Publicly Evaluable Pseudorandom Functions and Their Applications

Yu Chen *
yuchen.prc@gmail.com

Zongyang Zhang †
zongyang.zhang@gmail.com

December 1, 2014

Abstract

We put forth the notion of *publicly evaluable* pseudorandom functions (PEPRFs), which is a non-trivial extension of the standard pseudorandom functions (PRFs). Briefly, PEPRFs are defined over domain X containing an NP language L in which the witness is hard to extract on average, 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 cannot be distinguished from a uniform random function at randomly chosen inputs. The strengthened one is adaptively 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 scheme (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 constrained PRFs, and introduce the notion of publicly evaluable constrained PRFs, which, as an immediate application, implies predicate encryption.
- We propose a variant of PEPRFs, which we call publicly evaluable and verifiable functions (PEVFs). Compared to PEPRFs, PEVFs have an additional promising property named public verifiability while the best possible security degrades to being hard to compute on average. We show how to construct PEVFs from EHPS for publicly verifiable relation. Moreover, we justify the applicability of PEVFs by presenting a

*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China. Yu Chen is supported by the National Natural Science Foundation of China under Grant No. 61303257, No. 61379141, the IIE's Cryptography Research Project, the Strategic Priority Research Program of CAS under Grant No. XDA06010701, and the National 973 Program of China under Grant No. 2011CB302400.

†Research Institute for Secure Systems, National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan. Zongyang Zhang is an International Research Fellow of JSPS and supported by the National Natural Science Foundation of China under Grant No. 61303201.

simple construction of "hash-and-sign" signatures, both in the random oracle model and the standard model.

Keywords: publicly evaluable, PRF, HPS, EHPS, TDF

1 Introduction

Pseudorandom functions (PRFs) [GGM86] are a fundamental concept in modern cryptography. Loosely speaking, PRFs are a family of keyed functions $\mathsf{F}_{sk} : X \to Y$ such that: (1) it is easy to sample the functions and compute their values, i.e., given a 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 keys. In a publicly evaluable PRF, there exists an 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 corresponding public key pk and a witness w for $x \in L$. Regarding the security requirement for PEPRFs, we require weak pseudorandomness which ensures that no PPT adversary can distinguish F_{sk} from a real random function on uniformly distributed challenge points in L (this differs from the standard pseudorandomness for PRFs in which the challenge points are arbitrarily chosen by an adversary).

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 simple application is an elegant construction of private-key encryption as follows: the secret key sk of PRFs 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 whether PRFs also yield PKE in the same way. However, the above 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 violates the pseudorandomness of PRFs) and thus encrypting publicly is impossible. Moreover, since PRFs and one-way functions (OWFs) imply each other [GGM86, HILL99], the implications of PRFs are inherently confined in Minicrypt (the hypothetical world defined by Impagliazzo where one-way functions exist, but public-key cryptography does not [Imp95]). 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 of PRFs can translate the above construction into 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 conceptual extensions of standard PRFs. In a PEPRF, there is an 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 adaptively 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, a 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 a ciphertext $(x, m \oplus \mathsf{F}_{sk}(x))$. To decrypt, a 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 well-known ElGamal PKE can be explained neatly by weak-pseudorandom PEPRFs 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, a sender first produces a DEM key by picking $x \xleftarrow{R} X$,[3] then generates the associated ciphertext $\mathsf{TDF}_{ek}(x)$; while in the construction from PEPRFs, a sender first generates a ciphertext by picking $x \xleftarrow{R} X$, then produces the associated DEM key $\mathsf{F}_{sk}(x)$.
- In Section 5 and Section 6, we show that both smooth hash proof systems (HPS) and extractable hash proof systems (EHPS) yield weak-pseudorandom PEPRFs, while both smooth plus universal$_2$ HPS and all-but-one EHPS yield adaptively weak-pseudorandom PEPRFs, respectively. This means that the works on HPS and EHPS implicitly constructed PEPRFs. Therefore, PEPRFs are an abstraction of the common aspect of the HPS and EHPS which are 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 implies 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. This implication helps 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 8, we introduce an extension of PEPRFs named publicly evaluable constrained PRFs. An immediate application of publicly evaluable constrained PRFs is predicate encryption. We present a concrete construction based on recent attribute-based encryption from multilinear maps [GGH+13].

[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.

- In Section 9, we propose a variant of PEPRFs named publicly evaluable and verifiable functions (PEVFs). We show how to construct PEVFs from EHPS for publicly verifiable relations. Moreover, we demonstrate the utility of PEVFs by presenting a simple construction of "hash-and-sign" signatures, both in the random oracle model and the standard model.

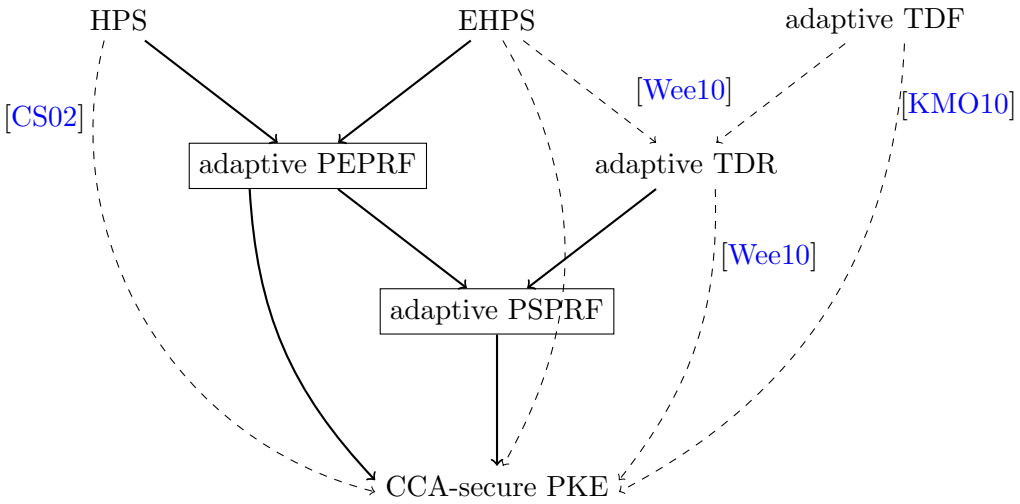

Figure 1: Summary of CCA-secure PKEs from paradigms and general assumptions. Here, HPS refers to smooth plus universal$_2$ HPS, and EHPS refers to its all-but-one variant. The bold lines and rectangles denote our contributions, while the dashed lines denote those that are straight-forward or from previous work. All of the constructions from general assumptions are black-box.

1.3 Related Work

General assumptions vs. Paradigms. We first try to explain the terms of "general assumption" and "paradigm" in the context of cryptography according to our understanding. Roughly speaking, a general assumption (also referred to generic cryptographic assumption or primitive) usually consists of one set of algorithms, which is used to fulfill some functionality and is expected to satisfy some desired security. Examples of general assumption include one-way (trapdoor) functions/permutations and pseudorandom generators/functions, etc. Different from general assumption, a paradigm usually consists of two sets of algorithms, where 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 assumptions are more abstract than paradigms.

CCA-secure PKE from general assumptions or paradigms. Except the effort on constructing CCA-secure PKE from specific assumptions [HK08, MH14] or from encryption schemes satisfying some weak security notions [NY90, DDN00, BCHK07, CHK10, HLW12, DS13, LT13], it is mostly of theoretical interest to build CCA-secure PKE from general assumptions and

paradigms. 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. Kurosawa and Desmedt [KD04] and Kiltz et al. [KPSY09] later improved upon the original HPS paradigm. Peikert and Waters [PW08] proposed lossy trapdoor functions (LTDFs) and showed a black-box construction of CCA-secure PKE from them. Rosen and Segev [RS10] introduced correlated-product secure trapdoor functions (CP-TDFs) and also showed a construction of CCA-secure PKE from them. Moreover, they showed that CP-TDFs are strictly weaker than LTDFs by giving a black-box separation between them. Kiltz et al. [KMO10] introduced (injective) adaptive trapdoor functions (ATDFs) which are strictly weaker than both LTDFs and CP-TDFs but suffice to imply CCA-secure PKE. Wee [Wee10] introduced extractable hash proof system (EHPS) and used it as a paradigm to construct CCA-secure PKE from various search assumptions. Wee also showed that both EHPS and ATDFs imply (injective) adaptive trapdoor relations (ATDRs), which are sufficient to imply CCA-secure PKE. To the best of our knowledge, ATDR is the weakest general assumption that implies CCA-secure PKE. Very recently, Sahai and Waters [SW14] successfully translated the PRF-based private-key encryption to PKE by using punctured program technique in conjunction with indistinguishability obfuscation ($i\mathcal{O}$). Due to the use of obfuscation, their construction is inherently non-black-box.

Constrained PRFs. Very recently, constrained 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 [BGI14] under the name of *functional PRFs*. In constrained PRFs, secret key admits delegation for a family of predicates, and the delegated key for predicate p enable one to compute the PRF value 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, and optimal private broadcast encryption.

Witness PRFs. Independently and concurrently of our work, Zhandry [Zha14] introduces the notion of *witness PRFs* (WPRFs), which are similar in concept to PEPRFs. In a nutshell, both WPRFs and PEPRFs are defined with respect to NP languages and extend the standard PRFs with the same extra functionality, i.e., one can publicly evaluate $\mathsf{F}_{sk}(x)$ for $x \in L$ with the knowledge of the corresponding witness. The main differences between WPRFs and our PEPRFs are as follows:

1. WPRFs can handle arbitrary NP languages, while PEPRFs are only for NP languages whose witness is hard to extract on average.
2. WPRFs require that $\mathsf{F}_{sk}(x)$ is pseudorandom for any adversarially chosen $x \in X \backslash L$, while PEPRFs only require that $\mathsf{F}_{sk}(x)$ is pseudorandom for randomly chosen $x \in L$.

WPRFs are introduced as a weaker primitive for several obfuscation-based applications. By utilizing the reduction from any NP language to the subset-sum problem, WPRFs can handle arbitrary NP languages. However, for applications of WPRFs whose functionalities rely on $\mathsf{F}_{sk}(x)$ for $x \in L$, such as CCA-secure encryption, non-interactive key exchange, and hardcore functions for any one-way function, the underlying NP languages have to be at least hard-on-average. This is because these applications usually need the indistinguishability between $x \xleftarrow{\mathrm{R}} L$ and $x \xleftarrow{\mathrm{R}} X \backslash L$ to argue $\mathsf{F}_{sk}(x)$ is computationally pseudorandom for $x \xleftarrow{\mathrm{R}} L$.

2 Preliminaries and Definitions

Notations. For a distribution or random variable X, we write $x \xleftarrow{R} X$ to denote the operation of sampling a random x according to X. For a set X, we use $x \xleftarrow{R} X$ 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, which 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 $z \leftarrow \mathcal{A}(x_1, \ldots, x_n)$. The statistical distance between two random variables X and Y having a common domain Ω is $\Delta[X, Y] = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$. We also define the conditional statistical distance as $\Delta_Z[X, Y] = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X = \omega|Z] - \Pr[Y = \omega|Z]|$. The min-entropy of a random variable X over domain Ω is defined as $\mathrm{H}_\infty = -\log_2(\max_{\omega\in\Omega} \Pr[X = \omega])$.

2.1 Pseudorandom Functions

We first recap the definition of standard PRFs in order to compare it with PEPRFs more clearly.

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

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters $pp = (\mathsf{F}, SK, X, Y)$ (the sets SK, X, and Y may be parameterized by κ), where $\mathsf{F} : SK \times X \to Y$ can be viewed as a keyed function indexed by SK, namely $\mathsf{F} = \{\mathsf{F}_{sk}\}_{sk\in SK}$.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key $sk \in SK$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, 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 PRFs is pseudorandomness. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against PRFs 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 $\mathcal{O}_{\mathsf{ror}}(0, x) = \mathsf{F}_{sk}(x)$, $\mathcal{O}_{\mathsf{ror}}(1, x) = \mathsf{H}(x)$ (here H is chosen uniformly at random from all the functions from X to Y[4]). Note that $\mathcal{A}$ can adaptively access the oracle $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ polynomial many 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.

[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 repeatedly are answered consistently.

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 Weak PRFs

In the original syntax of PRFs, the input-sampling algorithm is not implicitly given. Note that in the weak PRF security experiment the challenge points are sampled by the challenger, thus it is convenient to explicitly introduce the input-sampling algorithm to obtain more refined security notions for weak PRFs. Hereafter, let $\mathsf{SampDom}$ be an algorithm that takes as input random coins r and outputs an element $x \in X$. Without loss of generality, we assume the distribution of x induced by $\mathsf{SampDom}(r)$ conditioned on $r \xleftarrow{\mathrm{R}} R$ is statistically close to $x \xleftarrow{\mathrm{R}} X$. Hence, in the weak PRF security experiment the challenger can sample elements uniformly at random from X by running $\mathsf{SampDom}$ with random coins $r \xleftarrow{\mathrm{R}} R$. Depending on whether the random coins r can be made public, weak PRFs are further divided into secret-coin and public-coin weak PRFs [PS08]. Secret-coin weak PRFs require weak pseudorandomness holds if the random coins are kept secret, whereas public-coin weak PRFs requires the weak pseudorandomness holds even the random coins are made public. Clearly, whether a weak PRF is public-coin or just secret-coin secure depends on the input-sampling algorithm.

3 Publicly Evaluable PRFs

Here we define PEPRFs. We begin with the syntax and then define the security.

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

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp which include $(\mathsf{F}, PK, SK, X, L, W, Y)$, where $\mathsf{F} : SK \times X \to Y \cup \perp$ could be viewed as a keyed function indexed by SK, L is an NP language defined over X, and W is the set of associated witnesses.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key sk and an associated public key pk.[5]
- $\mathsf{SampLan}(r)$: 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:

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

[5] In standard PRF, it is also harmless to explicitly introduce public key, 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: $\mathsf{F}_{\vec{a}}(x) = (g^{a_0})^{\prod_{x_i=1} a_i}$, where $\vec{a} = (a_0, a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ is the secret key, g^{a_i} for $1 \le i \le n$ can be safely published as the public key. If no information can be made public, one can always assume $pk = \{\perp\}$.

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

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

where $p(\kappa)$ is any polynomial, $\mathcal{O}_{\mathsf{eval}}(x) = \mathsf{F}_{sk}(x)$, $\mathcal{O}_{\mathsf{ror}}(0, x) = \mathsf{F}_{sk}(x)$, $\mathcal{O}_{\mathsf{ror}}(1, x) = \mathsf{H}(x)$, 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.[6] The adaptive weak-pseudorandomness captures the security against active adversaries, who are given adaptive access to oracle $\mathcal{O}_{\mathsf{eval}}(\cdot)$. We also consider weak-pseudorandomness which captures the security against static adversaries, who is not given access to oracle $\mathcal{O}_{\mathsf{eval}}(\cdot)$.

On the basis of previous works [BK03, BC10, HLAWW13], one can also define more advanced security notions such as security against related-key attacks and leakage-resilience for PEPRFs.

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 cannot 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 that in the NP language L the extraction of witness must be hard to extract on average. In fact, this requirement is strictly weaker than the NP language L itself is hard on average, which is justified by the construction in following example 3.1.

Remark 3.2. In some scenarios, it is more convenient to work with a definition that slightly restricts an adversary's power, but is equivalent to Definition 3.1. That is, $p(\kappa)$ is fixed to 1. Due 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.

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 NP 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 the PEPRFs are weak-pseudorandom assuming the DDH assumption holds in $\mathbb{G}$. Looking ahead, when applying the construction shown in Section 4 to the PEPRFs, yields exactly the plain ElGamal PKE.

A possible relaxation. To be completely precise, it is not necessary to require the distribution of x induced by $\mathsf{SampLan}(r)$ conditioned on $r \xleftarrow{\mathrm{R}} R$ 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

[6] The readers should not confuse with adaptive PRFs [BH12], where "adaptive" means that instead of deciding the queries in advance, an adversary can adaptively make queries to $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ based on previous queries.

[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.

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 notion 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 Weak PRFs

It is easy to see that PEPRFs naturally imply secret-coin weak PRFs by letting the input-sampling algorithm simply run $(x, w) \leftarrow \mathsf{SampLan}(r)$ and only output x. In fact, PEPRFs can be viewed as a special case of secret-coin weak PRFs, where the weak PRF security completely breaks down if the random coins used to sample the challenge inputs are revealed. Also, every public-coin weak PRF is clearly a secret-coin weak PRF. We depict the relations among secret-coin, public-coin, and publicly evaluable PRFs in Figure 2. On one hand, Pietrzak and Sjödin [PS08] demonstrated that the existence of a secret-coin weak PRF which is not also a public-coin weak PRF implies the existence of two pass key-agreement and thus two pass public-key encryption. This result indicates that PEPRFs must be very artificial in Minicrypt. On the other hand, as we will see shortly, PEPRFs admit a black-box construction of PKE, which implies that PEPRFs are strictly stronger than PRFs (in a black-box sense). The results from the above two hands agree with each other.

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 and $\{\mathsf{H}_{sk_2} : X_2 \to Y_2\}_{sk_2 \in SK_2}$ is a PEPRF, 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 from $X_1 \times X_2$ to $Y_1 \times Y_2$ indexed by $SK_1 \times SK_2$.

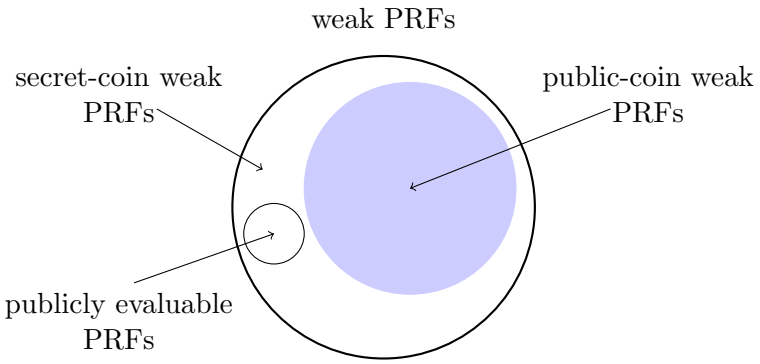

Figure 2: Relations among secret-coin, public-coin, and publicly evaluable PRFs.

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 A.1 for the definition and security notion of KEM.

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

Correctness of the above KEM construction follows immediately from correctness 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 IND-CPA security of the KEM to a distinguisher $\mathcal{B}$ against weak-pseudorandomness of PEPRFs. We proceed via a single game.

Game CPA: $\mathcal{B}$ receives (pp, pk) of PEPRFs, then simulates $\mathcal{A}$'s challenger in the IND-CPA experiment as follows:
Setup: $\mathcal{B}$ sends (pp, pk) to $\mathcal{A}$.
Challenge: $\mathcal{B}$ receives (x^*, y^*) from its own challenger, where $(x^*, w^*) \leftarrow \mathrm{PRF.}\mathsf{SampLan}(r^*)$ 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. Therefore, $\mathcal{B}$ can break weak-pseudorandomness of PEPRFs with advantage at least $\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}}(\kappa)$. 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 IND-CCA security of the KEM construction to a distinguisher $\mathcal{B}$ against adaptive weak-pseudorandomness of PEPRFs. We proceed via a single game.

Game CCA: $\mathcal{B}$ receives (pp, pk) of PEPRFs, then simulates $\mathcal{A}$'s challenger in the IND-CCA experiment as follows:
Setup: $\mathcal{B}$ sends (pp, 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 $(x^*, w^*) \leftarrow \mathrm{PRF.}\mathsf{Sample}(r^*)$ 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. Therefore, $\mathcal{B}$ can break the adaptively weak-pseudorandomness of PEPRFs with advantage at least $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}}(\kappa)$. This concludes the proof. □

The above results also hold if the underlying PEPRFs are (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 $(\Lambda, SK, PK, X, L, W, \Pi, \alpha)$, where $\Lambda : SK \times X \to \Pi$ can be viewed as a keyed function indexed by SK, L is an NP 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 $\pi = \Lambda_{sk}(x)$.
- $\mathsf{Pub}(pk, x, w)$: on input pk and $x \in L$ together with a witness w for x, output π such that $\pi = \Lambda_{sk}(x)$.

Following [CS02, KPSY09] we define the following properties and results as below.

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 Λ on input $x \in X \backslash L$.

Definition 5.2 (collision probability)**.** The collision probability of projective hash Λ is defined as:

$$\delta = \max_{x, x^* \in X \backslash L, x \neq x^*} (\Pr_{sk}[\Lambda_{sk}(x) = \Lambda_{sk}(x^*)]).$$

Definition 5.3 (universal$_1$)**.** The projective hash Λ is ϵ-universal$_1$ if for all $x \in X \backslash L$,

$$\Delta[(pk, \Lambda_{sk}(x)), (pk, \pi)] \leq \epsilon,$$

where in the above $sk \xleftarrow{R} SK$ conditioned on $\alpha(sk) = pk$ and $\pi \xleftarrow{R} \Pi$.

[CS02] introduced the following relaxation of the universal$_1$ property which only requires that it holds in the average case.

Definition 5.4 (smooth)**.** The projective hash Λ is ϵ-smooth if for $x \xleftarrow{R} X \backslash L$,

$$\Delta[(pk, \Lambda_{sk}(x)), (pk, \pi)] \leq \epsilon,$$

where in the above $sk \xleftarrow{R} SK$ conditioned on $\alpha(sk) = pk$ and $\pi \xleftarrow{R} \Pi$.

[KPSY09] introduced another relaxation of the universal$_1$ property which only requires that for all $x \in X \backslash L$, given $pk = \alpha(sk)$, $\mathsf{H}_{sk}(x)$ has high min-entropy.

Definition 5.5 (k-entropic)**.** The projective hash Λ is ϵ-k-entropic if for all $x \in X \backslash L$,

$$\Pr[\mathrm{H}_\infty(\Lambda_{sk}(x) | pk) \geq k] \geq 1 - \epsilon$$

where in the above $sk \xleftarrow{R} SK$ conditioned on $\alpha(sk) = pk$.

Lemma 5.1 ([KPSY09])**.** *Every ϵ-universal$_1$ projective hash is ϵ-k-entropic, for $k = \log_2 |\Pi|$.*

Definition 5.6 (universal$_2$)**.** The projective hash Λ is ϵ-universal$_2$ if for all $x, x^* \in X \backslash L$ with $x \neq x^*$,

$$\Delta[(pk, \Lambda_{sk}(x^*), \Lambda_{sk}(x)), (pk, \Lambda_{sk}(x^*), \pi)] \leq \epsilon$$

where in the above $sk \xleftarrow{R} SK$ conditioned on $\alpha(sk) = pk$ and $\pi \xleftarrow{R} \Pi$.

A Generic Transform from k-entropic to universal$_2$ HPSs. The following transformation was given in [KPSY09]. Given a HPS with projective hash $\Lambda : SK \times X \to \Pi$ regarding projection $\alpha : SK \to PK$ and a family of functions $H = \{\mathsf{H} : \Pi \to \{0,1\}^\ell\}$, we can define its hashed variant HPS^H with projective hash $\Lambda^H : SK^H \times X \to \{0,1\}^\ell$ regarding $\alpha^H : SK^H \to PK^H$ where $PK^H = PK \times H$, $SK^H = SK \times H$, $\Lambda^H((sk, \mathsf{H}), x) = \mathsf{H}(\Lambda(sk, x))$ and $\alpha^H(sk, \mathsf{H}) = (\alpha(sk), \mathsf{H})$. Note that X and L are the same for HPS and HPS^H.

Theorem 5.2 ([KPSY09])**.** *Assume HPS is ϵ_1 k-entropic with collision probability $\delta \leq 1/2$ and $H = \{\mathsf{H} : \Pi \to \{0,1\}^\ell\}$ (where $\ell \geq 6$) is a family of 4-wise independent hash functions, then HPS^H is ϵ_2-universal$_2$ for:*

$$\epsilon_2 = 2^{\ell - \frac{\kappa-1}{2}} + 3\epsilon_1 + \delta$$

It is easy to see that universal$_2$ property implies universal$_1$ property, while universal$_1$ property further implies smooth property. 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 5.7 (weak-universal$_2$)**.** The projective hash Λ is ϵ-weak-universal$_2$ if for $x^* \xleftarrow{\mathrm{R}} X \backslash L$ and all $x \in X \backslash L$ with $x \neq x^*$,

$$\Delta[(pk, \Lambda_{sk}(x^*), \Lambda_{sk}(x)), (pk, \Lambda_{sk}(x^*), \pi)] \leq \epsilon,$$

where in the above $sk \xleftarrow{\mathrm{R}} SK$ conditioned on $\alpha(sk) = pk$ and $\pi \xleftarrow{\mathrm{R}} \Pi$.

5.1 Construction from Smooth HPS

From smooth HPS, we construct weak-pseudorandom PEPRFs as follows:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, run HPS.$\mathsf{Setup}(\kappa)$ to generate an HPS instance $(\Lambda, PK, SK, X, L, \Pi, \alpha)$, then produces public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs from it, where $\mathsf{F} = \Lambda$, $Y = \Pi$. 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.$\mathsf{KeyGen}(pp)$.
- $\mathsf{SampLan}(r)$: 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 5.3. *If the underlying subset membership problem is hard, then the PEPRFs from smooth HPS are weak-pseudorandom.*

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 $(x^*, w^*) \leftarrow \text{HPS.}\mathsf{Sample}(r^*)$, then computes $\pi^* \leftarrow \Lambda_{sk}(x^*)$ via $\text{HPS.}\mathsf{Pub}(pk, x^*, w^*)$, sets $y_0^* = \pi^*$, 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 $\pi^* \leftarrow \Lambda_{sk}(x^*)$ via $\text{HPS.}\mathsf{Priv}(sk, x^*)$. 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 $\text{HPS.}\mathsf{Sample}'$ instead of $\text{HPS.}\mathsf{Sample}$. The sampling indistinguishability (based on the hardness of the subset membership problem) ensures that:

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

Game 3: same as Game 2 except that in the challenge stage x^* is set as π', where $\pi' \xleftarrow{R} \Pi$. It now follows directly from the ϵ-smooth property of HPS that:

$$|\Pr[S_3] - \Pr[S_2]| \leq \epsilon \tag{4}$$

It is evident from the definition of Game 3 that $\mathcal{A}$'s output b' is independent of the hidden bit b, therefore:

$$\Pr[S_3] = 1/2 \tag{5}$$

Putting all these above, the theorem immediately follows. □

5.2 Construction from Smooth and Weak-Universal$_2$ HPS

From smooth HPS and associated weak-universal$_2$ HPS, we can construct adaptively weak-pseudorandom PEPRFs as follows:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, run $\text{HPS}_1.\mathsf{Setup}(\kappa)$ to generate a smooth HPS instance $pp_1 = (\Lambda^1, PK_1, SK_1, \tilde{X}, \tilde{L}, W, \Pi_1, \alpha_1)$, run $\text{HPS}_2.\mathsf{Setup}(\kappa)$ to generate a weak-universal$_2$ HPS instance $pp_2 = (\Lambda^2, PK_2, SK_2, \tilde{X}, \tilde{L}, W, \Pi_2, \alpha_2)$, then build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs from pp_1 and pp_2, where $X = \tilde{X} \times \Pi_2$, $Y = \Pi_1 \cup \perp$, $PK = PK_1 \times PK_2$, $SK = SK_1 \times SK_2$, F, L, and W will be defined later. We assume the two HPSs share a common sampling algorithm and $pp = pp_1 \cup pp_2$.
- $\mathsf{KeyGen}(pp)$: on input $pp = pp_1 \cup pp_2$, run $\text{HPS}_1.\mathsf{KeyGen}(pp_1)$ and $\text{HPS}_2.\mathsf{KeyGen}(pp_2)$ to get (pk_1, sk_1) and (pk_2, sk_2) respectively, output $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$.
- $\mathsf{SampLan}(pk; r)$: on input $pk = (pk_1, pk_2)$ and random coins r, run $(\tilde{x}, w) \leftarrow \text{HPS}_1.\mathsf{Sample}(r)$, compute $\pi_2 \leftarrow \text{HPS}_2.\mathsf{Pub}(pk_2, \tilde{x}, w)$. This sampling algorithm defines a collection of language $L = \{L_{pk}\}_{pk \in PK}$ over $X = \tilde{X} \times \Pi_2$ where each $L_{pk} = \{(\tilde{x}, \pi_2) : \tilde{x} \in \tilde{L} \wedge \pi_2 = \text{HPS}_2.\mathsf{Pub}(pk_2, \tilde{x}, w)\}$. It is easy to see 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 $y \leftarrow \mathrm{HPS}_1.\mathsf{Pub}(pk_1, \tilde{x}, w)$.
- $\mathsf{PrivEval}(sk, x)$: on input $sk = (sk_1, sk_2)$ and $x = (\tilde{x}, \pi_2)$, output $y \leftarrow \mathrm{HPS}_1.\mathsf{Priv}(sk_1, \tilde{x})$ if $\pi_2 = \mathrm{HPS}_2.\mathsf{Priv}(sk_2, \tilde{x})$ and $\perp$ otherwise.

We have the following theorem of the above construction.

Theorem 5.4. *If the underlying subset membership problem is hard, then the PEPRFs from smooth and weak-universal$_2$ HPSs are adaptively weak-pseudorandom.*

Proof. The proof is similar to [CS02]. To base adaptively weak-pseudorandomness 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 guess 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 $\mathrm{HPS}_1.\mathsf{Setup}(\kappa)$ and $\mathrm{HPS}_2.\mathsf{Setup}(\kappa)$ 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 $(pk_1, sk_1) \leftarrow \mathrm{HPS}_1.\mathsf{KeyGen}(pp_1)$ and $(pk_2, sk_2) \leftarrow \mathrm{HPS}_2.\mathsf{KeyGen}(pp_2)$, 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 $\Lambda^1_{sk_1}(\tilde{x})$ if $\Lambda^2_{sk_2}(\tilde{x}) = \pi_2$ and $\perp$ otherwise.
- Challenge: $\mathcal{CH}$ picks $r^* \xleftarrow{\mathrm{R}} R$, sets $(\tilde{x}^*, w^*) \leftarrow \mathrm{HPS}_1.\mathsf{Sample}(r^*)$, computes $\pi_2^* = \Lambda^2_{sk_2}(\tilde{x}^*)$ via $\mathrm{HPS}_2.\mathsf{Pub}(pk_2, \tilde{x}^*, w^*)$, sets $x^* = (\tilde{x}^*, \pi_2^*)$. $\mathcal{CH}$ then computes $\pi_1^* = \Lambda^1_{sk_1}(\tilde{x}^*)$ via $\mathrm{HPS}_1.\mathsf{Pub}(pk_1, \tilde{x}^*, w^*)$, sets $y_0^* = \pi_1^*$, samples $y_1^* \xleftarrow{\mathrm{R}} Y$, picks a random bit $b \in \{0, 1\}$, and 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:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2| \tag{6}$$

Game 1: same as Game 0 except that in the challenge stage $\mathcal{CH}$ computes $\pi_2^* = \Lambda^2_{sk_2}(\tilde{x}^*)$ via $\mathrm{HPS}_2.\mathsf{Priv}(sk_2, \tilde{x}^*)$ and computes $\pi_1^* = \Lambda^1_{sk_1}(\tilde{x}^*)$ via $\mathrm{HPS}_1.\mathsf{Priv}(sk_2, \tilde{x}^*)$. According to the functionality of HPS.Priv and HPS.Pub, this change is perfectly hidden from $\mathcal{A}$. Thus, we have:

$$\Pr[S_1] = \Pr[S_0] \tag{7}$$

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

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\kappa) \tag{8}$$

Game 3: same as Game 2 except that when answering decapsulation queries $\langle x \rangle$ where $x = (\tilde{x}, \pi_2)$, $\mathcal{CH}$ returns $\perp$ when $\tilde{x} \notin \tilde{L}$ even $\Lambda^2_{sk_2}(\tilde{x}) = \pi_2$. For the ease of analysis, we denote by E 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

$\Lambda^2_{sk_2}(\tilde{x}) = \pi_2$. According to the weak-universal$_2$ property of HPS$_2$, we have $\Pr[E] \leq Q\epsilon$, where Q is the maximum number of PRF evaluation queries that $\mathcal{A}$ may make. Since ϵ is negligible in κ, we have:

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

Game 4: same as Game 3 except that in the challenge stage y_0^* is set as π_1', where $\pi_1' \xleftarrow{R} \Pi_1$. Now, let us condition on a fixed value of pk_2, b, and $\mathcal{A}$'s random coins. In this conditional probability space, since the action of $\Lambda^1_{sk_1}$ on $\tilde{L}$ is determined by pk_1, and all decapsulation queries $x = (\tilde{x}, \pi_2)$ with $\tilde{x} \in \tilde{L}$ will be rejected, it follows that $\mathcal{A}$'s view in Game 3 is completely determined as a function of $\tilde{x}^*$, pk_1, and π_1^*, while $\mathcal{A}$'s view in Game 4 is determined as the same function of $\tilde{x}^*$, pk_1, and π_1'. Moreover, by independence, the joint distributions of $(\tilde{x}^*, pk_1, \pi_1^*)$ and $(\tilde{x}^*, pk_1, \pi_1')$ do not change in passing from the original probability space to the conditional probability space. It now follows directly from the ϵ-smooth property of HPS$_1$, we have:

$$|\Pr[S_4] - \Pr[S_3]| \leq \epsilon$$

It is evident from the definition of Game 4 that $\mathcal{A}$'s output b' is independent of the hidden bit b; therefore,

$$\Pr[S_4] = 1/2$$

Putting all these above, the theorem immediately follows. □

5.3 Construction from Universal$_1$ HPS

From a universal$_1$ HPS, we can construct adaptively weak-pseudorandom PEPRFs as follows:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, run HPS.$\mathsf{Setup}(\kappa)$ to generate a universal$_1$ HPS instance $pp = (\Lambda, \tilde{PK}, \tilde{SK}, \tilde{X}, \tilde{L}, W, \Pi, \alpha)$, then use the transformation described above to obtain a universal$_2$ HPS^H, where H is a family of 4-wise independent hash functions from Π to $\Pi^H = \{0,1\}^\ell$; then build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs from pp and H, where $X = \tilde{X} \times \Pi^H$, $Y = \Pi \cup \perp$, $PK = \tilde{PK} \times H$, $SK = \tilde{SK} \times H$, F, L, and W will be defined later.
- $\mathsf{KeyGen}(pp)$: on input pp, run HPS.$\mathsf{KeyGen}(pp)$ to get $(\tilde{pk}, \tilde{sk})$, pick $\mathsf{H} \xleftarrow{R} H$, set $pk = (\tilde{pk}, \mathsf{H})$, $sk = (\tilde{sk}, \mathsf{H})$, output (pk, sk).
- $\mathsf{SampLan}(pk; r)$: on input $pk = (\tilde{pk}, \mathsf{H})$ and random coins r, run $(\tilde{x}, w) \leftarrow$ HPS.$\mathsf{Sample}(r)$, compute $\pi^H \leftarrow \mathsf{H}(\text{HPS.}\mathsf{Pub}(\tilde{pk}, \tilde{x}, w))$. This sampling algorithm defines a collection of language $L = \{L_{pk}\}_{pk \in PK}$ over $X = X \times \Pi^H$ where each $L_{pk} = \{(\tilde{x}, \pi^H) : \tilde{x} \in \tilde{L} \wedge \pi^H = \mathsf{H}(\text{HPS.}\mathsf{Pub}(\tilde{pk}, \tilde{x}, w))\}$. It is easy to see that a witness w for $\tilde{x} \in \tilde{L}$ is also a witness for $x = (\tilde{x}, \pi^H) \in L_{pk}$.
- $\mathsf{PubEval}(pk, x, w)$: on input $pk = (\tilde{pk}, \mathsf{H})$ and an element $x = (\tilde{x}, \pi^H) \in L_{pk}$ together with a witness w, output $y \leftarrow \mathsf{H}(\text{HPS.}\mathsf{Pub}(\tilde{pk}, \tilde{x}, w))$.
- $\mathsf{PrivEval}(sk, x)$: on input $sk = (\tilde{sk}, \mathsf{H})$ and $x = (\tilde{x}, \pi^H)$, output $y \leftarrow \text{HPS.}\mathsf{Priv}(\tilde{sk}, \tilde{x})$ if $\pi^H = \mathsf{H}(\text{HPS.}\mathsf{Priv}(\tilde{sk}, \tilde{x}))$ and $\perp$ otherwise.

In the above construction, we implicitly build HPS^H, which is universal$_2$ according to Theorem 5.2. We have the following theorem of the above construction.

Theorem 5.5. *If the underlying subset membership problem is hard, then the PEPRFs from universal$_1$ HPS are adaptively weak-pseudorandom.*

Proof. The proof is similar as that in section 5.2. To base adaptively weak-pseudorandomness on the properties universal$_1$ 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 adaptively weak-pseudorandomness game for PEPRFs as follows:

- Setup: $\mathcal{CH}$ runs HPS.$\mathsf{Setup}(\kappa)$ to generate a universal$_1$ HPS instance pp and chooses a family of 4-wise independent hash functions H from Π to $\{0,1\}^{\ell}$, then builds public parameters pp for PEPRFs from pp and H. $\mathcal{CH}$ then runs $(\tilde{pk}, \tilde{sk}) \leftarrow$ HPS.$\mathsf{KeyGen}(pp)$ picks $\mathsf{H} \xleftarrow{\mathrm{R}} H$, sets $pk = (\tilde{pk}, \mathsf{H})$ and $sk = (\tilde{sk}, \mathsf{H})$. $\mathcal{CH}$ 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 $sk = (\tilde{sk}, \mathsf{H})$. More precisely, $\mathcal{CH}$ parses x as $(\tilde{x}, \pi^H)$, then responds with $\Lambda_{\tilde{sk}}(\tilde{x})$ if $\mathsf{H}(\Lambda_{\tilde{sk}}(\tilde{x})) = \pi^H$ and $\perp$ otherwise.
- Challenge: $\mathcal{CH}$ picks $r^* \xleftarrow{\mathrm{R}} R$, sets $(\tilde{x}^*, w^*) \leftarrow$ HPS.$\mathsf{Sample}(r^*)$, computes $\pi^{H^*} = \mathsf{H}(\Lambda_{\tilde{sk}}(\tilde{x}^*))$ via $\mathsf{H}(\text{HPS.}\mathsf{Pub}(\tilde{pk}, \tilde{x}^*, w^*))$, sets $x^* = (\tilde{x}^*, \pi^{H*})$. $\mathcal{CH}$ then computes $\pi^* = \Lambda_{\tilde{sk}}(\tilde{x}^*)$ via HPS.$\mathsf{Pub}(\tilde{pk}, \tilde{x}^*, w^*)$, sets $y_0^* = \pi^*$, samples $y_1^* \xleftarrow{\mathrm{R}} Y$, picks a random bit $b \in \{0,1\}$, and 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:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2| \tag{9}$$

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

$$\Pr[S_1] = \Pr[S_0] \tag{10}$$

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:

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\kappa) \tag{11}$$

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

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

Game 4: same as Game 3 except that in the challenge stage y_0^* is set as π', where $\pi' \xleftarrow{\mathrm{R}} \Pi$. The rest reasoning is similar as that presented in 5.2. It now follows directly from the ϵ_1-universal$_1$ property of HPS, we have:

$$|\Pr[S_4] - \Pr[S_3]| \leq \epsilon_1$$

It is evident from the definition of Game 4 that $\mathcal{A}$'s output b' is independent of the hidden bit b; therefore,

$$\Pr[S_4] = 1/2$$

Putting all these above, the theorem immediately follows. □

Remark 5.1. Construction 5.1 is straightforward, while the construction 5.2 and construction 5.3 are technical involved. The basic idea underlying the latter two constructions are similar as the CCA-secure PKE from smooth plus universal$_2$ HPSs [CS02]. That is, using a weak HPS to generate a random DEM key, while using a strong HPS to eliminate "dangerous" decapsulation queries. As we analyzed above, the minimum requirement for the weak HPS is smoothness, while the minimum requirement for the strong HPS is weak-universal$_2$. The advantage of construction 5.2 is that both HPSs are case-tailored, while the disadvantage is that we have to assume there exists a strong HPS sharing the same X and L as that of the weak HPS. On the contrary, the advantage of construction 5.3 is that we can start from a weak HPS and then transform it into a strong one, while the disadvantage is that the weak HPS has to be universal$_1$, which is strictly stronger than smooth. It seems to us that the generic transformation [KPSY09] cannot convert a smooth HPS into a universal$_2$ one, or even weak-universal$_2$ one.

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 a security parameter κ, output public parameters pp which include an EHPS instance $(\mathsf{H}, PK, SK, S, U, \Pi)$, where $\mathsf{H} : PK \times U \to \Pi$ can be viewed as a keyed function indexed by PK. Let $\mathsf{hc}(\cdot) : S \to \{0,1\}^l$ be a hardcore function for one-way binary relation R over $S \times U$. We say that R is *efficiently verifiable* if there is an efficient algorithm Vefy that on input (s, u) outputs true iff $(s, u) \in \mathsf{R}$. We say that an EHPS is *publicly checkable* if there is an algorithm Check that on input (pk, u, π) outputs true iff $\pi = \mathsf{H}_{pk}(u)$.
- $\mathsf{KeyGen}(pp)$: on input public parameters pp, output a key pair (pk, sk).
- $\mathsf{KeyGen}'(pp)$: on input public parameters 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 PEPRFs as follows:

- $\mathsf{Setup}(\kappa)$: on input κ, run $\mathrm{EHPS}.\mathsf{Setup}(\kappa)$ to generate an EHPS instance (H, PK, SK, S, U, Π), and build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs 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 $(pk, sk) \leftarrow \mathrm{EHPS}.\mathsf{KeyGen}(pp)$.
- $\mathsf{SampLan}(r)$: on input r, compute $u \leftarrow \mathrm{EHPS}.\mathsf{SampRight}(r)$, and $\pi \leftarrow \mathrm{EHPS}.\mathsf{Pub}(pk, r)$, 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 $\mathrm{EHPS}.\mathsf{Sample}$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, compute $s \leftarrow \mathrm{EHPS}.\mathsf{SampLeft}(w)$, output $y \leftarrow \mathsf{hc}(s)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, parse x as (u, π), compute $s \leftarrow \mathrm{EHPS}.\mathsf{Ext}(sk, u, \pi)$, 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 PEPRFs from EHPS are weak-pseudorandom.*

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

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2| \tag{12}$$

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:

$$\Pr[S_1] = \Pr[S_0] \tag{13}$$

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 $\mathsf{KeyGen}'(pp)$ as well as the correctness of the hashing mode, we have:

$$\Pr[S_2] = \Pr[S_1] \tag{14}$$

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:

$$\mathsf{Adv}_{\mathcal{B}}(\kappa) = |\Pr[S_2] - 1/2| = \mathsf{Adv}_{\mathcal{A}}(\kappa) \tag{15}$$

The theorem immediately follows. □

Theorem 6.2. *If the underlying binary relation* R *is one-way, then the PEPRFs from ABO EHPS are adaptively weak-pseudorandom.*

Proof. The proof follows immediately from [Wee10]. To base adaptively weak-pseudorandomness of PEPRFs 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 guess 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 $r^* \xleftarrow{R} R$, 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 $y_1^* \xleftarrow{R} \{0,1\}^l$. $\mathcal{CH}$ 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 $r^* \xleftarrow{R} R$, 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 $\Pr[F] \leq Q_1/|X| \leq \mathsf{negl}(\kappa)$. By the difference lemma, we have:

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

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:

$$\Pr[S_3] = \Pr[S_2] \tag{16}$$

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 $y^* \xleftarrow{R} \{0,1\}^l$). $\mathcal{B}$ 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. □

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 $\mathsf{SampLan}(r)$ 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 PSPRFs and KEM imply each other by viewing PSPRF.PubSamp (resp. PSPRF.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 A.3). Briefly, TDFs are a family of functions that are easy to compute, invert with trapdoor but hard to invert on average 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 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.

[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 that case, we can view KEM.Decap as randomized PSPRFs.

- $\mathsf{Setup}(\kappa)$: 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: For any $pp \leftarrow \mathsf{Setup}(\kappa)$ any $(ek, td) \leftarrow \mathsf{TrapGen}(pp)$, 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:

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

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 are "many-to-one" (resp. "one-to-one") if the underlying TDFs are "many-to-one" (injective).

[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 $(s, u) = \mathsf{PubSamp}(ek; r_2)$.

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 $= (\mathsf{R}, S, U)$. Let $\mathsf{hc} : S \to \{0,1\}^l$ be a hardcore function for binary relation R, we construct PSPRFs from U to $\{0,1\}^l \cup \perp$ as follows:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, run $\mathrm{TDR}.\mathsf{Setup}(\kappa)$ to generate pp.
- $\mathsf{KeyGen}(pp)$: on input pp, compute $(ek, td) \leftarrow \mathrm{TDR}.\mathsf{TrapGen}(pp)$, set $pk = ek$ and $sk = td$, output (pk, sk).
- $\mathsf{PubSamp}(pk; r)$: on input pk and random coins r, compute $(s, u) \leftarrow \mathrm{TDR}.\mathsf{PubSamp}(pk; r)$, output $(u, \mathsf{hc}(s))$.
- $\mathsf{PrivEval}(sk, u)$: on input sk and u, compute $s \leftarrow \mathrm{TDR}.\mathsf{TdInv}(sk, u)$, 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 PSPRFs are (adaptively) weak-pseudorandom if the underlying TDRs are (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, 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, whether adaptive PSPRFs are strictly weaker than general ATDRs is also unclear to us. We left this as an open problem.

8 Publicly Evaluable Constrained PRFs

In this section, we introduce the notion of publicly evaluable constrained PRFs (PECPRFs), which is an extension of recent constrained PRFs [KPTZ13, BW13, BGI14] analogous to PEPRFs of PRFs. We begin with the precise definition, and then proceed to explore possible applications.

Definition 8.1 (Publicly Evaluable Constrained PRFs)**.** Publicly evaluable constrained 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 family of predicates $P = \{p : I \to \{0,1\}\}$, 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 constrained PRFs [KPTZ13, BW13, BGI14], here we explicitly define the domain as a Cartesian product of I and X. We also 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.
- $\mathsf{Constrain}(msk, p)$: on input msk and a predicate $p \in P$, output a secret key sk_p.
- $\mathsf{SampLan}(ind, r)$: 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:

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

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

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

where $\mathcal{O}_{\mathsf{eval}}(ind, x) = \mathsf{F}_{msk}(ind, x)$, $\mathcal{O}_{\mathsf{constrain}}(p) = \mathsf{Constrain}(msk, p)$. The adversary $\mathcal{A}$ is restricted from querying $\mathcal{O}_{\mathrm{constrain}}(\cdot)$ 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^*). PECPRF 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-adaptive weak-pseudorandomness.

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

8.1 Predicate KEM from Publicly Evaluable Constrained 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 A.2 for formal definition of predicate KEM.

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

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

Correctness of this construction follows directly from that of PECPRF. For security, we have the following results:

Theorem 8.1. *The predicate KEM is CPA-secure if the underlying PECPRF is semi-adaptively weak-pseudorandom.*

Theorem 8.2. *The predicate KEM is CCA-secure if the underlying PECPRF is adaptively weak-pseudorandom.*

We omit the proofs here for their triviality.

Remark 8.1. Considering PECPRF that supports a special family of predicates P where p are indexed by I and p_{ind} is defined as $p_{ind}(ind') = 1$ iff $ind = ind'$. It is easy to see that such PECPRF immediately implies identity-based key encapsulation mechanism (IB-KEM), while itself can be generically constructed from either identity-based hash proof system (IB-HPS) [ADN+10, CZLC12a] or identity-based extractable hash proof system (IB-EHPS) [CZLC12b].

8.2 A Construction of PECPRFs

Next we present a concrete construction of 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 A.4 for completeness. We refer the readers to Appendix A.5 for the definition of multilinear maps and related hardness assumption.

- $\mathsf{Setup}(\kappa, \ell)$: on input a 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 parameters 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.
- $\mathsf{Constrain}(msk, c)$: 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].

9 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*, while the best possible security degrades to "hard to compute" on average.

Definition 9.1 (Publicly Evaluable and Verifiable Functions)**.** A family of PEVFs consist of the following polynomial-time algorithms:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters $pp = (\mathsf{F}, SK, PK, X, L, W, Y)$, where $\mathsf{F} : SK \times X \to Y$ can be viewed as a keyed function indexed by SK.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key sk and an associated public key pk.
- $\mathsf{SampLan}(r)$: 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 true if $y = \mathsf{F}_{sk}(x)$ and false if not.

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

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

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.

9.1 PEVFs from EHPS

From EHPS, we construct PEVFs as follows:

- $\mathsf{Setup}(\kappa)$: on input κ, run $\mathrm{EHPS.}\mathsf{Setup}(\kappa)$ to generate an EHPS instance $(\mathsf{H}, PK, SK, S, U, \Pi)$, and build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEVFs from it, where $X = U \times \Pi$, $Y = S$, 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 $(pk, sk) \leftarrow \mathrm{EHPS.}\mathsf{KeyGen}(pp)$.
- $\mathsf{SampLan}(r)$: on input r, compute $u \leftarrow \mathrm{EHPS.}\mathsf{SampRight}(r)$, and $\pi \leftarrow \mathrm{EHPS.}\mathsf{Pub}(pk, r)$, 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 $\mathrm{EHPS.}\mathsf{Sample}$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, output $y \leftarrow \mathrm{EHPS.}\mathsf{SampLeft}(w)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, parse x as (u, π), output $y \leftarrow \mathrm{EHPS.}\mathsf{Ext}(sk, u, \pi)$. This algorithm defines $\mathsf{F}_{sk}(x)$ as $\mathsf{Ext}(sk, x)$.
- $\mathsf{PubVefy}(pk, x, y)$: on input pk, x, and y, parse x as (u, π), output $\mathrm{EHPS.}\mathsf{Vefy}(y, u)$. Here we always assume the second input $x \in L$, since it is efficiently recognizable if the underlying EHPS is efficiently checkable.

We have the following theorem about the above construction.

Theorem 9.1. *If the underlying binary relation* R *is one-way and efficiently verifiable, then the PEVFs from EHPS are hard to compute on average.*

We omit the proof here due to its straightforwardness.

9.2 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.$\mathsf{SampLan}$ and PEVF.$\mathsf{PubEval}$ are not used in the above construction, but will prove useful in security argument.

Theorem 9.2. *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 $j \in \{1, \ldots, Q_h\}$.
- 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 $r_i \xleftarrow{\mathrm{R}} R$, runs PEVF.$\mathsf{SampLan}(r_i)$ 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. □

Looking ahead, When applying the above generic construction to PEVFs based on the RSA assumption and the CDH assumption in bilinear groups presented in subsection 9.3, 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 cannot encompass the BLS signature accurately, cause there is no corresponding efficiently computable trapdoor function/permutation.

Replacing random oracle: Very recently, Hohenberger, Sahai, and Waters [HSW14] 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 $r \leftarrow \mathsf{PPRF}(k, m)$, $(x, w) \leftarrow \mathsf{PEVF.SampLan}(r)$, 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.

Remark 9.1. 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.

9.3 Instantiations of PEVFs

Here we construct PEVFs based on the RSA assumption (c.f. definition in A.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 $sk \xleftarrow{R} \mathbb{Z}_p$, 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.

Acknowledgment

We are grateful to Yi Deng, Qiong Huang, and Dennis Hofheinz for helpful discussions and advice. We also thank the SCN 2014 reviewers for many useful comments.

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.

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. 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.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *17th International Conference on Practice and Theory in Public-Key Cryptography, PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, 2014.

[BH12] Itay Berman and Iftach Haitner. From non-adaptive to adaptive pseudorandom functions. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012*, volume 7194 of *LNCS*, pages 357–368. Springer, 2012.

[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.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, 2003.

[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.

[CZLC12a] Yu Chen, Zongyang Zhang, Dongdai Lin, and Zhenfu Cao. Anonymous identity-based hash proof system and its applications. In *Provable Security - 6th International Conference, ProvSec 2012*, volume 7496 of *LNCS*, pages 143–160. Springer, 2012.

[CZLC12b] Yu Chen, Zongyang Zhang, Dongdai Lin, and Zhenfu Cao. Identity-based extractable hash proofs and their applications. In *International Conference on Applied Cryptography and Network Security - ACNS 2012*, volume 7341 of *LNCS*, pages 153–170. Springer, 2012.

[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 *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 126–135. IEEE Computer Society, 2001.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[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.

[HLAWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 160–176. Springer, 2013.

[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.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, 2014.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference STOC 1995*, pages 134–147. IEEE Computer Society, 1995.

[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. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013*, pages 669–684. ACM, 2013.

[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.

[LT13] Huijia Lin and Stefano Tessaro. Amplification of chosen-ciphertext security. In *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 503–519. Springer, 2013.

[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.

[Pei14] Chris Peikert. Lattice cryptography for the internet. IACR Cryptology ePrint Archive, Report 2014/070, 2014. http://eprint.iacr.org/2014/070.

[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.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014*, pages 475–484. ACM, 2014.

[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.

[Zha14] Mark Zhandry. How to avoid obfuscation using witness prfs. IACR Cryptology ePrint Archive, Report 2014/301, 2014. http://eprint.iacr.org/2014/301.

A Review of Standard Definitions and Assumptions

A.1 Key Encapsulation Mechanism

Following the treatment of [Pei14], we equip key encapsulation mechanism (KEM) with an explicit setup algorithm run by a trusted party, which generates some global public parameters shared by all parties. If no trusted party is available, then the setup algorithm can be run by individual parties as part of key generation algorithm, and the public parameters are included in the resulting public key. Formally, a KEM consists of four polynomial-time algorithms:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp. We assume that pp also includes the descriptions of ciphertext space C and DEM (data encapsulation mechanism) key space K. pp will be used as an implicit input for algorithms Encap and Decap.
- $\mathsf{KeyGen}(pp)$: on input pp, 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 $pp \leftarrow \mathsf{Setup}(\kappa)$, any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, 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:

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

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 $\mathcal{O}_{\mathsf{dec}}(\cdot)$.

A.2 Predicate Key Encapsulation Mechanism

A predicate KEM consists of five polynomial-time algorithms:

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters pp. We assume that pp include the descriptions of index set I (index is also usually referred to as attribute), a family of predicates P, ciphertext space C, and DEM (data encapsulation mechanism) key space K. pp will be used as an implicit input for algorithms KeyGen, Encap, and Decap.
- $\mathsf{KeyGen}(pp)$: on input pp, 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:

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

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 $\mathcal{O}_{\mathsf{dec}}(\cdot,\cdot)$.

A.3 Trapdoor Functions

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

- $\mathsf{Setup}(\kappa)$: on input a security parameter κ, output public parameters $pp = (\mathsf{TDF}, EK, TD, X, Y)$, where $\mathsf{TDF} : EK \times X \to Y$ can be viewed as a keyed function indexed by EK.
- $\mathsf{TrapGen}(pp)$: on input pp, output $(ek, td) \in EK \times TD$.
- $\mathsf{SampDom}(r)$: 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: 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 TDFs and define its advantage as:

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

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 TDFs 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.

A.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]

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 $\mathit{Inputs} = \{1, \ldots, n\}$, $\mathit{Wires} = \{1, \ldots, n+q\}$, $\mathit{Gates} = \{n+1, \ldots, n+q\}$. The wire $n+q$ is the designated output wire. $\mathsf{A} : \mathit{Gates} \to \mathit{Wires}$ is a function where $\mathsf{A}(w)$ identifies w's first incoming wire and $\mathsf{B} : \mathit{Gates} \to \mathit{Wires}$ is a function where $\mathsf{B}(w)$ identifies w's second incoming wire. Finally, $\mathsf{GateType} : \mathit{Gates} \to \{\mathrm{AND}, \mathrm{OR}\}$ is a function that identifies a gate as either an AND or OR gate.

For any $w \in \mathit{Gates}$ we require that $w > B(w) > A(w)$. We also define a function $\mathsf{depth}(w)$ where if $w \in \mathit{Inputs}$ 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 $w \in \mathit{Gates}$ 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.

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

[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.

A.6 RSA Assumption

Assumption A.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 $\phi(N) = (p-1)(q-1)$, $y \xleftarrow{R} \mathbb{Z}_N^*$.