td, y)$, and $\mathcal{A}_2$ is not allowed to query $\mathcal{O}_{\mathsf{inv}}(\cdot)$ for the challenge u^*. We say TDRs are adaptively one-way (or simply adaptively) if for any PPT inverter its advantage is negligible in κ. The standard one-wayness can be defined similarly as above except that the adversary is not given access to the inversion oracle.

Construction from TDFs. It is easy to see that TDFs imply TDRs. TDRs can be constructed from TDFs as below:

[9] We say u is well-defined with respect to td if there exists ek and random coins r_1, r_2 such that $(ek, td) = \mathsf{KeyGen}(pp; r_1)$ and $(s, u) = \mathsf{PubSamp}(ek; r_2)$.

- $\mathsf{Setup}(\kappa)$: run TDF.$\mathsf{Setup}(\kappa)$ to generate public parameters pp, set $S = X$, $U = Y$.
- $\mathsf{KeyGen}(\kappa)$: run TDF.$\mathsf{TrapGen}(pp)$ to generate (ek, td).
- $\mathsf{PubSamp}(ek; r)$: run TDF.$\mathsf{SampleDom}(r)$ to sample a random element $s \in S$, compute $u \leftarrow$ TDF.$\mathsf{Eval}(ek, s)$, then output (s, u).
- $\mathsf{TdInv}(td, u)$: output TDF.$\mathsf{TdInv}(td, u)$.

The correctness and security of the above construction follows immediately from that of TDFs. We omit the details here for triviality. Obviously, the resulting TDRs is "many-to-one" (resp. "one-to-one") if the underlying is "many-to-one" (injective).

7.2 Publicly Samplable PRFs from TDRs

Construction from TDRs. We show a simple construction of PSPRFs from "one-to-many" or "one-to-one" TDRs $= (\mathsf{TDR}, S, U)$. Let $\mathsf{hc} : S \to \{0,1\}^l$ be a hardcore function for TDR, we construct a PSPRF from U to $\{0,1\}^l \cup \perp$ as follows:

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, run TDR.$\mathsf{Setup}(\kappa)$ to generate pp.
- $\mathsf{KeyGen}(pp)$: on input pp, compute $(ek, td) \leftarrow$ TDR.$\mathsf{TrapGen}(pp)$, set $pk = ek$ and $sk = td$, output (pk, sk).
- $\mathsf{PubSamp}(pk; r)$: on input pk and random coins r, compute $(s, u) \leftarrow$ TDR.$\mathsf{PubSamp}(pk; r)$, output $(u, \mathsf{hc}(s))$.
- $\mathsf{PrivEval}(sk, u)$: on input sk and u, compute $s \leftarrow$ TDR.$\mathsf{TdInv}(sk, u)$, if $s = \perp$ output $\perp$, else output $\mathsf{hc}(s)$.

The correctness of the above construction is easy to verify. For the security, we have the following result:

Theorem 7.1. *The resulting PSPRF is (adaptively) weak pseudorandom if the underlying TDR is (adaptively) one-way.*

We omit the proof for its straightforwardness. The above result indicates that adaptive PSPRFs are implied by adaptive TDFs. By the separation result due to Gertner, Malkin, and Reingold [GMR01] that it is impossible of basing TDFs on trapdoor predicates, as well as the equivalence among trapdoor predicates and CPA-secure PKEs and PSPRFs, we conclude that PSPRFs are strictly weaker than TDFs in a black-box sense. We conjecture a similar separation result also exists between adaptive PSPRFs and ATDFs. Besides, if adaptive PSPRFs are strictly weaker than general ATDRs is also unclear to us. We left this as an open problem.

References

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 113–134. Springer, 2010.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computation*, 36(5):1301–1328, 2007.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401, 2013. http://eprint.iacr.org/.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM Conference on Computer and Communications Security, CCS 2012*, pages 784–796. ACM, 2012.

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their ralation to public-key cryptosystems. In *CRYPTO 1998*, volume 1462 of *LNCS*, pages 283–298. Springer, 1998.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532, 2001.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *CCS 2005*, pages 320–329. ACM, 2005.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *Advances in Cryptology - EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 399–416, 1996.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 280–300. Springer, 2013.

[CHK10] Ronald Cramer, Dennis Hofheinz, and Eike Kiltz. A twist on the naor-yung paradigm and its application to efficient cca-secure encryption from hard search problems. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010*, volume 5978 of *LNCS*, pages 146–164. Springer, 2010.

[CKS09] David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. *J. Cryptology*, 22(4):470–504, 2009.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO 1998*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, 2002.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33:167–226, 2003.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DS13] Dana Dachman-Soled. A black-box construction of a cca2 encryption scheme from a plaintext aware encryption scheme. *IACR Cryptology ePrint Archive*, 2013:680, 2013.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *Advances in Cryptology - CRYPTO 2013*, volume 8043 of *LNCS*, pages 479–499. Springer, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS 2001*, pages 126–135. IEEE Computer Society, 2001.

[HJKS10] Kristiyan Haralambiev, Tibor Jager, Eike Kiltz, and Victor Shoup. Simple and efficient public-key encryption from computational diffie-hellman in the standard model. In *Public Key Cryptography - PKC 2010*, volume 6056 of *LNCS*, pages 1–18. Springer, 2010.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer, 2007.

[HK08] Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational diffie-hellman assumption. In *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 308–325. Springer, 2008.

[HK09] Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 313–332. Springer, 2009.

[HLW12] Susan Hohenberger, Allison B. Lewko, and Brent Waters. Detecting dangerous queries: A new approach for chosen ciphertext security. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 663–681. Springer, 2012.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. 2013. http://eprint.iacr.org/2013/509.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, STOC 1989*, pages 44–61. ACM, 1989.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442. Springer, 2004.

[Kil06] Eike Kiltz. On the limitations of the spread of an ibe-to-pke transformation. In *Public Key Cryptography - PKC 2006*, volume 3958 of *LNCS*, pages 274–289. Springer, 2006.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, 2010.

[KPSY09] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A new randomness extraction paradigm for hybrid encryption. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 590–609. Springer, 2009.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. Cryptology ePrint Archive, 2013. http://eprint.iacr.org/2013/379.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.

[KV08] Eike Kiltz and Yevgeniy Vahlis. Cca2 secure ibe: Standard model efficiency through authenticated symmetric encryption. In *CT-RSA*, volume 4964 of *LNCS*, pages 221–238. Springer, 2008.

[MH14] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via point obfuscation. In *TCC 2014*, volume 8349 of *LNCS*, pages 95–120. Springer, 2014.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing, STOC 1990*, pages 427–437. ACM, 1990.

[PS08] Krzysztof Pietrzak and Johan Sjödin. Weak pseudorandom functions in minicrypt. In *Automata, Languages and Programming, 35th International Colloquium, ICALP (2) 2008*, volume 5126 of *LNCS*, pages 423–436. Springer, 2008.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, pages 187–196. ACM, 2008.

[Rab81] Michael Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computation*, 9:273–280, 1981.

[RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.

[RSA78] Ron Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. 2013. http://eprint.iacr.org/2013/454.

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332. Springer, 2010.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91. IEEE Computer Society, 1982.

A Publicly Evaluable Predicate PRFs

In this section, we investigate a similar extension on recent predicate PRFs [BGI13, BW13, KPTZ13] as we did on standard PRFs. We first give a precise definition of publicly evaluable predicate PRFs, then define the security requirement and explore possible applications.

Definition A.1 (Publicly Evaluable Predicate PRFs)**.** Publicly evaluable predicate PRFs consists of six polynomial-time algorithms as follows:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp which includes finite sets MPK, MSK, I, X, Y, a predicate family $P : I \to \{0, 1\}$, a collection of languages $L = \{L_{ind}\}_{ind \in I}$ defined over X and a witness set W, as well as a PRF family $F = \{\mathsf{F}_{msk} : I \times X \to Y\}_{msk \in MSK}$. Unlike the syntax of predicate PRF [BGI13, BW13, KPTZ13], here we explicitly define the domain as a Cartesian product of I and X. By the convention of predicate encryption, we assume that for any $ind \in I$ one can efficiently find a predicate $p \in P$ satisfying $p(ind) = 1$.
- $\mathsf{KeyGen}(pp)$: on input pp, output master public key mpk and master secret key msk.
- $\mathsf{Extract}(msk, p)$: on input msk and a predicate $p \in P$, output a secret key sk_p.
- $\mathsf{Sample}(ind, r)$: on input $ind \in I$ and random coins r, output a random $x \in L_{ind}$ and a witness w for x.
- $\mathsf{PubEval}(ind, x, w)$: on input $ind \in I$ and $x \in L_{ind}$ together with a witness $w \in W$ for x, output $y \in Y$.
- $\mathsf{PrivEval}(sk_p, x)$: on input secret key sk_p and $x \in X$, output $y \in Y$.

Correctness: For any $pp \leftarrow \mathsf{Setup}(\kappa)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any $ind \in I$, and any $x \in L_{ind}$, it holds that:

$$\begin{aligned} \forall sk_p \leftarrow \mathsf{KeyGen}(msk, p) \text{ such that } p(ind) = 1 : \quad & \mathsf{F}_{msk}(ind, x) = \mathsf{PrivEval}(sk_p, x). \\ \text{a witness } w \in W \text{ for } x \in L_{ind} : \quad & \mathsf{F}_{msk}(ind, x) = \mathsf{PubEval}(ind, x, w). \end{aligned}$$

Pseudorandomness: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against publicly evaluable predicate PRFs and defines its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ (mpk, msk) \leftarrow \mathsf{KeyGen}(pp); \\ (state, ind^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{eval}}(\cdot,\cdot), \mathcal{O}_{\mathsf{extract}}(\cdot)}(pp, mpk); \\ (x^*, w^*) \leftarrow \mathsf{Sample}(ind^*, r^*), r^* \xleftarrow{\text{R}} R; \\ y_0^* \leftarrow \mathsf{F}_{msk}(ind^*, x^*), y_1^* \xleftarrow{\text{R}} Y; \\ b \xleftarrow{\text{R}} \{0, 1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{eval}}(\cdot,\cdot), \mathcal{O}_{\mathsf{extract}}(\cdot)}(state, x^*, y_b^*) \end{array} \right] - \frac{1}{2},$$

where $\mathcal{O}_{\mathsf{eval}}(ind, x) = \mathsf{F}_{msk}(ind, x)$, $\mathcal{O}_{\mathsf{extract}}(p) = \mathsf{Extract}(msk, p)$. The adversary $\mathcal{A}$ is restricted from querying $\mathcal{O}_{\mathsf{extract}}(\cdot)$ with p such that $p(id^*) = 1$, and the $\mathcal{A}_2$ is restricted from querying $\mathcal{O}_{\mathsf{eval}}(\cdot, \cdot)$ with (ind^*, x^*). Publicly evaluable predicate PRF is said to be adaptively weak pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}$ is negligible in κ.

We can define a weaker security notion by considering adversaries who only adaptively query oracle $\mathcal{O}_{\mathsf{extract}}(\cdot)$ but never query oracle $\mathcal{O}_{\mathsf{eval}}(\cdot, \cdot)$. We refer to the corresponding security notion as semi-adaptively weak pseudorandomness.

It is furthermore possible and straightforward to give an analogous relaxation of publicly evaluable predicate PRFs, namely requiring that $\mathsf{F}_{msk}(ind, \cdot)$ is publicly samplable instead of publicly evaluable.

A.1 Predicate KEM from Publicly Evaluable Predicate PRFs

Predicate encryption was introduced by Katz et al. [KSW08] as a generalization of identity-based encryption and attribute-based encryption. Similar to the situation in the public-key setting and identity-based setting, there are numerous practical reasons to prefer a predicate key encapsulation mechanism over a predicate encryption. We refer the readers to D.3 for formal definition of predicate KEM.

The construction of a predicate KEM from a publicly evaluable predicate PRF is almost immediate, by simply using the PRF value as a DEM key. In particular, given a publicly evaluable predicate PRF where the range Y, we construct a predicate KEM with the same index set I and DEM key set $K = Y$. The Setup, KeyGen, and Extract algorithms are the same as that of publicly evaluable predicate PRF and Encap, Decap are defined by:

- $\mathsf{Encap}(mpk, ind)$: on input mpk and $ind \in I$, run PRF.$\mathsf{Sample}(ind, r)$ with fresh random coins r to sample a random $x \in L_{ind}$ with a witness w for x, compute $y \leftarrow$ PRF.$\mathsf{PubEval}(ind, x, w)$, set $c = x$, $k = y$ and output (c, k).
- $\mathsf{Decap}(sk_p, c)$: on input sk_p and c, output $k \leftarrow$ PRF.$\mathsf{PrivEval}(sk_p, c)$.

Correctness of this construction follows directly from that of the publicly evaluable predicate PRF. For security, we have the following results:

Theorem A.1. *The predicate encryption is CPA-secure if the underlying publicly evaluable predicate PRF is semi-adaptively weak pseudorandom.*

Theorem A.2. *The predicate encryption is CCA-secure if the underlying publicly evaluable predicate PRF is adaptively weak pseudorandom.*

We omit the proofs here for their triviality.

A.2 A Construction of Publicly Evaluable Predicate PRFs

Next we present a concrete publicly evaluable predicate PRFs for general circuits from multilinear maps based on recent attribute-based encryption [GGH+13]. We use the same notation for circuits as in [GGH+13], which is included in Appendix D.4 for completeness. We refer the readers to Appendix D.5 for the definition of multilinear maps and related hardness assumption.

- $\mathsf{Setup}(\kappa, \ell)$: on input security parameter κ, the maximum depth ℓ of a circuit, and the number of boolean inputs n, run $\mathsf{MLGroupGen}(\kappa, k = \ell + 1)$ to generate multilinear groups $(p, \{\mathbb{G}_i\}_{i\in[k]}, \{e_{i,j}\}_{i,j\geq 1, i+j\leq k})$ with canonical generators $g_1, \ldots, g_k$ as public parameter pp.
- $\mathsf{KeyGen}(pp)$: on input public parameter pp, choose $\alpha \xleftarrow{R} \mathbb{Z}_p$ and $h_1, \ldots, h_n \xleftarrow{R} \mathbb{G}_1$, output $mpk = (g_k^\alpha, h_1, \ldots, h_n)$ and $msk = (g_{k-1})^\alpha$. We let $I = \{0,1\}^n$, $X = \mathbb{G}_1^n$. For any $ind \in \{0,1\}^n$, let S_{ind} be the set of subscript indices i such that $ind_i = 1$. We define a collection of languages $L_{ind} = \{(g_1^r, \{h_i^r\}_{i\in S_{ind}})\}_{ind \in I}$ indexed by I, where $r \in \mathbb{Z}_p$ serves as the witness.
- $\mathsf{Extract}(msk, c)$: on input $msk = (g_{k-1})^\alpha$ and a circuit c, choose $s_1, \ldots, s_{n+q} \xleftarrow{R} \mathbb{Z}_p$ (where randomness s_w is associated with wire w), first produce a "header" component $sk_h = (g_{k-1})^{\alpha - s_{n+q}}$, then produce key components for every wire w. The structure of the key components depends upon if w is an input wire, an AND gate, or an OR gate. We describe each case as below:
  - *Input wire*
    By our convention if $w \in [1, n]$ then it corresponds to the w-th input. Pick $t_w \xleftarrow{R} \mathbb{Z}_p$, and create key component $sk_w = (sk_{w,1}, sk_{w,2})$, where
    $$sk_{w,1} = g_1^{s_w} h_w^{t_w}, sk_{w,2} = g_1^{-t_w}$$
  - *OR gate*
    Suppose that wire $w \in Gates$ and $\mathsf{GateType}(w) = \text{OR}$. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w. Pick $a_w, b_w \xleftarrow{R} \mathbb{Z}_p$, and create key component $sk_w = (sk_{w,1}, sk_{w,2}, sk_{w,3}, sk_{w,4})$, where
    $$sk_{w,1} = g_1^{a_w}, sk_{w,2} = g_1^{b_w}, sk_{w,3} = g_j^{s_w - a_w \cdot s_{\mathsf{A}(w)}}, sk_{w,4} = g_j^{s_w - b_w \cdot s_{\mathsf{B}(w)}}$$
  - *AND gate*
    Suppose that wire $w \in Gates$ and that $\mathsf{GateType}(w) = \text{AND}$. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w. Pick $a_w, b_w \leftarrow \mathbb{Z}_p$, and create key component $sk_w = (sk_{w,1}, sk_{w,2}, sk_{w,3})$, where
    $$sk_{w,1} = g_1^{a_w}, sk_{w,2} = g_1^{b_w}, sk_{w,3} = g_j^{s_w - a_w \cdot s_{\mathsf{A}(w)} - b_w \cdot s_{\mathsf{B}(w)}}$$
- $\mathsf{Sample}(ind, r)$: on input $ind \in \{0,1\}^n$ and random coins $r \in \mathbb{Z}_p$, set $x_0 = g_1^r$, $x_i = h_i^r$ for $i \in S_{ind}$, and output $x = (x_0, \{x_i\}_{i\in S_{ind}}) \in L_{ind}$ and witness $w = r$.
- $\mathsf{PubEval}(ind, x, r)$: on input $ind \in \{0,1\}^n$, $x \in L_{ind}$, and a witness $r \in \mathbb{Z}_p$, output $y = (g_k^\alpha)^r$.

- $\mathsf{PrivEval}(sk_c, x)$: on input a secret key sk_c for a circuit $c = (n, q, \mathsf{A}, \mathsf{B}, \mathsf{GateType})$ and x (the associated ind can be simply recovered from x), first compute $y' = e(sk_h, g_1^r) = e(g_{k-1}^{\alpha - s_{n+q}}, g_1^r) = g_k^{\alpha r} g_k^{-s_{n+q} \cdot r}$, then evaluate the circuit from the bottom up: consider wire w at depth j, if $c_w(ind) = 1$ then computes $y_w = (g_{j+1})^{s_w \cdot r}$, else nothing needs to be computed for this wire. The evaluation proceeds iteratively starting from y_1 to finally y_{n+q}, with the purpose of the computation on a depth $j-1$ wire (that evaluates to 1) will be defined before computing for a depth j wire. We show how to compute y_w for all w where $c_w(ind) = 1$, again according to whether the wire is an input, AND or OR gate.
  - *Input wire*
    By our convention if $w \in [1, n]$ then it corresponds to the w-th input. Suppose that $c_w(ind) = 1$. The algorithm computes:
    $$y_w = e(sk_{w,1}, g_1^r) \cdot e(sk_{w,2}, x_w) = e(g_1^{s_w} h_w^{t_w}, g_1^r) \cdot e(g_1^{-t_w}, h_w^r) = g_2^{r s_w}$$
  - *OR gate*
    Suppose that wire $w \in Gates$ and that $\mathsf{GateType}(w) = \text{OR}$. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w. Suppose that $c_w(ind) = 1$. If $c_{\mathsf{A}(w)}(ind) = 1$ (the first input evaluated to 1), then we compute:
    $$y_w = e(y_{\mathsf{A}(w)}, sk_{w,1}) \cdot e(sk_{w,3}, g_1^r) = e(g_j^{r s_{\mathsf{A}(w)}}, g_1^{a_w}) \cdot e(g_j^{s_w - a_w \cdot s_{\mathsf{A}(w)}}, g_1^r) = (g_{j+1})^{r s_w}$$
    Alternatively, if $c_{\mathsf{A}(w)}(ind) = 0$, but $c_{\mathsf{B}(w)}(ind) = 1$, then we compute:
    $$y_w = e(y_{\mathsf{B}(w)}, sk_{w,2}) \cdot e(sk_{w,4}, g_1^r) = e(g_j^{r s_{\mathsf{B}(w)}}, g_1^{b_w}) \cdot e(g_j^{s_w - b_w \cdot s_{\mathsf{B}(w)}}, g_1^r) = (g_{j+1})^{r s_w}$$
  - *AND gate*
    Suppose that wire $w \in Gates$ and that $\mathsf{GateType}(w) = \text{AND}$. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w. Suppose that $c_w(ind) = 1$. Then $c_{\mathsf{A}(w)}(ind) = c_{\mathsf{B}(w)}(ind) = 1$ and we compute:
    $$\begin{aligned} y_w &= e(y_{\mathsf{A}(w)}, sk_{w,1}) \cdot e(y_{\mathsf{B}}(w), sk_{w,2}) \cdot e(sk_{w,3}, g_1^r) \\ &= e(g_j^{r s_{\mathsf{A}(w)}}, g_1^{a_w}) \cdot e(g_j^{r s_{\mathsf{B}(w)}}, g_1^{b_w}) \cdot e(g_j^{s_w - a_w \cdot s_{\mathsf{A}(w)} - b_w \cdot s_{\mathsf{B}}(w)}, g_1^r) = (g_{j+1})^{r s_w} \end{aligned}$$

  Finally, output $y' \cdot y_{n+q}$. The correctness is easy to verify by observing that if $c(ind) = c_{n+q}(ind) = 1$, then $y_{n+q} = g_k^{r \cdot s_{n+q}}$ and the final output is $g_k^{\alpha r}$.

The security of the above construction is based on the MDDH assumption, which follows immediately from the analysis of attribute-based encryption [GGH+13].

B Publicly Evaluable and Verifiable Functions

In this section, we introduce a variant of PEPRFs, which we call publicly evaluable and verifiable functions (PEVFs). Compared to PEPRFs, PEVFs have an additional property named *public verifiability*. As a consequence, the best possible security for PEVFs degrades to "hard to compute" on average.

Definition B.1 (Publicly Evaluable and Verifiable Functions)**.** PEVFs consist of the following polynomial-time algorithms:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp which includes finite sets SK, PK, X, L, W, Y (these sets may be parameterized by κ), as well as a publicly evaluable and verifiable functions family $F = \{\mathsf{F}_{sk} : X \to Y\}_{sk \in SK}$.

- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key sk and an associated public key pk.
- $\mathsf{Sample}(r)$: on input random coins r, output a random $x \in L$ along with a witness $w \in W$ for x.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, output $y = \mathsf{F}_{sk}(x)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y = \mathsf{F}_{sk}(x)$.
- $\mathsf{PubVefy}(pk, x, y)$: on input pk, x, and y, output 1 if $y = \mathsf{F}_{sk}(x)$ and 0 if not.

Security: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against PEVFs and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[\mathsf{PubVefy}(pk, x^*, y) = 1 : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ state \leftarrow \mathcal{A}_1(pp, pk); \\ x^* \xleftarrow{\text{R}} L; \\ y \leftarrow \mathcal{A}_2(state, x^*); \end{array}\right],$$

We say that PEVFs are hard to compute on average if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

B.1 Instantiations of PEVFs

Here we construct PEVFs based on the RSA assumption (c.f. definition in D.6) and the CDH assumption in bilinear groups, respectively.

PEVFs based on the RSA assumption

- $\mathsf{Setup}(\kappa)$: run $\mathsf{RSAGen}(\kappa)$ to generate (N, p, q), set $X = L = W = Y = \mathbb{Z}_N^*$, set PK and SK as the set of integers that are less than and relatively prime to $\phi(N)$.
- $\mathsf{KeyGen}(pp)$: randomly pick an integer e between 1 and $\phi(N)$ such that $\gcd(\phi(N), e) = 1$, compute $d \equiv e^{-1} \bmod \phi(N)$; output $pk = e$, $sk = d$.
- $\mathsf{Sample}(r)$: on input random coins r, output $x \in \mathbb{Z}_N^*$ together with a witness w such that $w^e \equiv x \bmod N$.
- $\mathsf{PubEval}(pk, x, w)$: on input $pk = e$, x and w, output w.
- $\mathsf{PrivEval}(sk, x)$: on input $sk = d$ and x, output x^d.
- $\mathsf{PubVefy}(pk, x, y)$: on input $pk = e$, x and y, output 1 if $x \equiv y^e \bmod N$ and 0 if not.

PEVFs based on the CDH assumption in bilinear groups

- $\mathsf{Setup}(\kappa)$: run $\mathsf{BLGroupGen}(\kappa)$ to generate $(e, g, \mathbb{G}, \mathbb{G}_T)$, set $X = L = Y = PK = \mathbb{G}$, $SK = W = \mathbb{Z}_p$.
- $\mathsf{KeyGen}(pp)$: randomly pick $sk \xleftarrow{\text{R}} \mathbb{Z}_p$, compute $pk = g^{sk}$.
- $\mathsf{Sample}(r)$: on input random coins r, output $x \in \mathbb{G}$ together with a witness w such that $x = g^w$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk, x and w, output $y = pk^w$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, output $y = x^{sk}$.
- $\mathsf{PubVefy}(pk, x, y)$: on input pk, x and y, output 1 if $e(pk, x) = e(g, y)$ and 0 if not.

B.2 Signature from PEVFs

Now we show a simple and intuitive construction of "hash-and-sign" signatures from PEVFs.

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, run PEVF.$\mathsf{Setup}(\kappa)$ to generate public parameters pp. Let M be the message space, $\mathsf{H} : M \to L$ be a hash function.
- $\mathsf{KeyGen}(pp)$: run PEVF.$\mathsf{KeyGen}(pp)$ to generate (pk, sk), output $vk = (pk, \mathsf{H})$ as the verification key and sk as the signing key.
- $\mathsf{Sign}(sk, m)$: on input sk and m, output signature $\sigma \leftarrow$ PEVF.$\mathsf{PrivEval}(sk, \mathsf{H}(m))$.
- $\mathsf{Verify}(vk, m, \sigma)$: on input $vk = (pk, \mathsf{H})$, m and σ, output PEVF.$\mathsf{PubVefy}(pk, \mathsf{H}(m), \sigma)$.

Note that algorithms PEVF.Sample and PEVF.$\mathsf{PubEval}$ are not used in the above construction, but will prove useful in security argument.

Theorem B.1. *The above signature is strongly unforgeable under adaptive chosen-message attack in the random oracle model if the underlying PEVFs are hard to compute on average. Suppose* H *is a random oracle, for any adversary* $\mathcal{A}$ *breaking the signature with advantage* $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ *that makes at most* Q_h *random oracle queries to* H*, there is an algorithm* $\mathcal{B}$ *that breaks the security of PEVFs with advantage at least* $\mathsf{Adv}_{\mathcal{A}}(\kappa)/Q_h$*.*

Proof. We prove this theorem by showing how to transform an adversary $\mathcal{A}$ against the signature into an algorithm $\mathcal{B}$ breaking the security of the underlying PEVFs. Without loss of generality, we assume that $\mathcal{A}$: (1) always queries the random oracle H with distinct messages; (2) first queries $\mathsf{H}(m)$ before querying the signing oracle with a message m; (3) first queries $\mathsf{H}(m)$ before outputting a forgery (m, σ). Given pk and the challenged point x^*, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows, with the aim to compute $\mathsf{F}_{sk}(x^*)$.

- Setup: $\mathcal{B}$ sends $vk = (pk, \mathsf{H})$ to $\mathcal{A}$. $\mathcal{B}$ randomly picks $j \in \{1, \dots, Q_h\}$.
- Random oracle queries: To process random oracle queries, $\mathcal{B}$ maintains a list H which is initially empty. Each entry in H is of the form (m, x, w), where $m \in M$, $x \in L$ and $w \in W$. When i-th random query on message m_i comes, $\mathcal{B}$ responds as follows:
  - If $i \neq j$, $\mathcal{B}$ picks $r_i \xleftarrow{\text{R}} R$, runs PEVF.$\mathsf{Sample}(r_i)$ to obtain (x_i, w_i), adds the entry (m_i, x_i, w_i) to the H list, returns x_i to $\mathcal{A}$.
  - If $i = j$, $\mathcal{B}$ adds the entry $(m_j, x^*, \perp)$ to the H list, returns x^* to $\mathcal{A}$.
- Signing queries: Upon receiving the singing query on message m_i, $\mathcal{B}$ responds as follows:
  - If $i \neq j$, $\mathcal{B}$ responds with $\sigma_i \leftarrow$ PEVF.$\mathsf{PubEval}(pk, x_i, w_i)$.
  - If $i = j$, $\mathcal{B}$ aborts.
- Forgery: Finally, $\mathcal{A}$ outputs a forgery (m_i, σ_i). If $i = j$, $\mathcal{B}$ forwards σ_i to its own challenger. Else, $\mathcal{B}$ aborts.

It is not difficult to verify that, unless $\mathcal{B}$ aborts, the simulation provided for $\mathcal{A}$ is perfect and $\mathcal{B}$ correctly computes the PEVF value at point x^* if $\mathcal{A}$ outputs a valid signature for m^*. It is easy to show the probability that $\mathcal{B}$ does not abort is $1/Q_h$. The theorem immediately follows. □

When applying the above generic construction to PEVFs based on the RSA assumption and the CDH assumption in bilinear groups presented in B.1, yields precisely the Bellare-Rogaway signature [BR96] and the Boneh-Lynn-Shacham (BLS) signature [BLS01]. We also note that the "full domain hash" (FDH) framework can not encompass the BLS signature accurately, cause there is no corresponding efficiently computable trapdoor function/permutation.

Replacing random oracle: Very recently, Hohenberger, Sahai, and Waters [HSW13] utilized indistinguishability obfuscation ($i\mathcal{O}$) to give a way to instantiate the random oracle with a concrete hash function in FDH applications. We extend their techniques to replacing the random oracle in the above construction. In what follows, we sketch how to instantiate H and establish the security. Let $\mathsf{PPRF} : K \times M \to R$ be a puncturable PRF. To build a concrete hash function for H, the user first picks a master key k for PPRF, then sets the hash function as an obfuscation of the program which on input m computes $r \leftarrow \mathsf{PPRF}(k, m)$, $(x, w) \leftarrow \mathsf{PEVF.Sample}(r)$, and outputs x. The security proof will proceed via a sequence of games. Let Game 0 be the standard selective security game with hash function instantiated as described above. In Game 1, we replace the original program with an obfuscation of a "puncturable program" which on input $m \neq m^*$ (here m^* is the message that $\mathcal{A}$ commits to attack before seeing the vk) outputs $\mathsf{PPRF}(k(\{m^*\}), m^*)$, and on input m^* is hardwired to output $z^* \leftarrow \mathsf{PPRF}(k, m^*)$. Since the input/output behavior is identical, Game 0 and Game 1 are computationally indistinguishable by the security of $i\mathcal{O}$. In Game 2, we replace z^* with a random element chosen from L. Due to the security of PPRF, Game 1 and Game 2 are computationally indistinguishable. At this moment, we can build an algorithm $\mathcal{B}$ that breaks the security of PEVFs by invoking $\mathcal{A}$ in Game 2. $\mathcal{B}$ receives PEVF challenge x^* and hardwires it as the output of $\mathsf{H}(m^*)$. During Game 2, $\mathcal{B}$ can use $k(\{m^*\})$ to create a valid signature for any messages other than m^* without knowing sk. Finally, $\mathcal{B}$ simply forwards the signature σ^* on m^* as the solution of $\mathsf{F}_{sk}(x^*)$. We note that one could use the usual complexity leveraging arguments to claim adaptive security.

We remark that our security proofs both in the random oracle model and the standard model share some of the spirit of the general RO security proof for FDH signatures, where the reduction programs the challenge at one point and it is able to produce valid signatures at all others points. A distinguished aspect is that the reduction creates the signature σ for message m via different methods. In the general RO security proof for FDH signatures, the reduction first picks σ randomly from domain, then programs $\mathsf{H}(m)$ to its trapdoor permutation $\mathsf{TDP}(\sigma)$. In contrast, in our security proofs for PEVF signatures, the reductions first programs $\mathsf{H}(m)$ to a random element x with extra information – its witness w, then computes its signature σ as $\mathsf{F}_{sk}(\mathsf{H}(m))$ using the publicly evaluable property.

Randomized PEVFs. We can further generalize the notion of PEVFs to randomized publicly evaluable and verifiable functions (RPEVFs). Briefly, RPEVFs are PEVFs whose evaluation is randomized, and the randomness is added to the image. We believe RPEVFs are suitable for admitting more applications, such as probabilistic "hash-and-sign" signatures.

C Missing Proofs

C.1 Proof of Theorem 5.1

Proof. The proof is similar to [CS02]. To establish the weak pseudorandomness based on the properties of smooth HPS and the hardness of underlying subset membership problem, we proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ outputs the right guess in Game i.

Game 0: $\mathcal{CH}$ interacts with $\mathcal{A}$ in the weak pseudorandomness game for PEPRFs as follows:

- Setup: $\mathcal{CH}$ runs HPS.$\mathsf{Setup}(\kappa)$ to build public parameters pp for PEPRFs, then runs HPS.$\mathsf{KeyGen}(pp)$ to generate public/secret key pair (pk, sk). $\mathcal{CH}$ gives (pp, pk) to $\mathcal{A}$.
- Challenge: $\mathcal{CH}$ picks $r^* \xleftarrow{R} R$, sets $(x^*, w^*) \leftarrow$ HPS.$\mathsf{Sample}(r^*)$, then computes $y_0^* \leftarrow \mathsf{H}_{sk}(x^*)$ via HPS.$\mathsf{Pub}(pk, x^*, w^*)$, picks $y_1^* \xleftarrow{R} Y$, picks a random bit $b \in \{0, 1\}$, then sends (x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b' = b$.

According to the definition of $\mathcal{A}$, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2| \tag{1}$$

Game 1: same as Game 0 except that in the challenge stage $\mathcal{CH}$ computes $y_0^* \leftarrow \mathsf{H}_{sk}(x^*)$ via HPS.$\mathsf{Priv}(sk, x^*)$. According to the functionality of Priv and Pub, this change is perfectly hidden from the adversary. Thus, we have:

$$\Pr[S_1] = \Pr[S_0] \tag{2}$$

Game 2: same as Game 1 except that in the challenge stage $\mathcal{CH}$ samples x^* via algorithm HPS.Sample' instead of HPS.Sample. The sampling indistinguishability (based on the hardness of the subset membership problem) ensures that:

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\kappa) \tag{3}$$

In Game 2, according to the smoothness of HPS, we have:

$$\Pr[S_2] = 1/2 \tag{4}$$

Putting all these above, the theorem immediately follows. □

C.2 Proof of Theorem 5.2

Proof. The proof is similar to [CS02]. To establish the adaptively weak pseudorandomness based on the properties of smooth and weak universal$_2$ HPSs and the hardness of underlying subset membership problem, we proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ outputs the right bit in Game i.

Game 0: $\mathcal{CH}$ interacts with $\mathcal{A}$ in the adaptively weak pseudorandomness game for PEPRFs as follows:

- Setup: $\mathcal{CH}$ runs HPS$_1$.$\mathsf{Setup}(\kappa)$ and HPS$_2$.$\mathsf{Setup}(\kappa)$ to generate a smooth HPS instance pp_1 and a weak universal$_2$ HPS instance pp_2 respectively, then builds public parameters pp for PEPRFs from pp_1 and pp_2. $\mathcal{CH}$ then runs $(pk_1, sk_1) \leftarrow$ HPS$_1$.$\mathsf{KeyGen}(pp_1)$ and $(pk_2, sk_2) \leftarrow$ HPS$_2$.$\mathsf{KeyGen}(pp_2)$, sets $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$. $\mathcal{CH}$ sends (pp, pk) to $\mathcal{A}$ as the challenge.
- Phase 1 - Evaluation query: When $\mathcal{A}$ queries the PRF value at point x, $\mathcal{CH}$ responds normally with $sk = (sk_1, sk_2)$. More precisely, $\mathcal{CH}$ parses x as $(\tilde{x}, \pi_2)$, then responds with $\tilde{\mathsf{H}}_{sk_1}(\tilde{x})$ if $\hat{\mathsf{H}}_{sk_2}(\tilde{x}) = \pi_2$ and $\perp$ otherwise.
- Challenge: $\mathcal{CH}$ picks $r^* \xleftarrow{\text{R}} R$, sets $(\tilde{x}^*, w^*) \leftarrow$ HPS$_1$.$\mathsf{Sample}(r^*)$, computes $\pi_2^* = \hat{\mathsf{H}}_{sk_2}(\tilde{x}^*)$ via HPS$_2$.$\mathsf{Pub}(pk_2, \tilde{x}^*, w^*)$, sets $x^* = (\tilde{x}^*, \pi_2^*)$. It then computes $y_0^* = \tilde{\mathsf{H}}_{sk_1}(\tilde{x}^*)$ via HPS$_1$.$\mathsf{Pub}(pk_1, \tilde{x}^*, w^*)$, samples $y_1^* \xleftarrow{\text{R}} Y$, picks a random bit $b \in \{0, 1\}$, then sends (x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Phase 2 - Evaluation query: same as in Phase 1 except that the query $\langle x^* \rangle$ is not allowed.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b' = b$.

According to the definition of $\mathcal{A}$, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2| \tag{5}$$

Game 1: same as Game 0 except that in the challenge stage $\mathcal{CH}$ computes $\pi_2^* = \hat{\mathsf{H}}_{sk_2}(\tilde{x}^*)$ via HPS$_2$.$\mathsf{Priv}(sk_2, \tilde{x}^*)$ and computes $y_0^* = \tilde{\mathsf{H}}_{sk_1}(\tilde{x}^*)$ via HPS$_1$.$\mathsf{Priv}(sk_2, \tilde{x}^*)$. According to the functionality of HPS.Priv and HPS.Pub, this change is perfectly hidden from $\mathcal{A}$. Thus, we have:

$$\Pr[S_1] = \Pr[S_0] \tag{6}$$

Game 2: same as Game 1 except that in the challenge stage $\mathcal{CH}$ samples $\tilde{x}^*$ via algorithm HPS.Sample' instead of HPS.Sample. The sampling indistinguishability (based on the hardness of the subset membership problem) ensures that:

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\kappa) \tag{7}$$

Game 3: same as Game 2 except that when answering evaluation queries $\langle x \rangle$ where $x = (\tilde{x}, \pi_2)$, $\mathcal{CH}$ returns $\perp$ when $\tilde{x} \notin \tilde{L}$ even $\hat{\mathsf{H}}_{sk_2}(\tilde{x}) = \pi_2$. For the ease of analysis, we denote by F the event that $\mathcal{A}$ submits some evaluation queries $\langle x \rangle$ where $x = (\tilde{x}, \pi_2)$ such that $\tilde{x} \notin L$ but $\hat{\mathsf{H}}_{sk_2}(\tilde{x}) = \pi_2$. According to the weak universal$_2$ property of HPS$_2$, we have $\Pr[F] \leq Q\epsilon$, where Q is the maximum number of PRF evaluation queries that $\mathcal{A}$ may make. Since ϵ is negligible in κ, we have:

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F] \leq \mathsf{negl}(\kappa)$$

In Game 3, $\mathcal{A}$'s view (the responses to evaluation queries and the challenge ciphertext) can be expressed as some function of (pk, x^*). Therefore, according to the smooth property of HPS$_1$, we have:

$$\Pr[S_3] = 1/2$$

Putting all these above, the theorem immediately follows. □

C.3 Proof of Theorem 6.1

Proof. The proof is similar to [Wee10]. To establish the weak pseudorandomness based on the properties of EHPS and one-wayness of R, we proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ outputs the right bit in Game i.

Game 0: $\mathcal{CH}$ interacts with $\mathcal{A}$ in the weak pseudorandomness game for PEPRFs by operating EHPS in the extraction mode.

- Setup: $\mathcal{CH}$ runs EHPS.$\mathsf{Setup}(\kappa)$ to generate public parameters pp for PEPRFs, runs EHPS.$\mathsf{KeyGen}(pp)$ to generate (pk, sk). $\mathcal{CH}$ sends (pp, pk) to $\mathcal{A}$.
- Challenge: $\mathcal{CH}$ picks random coins $r^* \xleftarrow{R} R$, samples $s^* \leftarrow$ EHPS.$\mathsf{SampLeft}(r^*)$ and $u^* \leftarrow$ EHPS.$\mathsf{SampRight}(r^*)$, computes $\pi^* = \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Pub}(pk, r^*)$, sets $x^* = (u^*, \pi^*)$, computes $y_0^* \leftarrow \mathsf{hc}(s^*)$, picks $y_1^* \xleftarrow{R} \{0,1\}^l$. $\mathcal{CH}$ picks a random bit $b \in \{0,1\}$, sends (x^*, y_b^*) to $\mathcal{A}$.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b = b'$.

According to the definition, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2| \tag{8}$$

Game 1: same as Game 0 except that $\mathcal{CH}$ prepares the challenge at the setup stage: picks $r^* \xleftarrow{R} R$, runs EHPS.$\mathsf{SampLeft}(r^*)$ and EHPS.$\mathsf{SampRight}(r^*)$ to obtain (s^*, u^*), and then computes $\pi^* = \mathsf{H}_{pk}(x^*)$ via EHPS.$\mathsf{Pub}(pk, r^*)$. This change is only conceptual, thus we have:

$$\Pr[S_1] = \Pr[S_0] \tag{9}$$

Game 2: $\mathcal{CH}$ interacts with $\mathcal{A}$ in the weak pseudorandomness game for PEPRFs by operating EHPS in the hashing mode. The differences to Game 1 are that in the setup phase, $\mathcal{CH}$ runs EHPS.$\mathsf{KeyGen}'(pp)$ to generate (pk, sk') and prepares the value $\pi^* = \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Priv}(sk', u^*)$. According to indistinguishability between $\mathsf{KeyGen}(pp)$ and $\mathsf{KeyGen}'(pp)$ and the correctness of the hashing mode, we have:

$$\Pr[S_2] = \Pr[S_1] \tag{10}$$

We then show that no PPT adversary has non-negligible advantage in Game 2. We prove this by showing that if such an adversary $\mathcal{A}$ exists, then we can construct an algorithm $\mathcal{B}$ that breaks the one-wayness of the underlying binary relation with non-negligible advantage. Given (y^*, u^*), $\mathcal{B}$ determines if $b = 0$ (meaning that $y^* = \mathsf{hc}(s^*)$ where $(s^*, u^*) \in \mathsf{R}$) or $b = 1$ (meaning that y^* is randomly picked from $\{0,1\}^l$). $\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 3 as follows:

- Setup: $\mathcal{B}$ runs EHPS.$\mathsf{Setup}(\kappa)$ to generate public parameters pp, runs EHPS.$\mathsf{KeyGen}'(pp)$ to generate (pk, sk'), computes $\pi^* = \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Priv}(sk', u^*)$, then sends (pp, pk) to $\mathcal{A}$.
- Challenge: $\mathcal{B}$ sets $x^* = (u^*, \pi^*)$, sends (x^*, y^*) to $\mathcal{A}$ as the challenge.
- Guess: $\mathcal{A}$ outputs its guess b' and $\mathcal{B}$ forwards b' to its own challenger.

Clearly, $\mathcal{B}$'s simulation is perfect. Therefore, we have:

$$\mathsf{Adv}_{\mathcal{B}}(\kappa) = |\Pr[S_2] - 1/2| = \mathsf{Adv}_{\mathcal{A}}(\kappa) \tag{11}$$

The theorem immediately follows. □

C.4 Proof of Theorem 6.2

Proof. The proof follows immediately from [Wee10]. To establish the adaptively weak pseudorandomness based on the properties of EHPS and one-wayness of R, we proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ outputs the right bit in Game i.

Game 0: $\mathcal{CH}$ interacts with $\mathcal{A}$ in the adaptively weak pseudorandomness game for PEPRFs by operating the ABO EHPS in the extraction mode.

- Setup: $\mathcal{CH}$ runs EHPS.$\mathsf{Setup}(\kappa)$ to generate public parameters pp for PEPRFs, then runs EHPS.$\mathsf{KeyGen}(pp)$ to generate (pk, sk). $\mathcal{CH}$ sends (pp, pk) to $\mathcal{A}$.
- Phase 1 - Evaluation queries: When $\mathcal{A}$ issues evaluation query at point x, $\mathcal{CH}$ parses x as (u, π) then computes $s \leftarrow$ EHPS.$\mathsf{Ext}(sk, u, \pi)$ and responds with $\mathsf{hc}(s)$.
- Challenge: $\mathcal{CH}$ picks random coins $r^* \xleftarrow{\text{R}} R$, samples $s^* \leftarrow$ EHPS.$\mathsf{SampLeft}(r^*)$ and $u^* \leftarrow$ EHPS.$\mathsf{SampRight}(r^*)$, computes $\pi^* = \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Pub}(pk, r^*)$, sets $x^* = (u^*, \pi^*)$, computes $y_0^* \leftarrow \mathsf{hc}(s^*)$, and picks $y_1^* \xleftarrow{\text{R}} \{0,1\}^l$. $\mathcal{CH}$ then picks a random bit $b \in \{0,1\}$, sends (x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Phase 2 - Evaluation queries: same as in Phase 1 except that the query for x^* is not allowed.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b = b'$.

According to the definition of $\mathcal{A}$, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game 1: The same as Game 0 except that $\mathcal{CH}$ prepares the challenge at the setup stage: picks $r^* \xleftarrow{\text{R}} R$, runs EHPS.$\mathsf{SampLeft}(r^*)$ and EHPS.$\mathsf{SampRight}(r^*)$ to obtain (s^*, u^*), and then computes $\pi^* \leftarrow \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Pub}(pk, r^*)$. This change is only conceptual, thus we have:

$$\Pr[S_1] = \Pr[S_0]$$

Game 2: The same as Game 1 except that $\mathcal{CH}$ will abort if $\mathcal{A}$ issues evaluation query at point $x^* = (u^*, \pi^*)$ in Phase 1. We denote the event as F. Suppose Q_1 is the maximum evaluation queries that $\mathcal{A}$ may make in Phase 1. Note that u^* is totally hidden from $\mathcal{A}$ in Phase 1, thus we have $\Pr[F] \leq Q_1/|X| \leq \mathsf{negl}(\kappa)$. Obviously, By the difference lemma, we have:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[F] \leq \mathsf{negl}(\kappa)$$

Game 3: $\mathcal{CH}$ interacts with $\mathcal{A}$ in the security experiment for PEPRFs by operating the ABO EHPS in the ABO hashing mode.

- Setup: same as in Game 2 except that $\mathcal{CH}$ runs EHPS.$\mathsf{KeyGen}'(pp, u^*)$ to generate (pk, sk') and prepares the value $\pi^* \leftarrow \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Pub}(sk', u^*)$.
- Phase 1 - Evaluation queries: When $\mathcal{A}$ issues evaluation query at point $x = (u, \pi)$, $\mathcal{CH}$ responds as follows:
  - If $x = x^*$, then $\mathcal{CH}$ aborts.
  - If $u \neq u^*$ then computes $s = $ EHPS.$\mathsf{Ext}'(sk', u, \pi)$, and responds with $\mathsf{hc}(s)$.
  - If $u = u^*$ but $\pi \neq \pi^*$, $\mathcal{CH}$ responds with $\perp$.
- Challenge: $\mathcal{CH}$ computes $y_0^* = \mathsf{hc}(s^*)$ and picks $y_1^* \leftarrow \{0,1\}^l$, then picks a random bit $b \in \{0,1\}$ and sends (x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Phase 2 - Evaluation queries: The same as in Phase 1 except that the evaluation query at point x^* is not allowed.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b' = b$.

According to the indistinguishability between $\mathsf{KeyGen}(pp)$ and $\mathsf{KeyGen}'(pp, x^*)$ and the correctness of the ABO hashing mode, we have:

$$\Pr[S_3] = \Pr[S_2] \tag{12}$$

We then show that no PPT adversary has non-negligible advantage in Game 3. We prove this by showing that if such an adversary $\mathcal{A}$ exists, then we can construct an algorithm $\mathcal{B}$ that can break the one-wayness of the underlying binary relation with non-negligible advantage. Given (y^*, u^*), $\mathcal{B}$ is asked to determine if $b = 0$ (meaning that $y^* = \mathsf{hc}(s^*)$ where $(s^*, u^*) \in \mathsf{R}$) or $b = 1$ (meaning that $y^* \xleftarrow{\text{R}} \{0,1\}^l$). $\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 3 as follows:

- Setup: $\mathcal{B}$ runs EHPS.$\mathsf{Setup}(\kappa)$ to generate public parameters pp, runs EHPS.$\mathsf{KeyGen}'(pp, u^*)$ to obtain (pk, sk'), computes $\pi^* \leftarrow \mathsf{H}_{pk}(u^*)$ via EHPS.$\mathsf{Priv}(sk', u^*)$. $\mathcal{B}$ sends (pp, pk) to $\mathcal{A}$.
- Phase 1 - Evaluation queries: $\mathcal{B}$ proceeds the same way as $\mathcal{CH}$ does in Game 3.
- Challenge: $\mathcal{CH}$ sets $x^* = (u^*, \pi^*)$, sends (x^*, y^*) to $\mathcal{A}$.
- Phase 2 - Evaluation queries: The same as in Phase 1 except that the evaluation query $\langle x^* \rangle$ is not allowed.
- Guess: $\mathcal{A}$ outputs its guess b' and $\mathcal{B}$ forwards b' to its own challenger.

Obviously, $\mathcal{B}$'s simulation is perfect. Therefore, we have:

$$\mathsf{Adv}_{\mathcal{B}}(\kappa) = |\Pr[S_3] - 1/2| \approx \mathsf{Adv}_{\mathcal{A}}(\kappa)$$

The theorem immediately follows. □

D Review of Standard Definitions

D.1 Trapdoor Functions

A family of trapdoor functions consists of five polynomial-time algorithms as below.

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, output public parameters pp which includes finite sets EK, TD, X, Y (these sets are parameterized by κ).
- $\mathsf{TrapGen}(pp)$: on input pp, output $(ek, td) \in EK \times TD$.
- $\mathsf{SampleDom}(r)$: on input ek and random coins r, output a random $x \in X$.

- $\mathsf{Eval}(ek, x)$: on input ek and $x \in X$, output $\mathsf{TDF}_{ek}(x)$.
- $\mathsf{TdInv}(td, y)$: on input td and $y \in Y$, output $x \in X$ or a distinguished symbol $\perp$ indicating y does not have pre-image.

Correctness: We require that for any $pp \leftarrow \mathsf{Setup}(\kappa)$ any $(ek, td) \leftarrow \mathsf{TrapGen}(pp)$, and any $y = \mathsf{Eval}(ek, x)$, it holds that:

$$\Pr[\mathsf{Eval}(ek, \mathsf{TdInv}(td, y)) = y] = 1$$

Adaptive One-wayness: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an inverter against trapdoor functions and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[x \in \mathsf{TDF}_{ek}^{-1}(y^*) : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ (ek, td) \leftarrow \mathsf{TrapGen}(pp); \\ state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{inv}}(\cdot)}(pp, ek); \\ y^* \leftarrow \mathsf{Eval}(ek, x^*), x^* \leftarrow \mathsf{SampleDom}(r^*); \\ x \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{inv}}(\cdot)}(state, y^*) \end{array} \right],$$

where $\mathcal{O}_{\mathsf{inv}}(y) = \mathsf{TdInv}(td, y)$, and $\mathcal{A}_2$ is not allowed to query $\mathcal{O}_{\mathsf{inv}}(\cdot)$ for the challenge y^*. We say the trapdoor relations is adaptively one-way (or simply adaptively) if for any PPT inverter its advantage is negligible in κ. The standard one-wayness can be defined similarly as above except that the adversary is not given access to the inversion oracle.

D.2 Key Encapsulation Mechanism

A key encapsulation mechanism (KEM) consists of four polynomial-time algorithms:

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, output master public key mpk which includes global public parameters. We assume that mpk also includes the descriptions of ciphertext space C and DEM (data encapsulation mechanism) key space K. mpk will be used as an implicit input for algorithms Encap and Decap,
- $\mathsf{KeyGen}(mpk)$: on input mpk, output a public/secret key pair (pk, sk).
- $\mathsf{Encap}(pk)$: on input public key pk, output a ciphertext c and a DEM key $k \in K$.
- $\mathsf{Decap}(sk, c)$: on input secret key sk and a ciphertext $c \in C$, output a DEM key k or a reject symbol $\perp$ indicating c is invalid.

Correctness: We require that for any $mpk \leftarrow \mathsf{Setup}(\kappa)$, any $(pk, sk) \leftarrow \mathsf{KeyGen}(mpk)$, and any $(c, k) \leftarrow \mathsf{Encap}(pk)$, it holds that:

$$\Pr[\mathsf{Decap}(sk, c) = k] = 1$$

Security: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against KEM and define its advantage as

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[b = b' : \begin{array}{l} mpk \leftarrow \mathsf{Setup}(\kappa); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(mpk); \\ state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{dec}}(\cdot)}(mpk, pk); \\ (c^*, k_0^*) \leftarrow \mathsf{Encap}(pk), k_1^* \xleftarrow{\text{R}} K; \\ b \xleftarrow{\text{R}} \{0, 1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{dec}}(\cdot)}(state, c^*, k_b^*); \end{array} \right] - \frac{1}{2},$$

where $\mathcal{O}_{\mathsf{dec}}(c) = \mathsf{Decap}(sk, c)$, and $\mathcal{A}_2$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}(\cdot)$ for the challenge ciphertext c^*. A KEM is said to be IND-CCA secure if for any PPT adversary $\mathcal{A}$, its advantage defined as above is negligible in κ. The IND-CPA security for KEM can be defined similarly except that the adversary is not allowed to access $\mathcal{O}_{\mathsf{dec}}(\cdot)$.

D.3 Predicate Key Encapsulation Mechanism

A predicate KEM consists of five polynomial-time algorithms:

- $\mathsf{Setup}(\kappa)$: on input security parameter κ, output public parameter pp. We assume that pp includes the descriptions of index set I (index is also usually referred to as attribute), predicate family P, ciphertext space C, and DEM (data encapsulation mechanism) key space K. mpk will be used as an implicit input for algorithms KeyGen, Encap, and Decap.
- $\mathsf{KeyGen}(pp)$: on input mpk, output a master public/secret key pair (mpk, msk).
- $\mathsf{Extract}(msk, p)$: on input msk and $p \in P$, output a secret key sk_p for p.
- $\mathsf{Encap}(ind; r)$: on input $ind \in I$, output a ciphertext c and a DEM key $k \in K$.
- $\mathsf{Decap}(sk_p, c)$: on input a secret key sk_p and ciphertext $c \in C$, output a DEM key $k \in K$ or a reject symbol $\perp$ indicating c is invalid.

Correctness: We require that for any $pp \leftarrow \mathsf{Setup}(\kappa)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any $ind \in I$, any $(c, k) \leftarrow \mathsf{Encap}(ind)$, and any $sk_p \leftarrow \mathsf{KeyGen}(msk, p)$ satisfying $p(ind) = 1$, it holds that: $\Pr[\mathsf{Decap}(sk_p, c) = k] = 1$.

Security: Here we only focus on a "basic" level of security named payload hiding which guarantees, intuitively, that a ciphertext associated with index ind hides all information about the underlying message m unless one is in possession of a secret key giving the explicit ability to decrypt, i.e., if an adversary $\mathcal{A}$ holds keys $sk_{p_1}, \ldots, sk_{p_l}$, then $\mathcal{A}$ learns nothing about the message if $p_1(ind) = \cdots = p_l(ind) = 0$. Formally, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against predicate KEM and define its advantage in the following experiment:

$$\mathrm{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\kappa); \\ (mpk, msk) \leftarrow \mathsf{KeyGen}(pp); \\ (state, ind^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{keygen}}(\cdot), \mathcal{O}_{\mathsf{dec}}(\cdot,\cdot)}(mpk); \\ (k_0^*, c^*) \leftarrow \mathsf{Encap}(ind^*), k_1^* \xleftarrow{\text{R}} K; \\ b \xleftarrow{\text{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{keygen}}(\cdot), \mathcal{O}_{\mathsf{dec}}(\cdot,\cdot)}(state, c^*, k_b^*); \end{array} \right] - \frac{1}{2},$$

where $\mathcal{O}_{\mathsf{dec}}(ind, c) = \mathsf{Decap}(sk_p, c)$ where $p(ind) = 1$, and in the second phase $\mathcal{A}_2$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}(\cdot, \cdot)$ with the challenge (ind^*, c^*); $\mathcal{O}_{\mathsf{keygen}}(p) = \mathsf{KeyGen}(msk, p)$, and throughout the experiment $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{keygen}}(\cdot)$ with the predicate p such that $p(ind^*) = 1$. A predicate KEM is said to be IND-CCA secure if for any PPT adversary $\mathcal{A}$, its advantage defined as above is negligible in κ. The IND-CPA security for predicate KEM can be defined similarly except that the adversary is allowed to access $\mathcal{O}_{\mathsf{dec}}(\cdot, \cdot)$.

D.4 Circuit Notation

We now define our notation for circuits that adapts the model and notation of Bellare, Hoang, and Rogaway [BHR12, Section 2.3]. For our application we restrict our attention to certain classes of boolean circuits. First, our circuits have a single output gate. Next, we only consider layered circuits. In a layered circuit a gate at depth j will receive both of its inputs from wires at depth $j - 1$. Finally, we will restrict to monotonic circuits where gates are either AND or OR gates of two inputs.[10]

[10]These restrictions are mostly useful for exposition and do not impact functionality. General circuits can be built from non-monotonic circuits. In addition, given a circuit an equivalent layered exists that is larger by at most a polynomial factor.

Our circuits is a five tuple $c = (n, q, \mathsf{A}, \mathsf{B}, \mathsf{GateType})$. We let n be the number of inputs and q be the number of gates. We define $Inputs = \{1, \ldots, n\}$, $Wires = \{1, \ldots, n+q\}$, $Gates = \{n+1, \ldots, n+q\}$. The wire $n+q$ is the designated output wire. $\mathsf{A} : Gates \rightarrow Wires$ is a function where $\mathsf{A}(w)$ identifies w's first incoming wire and $\mathsf{B} : Gates \rightarrow Wires$ is a function where $\mathsf{B}(w)$ identifies w's second incoming wire. Finally, $\mathsf{GateType} : Gates \rightarrow \{\mathrm{AND}, \mathrm{OR}\}$ is a function that identifies a gate as either an AND or OR gate.

For any $w \in Gates$ we require that $w > B(w) > A(w)$. We also define a function $\mathsf{depth}(w)$ where if $w \in Inputs$ then $\mathsf{depth}(w) = 1$ and in general $\mathsf{depth}(w)$ of wire w is equal to the shortest path to an input wire plus 1. Since our circuits is layered we require that for all $w \in Gates$ that if $\mathsf{depth}(w) = j$ then $\mathsf{depth}(\mathsf{A}(w)) = \mathsf{depth}(\mathsf{B}(w)) = j - 1$. We will abuse notation and let $c(x)$ be the evaluation of the circuit c on input $x \in \{0,1\}^n$. In addition, we let $c_w(x)$ be the value of wire w of the circuit on input x.

D.5 Multilinear Maps and the MDDH Assumption

A k-group system consists of k cyclic groups $\mathbb{G}_1, \cdots, \mathbb{G}_k$ of prime order p, along with bilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \rightarrow \mathbb{G}_{i+j}$ for all $i, j \geq 1$ and $i + j \leq k$. Let g_i be a canonical generator of $\mathbb{G}_i$ (included in the group's description). For any $a, b \in \mathbb{Z}_p$ the map $e_{i,j}$ satisfies $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$. When i, j are clear, we will simply write e instead of $e_{i,j}$. It will also be convenient to abbreviate $e(h_1, \ldots, h_j) := e(h_1, e(h_2, \ldots, e(h_{j-1}, h_j)))$ for $h_j \in \mathbb{G}_{i_j}$ and $i_1 + \cdots + i_j \leq k$. By induction, it is easy to see that this map j-linear. Additionally, we define $e(g) := g$. Finally, it will also be useful to define the group $\mathbb{G}_0 = \mathbb{Z}_p$ of exponents to which this pairing family naturally extends. In the following, we denote $\mathsf{MLGroupGen}$ by multilinear maps parameter generator which on input security parameter κ and level k, output a k-group system $(p, \{\mathbb{G}_i\}_{i \in [k]}, \{e_{i,j}\}_{i,j \geq 1, i+j \leq k})$.

Assumption D.1 (k-Multilinear Decisional Diffie-Hellman Assumption: k-MDDH)**.** Let $(p, \{\mathbb{G}_i\}_{i \in [k]}, \{e_{i,j}\}_{i,j \geq 1, i+j \leq k})$ be a k-group system generated by $\mathsf{MLGroupGen}(\kappa, k)$. For any PPT adversary $\mathcal{A}$ it holds that:

$$|\Pr[\mathcal{A}(g, g^{c_1}, \ldots, g^{c_{k+1}}, T_b) = 1] - 1/2| \leq \mathsf{negl}(\kappa)$$

where $T_0 = g_k^{\prod_{j=1}^{k+1} c_j} \in \mathbb{G}_k$, $T_1 \xleftarrow{\mathrm{R}} \mathbb{G}_k$. The probability is taken over the random coins used by $\mathsf{MLGroupGen}(\kappa, k)$ and picking $g \xleftarrow{\mathrm{R}} \mathbb{G}_1$, $c_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, $b \xleftarrow{\mathrm{R}} \{0,1\}$.

D.6 RSA Assumption

Assumption D.2 (RSA Assumption [RSA78])**.** Let (N, p, q) be a RSA parameter generated by $\mathsf{RSAGen}(\kappa)$, where N is the product of two κ-bit, distinct odd primes p, q. For any PPT adversary $\mathcal{A}$ it holds that:

$$|\Pr[\mathcal{A}(N, e, x) = y \text{ s.t. } y^e \equiv x \bmod N]| \leq \mathsf{negl}(\kappa)$$

where e be randomly chosen positive integer less than and relatively prime to $\phi(N) = (p - 1)(q - 1)$, $y \xleftarrow{\mathrm{R}} \mathbb{Z}_N^*$.