Publicly Evaluable Pseudorandom Functions and Their Applications

Yu Chen *
yuchen.prc@gmail.com

Zongyang Zhang †
zongyang.zhang@gmail.com

Abstract

We put forth the notion of *publicly evaluable* pseudorandom functions (PEPRFs), which can be viewed as a counterpart of standard pseudorandom functions (PRFs) in the public-key setting. Briefly, PEPRFs are defined over domain X containing a language L associated with a hard relation R_L, and each secret key sk is associated with a public key pk. For any $x \in L$, in addition to evaluate $\mathsf{F}_{sk}(x)$ using sk as 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 a PEPRF cannot be distinguished from a real random function on uniformly random chosen inputs. The strengthened one is adaptive weak pseudorandomness which requires a PEPRF remains weak pseudorandom even when an adversary is given adaptive access to an evaluation oracle. We conduct a formal study of PEPRFs, focusing on applications, constructions, and extensions.

- We show how to construct chosen-plaintext secure (CPA) and chosen-ciphertext secure (CCA) public-key encryption (PKE) schemes 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 constructions from injective trapdoor functions, hash proof systems, extractable hash proof systems, as well as a construction from puncturable PRFs with program obfuscation.
- We introduce the notion of publicly sampleable PRFs (PSPRFs), which is a relaxation of PEPRFs, but nonetheless imply PKE. We show (adaptive) PSPRFs are implied by (adaptive) trapdoor relations. This helps us to unify and clarify many PKE schemes from seemingly unrelated general assumptions and paradigms under the notion of PSPRFs.
- We explore similar extension on recently emerging constrained PRFs, and introduce the notion of publicly evaluable constrained PRFs, which, as an immediate application, implies attribute-based encryption.
- We propose a twist on 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 unpredictability. We justify the applicability of PEVFs by presenting a simple construction of "hash-and-sign" signatures, both in the random oracle model and the standard model.

*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China. & State Key Laboratory of Cryptology, Beijing, China.

†Corresponding Author. Advanced Cryptosystems Research Group, Information Technology Research Institute (ITRI), National Institute of Advanced Industrial Science and Technology, Tokyo, Japan.

1 Introduction

Pseudorandom functions (PRFs) [GGM86] are a fundamental concept in modern cryptography. Loosely speaking, PRFs are a family of keyed functions $\mathsf{F} : SK \times X \to Y$ such that: (1) it is easy to sample the functions and compute their values, i.e., given a secret key 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 a language $L \subseteq X$ with a hard relation R_L, and each secret key sk is associated with a public key pk. In addition, for any $x \in L$, except via a private evaluation algorithm with sk, one can also efficiently compute the value of $\mathsf{F}_{sk}(x)$ via a public evaluation algorithm 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 challenge points are arbitrarily chosen by an adversary).

While PEPRFs are a conceptually simple extension of 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, a sender first chooses a random $x \in X$, and then outputs a 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 it is impossible to encrypt publicly. Moreover, since PRFs and one-way functions (OWFs) imply each other [GGM86, HILL99], the implications of PRFs are inherently confined in Minicrypt.[1] 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 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].[2] However, the celebrated Goldwasser-Micali PKE [GM84] and ElGamal PKE [ElG85] do not fit into any known paradigms or general assumptions.

Recently, indistinguishability obfuscation ($i\mathcal{O}$) together with puncturable PRFs (PPRFs) have proven to be an extremely powerful combination and have been used to construct a variety of cryptographic primitives, such as PKE, short signature, non-interactive zero knowledge proof, oblivious transfer [SW14], multiparty key exchange and traitor tracing [BZ14], etc. On one hand, while these results are exciting, existing instantiations of candidate obfuscators [GGH+13b,

[1] According to [Imp95], Minicrypt refers to the world where one-way functions exist, but public-key cryptography does not; while Cryptomania refers to the world where public-key cryptography is possible.

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

BGK+14, PST14, AGIS14] have several drawbacks in terms of efficiency [Zha16] and inherently rely on multilinear maps. On the other hand, we observe that in the constructions of PKE and multiparty key exchange $i\mathcal{O}$ is in fact compiling PPRFs into some kind of "publicly computable" PRFs as an intermediate primitive, which are in turn sufficient for the applications. However, no such "publicly computable" PRFs is known to date.

Motivated by the above discussion, we ask the following intriguing questions:

Can we adapt the concept of PRFs to the public-key setting? If so: Can it enable the above PRF-based private-key encryption to work the public-key setting? Can it be instantiated from standard assumptions? Can it be used to explain unclassified PKE schemes?

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 is a counterpart of standard PRFs in the public-key setting. In a PEPRF, there is a language $L \subseteq X$ defined by a hard relation R_L, and each secret key sk is associated with a public key pk. For any $x \in L$, except via a private evaluation algorithm with sk, one can efficiently evaluate $\mathsf{F}_{sk}(x)$ using pk and a witness w for $x \in L$. We also formalize security notions for PEPRFs, namely weak pseudorandomness and adaptive weak pseudorandomness.
- In Section 4, we demonstrate the power of PEPRFs by showing that they enable the PRF-based 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.[3] In particular, in Section 3.1 we show that the celebrated Goldwasser-Micali PKE and ElGamal PKE can be explained neatly by weak pseudorandom PEPRFs based on the quadratic residuosity assumption and the decisional Diffie-Hellman assumption, respectively.
- In Section 5, Section 6 and Section 7, we show how to construct PEPRFs from trapdoor functions (TDFs), hash proof systems (HPSs), and extractable hash proof systems (EHPSs), respectively. This indicates that previous works on TDFs, HPSs and EHPSs implicitly constructed PEPRFs. Therefore, PEPRFs are abstraction of the common aspect of HPSs and EHPSs which are not formalized before. Moreover, existing constructions of TDFs, HPS and EHPS imply that PEPRFs can be instantiated from a variety of number-theoretic assumptions. In Section 8, we show that recent work [SW14] essentially suggests a way to compile puncturable PRFs to adaptive weak pseudorandom PEPRFs via indistinguishability obfuscator. This result reinforces the intuition that in several obfuscation-based constructions the indistinguishability obfuscator is in fact compiling puncturable PRFs into some kind of publicly computable PRFs as an intermediate object.
- In Section 9, we introduce the notion of publicly sampleable 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 TDFs imply "one-to-one" TDRs, while the latter further imply PSPRFs. This implication helps us to unify and

[3] 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].

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 10, we introduce an extension of PEPRFs named publicly evaluable constrained PRFs (PECPRFs). An immediate application of PECPRFs is attribute-based encryption. We also present an instantiation of PECPRFs based on an attribute-based encryption scheme from multilinear maps [GGH+13c].
- In Section 11, we propose a twist on PEPRFs named publicly evaluable and verifiable functions (PEVFs), which enjoy an additional property named public verifiability and only require unpredictability rather than weak pseudorandomness. 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.

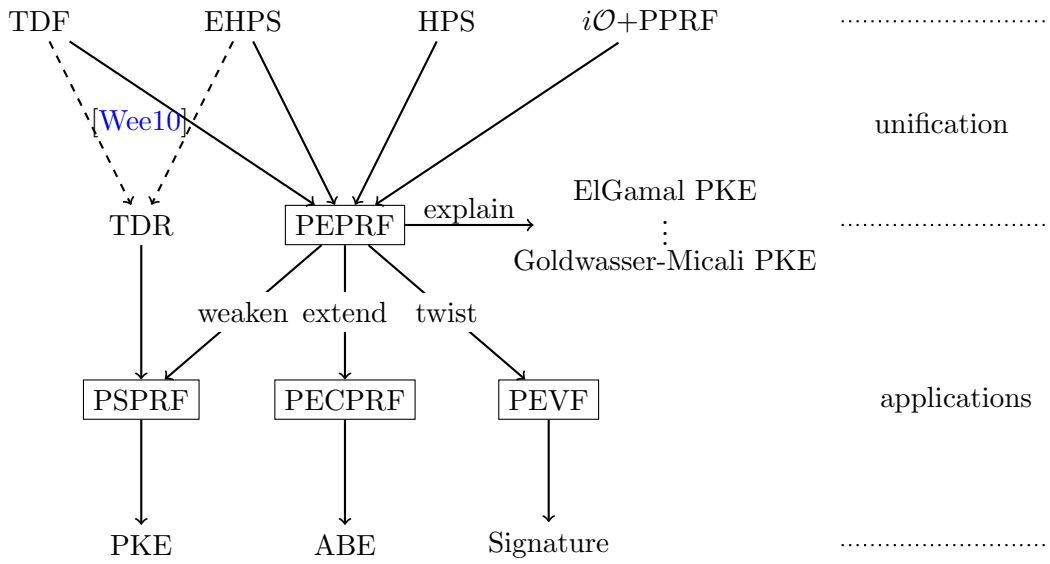

Figure 1: Summary of main results in this work. The bold lines and rectangles denote our contributions and our new concepts, while the dashed lines denote those from existing works.

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 a 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 assumptions include one-way (trapdoor) functions/permutations and pseudorandom generators/functions, etc. In contrast, a paradigm usually consists of two modes, where one mode is used to fulfill some functionality, while the other one is used to argue the first one satisfies some desired security. Examples of paradigms include hash proof system and extractable hash proof system, etc. Both a general assumption and a 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 of great theoretical interest to build CCA-secure PKE from general assumptions and paradigms. Cramer and Shoup [CS02] generalized their CCA-secure PKE construction [CS98] to the notion of hash proof systems (HPSs) 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 the notion of extractable hash proof systems (EHPSs) 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, the notion of ATDRs is the weakest general assumption that implies CCA-secure PKE.

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, the secret key admits a delegation for a family of circuits, and the delegated key for circuit f enable one to compute the PRF value at any point x such that $f(x) = 1$. This natural extension turns out to be useful since it has powerful applications outside 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 [Zha16] introduces the notion of *witness PRFs* (WPRFs), which is similar in concept to PEPRFs. In a nutshell, both WPRFs and PEPRFs are defined with respect to a language and extend standard PRFs with the same extra functionality, i.e., one can publicly evaluate $\mathsf{F}_{sk}(x)$ for $x \in L$ with its witness. The main differences between WPRFs and PEPRFs are as follows:

1. In WPRFs the associated relation of the language is required to be polynomially bounded, while in PEPRFs the associated relation of the language is required to be hard.
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 to replace indistinguishability obfuscation for several obfuscation-based applications. By utilizing the reduction from any $\mathcal{NP}$ language to the subset-sum problem, WPRFs can handle arbitrary $\mathcal{NP}$ languages. However, for applications of WPRFs whose functionalities rely on $\mathsf{F}_{sk}(x)$ for $x \in L$, such as public-key encryption, non-interactive key exchange, and hardcore functions for any one-way function, the underlying $\mathcal{NP}$ languages have to be at least hard-on-average (i.e., the associated relation is hard). 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

2.1 Basic Notations

We review the basic terminology used throughout the paper. For a distribution or a set S, we write $s \xleftarrow{R} S$ to denote the operation of sampling s according to S or uniformly at random from S, and use $|S|$ to denote its size. For a set S, we let U_S denote the uniform distribution over S.

We let $\mathbb{N}$ denote the positive integers. The natural security parameter throughout the paper is λ, and all other quantities as well as all algorithms (including the adversary) are implicitly functions of λ. We write $\mathsf{poly}(\lambda)$ to denote an arbitrary polynomial function in λ. We write $\mathsf{negl}(\lambda)$ to denote an arbitrary negligible function in λ, which vanishes faster than the inverse of any polynomial. We say a probability is overwhelming if it is $1-\mathsf{negl}(\lambda)$, and said to be noticeable if it is $1/\mathsf{poly}(\lambda)$. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm that runs in time $\mathsf{poly}(\lambda)$. If $\mathcal{A}$ is a randomized algorithm, we write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n; r)$ We will omit r and write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n)$. In addition, we assume an algorithm always outputs $\perp$ when its input is $\perp$.

The statistical distance between two random variables X and Y having a common domain S is $\Delta[X, Y] = \frac{1}{2}\sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$. We write $\approx_c$ as shorthand for computationally indistinguishable.

2.2 Languages and Relations

Cryptographic hard problems are usually given by a set of instances, which we denote by X. We call a subset L of X as a language, in which all its elements meet some property. Typically, a language L can be associated with a binary relation $\mathsf{R}_L \subseteq X \times W$, i.e., $x \in L \Leftrightarrow \exists w \in W$ s.t. $(x, w) \in \mathsf{R}_L$. The only restriction is that if $(x, w) \in \mathsf{R}_L$, then $|w| \leq \mathsf{poly}(|x|)$. We call $w \in W$ a witness for the membership of $x \in L$. We recall some properties of relations as below.

1. *Easy to sample a tuple:* There exists a PPT algorithm $\mathsf{SampRel}$ that on input a public description pp for R_L and fresh random coins r, outputs a random tuple $(x, w) \in \mathsf{R}_L$. We can further decompose $\mathsf{SampRel}$ to $\mathsf{SampLan}$ and $\mathsf{SampWit}$. The former on input pp and r outputs $x \in L$, while the latter on input pp and r outputs $w \in W$.[4] Hereafter, let R be the random coins space of $\mathsf{SampRel}$, $\mathsf{SampLan}$, and $\mathsf{SampWit}$. We require that for any $r \in R$, it holds that $(\mathsf{SampLan}(r), \mathsf{SampWit}(r)) \in \mathsf{R}_L$.
2. *Hard to find a witness:* Given x where $(x, w) \leftarrow \mathsf{SampRel}(pp; r)$, no PPT algorithm can compute w' such that $(x, w') \in \mathsf{R}_L$ with non-negligible probability.
3. *Easy to verify a tuple:* R_L is polynomially-verifiable (i.e., $\mathsf{R}_L \in \mathcal{P}$).

We say a relation R_L is *hard* or *one-way* if it satisfies the first two properties. We say a relation R_L is *polynomially bounded* if it satisfies the third property. Polynomially bounded relations define the usual $\mathcal{NP}$ language.

Definition 2.1 (Subset membership problem)**.** Let $L \subset X$ be a language, and both L and $X \backslash L$ allow efficiently uniform sampling (let $\mathsf{SampYes}$ and SampNo be the sampling algorithms respectively[5]). We say subset membership problem w.r.t. (X, L) is hard if:

$$U_{X \backslash L} \approx_c U_L$$

[4] When the context is clear, we usually omit pp from the input of $\mathsf{SampRel}$.

[5] Without loss of generality, we assume $\mathsf{SampYes}$ and $\mathsf{SampLan}$ induce identical distribution over L.

2.3 Pseudorandom Functions

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

Definition 2.2 (PRFs [GGM86])**.** A family of PRFs consists of three polynomial-time algorithms as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters $pp = (\mathsf{F}, SK, X, Y)$ (the sets SK, X, and Y may be parameterized by λ), 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 $y \in Y$.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $sk \leftarrow \mathsf{KeyGen}(pp)$ and any $x \in X$, we have $\mathsf{F}_{sk}(x) = \mathsf{PrivEval}(sk, x)$.

Security: The standard security requirement for PRFs is pseudorandomness. Let $\mathcal{A}$ be an adversary against PRFs and define its advantage as:

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

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[6]). 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 $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible in λ. We refer to such security as full PRF security.

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

2.4 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 challenge points are sampled by the challenger, thus it is convenient to explicitly introduce an 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 weak pseudorandomness holds even the random coins are made public. Clearly, whether a weak PRF is public-coin or just secret-coin depends on the input-sampling algorithm.

[6] 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.

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 four polynomial-time algorithms as below:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters pp which includes $(\mathsf{F}, PK, SK, X, L, W, Y)$, where $\mathsf{F} : SK \times X \to Y \cup \perp$ could be viewed as a keyed function indexed by SK, $L \subseteq X$ is a language defined by some hard relation R_L, and W is the set of associated witnesses. We assume R_L is efficiently sampleable, that is, there exists a PPT algorithm $\mathsf{SampRel}$ which on input random coins r, output a random tuple $(x, w) \in \mathsf{R}_L$.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key sk and an associated public key pk.[7]
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \in Y \cup \perp$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$, output $y \in Y$.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, we have:

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

Security: Let $\mathcal{A}$ be an adversary against PEPRFs and define its advantage as:

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

where $p(\lambda)$ 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 $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{eval}}(\cdot)$ with any x_i^*. We say that PEPRFs are adaptive weak pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible in λ.[8] The adaptive weak pseudorandomness captures the security against active adversaries, who are given adaptive access to the 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 security against key-leakage attacks for PEPRFs.

Remark 3.1. Different from standard PRFs, PEPRFs require the existence of a language $L \subseteq X$ as well as 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.[9] Note that the weak PRF security implicitly requires that the associated relation R_L is hard.

[7] In a standard PRF, it is also harmless to explicitly introduce a 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 $pk = \{\perp\}$.

[8] The readers should not be confused 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.

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

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, the query times $p(\lambda)$ by an adversary 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.

A possible relaxation. To be completely precise, it is not necessary to require the distribution of x induced by $\mathsf{SampRel}(r)$ 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 public key rather than a fixed language L. Correspondingly, the sampling algorithm $\mathsf{SampRel}$ takes pk as an extra input to sample a random tuple $(x, w) \in \mathsf{R}_{L_{pk}}$. We refer to such generalized notion as *PEPRFs for public-key dependent languages*, and we will work with it when constructing adaptive PEPRFs from hash proof systems.

3.1 Two Simple PEPRFs from Number-Theoretic Assumptions

In what follows, we present two illustrative constructions of PEPRFs from the quadratic residuosity (QR) assumption (c.f. Appendix A.6) and the decisional Diffie-Hellman (DDH) assumption (c.f. Appendix A.7), respectively.

Construction 3.1. We build PEPRFs from the QR assumption as follows:

- $\mathsf{Setup}(1^\lambda)$: run $(N, p, q) \leftarrow \mathsf{GenModulus}(1^\lambda)$, set $X = \mathbb{Z}_N^*$, $Y = \{0, 1\}$, $PK = \{N\} \times QNR_N^{+1}$, $SK = \{p\} \times \{q\}$, $W = \mathbb{Z}_N^*$, $\mathsf{F} : SK \times X \to Y$ will be defined later; L be the set of elements in $\mathbb{Z}_N^*$ having Jacobi symbol $+1$. Let z be an element in QNR_N^{+1}, we can define $L = \{x : \exists w \in W \text{ s.t. } x = w^2 \bmod N \vee x = zw^2 \bmod N\}$, the corresponding sampling algorithm $\mathsf{SampRel}$ on input 1^λ, public key (N, z), and random coins r, picks $w \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes either $x = w^2 \bmod N$ or $x = zw^2 \bmod N$, then outputs (x, w).
- $\mathsf{KeyGen}(pp)$: pick $z \xleftarrow{\mathrm{R}} QNR_N^{+1}$, output $pk = (N, z)$ and $sk = (p, q)$.
- $\mathsf{PrivEval}(sk, x)$: output 1 if $\mathsf{Jacobi}_p(x) = \mathsf{Jacobi}_q(x) = 1$ and 0 otherwise. This algorithm defines $\mathsf{F} : SK \times X \to Y$.
- $\mathsf{PubEval}(pk, x, w)$: output 1 if $x = w^2 \bmod N$ and 0 if $x = zw^2 \bmod N$.

It is easy to verify that the above PEPRFs are weak pseudorandom based on the QR assumption. Looking ahead, applying the construction shown in Section 4 to the PEPRFs yields the Goldwasser-Micali PKE [GM84].

Construction 3.2. We build PEPRFs from the DDH assumption as follows:

- $\mathsf{Setup}(1^\lambda)$: run $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, set $X = Y = PK = L = \mathbb{G}$, $SK = W = \mathbb{Z}_p$, $\mathsf{F} : SK \times X \to Y$ is defined as $\mathsf{F}_{sk}(x) = x^{sk}$; we can define $L = \{x : \exists w \in W \text{ s.t. } x = g^w\}$; the corresponding sampling algorithm $\mathsf{SampRel}$ on input λ and random coins r, picks $w \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, computes $x = g^w$, then outputs (x, w).
- $\mathsf{KeyGen}(pp)$: pick $sk \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, compute $pk = g^{sk}$, output (pk, sk).
- $\mathsf{PrivEval}(sk, x)$: output x^{sk}.
- $\mathsf{PubEval}(pk, x, w)$: output pk^w.

It is easy to verify that the above PEPRFs are weak pseudorandom based on the DDH assumption. Looking ahead, applying the construction shown in Section 4 to the PEPRFs yields exactly the plain ElGamal PKE [ElG85].

3.2 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 $(x, w) \leftarrow \mathsf{SampRel}(r)$ 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).

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 $SK_1 \times SK_2$.

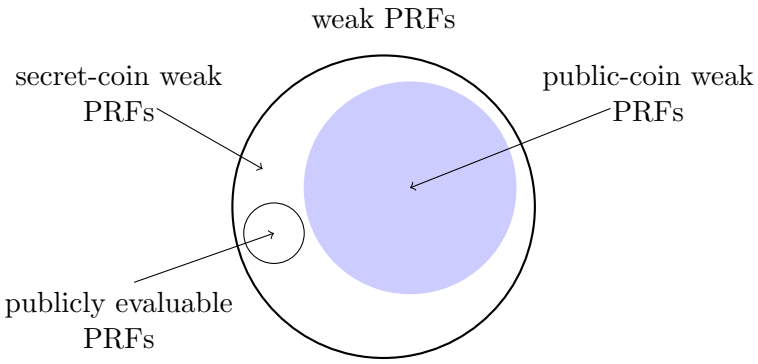

Figure 2: Relations among secret-coin, public-coin, and publicly evaluable PRFs.

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.

- $\mathsf{Setup}(1^\lambda)$: run $\mathsf{PEPRF.Setup}(1^\lambda)$ to generate pp as public parameters.
- $\mathsf{KeyGen}(pp)$: run $\mathsf{PEPRF.KeyGen}(pp)$ to generate (pk, sk).
- $\mathsf{Encaps}(pk; r)$: run $\mathsf{SampRel}(r)$ to generate a random tuple $(x, w) \in \mathsf{R}_L$, set x as the ciphertext c and compute $\mathsf{PEPRF.PubEval}(pk, x, w)$ as the DEM key k, output (c, k).
- $\mathsf{Decaps}(sk, c)$: output $\mathsf{PEPRF.PrivEval}(sk, c)$.

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 (resp. CCA-secure) if the underlying PEPRFs are weak pseudorandom (resp. adaptive weak pseudorandom).*

Proof. The proof is straightforward, which transforms an adversary $\mathcal{A}$ against IND-CPA security (resp. IND-CCA security) of the KEM construction to a distinguisher $\mathcal{B}$ against weak

pseudorandomness (resp. adaptive weak pseudorandomness) of PEPRFs. Here, we only prove the IND-CCA case and the IND-CPA case immediately follows.

$\mathcal{B}$ simulates $\mathcal{A}$'s challenger in the standard IND-CCA experiment.

- Setup and challenge: $\mathcal{B}$ receives (pp, pk, x^*, y_b^*) from its own challenger, where $pp \leftarrow \mathrm{PEPRF.}\mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathrm{PEPRF.}\mathsf{KeyGen}(pp)$, $(x^*, w^*) \leftarrow \mathsf{SampRel}(r^*)$ for some random coins r^*, and y_b^* is $\mathsf{F}_{sk}(x^*)$ if $b = 0$ or randomly chosen from Y if $b = 1$. $\mathcal{B}$ sets $c^* = x^*$, $k_b^* = y_b^*$, and sends (pp, pk, c^*, k_b^*) to $\mathcal{A}$ as the challenge.
- Decapsulation queries: on decapsulation query $\langle x \rangle$ where $x \neq x^*$, $\mathcal{B}$ submits evaluation query on point x to its own challenger and forwards the reply to $\mathcal{A}$.
- Guess: $\mathcal{A}$ outputs its guess b' for b and $\mathcal{B}$ forwards b' to its own challenger.

Clearly, $\mathcal{B}$ simulates the IND-CCA experiment perfectly. Therefore, $\mathcal{B}$ can break the adaptive weak pseudorandomness of PEPRFs with advantage at least $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}}(\lambda)$. Assuming the adaptive weak pseudorandomness of the underlying PEPRFs, $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A}}(\lambda)$ is negligible in λ. This concludes the proof. □

The above results also hold if the underlying PEPRFs are (adaptive) χ-weak pseudorandom.

5 Connection to Trapdoor Functions

Trapdoor functions (TDFs) are a fundamental primitive introduced by Diffie and Hellman [DH76]. In the following, we recall the syntax and security notions of TDFs and then show how to construct PEPRFs from them.

Definition 5.1 (Trapdoor Functions)**.** A family of trapdoor functions is given by four polynomial-time algorithms as below.

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters $pp = (\mathsf{TDF}, PK, SK, S, U)$, where $\mathsf{TDF} : PK \times S \to U$ can be viewed as a keyed function indexed by PK. We assume the domain S is efficiently sampleable, that is, there exists a PPT algorithm $\mathsf{SampDom}$, which on input random coins $r \xleftarrow{R} R$, output a random element $s \in S$.
- $\mathsf{KeyGen}(pp)$: on input pp, output a public key pk (serve as a key for evaluation) and a corresponding secret key sk (serve as a trapdoor for inversion).
- $\mathsf{Eval}(pk, s)$: on input pk and $s \in S$, output $u \leftarrow \mathsf{TDF}_{pk}(s)$.
- $\mathsf{TdInv}(sk, u)$: on input sk and $u \in U$, output $s \in S$ or a distinguished symbol $\perp$ indicating u does not have pre-image.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$ and any $s = \mathsf{Eval}(pk, u)$, we have $\mathsf{Eval}(pk, \mathsf{TdInv}(sk, s)) = s$.

(Adaptive) One-wayness: Let $\mathcal{A}$ be an inverter against TDFs and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[s \in \mathsf{TDF}_{pk}^{-1}(u^*) : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ u^* \leftarrow \mathsf{Eval}(pk, s^*), s^* \leftarrow \mathsf{SampDom}(r^*); \\ s \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{inv}}(\cdot)}(pp, pk, u^*) \end{array} \right],$$

where $\mathcal{O}_{\mathsf{inv}}(u) = \mathsf{TdInv}(sk, u)$, and $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{inv}}(\cdot)$ for the challenge u^*. We say that TDFs are adaptive one-way (or simply adaptive) [KMO10] if for any PPT inverter its advantage is negligible in λ. The standard one-wayness can be defined similarly as above except that the adversary is not given access to the inversion oracle.

Definition 5.2 (Hardcore Functions)**.** A polynomial-time algorithm $\mathsf{hc} : S \to Y$ is said to be a hardcore function of a family of efficiently computable functions $\mathsf{F} : PK \times S \to U$ if for any PPT algorithm $\mathcal{A}$, the following two distributions are computationally indistinguishable:

$$(pp, pk, \mathsf{F}_{pk}(s), \mathsf{hc}(s)) \approx_c (pp, pk, \mathsf{F}_{pk}(s), y)$$

where $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $pk \leftarrow \mathsf{KeyGen}(pp)$, $s \leftarrow \mathsf{SampDom}(r)$, $y \xleftarrow{\mathrm{R}} Y$.

Corollary 1. *Let* F *be a family of efficiently computable injective functions.* F *is one-way if and only if* F *has a hardcore function* hc*.*

5.1 Construction from (Adaptive) Injective TDFs

From (adaptive) injective TDFs, we construct PEPRFs as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, run $\mathrm{TDF}.\mathsf{Setup}(\lambda)$ to generate $pp = (\mathsf{TDF}, PK, SK, S, U)$ for TDFs, and let $\mathsf{hc} : S \to Y$ be a corresponding hardcore function; then create public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs as follows: set PK and SK the same as that of TDFs, set Y the same as that of the hardcore function, set $X = U$, $W = S$, F will be defined latter. The algorithm $\mathrm{TDF}.\mathsf{Eval}$ induces a collection of languages $L = \{L_{pk}\}_{pk \in PK}$ over X where each $L_{pk} = \{x : \exists w \in W \text{ s.t. } x = \mathrm{TDF}.\mathsf{Eval}(pk, w)\}$. The sampling algorithm $\mathsf{SampRel}$ on input r, runs $s \leftarrow \mathsf{SampDom}(r)$, computes $u \leftarrow \mathrm{TDF}.\mathsf{Eval}(pk, s)$, and then outputs $x = u$ and $w = s$. We assume the public parameters pp of TDFs and PEPRFs contain essentially the same information.
- $\mathsf{KeyGen}(pp)$: on input pp, output $(pk, sk) \leftarrow \mathrm{TDF}.\mathsf{KeyGen}(pp)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \leftarrow \mathsf{hc}(\mathrm{TDF}.\mathsf{TdInv}(sk, x))$. This algorithm defines $\mathsf{F} : SK \times X \to Y \cup \perp$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L_{pk}$ together with a witness w, output $y \leftarrow \mathsf{hc}(w)$.

The correctness of the above construction follows from the correctness and injectivity of TDFs. For the security, we have the following theorem.

Theorem 5.1. *PEPRFs are weak pseudorandom (resp. adaptive weak pseudorandom) if the underlying TDFs are one-way (resp. adaptive one-way).*

Proof. The proof is straightforward, which transforms an adversary $\mathcal{A}$ against weak pseudorandomness (resp. adaptive weak pseudorandomness) of PEPRFs to an adversary $\mathcal{B}$ against pseudorandomness (resp. adaptive pseudorandomness) of hc, and thus contradicts to the assumed one-wayness (resp. adaptive one-wayness) of TDFs. Here, we only prove the adaptive weak pseudorandomness case and the weak pseudorandomness case immediately follows.

$\mathcal{B}$ simulates $\mathcal{A}$'s challenger in the adaptive weak pseudorandomness experiment for PEPRFs.

- Setup and challenge: $\mathcal{B}$ receives (pp, pk, u^*, y_b^*) from its own challenger, where $pp \leftarrow \mathrm{TDF}.\mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathrm{TDF}.\mathsf{KeyGen}(pp)$, $u^* \leftarrow \mathsf{TDF}.\mathsf{Eval}(pk, s^*)$ for some $s^* \xleftarrow{\mathrm{R}} S$, and y_b^* is $\mathsf{hc}(s^*)$ if $b = 0$ or randomly chosen from Y if $b = 1$. $\mathcal{B}$ sets $x^* = u^*$, sends (pp, pk, x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Evaluation queries: on evaluation query $x \neq x^*$, $\mathcal{B}$ queries the inversion oracle at point x and gets the response s, $\mathcal{B}$ then responds with $\mathsf{hc}(s)$.
- Guess: $\mathcal{A}$ outputs its guess b' for b and $\mathcal{B}$ forwards b' to its own challenger.

Clearly, $\mathcal{B}$ simulates the adaptive weak pseudorandomness experiment perfectly. Therefore, $\mathcal{B}$ can break pseudorandomness of hc with advantage at least $\mathsf{Adv}_{\mathcal{A}}(\lambda)$. Assuming the one-wayness of the underlying TDFs, $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible in λ. This concludes the proof. □

6 Connection to Hash Proof Systems

Hash proof systems are 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 them.

Definition 6.1 (Hash Proof System)**.** An HPS consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, 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 a language defined over X, W is the associated witness set, and α is a projection from SK to PK.
- $\mathsf{KeyGen}(pp)$: on input pp, pick $sk \xleftarrow{R} SK$, compute $pk \leftarrow \alpha(sk)$, output (pk, sk).
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, output $\pi = \Lambda_{sk}(x)$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness w, output $\pi = \Lambda_{sk}(x)$.

Following [CS02, KPSY09] we introduce the following notions and properties for Λ on $X \backslash L$.

Collision probability. The collision probability of Λ is defined as:

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

Universal$_1$. Λ is ϵ-universal$_1$ if for all $x \in X \backslash L$,

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

where $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and $\pi \xleftarrow{R} \Pi$.

[CS02] introduced a relaxation of universal$_1$ property, named smoothness, which only requires the universal$_1$ property holds in the average case.

Smoothness. Λ is ϵ-smooth if for $x \xleftarrow{R} X \backslash L$,

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

where $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and $\pi \xleftarrow{R} \Pi$.

[CS02] also introduced a strengthening of universal$_1$ property, named universal$_2$ property.

Universal$_2$. 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 $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and $\pi \xleftarrow{R} \Pi$.

Kiltz, Pietrzak, Stam, and Yung [KPSY09] (henceforth KPSY) provides a generic transform from universal$_1$ to universal$_2$ HPS. Given a HPS with hash $\Lambda : SK \times X \to \Pi$ regarding projection $\alpha : SK \to PK$ and a family of functions $\mathsf{H} = \{\mathsf{h} : \Pi \to \{0,1\}^\ell\}$, we can define its hashed variant $\mathrm{HPS}^{\mathsf{H}}$ with hash $\Lambda^{\mathsf{H}} : SK^{\mathsf{H}} \times X \to \{0,1\}^\ell$ regarding $\alpha^{\mathsf{H}} : SK^{\mathsf{H}} \to PK^{\mathsf{H}}$ where $PK^{\mathsf{H}} = PK \times \mathsf{H}$, $SK^{\mathsf{H}} = SK \times \mathsf{H}$, $\Lambda^{\mathsf{H}}((sk, \mathsf{h}), x) = \mathsf{h}(\Lambda(sk, x))$ and $\alpha^{\mathsf{H}}(sk, \mathsf{h}) = (\alpha(sk), \mathsf{h})$. Note that X and L are the same for HPS and $\mathrm{HPS}^{\mathsf{H}}$.

Theorem 6.1 ([KPSY09])**.** *Assume Λ is ϵ_1-universal$_1$ with collision probability $\delta \leq 1/2$ and $\mathsf{H} = \{\mathsf{h} : \Pi \to \{0,1\}^\ell\}$ (where $\ell \geq 6$) is a family of 4-wise independent hash functions, then Λ^{H} is ϵ_2-universal$_2$ for:*

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

6.1 Construction from Smooth HPS

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

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, run $\text{HPS.}\mathsf{Setup}(1^\lambda)$ to generate a smooth HPS instance $pp = (\Lambda, PK, SK, X, L, W, \Pi, \alpha)$, then create $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs as follows: set PK, SK, X, L, and W the same as that of HPS, set $\mathsf{F} = \Lambda$, $Y = \Pi$. We assume the public parameters pp of HPS and PEPRFs contain essentially the same information.
- $\mathsf{KeyGen}(pp)$: on input pp, output $(pk, sk) \leftarrow \text{HPS.}\mathsf{KeyGen}(pp)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \leftarrow \text{HPS.}\mathsf{PrivEval}(sk, x)$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, output $y \leftarrow \text{HPS.}\mathsf{PubEval}(pk, x, w)$.

We have the following theorem about the above construction.

Theorem 6.2. *If the underlying subset membership problem is hard, then the PEPRFs from the smooth HPSs are weak pseudorandom.*

Proof. The proof is similar to [CS02]. To establish weak pseudorandomness based on 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 (standard weak pseudorandomness experiment for PEPRFs)

- Setup and challenge: $\mathcal{CH}$ runs $\text{HPS.}\mathsf{Setup}(1^\lambda)$ to build public parameters pp for PEPRFs, and runs $\text{HPS.}\mathsf{KeyGen}(pp)$ to generate a key pair (pk, sk). $\mathcal{CH}$ then runs $(x^*, w^*) \leftarrow \mathsf{SampRel}(r^*)$ for $r^* \xleftarrow{R} R$, computes $\pi^* \leftarrow \Lambda_{sk}(x^*)$ via $\text{HPS.}\mathsf{PubEval}(pk, x^*, w^*)$, sets $y_0^* = \pi^*$, and picks $y_1^* \xleftarrow{R} Y$. Finally, $\mathcal{CH}$ picks $b \xleftarrow{R} \{0, 1\}$, and sends (pp, pk, 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:

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

Game 1: same as Game 0 except that $\mathcal{CH}$ computes $\pi^* \leftarrow \Lambda_{sk}(x^*)$ via $\text{HPS.}\mathsf{PrivEval}(sk, x^*)$. According to the functionality of $\mathsf{PrivEval}$ and $\mathsf{PubEval}$, 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 $\mathcal{CH}$ samples x^* via algorithm SampNo instead of $\mathsf{SampRel}$. The sampling indistinguishability (based on the hardness of the subset membership problem) ensures that:

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

Game 3: same as Game 2 except that $\mathcal{CH}$ sets y_0^* 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 that in Game 3 $\mathcal{A}$'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. □

6.2 Construction from Smooth and Universal$_2$ HPS

From a smooth HPS and an universal$_2$ HPS for the same language $\tilde{L} \subset \tilde{X}$, we can construct adaptive weak pseudorandom PEPRFs as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, run $\mathrm{HPS}_1.\mathsf{Setup}(1^\lambda)$ to generate a smooth HPS instance $pp_1 = (\Lambda^1, PK_1, SK_1, \tilde{X}, \tilde{L}, \tilde{W}, \Pi_1, \alpha_1)$, run $\mathrm{HPS}_2.\mathsf{Setup}(1^\lambda)$ to generate an universal$_2$ HPS instance $pp_2 = (\Lambda^2, PK_2, SK_2, \tilde{X}, \tilde{L}, \tilde{W}, \Pi_2, \alpha_2)$, then build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs from pp_1 and pp_2 as follows, sets $X = \tilde{X} \times \Pi_2$, $Y = \Pi_1 \cup \perp$, $PK = PK_1 \times PK_2$, $SK = SK_1 \times SK_2$, $W = \tilde{W}$, F will be defined later. The algorithm $\mathrm{HPS}_2.\mathsf{PubEval}$ induces a collection of languages $L = \{L_{pk}\}_{pk \in PK}$ over $X = \tilde{X} \times \Pi_2$, where each $L_{pk} = \{x = (\tilde{x}, \pi_2) : \exists w \in W \text{ s.t. } \tilde{x} \in \tilde{L} \wedge \pi_2 = \mathrm{HPS}_2.\mathsf{PubEval}(pk_2, \tilde{x}, w)\}$. The corresponding sampling algorithm $L.\mathsf{SampRel}$ on input $pk = (pk_1, pk_2)$ and random coins r, runs $(\tilde{x}, \tilde{w}) \leftarrow \tilde{L}.\mathsf{SampRel}(r)$, computes $\pi_2 \leftarrow \mathrm{HPS}_2.\mathsf{PubEval}(pk_2, \tilde{x}, \tilde{w})$, sets $x = (\tilde{x}, \pi_2)$, $w = \tilde{w}$, outputs (x, w). 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 $\mathrm{HPS}_1.\mathsf{KeyGen}(pp_1)$ and $\mathrm{HPS}_2.\mathsf{KeyGen}(pp_2)$ to get (pk_1, sk_1) and (pk_2, sk_2) respectively, output $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$.
- $\mathsf{PrivEval}(sk, x)$: on input $sk = (sk_1, sk_2)$ and $x = (\tilde{x}, \pi_2)$, output $y \leftarrow \mathrm{HPS}_1.\mathsf{PrivEval}(sk_1, \tilde{x})$ if $\pi_2 = \mathrm{HPS}_2.\mathsf{PrivEval}(sk_2, \tilde{x})$ and $\perp$ otherwise. This algorithm defines $\mathsf{F} : SK \times X \to Y \cup \perp$.
- $\mathsf{PubEval}(pk, x, w)$: on input $pk = (pk_1, pk_2)$ and an element $x = (\tilde{x}, \pi_2) \in L_{pk}$ together with a witness w, output $y \leftarrow \mathrm{HPS}_1.\mathsf{PubEval}(pk_1, \tilde{x}, w)$.

We have the following theorem about the above construction.

Theorem 6.3. *If the underlying subset membership problem is hard, then the PEPRFs from the smooth and universal$_2$ HPSs are adaptive weak pseudorandom.*

Proof. The proof is similar to [CS02]. To base adaptive weak pseudorandomness on 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 (standard adaptive weak pseudorandomness experiment for PEPRFs)

- Setup and challenge: $\mathcal{CH}$ prepares the challenge via the following steps:
  1. run $\mathrm{HPS}_1.\mathsf{Setup}(1^\lambda)$ and $\mathrm{HPS}_2.\mathsf{Setup}(1^\lambda)$ to generate pp_1 and pp_2 for a smooth HPS instance and an universal$_2$ HPS instance respectively, then build public parameters pp for PEPRFs from pp_1 and pp_2; run $(pk_1, sk_1) \leftarrow \mathrm{HPS}_1.\mathsf{KeyGen}(pp_1)$ and $(pk_2, sk_2) \leftarrow \mathrm{HPS}_2.\mathsf{KeyGen}(pp_2)$, set $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$.
  2. sample $(\tilde{x}^*, \tilde{w}^*) \leftarrow \tilde{L}.\mathsf{SampRel}(r^*)$ with fresh random coins $r^* \xleftarrow{R} R$, compute $\pi_2^* = \Lambda^2_{sk_2}(\tilde{x}^*)$ via $\mathrm{HPS}_2.\mathsf{PubEval}(pk_2, \tilde{x}^*, \tilde{w}^*)$, set $x^* = (\tilde{x}^*, \pi_2^*)$; compute $\pi_1^* = \Lambda^1_{sk_1}(\tilde{x}^*)$ via $\mathrm{HPS}_1.\mathsf{PubEval}(pk_1, \tilde{x}^*, \tilde{w}^*)$, set $y_0^* = \pi_1^*$, sample $y_1^* \xleftarrow{R} Y$, pick $b \xleftarrow{R} \{0, 1\}$, and send (pp, pk, x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Evaluation query: on evaluation query at point $x \neq x^*$, $\mathcal{CH}$ 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.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b' = b$.

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

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

Game 1: same as Game 0 except that $\mathcal{CH}$ computes $\pi_2^* = \Lambda_{sk_2}^2(\tilde{x}^*)$ via $\mathrm{HPS}_2.\mathsf{PrivEval}(sk_2, \tilde{x}^*)$ and computes $\pi_1^* = \Lambda_{sk_1}^1(\tilde{x}^*)$ via $\mathrm{HPS}_1.\mathsf{PrivEval}(sk_2, \tilde{x}^*)$. According to the functionality of $\mathsf{PrivEval}$ and $\mathsf{PubEval}$, 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 $\mathcal{CH}$ samples $\tilde{x}^*$ via SampNo instead of $\mathsf{SampRel}$. The sampling indistinguishability (based on the subset membership assumption) ensures that:

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

Game 3: same as Game 2 except that when answering evaluation queries $\langle x \rangle$ where $x = (\tilde{x}, \pi_2)$, $\mathcal{CH}$ returns $\perp$ if $\tilde{x} \notin \tilde{L}$ even when $\Lambda_{sk_2}^2(\tilde{x}) = \pi_2$. For the ease of analysis, we denote by E the event that $\mathcal{A}$ submits an evaluation query $x = (\tilde{x}, \pi_2)$ such that $\tilde{x} \notin L$ but $\Lambda_{sk_2}^2(\tilde{x}) = \pi_2$. According to the $\mathrm{universal}_2$ property of HPS_2, we have $\Pr[E] \leq Q\epsilon$, where Q is the maximum number of evaluation queries that $\mathcal{A}$ may make. Since ϵ is negligible in λ, we have:

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

Game 4: same as Game 3 except that $\mathcal{CH}$ sets y_0^* as π_1', where $\pi_1' \xleftarrow{\mathrm{R}} \Pi_1$. 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 $\Lambda_{sk_1}^1$ on $\tilde{L}$ is determined by pk_1, and all evaluation queries $x = (\tilde{x}, \pi_2)$ with $\tilde{x} \in \tilde{L}$ are responded with $\perp$, 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 Λ^1, we have:

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

It is evident that in Game 4 $\mathcal{A}$'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 6.1. To ensure the adaptive weak pseudorandomness of the above construction, we can weaken $\mathrm{universal}_2$ property to weak $\mathrm{universal}_2$ property, which requires the former property holds when $x^* \xleftarrow{\mathrm{R}} X \backslash L$. This is because adaptive weak pseudorandomness only stipulates pseudorandomness holds on the average case.

6.3 Construction from $\mathrm{Universal}_1$ HPS

From an $\mathrm{universal}_1$ HPS, we can construct adaptive weak pseudorandom PEPRFs as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, run $\mathrm{HPS}.\mathsf{Setup}(1^\lambda)$ to generate pp_1 for an $\mathrm{universal}_1$ HPS instance $(\Lambda, \tilde{PK}, \tilde{SK}, \tilde{X}, \tilde{L}, \tilde{W}, \Pi, \alpha)$, then apply the KPSY-transform [KPSY09] to obtain an $\mathrm{universal}_2$ $\mathrm{HPS}^{\mathsf{H}}$, where H is a family of 4-wise independent hash functions from Π to Π^{H}; then build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs from pp_1 and H as follows, where $X = \tilde{X} \times \Pi^{\mathsf{H}}$, $Y = \Pi \cup \perp$, $PK = \tilde{PK} \times \tilde{PK} \times \mathsf{H}$, $SK = \tilde{SK} \times \tilde{SK} \times \mathsf{H}$, $W = \tilde{W}$, F will be defined later. The algorithm $\mathrm{HPS}.\mathsf{PubEval}$ defines a collection of languages $L = \{L_{pk}\}_{pk \in PK}$ over $X = \tilde{X} \times \Pi^{\mathsf{H}}$ where each $L_{pk} = \{x = (\tilde{x}, \pi^{\mathsf{h}}) : \tilde{x} \in \tilde{L} \wedge \pi^{\mathsf{h}} = \mathsf{h}(\mathrm{HPS}.\mathsf{PubEval}(\tilde{pk}_2, \tilde{x}, \tilde{w}))\}$. It is easy to see that a witness $\tilde{w}$ for $\tilde{x} \in \tilde{L}$ is also a witness for $x = (\tilde{x}, \pi^{\mathsf{h}}) \in L_{pk}$. The corresponding sampling algorithm on input $pk = (\tilde{pk}_1, \tilde{pk}_2, \mathsf{h})$ and random coins r, samples $(\tilde{x}, \tilde{w}) \leftarrow \tilde{L}.\mathsf{SampRel}(r)$, computes $\pi^{\mathsf{h}} \leftarrow \mathsf{h}(\mathrm{HPS}.\mathsf{PubEval}(\tilde{pk}_2, \tilde{x}, \tilde{w}))$, sets $x = (\tilde{x}, \pi^{\mathsf{h}})$ and $w = \tilde{w}$, outputs (x, w).

- $\mathsf{KeyGen}(pp)$: on input $pp = (pp_1, \mathsf{H})$, run $\mathrm{HPS}.\mathsf{KeyGen}(pp_1)$ twice independently to get $(\tilde{pk}_1, \tilde{sk}_1)$ and $(\tilde{pk}_2, \tilde{sk}_2)$, pick $\mathsf{h} \xleftarrow{R} \mathsf{H}$, set $pk = (\tilde{pk}_1, \tilde{pk}_2, \mathsf{h})$, $sk = (\tilde{sk}_1, \tilde{sk}_2, \mathsf{h})$, output (pk, sk).
- $\mathsf{PrivEval}(sk, x)$: on input $sk = (\tilde{sk}_1, \tilde{sk}_2, \mathsf{h})$ and $x = (\tilde{x}, \pi^{\mathsf{h}})$, output $y \leftarrow \mathrm{HPS}.\mathsf{PrivEval}(\tilde{sk}_1, \tilde{x})$ if $\pi^{\mathsf{h}} = \mathsf{h}(\mathrm{HPS}.\mathsf{PrivEval}(\tilde{sk}_2, \tilde{x}))$ and $\perp$ otherwise. This algorithm defines $\mathsf{F} : SK \times X \to Y \cup \perp$.
- $\mathsf{PubEval}(pk, x, w)$: on input $pk = (\tilde{pk}_1, \tilde{pk}_2, \mathsf{h})$ and an element $x = (\tilde{x}, \pi^{\mathsf{h}}) \in L_{pk}$ together with a witness w, output $y \leftarrow \mathrm{HPS}.\mathsf{PubEval}(\tilde{pk}_1, \tilde{x}, w)$.

In the above construction, we implicitly build $\mathrm{HPS}^{\mathsf{H}}$, which is universal$_2$ according to Theorem 6.1. We have the following theorem about the above construction.

Theorem 6.4. *If the underlying subset membership problem is hard, then the PEPRFs from the universal$_1$ HPSs are adaptive weak pseudorandom.*

Proof. The proof is similar as that in Section 6.2. To base adaptive weak pseudorandomness on the universal$_1$ property of Λ and the subset membership problem assumption, 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 (standard adaptive weak pseudorandomness game)

- Setup and challenge: $\mathcal{CH}$ prepares the challenge via the following steps:
  1. run $\mathrm{HPS}.\mathsf{Setup}(1^\lambda)$ to generate pp_1 for an universal$_1$ HPS instance and choose a family of 4-wise independent hash functions H from Π to Π^{H}, then build public parameters pp for PEPRFs from pp and H; runs $\mathrm{HPS}.\mathsf{KeyGen}(pp)$ twice independently to generate $(\tilde{pk}_1, \tilde{sk}_1)$ and $(\tilde{pk}_2, \tilde{sk}_2)$, pick $\mathsf{h} \xleftarrow{R} \mathsf{H}$, sets $pk = (\tilde{pk}_1, \tilde{pk}_2, \mathsf{h})$ and $sk = (\tilde{sk}_1, \tilde{sk}_2, \mathsf{h})$.
  2. sample $(\tilde{x}^*, \tilde{w}^*) \leftarrow \tilde{L}.\mathsf{SampRel}(r^*)$ with fresh random coins $r^* \xleftarrow{R} R$, compute $\pi^{\mathsf{h}^*} = \mathsf{h}(\Lambda_{\tilde{sk}_2}(\tilde{x}^*))$ via $\mathsf{h}(\mathrm{HPS}.\mathsf{PubEval}(\tilde{pk}_2, \tilde{x}^*, \tilde{w}^*))$, set $x^* = (\tilde{x}^*, \pi^{\mathsf{h}^*})$; computes $\pi^* = \Lambda_{\tilde{sk}_1}(\tilde{x}^*)$ via $\mathrm{HPS}.\mathsf{PubEval}(\tilde{pk}_1, \tilde{x}^*, \tilde{w}^*)$, set $y_0^* = \pi^*$, pick $y_1^* \xleftarrow{R} Y$, pick $b \xleftarrow{R} \{0, 1\}$, and send (pp, pk, x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Evaluation query: on evaluation queries at point $x \neq x^*$, $\mathcal{CH}$ responds normally with $sk = (\tilde{sk}_1, \tilde{sk}_2, \mathsf{h})$. More precisely, $\mathcal{CH}$ parses x as $(\tilde{x}, \pi^{\mathsf{h}})$, then responds with $\Lambda_{\tilde{sk}_1}(\tilde{x})$ if $\mathsf{h}(\Lambda_{\tilde{sk}_2}(\tilde{x})) = \pi^{\mathsf{h}}$ and $\perp$ otherwise.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b' = b$.

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

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

Game 1: same as Game 0 except $\mathcal{CH}$ computes $\pi^{\mathsf{h}^*} = \mathsf{h}(\Lambda_{\tilde{sk}_2}(\tilde{x}^*))$ via $\mathsf{h}(\mathrm{HPS}.\mathsf{PrivEval}(\tilde{sk}_2, \tilde{x}^*))$ and computes $y_0^* = \Lambda_{\tilde{sk}_1}(\tilde{x}^*)$ via $\mathrm{HPS}.\mathsf{PrivEval}(\tilde{sk}_1, \tilde{x}^*)$. According to the functionality of $\mathsf{PrivEval}$ and $\mathsf{PubEval}$, 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 $\mathcal{CH}$ samples $\tilde{x}^*$ via SampNo instead of $\mathsf{SampRel}$. The sampling indistinguishability (based on the subset membership assumption) ensures that:

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

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

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

Game 4: same as Game 3 except that $\mathcal{CH}$ sets y_0^* as π', where $\pi' \xleftarrow{\mathrm{R}} \Pi$. The rest reasoning is similar as that presented in Section 6.2. It now follows directly from the ϵ_1-universal$_1$ property of Λ, we have:

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

It is evident that in Game 4 $\mathcal{A}$'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 6.2. Construction 6.1 is straightforward, while the construction 6.2 and construction 6.3 are more technically 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 being universal$_2$. The advantage of construction 6.2 is that both HPSs exactly satisfy the respective minimum requirements, while the disadvantage is that we have to assume that there exists a strong HPS sharing the same X and L as that of the weak HPS. On the contrary, the advantage of construction 6.3 is that we can start from a single weak HPS and then build a strong HPS from it, 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 KPSY transform cannot convert a smooth HPS into an universal$_2$ one.

7 Connection to Extractable Hash Proof Systems

Extractable hash proof systems (EHPS) were introduced by Wee [Wee10] as a paradigm of constructing PKE from search problems associated with hard relations. In the following, we recall the notion of EHPS and then show how to construct PEPRF from them.

Definition 7.1 (Extractable Hash Proof System)**.** An EHPS consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyGen}', \mathsf{PubEval}, \mathsf{PrivEval}, \mathsf{Ext})$ as below:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters pp which include an EHPS instance $(\mathsf{H}, PK, SK, L, W, \Pi)$, where $\mathsf{H} : PK \times L \to \Pi$ can be viewed as a keyed function indexed by PK. Let R_L be a corresponding hard relation for L, and $\mathsf{hc} : W \to Y$ be a hardcore function for R_L.
- $\mathsf{KeyGen}(pp)$: 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').
- $\mathsf{PubEval}(pk, x, r)$: on input pk, x and r, output $\pi = \mathsf{H}_{pk}(x)$ where $x \leftarrow \mathsf{SampLan}(r)$.
- $\mathsf{PrivEval}(sk', x)$: on input sk' and $x \in X$, output $\pi \leftarrow \mathsf{H}_{pk}(x)$.
- $\mathsf{Ext}(sk, x, \pi)$: on input sk, $x \in X$, and $\pi \in \Pi$, output $w \in W$ such that $(x, w) \in \mathsf{R}_L$ if $\pi = \mathsf{H}_{pk}(x)$ and $w = \perp$ if not.

In an EHPS, KeyGen' and $\mathsf{PrivEval}$ work in the hashing mode, which are only used to establish security. An EHPS satisfies the following property:

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

Definition 7.2 (All-but-One Extractable Hash Proof System)**.** An all-but-one (ABO) EHPS is a richer abstraction of EHPS, besides algorithms (Setup, KeyGen, KeyGen', $\mathsf{PubEval}$, $\mathsf{PrivEval}$, Ext), it has an additional algorithm Ext'.

- $\mathsf{KeyGen}'(pp, x^*)$: on input public parameters pp and an arbitrary $x^* \in X$, output a key pair (pk, sk').
- $\mathsf{PrivEval}(sk', x^*)$: on input sk' and x^*, output $\pi^* = \mathsf{H}_{pk}(x^*)$.
- $\mathsf{Ext}'(sk', x, \pi)$: on input sk', $x \in X$ such that $x \neq x^*$, and $\pi \in \Pi$, output $w \in W$ such that $(x, w) \in \mathsf{R}_L$ if $\pi = \mathsf{H}_{pk}(x)$ and $w = \perp$ otherwise.

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

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

7.1 Construction from (All-But-One) EHPS

From an (ABO) EHPS, we construct PEPRFs as follows:

- $\mathsf{Setup}(1^\lambda)$: run EHPS.$\mathsf{Setup}(1^\lambda)$ to generate an EHPS instance $pp = (\mathsf{H}, PK, SK, \tilde{L}, \tilde{W}, \Pi)$, and let $\mathsf{hc} : \tilde{W} \to Y$ be a hardcore function for $\mathsf{R}_{\tilde{L}}$; build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs as follows: sets $X = \tilde{X} \times \Pi$, $Y = Y$, $W = R$, and F will be defined later. The algorithm EHPS.$\mathsf{PubEval}$ induces a collection of languages $L = \{L_{pk}\}_{pk \in PK}$ over $X = \tilde{X} \times \Pi$ where each $L_{pk} = \{x = (\tilde{x}, \pi) : \exists w \in W \text{ s.t. } \tilde{x} = \mathsf{SampLan}(w) \wedge \pi = \text{EHPS.}\mathsf{PubEval}(pk, \tilde{x}, w)\}$. The corresponding sampling algorithm $\mathsf{SampRel}$ on input r, computes $\tilde{x} \leftarrow \mathsf{SampLan}(r)$ and $\pi \leftarrow \text{EHPS.}\mathsf{PubEval}(pk, \tilde{x}, r)$, outputs $x = (\tilde{x}, \pi)$ and $w = r$. We assume the public parameters pp of PEPRF and EHPS essentially contains the same information.
- $\mathsf{KeyGen}(pp)$: on input pp, output $(pk, sk) \leftarrow \text{EHPS.}\mathsf{KeyGen}(pp)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, parse x as $(\tilde{x}, \pi)$, compute $\tilde{w} \leftarrow \text{EHPS.}\mathsf{Ext}(sk, \tilde{x}, \pi)$, output $y \leftarrow \mathsf{hc}(\tilde{w})$. This algorithm defines $\mathsf{F}_{sk}(x)$ as $\mathsf{hc}(\text{EHPS.}\mathsf{Ext}(sk, x))$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, compute $\tilde{w} \leftarrow \mathsf{SampWit}(w)$, output $y \leftarrow \mathsf{hc}(\tilde{w})$.

We have the following two theorems about the above construction.

Theorem 7.1. *If the underlying binary relation $\mathsf{R}_{\tilde{L}}$ is one-way, then the PEPRFs from the EHPSs are weak pseudorandom.*

Proof. The proof is similar to [Wee10]. To establish the weak pseudorandomness based on the one-wayness of $\mathsf{R}_{\tilde{L}}$, 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 (standard weak pseudorandomness experiment for PEPRFs)

- Setup and challenge: $\mathcal{CH}$ prepares the challenge by operating the EHPS in the extraction mode in the following steps.

1. run EHPS.$\mathsf{Setup}(1^\lambda)$ to generate an EHPS instance and build public parameters pp for PEPRFs from it, then run EHPS.$\mathsf{KeyGen}(pp)$ to generate (pk, sk).
2. sample $\tilde{x}^* \leftarrow \mathsf{SampLan}(r^*)$ and $\tilde{w}^* \leftarrow \mathsf{SampWit}(r^*)$ using fresh random coins $r^* \xleftarrow{\mathrm{R}} R$, compute $\pi^* \leftarrow \mathsf{H}_{pk}(\tilde{x}^*)$ via EHPS.$\mathsf{PubEval}(pk, \tilde{x}^*, r^*)$, set $x^* = (\tilde{x}^*, \pi^*)$, compute $y_0^* \leftarrow \mathsf{hc}(\tilde{w}^*)$, pick $y_1^* \xleftarrow{\mathrm{R}} Y$; finally, pick $b \xleftarrow{\mathrm{R}} \{0,1\}$, and send (pp, pk, 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, we have:

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

Game 1: The differences to Game 0 are that $\mathcal{CH}$ preparing the challenge instance by operating EHPS in the hashing mode. That is, $\mathcal{CH}$ runs EHPS.$\mathsf{KeyGen}'(pp)$ to generate (pk, sk') and computes the value $\pi^* \leftarrow \mathsf{H}_{pk}(\tilde{x}^*)$ via EHPS.$\mathsf{PrivEval}(sk', \tilde{x}^*)$. According to indistinguishability between $\mathsf{KeyGen}(pp)$ and $\mathsf{KeyGen}'(pp)$ as well as the correctness of the hashing mode, we have:

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

We then show that no PPT adversary has non-negligible advantage in Game 1. 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 R_L also with non-negligible advantage.

$\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 1 as below.

- Setup and challenge: $\mathcal{B}$ proceeds the same way as $\mathcal{CH}$ does in Game 1 except that it receives $(\tilde{x}^*, y_b^*)$ from its own challenger.
- 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:

$$\mathsf{Adv}_{\mathcal{B}}(\lambda) = |\Pr[S_2] - 1/2| = \mathsf{Adv}_{\mathcal{A}}(\lambda) \tag{14}$$

The theorem immediately follows. □

Theorem 7.2. *If the underlying binary relation $\mathsf{R}_{\tilde{L}}$ is hard, then the PEPRFs from the ABO EHPSs are adaptive weak pseudorandom.*

Proof. The proof follows immediately from [Wee10]. To base adaptive weak pseudorandomness of PEPRFs on the properties of EHPS and one-wayness of $\mathsf{R}_{\tilde{L}}$, 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 (standard adaptive weak pseudorandomness experiment for PEPRFs)

- Setup and challenge: $\mathcal{CH}$ prepare the challenge instance by operating the ABO EHPS in the extraction mode in the following steps.
  1. run EHPS.$\mathsf{Setup}(\lambda)$ to generate an EHPS instance and build public parameters pp for PEPRFs from it, then runs EHPS.$\mathsf{KeyGen}(pp)$ to generate (pk, sk).
  2. sample $\tilde{x}^* \leftarrow \mathsf{SampLan}(r^*)$ and $\tilde{w}^* \leftarrow \mathsf{SampWit}(r^*)$ with fresh random coins $r^* \xleftarrow{\mathrm{R}} R$, compute $\pi^* = \mathsf{H}_{pk}(\tilde{x}^*)$ via EHPS.$\mathsf{PubEval}(pk, r^*)$, set $x^* = (\tilde{x}^*, \pi^*)$, compute $y_0^* \leftarrow \mathsf{hc}(\tilde{w}^*)$, and pick $y_1^* \xleftarrow{\mathrm{R}} Y$; finally, pick $b \xleftarrow{\mathrm{R}} \{0,1\}$ and send (pp, pk, x^*, y_b^*) to $\mathcal{A}$ as the challenge.

- Evaluation queries: When $\mathcal{A}$ issues evaluation query at point $x \neq x^*$, $\mathcal{CH}$ parses x as $(\tilde{x}, \pi)$, then computes $\tilde{w} \leftarrow \mathsf{EHPS.Ext}(sk, \tilde{x}, \pi)$ and responds with $\mathsf{hc}(\tilde{w})$.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b = b'$.

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

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

Game 1: The differences to Game 0 is that $\mathcal{CH}$ prepares the challenge and handles evaluation queries by operating ABO EHPS in the ABO hashing mode.

- Setup and challenge: $\mathcal{CH}$ prepares the challenge via the following steps.
  1. run $\mathsf{EHPS.Setup}(1^\lambda)$ to generate an ABO EHPS instance and build public parameters pp for PEPRFs from it; sample $\tilde{x}^* \leftarrow \mathsf{SampLan}(r^*)$ and $\tilde{w}^* \leftarrow \mathsf{SampWit}(r^*)$ using fresh random coins $r^* \xleftarrow{R} R$, then run $\mathsf{EHPS.KeyGen}'(pp, \tilde{x}^*)$ to generate (pk, sk').
  2. compute $\pi^* \leftarrow \mathsf{H}_{pk}(\tilde{x}^*)$ via $\mathsf{EHPS.PrivEval}(sk', \tilde{x}^*)$, set $x^* = (\tilde{x}^*, \pi^*)$; compute $y_0^* = \mathsf{hc}(\tilde{w}^*)$ and picks $y_1^* \xleftarrow{R} Y$; finally, pick $b \xleftarrow{R} \{0,1\}$ and send (pp, pk, x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Evaluation queries: When $\mathcal{A}$ issues evaluation query at point $x \neq x^*$, $\mathcal{CH}$ parses x as $(\tilde{x}, \pi)$, then computes $\tilde{w} \leftarrow \mathsf{EHPS.Ext}'(sk', \tilde{x}, \pi)$ and responds with $\mathsf{hc}(\tilde{w})$.
- 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:

$$\Pr[S_1] = \Pr[S_0] \tag{15}$$

We then show that no PPT adversary has non-negligible advantage in Game 1. 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.

$\mathcal{B}$ simulates $\mathcal{A}$'s challenger in Game 1 as below.

- Setup and challenge: $\mathcal{B}$ proceeds the same way as $\mathcal{CH}$ does in Game 1 except that it receives $(\tilde{x}^*, y_b^*)$ from its own challenger.
- Evaluation queries: $\mathcal{B}$ proceeds the same way as $\mathcal{CH}$ does in Game 1.
- 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:

$$\mathsf{Adv}_{\mathcal{B}}(\lambda) = |\Pr[S_1] - 1/2| \approx \mathsf{Adv}_{\mathcal{A}}(\lambda)$$

The theorem immediately follows. □

8 Connection to Puncturable PRFs

Puncturable PRFs (PPRFs) are a special type of constrained PRFs, where the constrained key is associated with a single point $x^* \in X$. Such constrained key allows evaluation at all points $x \neq x^*$. As noted by [KPTZ13, BW13, BGI14], the GGM tree-based construction of PRFs from one-way functions (OWFs) [GGM86] can be modified to construct PPRFs.

8.1 Construction from Puncturable PRFs and Obfuscation

Sahai and Waters [SW14] showed how to construct PKE/KEM from indistinguishability obfuscator ($i\mathcal{O}$), pseudorandom generator (PRG), and PPRFs. We refer to [SW14] for the definitions of $i\mathcal{O}$ and PPRFs. We observe that a slight modification of their construction essentially gives us a way to compile PPRFs to PEPRFs via $i\mathcal{O}$, which proceeds as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, generate an $i\mathcal{O}$ for circuit class $\mathcal{C}_\lambda$, a length-doubling pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$, and a puncturable PRF $\mathsf{PPRF} : K \times \{0,1\}^{2\lambda} \to Y$; then build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PEPRFs, where $PK = i\mathcal{O}(\mathcal{C}_\lambda)$, $SK = K$, $X = \{0,1\}^{2\lambda}$, $\mathsf{F} = \mathsf{PPRF}$, $W = \{0,1\}^\lambda$, and $L = \{x : \exists w \in W \text{ s.t. } x = \mathsf{PRG}(w)\}$. The corresponding sampling algorithm $\mathsf{SampRel}$ on input $r \in \{0,1\}^\lambda$, outputs $x \leftarrow \mathsf{PRG}(r)$ and $w = r$.
- $\mathsf{KeyGen}(pp)$: on input pp, pick a puncturable PRF key $k \in K$ as the secret key sk, create an obfuscation of the program of Figure 3 (the size of the program is padded to be the maximum of itself and of Figure 4) as the public key pk.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, output $y \leftarrow \mathsf{PPRF}(sk, x)$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk, an element $x \in L$ together with its witness w, output $y \leftarrow pk(x, w)$.

Public Evaluation

Constants: puncturable PRF key k.
Input: element $x \in \{0,1\}^{2\lambda}$, witness $w \in \{0,1\}^\lambda$.

1. If $x = \mathsf{PRG}(w)$, output $\mathsf{PPRF}(k, x)$.
2. Else, output $\perp$.

Figure 3: Program Public Evaluation

Public Evaluation*

Constants: puncturable PRF key $k(\{x^*\})$.
Input: element $x \in \{0,1\}^{2\lambda}$, witness $w \in \{0,1\}^\lambda$.

1. If $x = \mathsf{PRG}(w)$, output $\mathsf{PPRF}(k(\{x^*\}), x)$.
2. Else, output $\perp$.

Figure 4: Program Public Evaluation*

Theorem 8.1. *If the $i\mathcal{O}$ is indistinguishably secure, PRG is a secure pseudorandom generator, and PPRF is selective pseudorandom, then the PEPRFs are adaptive weak pseudorandom.*

Proof. The proof follows immediately from [SW14]. Here we only describe the proof in sketch. To base adaptive weak pseudorandomness of PEPRFs on the security of PRG, $i\mathcal{O}$, and PPRF, we proceed via a sequence of games where the first game corresponds to the original adaptive weak pseudorandom game for PEPRFs. We prove that $\mathcal{A}$'s advantage must be negligible close between each successive game and that $\mathcal{A}$ has zero advantage in the final game.

Game 0 (standard adaptive weak pseudorandomness experiment for PEPRFs)

- Setup and challenge: $\mathcal{CH}$ runs PEPRF.$\mathsf{Setup}(1^\lambda)$ to generate public parameters pp for PEPRFs, runs PEPRF.$\mathsf{KeyGen}(pp)$ to generate (pk, sk). $\mathcal{CH}$ then picks $r^* \xleftarrow{R} \{0,1\}^\lambda$, computes $x^* \leftarrow \mathsf{PRG}(r^*)$, sets $y_0^* = \mathsf{PrivEval}(sk, x^*)$ and $y_1^* \xleftarrow{R} Y$, picks a random bit $b \in \{0,1\}$, and sends (pp, pk, x^*, y_b^*) to $\mathcal{A}$ as the challenge.
- Evaluation queries: For evaluation queries $x \neq x^*$, $\mathcal{CH}$ answers with PEPRF.$\mathsf{PrivEval}(sk, x)$.
- Guess: $\mathcal{A}$ outputs its guess b' and wins if $b' = b$.

Game 1: same as Game 0 with the exception that x^* is chosen randomly from $\{0,1\}^{2\lambda}$. Note that r^* is no longer in $\mathcal{A}$'s view and does not need to be generated.
Game 2: same as Game 1 except that the public key is created as an obfuscation of the program Public Evaluation* of Figure 4.
Game 3: same as Game 2 except that y_0^* is randomly chosen from Y.

Game 0 and Game 1 are computationally indistinguishable assuming the security of the PRG. Game 1 and Game 2 are computationally indistinguishable assuming the security of the underlying obfuscation. Game 2 and Game 3 are computationally indistinguishable assuming the selective pseudorandomness of the puncturable PRF. Finally, we observe that any adversary's advantage in Game 3 must be zero, since both y_0^* and y_1^* are randomly chosen from Y. Putting all these together, the desire result immediately follows because the advantage of all PPT adversaries are negligibly close in each successive game. □

9 Publicly Sampleable PRFs

In this section, we consider a relaxation of the functionality for PEPRFs, that is, instead of requiring $\mathsf{F}_{sk}(x)$ is publicly evaluable on L, we only require that the distribution $(x, \mathsf{F}_{sk}(x))$ is efficiently sampleable over $L \times Y$. More precisely, we require there exist a PPT algorithm $\mathsf{PubSamp}$ that on input fresh random coins r, outputs $(x, y) \in L \times Y$ such that $y = \mathsf{F}_{sk}(x)$. We refer to this relaxed notion as publicly sampleable PRFs (PSPRFs). Clearly, PEPRFs imply PSPRFs. The (adaptive) weak pseudorandomness for PSPRFs can be defined analogously. It is easy to verify that PSPRFs and KEM imply each other by viewing PSPRF.$\mathsf{PubSamp}$ (resp. PSPRF.$\mathsf{PrivEval}$) as KEM.Encaps (resp. KEM.Decaps).[10] In light of this observation, we view PSPRF 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.

9.1 Trapdoor Relations

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 EHPSs.

Definition 9.1 (Trapdoor Relations)**.** A family of trapdoor relations consists of four polynomial-time algorithms as below.

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters $pp = (\mathsf{R}, U, S, PK, SK)$ (these sets are parameterized by λ), and a collection of binary relations $\mathsf{R} : U \times S$ indexed by PK. We say R is one-to-one if for any $pk \in PK$, we have: 1) for any $s \in S$ there exists one and

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

only one $u \in U$ such that $(u, s) \in \mathsf{R}_{pk}$; 2) for any $u \in U$ there exists at most one $s \in S$ such that $(u, s) \in \mathsf{R}_{pk}$.

- $\mathsf{KeyGen}(pp)$: on input pp, output a public key pk (serve as a key to sample a relation) and a secrey key sk (serve as a trapdoor to find a match).
- $\mathsf{TdInv}(td, u)$: on input td and $u \in U$, output $s \in S$ or a distinguished symbol $\perp$ indicating u does not have a match under R_{pk}.
- $\mathsf{SampRel}(pk; r)$: on input pk and random coins r, output a tuple $(u, s) \in \mathsf{R}_{pk}$.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, and any $(u, s) \leftarrow \mathsf{SampRel}(pk)$, we have $(u, \mathsf{TdInv}(td, u)) \in \mathsf{R}_{pk}$.

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

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[(u^*, s) \in \mathsf{R}_{pk} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (u^*, s^*) \leftarrow \mathsf{SampRel}(pk); \\ s \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{inv}}(\cdot)}(pp, pk, u^*) \end{array}\right],$$

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

9.2 PSPRFs from TDRs

From one-to-one TDRs, we construct PSPRFs as follows:

- $\mathsf{Setup}(1^\lambda)$: run $\mathrm{TDR}.\mathsf{Setup}(1^\lambda)$ to generate $pp = (\mathsf{R}, U, S, PK, SK)$, and let $\mathsf{hc} : S \to Y$ be a hardcore function for R; build public parameters $pp = (\mathsf{F}, PK, SK, X, L, W, Y)$ for PSPRFs as follows: set PK and SK the same as that of TDRs, set $X = U$, $W = S$, $Y = Y$, and F will be defined later. $\mathsf{R} = \{\mathsf{R}_{pk}\}_{pk \in PK}$ induces a collection of languages $L = \{L_{pk}\}_{pk \in PK}$ over $X = U$ where each $L_{pk} = \{x : \exists w \in W \text{ s.t. } (x, w) \in \mathsf{R}_{pk}\}$. The corresponding sampling algorithm is the same as that of TDRs. We assume the public parameters pp of TDRs and PEPRFs essentially contains same information.
- $\mathsf{KeyGen}(pp)$: on input pp, output $(pk, sk) \leftarrow \mathrm{TDR}.\mathsf{KeyGen}(pp)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, output $y \leftarrow \mathsf{hc}(\mathrm{TDR}.\mathsf{TdInv}(sk, x))$. This algorithm defines $\mathsf{F}_{sk} : X \to Y \cup \perp$.
- $\mathsf{PubSamp}(pk; r)$: on input pk and random coins r, compute $(x, w) \leftarrow \mathsf{SampRel}(pk; r)$, output $(x, \mathsf{hc}(w))$.

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

Theorem 9.1. *The resulting PSPRFs are (adaptive) weak pseudorandom if the underlying TDRs are (adaptive) 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.

10 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 a formal definition, and then proceed to explore possible applications.

Definition 10.1 (Publicly Evaluable Constrained PRFs)**.** Publicly evaluable constrained PRFs consists of six polynomial-time algorithms as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters pp which include finite sets MPK, MSK, I, X, Y, a class of circuits $\mathcal{F} = \{f : I \to \{0,1\}\}$, a collection of languages $L = \{L_{ind}\}_{ind \in I}$ defined over X, a witness set W, as well as a PRF family $\mathsf{F} = \{\mathsf{F}_{msk} : I \times X \to Y\}_{msk \in MSK}$. Unlike the syntax of CPRFs [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 circuit $f \in \mathcal{F}$ satisfying $f(ind) = 1$.
- $\mathsf{KeyGen}(pp)$: on input pp, output master public key mpk and master secret key msk.
- $\mathsf{Constrain}(msk, f)$: on input msk and a circuit $f \in \mathcal{F}$, output a secret key sk_f.
- $\mathsf{SampRel}(ind, r)$: on input $ind \in I$ and fresh random coins r, output a random tuple $(x, w) \in \mathsf{R}_{L_{ind}}$.
- $\mathsf{PubEval}(ind, x, w)$: on input $ind \in I$ and $x \in L_{ind}$ together with a witness $w \in W$ for x, output $y \in Y$.
- $\mathsf{PrivEval}(sk_f, x)$: on input secret key sk_f and $x \in X$, output $y \in Y$.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any $ind \in I$, and any $x \in L_{ind}$, it holds that:

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

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

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

where $\mathcal{O}_{\mathsf{eval}}(ind, x) = \mathsf{F}_{msk}(ind, x)$, $\mathcal{O}_{\mathsf{constrain}}(f) = \mathsf{Constrain}(msk, f)$. The adversary $\mathcal{A}$ is restricted from querying $\mathcal{O}_{\mathrm{constrain}}(\cdot)$ with f such that $f(ind^*) = 1$, and the $\mathcal{A}_2$ is restricted from querying $\mathcal{O}_{\mathsf{eval}}(\cdot,\cdot)$ with (ind^*, x^*). PECPRFs are said to be adaptive weak pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}$ is negligible in λ.

We can define a weaker security notion by considering adversaries who only adaptively query oracle $\mathcal{O}_{\mathsf{constrain}}(\cdot)$ 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 sampleable instead of publicly evaluable.

10.1 Attribute-Based KEM from Publicly Evaluable Constrained PRFs

Sahai and Waters [SW05] introduced the notion of attribute-based encryption (ABE), which admits one to implement fine-grained yet flexible access control over sensitive data. As in the public-key setting and identity-based setting, there are numerous practical reasons to prefer an attribute-based key encapsulation mechanism (AB-KEM) over an ABE. Combining a data encapsulation mechanism (DEM) with appropriate security properties with an AB-KEM yields a fully functional ABE. For IBE this was formally proven in [BFMLS08]. The proof can easily be adapted to the ABE setting. We refer the readers to A.2 for the formal definition of AB-KEM.

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

- $\mathsf{Encaps}(mpk, ind)$: on input mpk and $ind \in I$, run $\text{PECPRF}.\mathsf{SampRel}(ind, r)$ with fresh random coins r to sample a random $x \in L_{ind}$ with a witness w for x, compute $y \leftarrow \text{PECPRF}.\mathsf{PubEval}(ind, x, w)$, set $c = x$, $k = y$ and output (c, k).
- $\mathsf{Decaps}(sk_f, c)$: on input sk_f and c, output $k \leftarrow \text{PECPRF}.\mathsf{PrivEval}(sk_f, c)$.

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

Theorem 10.1. *The AB-KEM is CPA-secure (resp. CCA-secure) if the underlying PECPRFs are semi-adaptive weak pseudorandom (adaptive weak pseudorandom).*

We omit the proofs here for their triviality.

Remark 10.1. Considering PECPRFs that supports a special class of circuits $\mathcal{F}$ (point circuits) where each f is indexed by I and f_{ind} is defined as $f_{ind}(ind')$ iff $ind = ind'$. It is easy to see that such PECPRFs immediately implies an identity-based key encapsulation mechanism (IB-KEM), while itself can be generically constructed from either an identity-based hash proof system (IB-HPS) [ADN+10, CZLC12a] or an identity-based extractable hash proof system (IB-EHPS) [CZLC12b].

10.2 Instantiations of PECPRFs

Next we provide an instantiation of PECPRFs for general circuits from multilinear maps based on the attribute-based encryption scheme [GGH+13c]. We use the same notation for circuits as in [GGH+13c], which is included in Appendix A.3 for completeness. We refer the readers to Appendix A.4 for the definition of multilinear maps and the related hardness assumption.

- $\mathsf{Setup}(1^\lambda, \ell)$: on input 1^λ and the maximum depth ℓ of a circuit, run $\mathsf{MLGroupGen}(1^\lambda, k = \ell + 1)$ 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 parameters 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.
- $\mathsf{Constrain}(msk, f)$: on input $msk = (g_{k-1})^\alpha$ and a circuit f, choose $s_1, \ldots, s_{n+q} \xleftarrow{R} \mathbb{Z}_p$ (where random coins 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
  $$sk_{w,1} = g_1^{a_w}, sk_{w,2} = g_1^{b_w}, sk_{w,3} = g_j^{s_w - a_w \cdot s_{\mathsf{A}(w)} - b_w \cdot s_{\mathsf{B}(w)}}$$

- $\mathsf{SampRel}(ind, r)$: 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 $y = (g_k^{\alpha})^r$.
- $\mathsf{PrivEval}(sk_f, x)$: on input a secret key sk_f for a circuit $f = (n, q, \mathsf{A}, \mathsf{B}, \mathsf{GateType})$ 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 $f_w(ind) = 1$ 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 $f_w(ind) = 1$, 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 $f_w(ind) = 1$. 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 $f_w(ind) = 1$. If $f_{\mathsf{A}(w)}(ind) = 1$ (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 $f_{\mathsf{A}(w)}(ind) = 0$, but $f_{\mathsf{B}(w)}(ind) = 1$, 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 $f_w(ind) = 1$. Then $f_{\mathsf{A}(w)}(ind) = f_{\mathsf{B}(w)}(ind) = 1$ and we compute:

$$\begin{aligned} y_w &= e(y_{\mathsf{A}(w)}, sk_{w,1}) \cdot e(y_{\mathsf{B}}(w), sk_{w,2}) \cdot e(sk_{w,3}, g_1^r) \\ &= e(g_j^{rs_{\mathsf{A}(w)}}, g_1^{a_w}) \cdot e(g_j^{rs_{\mathsf{B}(w)}}, g_1^{b_w}) \cdot e(g_j^{s_w - a_w \cdot s_{\mathsf{A}(w)} - b_w \cdot s_{\mathsf{B}}(w)}, g_1^r) = (g_{j+1})^{rs_w} \end{aligned}$$

Finally, output $y' \cdot y_{n+q}$. The correctness is easy to verify by observing that if $f(ind) = f_{n+q}(ind) = 1$, 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+13c]. Applying the generic construction presented in Section 10.1 to the above PECPRF, we recover exactly the ABE scheme [GGH+13c].

11 Publicly Evaluable and Verifiable Functions

In this section, we introduce a twist on 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 from weak pseudorandomness to unpredictability. In addition, in PEVFs $L = X$ while in PEPRFs $L \subseteq X$.

Definition 11.1 (Publicly Evaluable and Verifiable Functions)**.** A family of PEVFs consist of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters $pp = (\mathsf{F}, SK, PK, L, W, Y)$, where $\mathsf{F} : SK \times L \to Y$ 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.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in L$, output $y = \mathsf{F}_{sk}(x)$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$ for x, output $y = \mathsf{F}_{sk}(x)$.
- $\mathsf{PubVefy}(pk, x, y)$: on input pk, x and y, output true if $y = \mathsf{F}_{sk}(x)$ and false if not.

Security: Let $\mathcal{A}$ be an adversary against PEVFs and define its advantage as:

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

We say that PEVFs are unpredictable if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible in λ.

Remark 11.1. Analogous to the generalization on PEPRFs, we can also generalize PEVFs by considering public-key dependent languages languages $\{L_{pk}\}_{pk \in PK}$ rather than a fixed language L. Correspondingly, the sampling algorithm $\mathsf{SampRel}$ takes pk as an extra input to sample a random tuple $(x, w) \in \mathsf{R}_{L_{pk}}$. The instantiation based on the RSA assumption presented in Section 11.2 exemplifies the usefulness of such generalization.

11.1 Signature from PEVFs

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

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, run PEVF.$\mathsf{Setup}(1^\lambda)$ 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 a message m, compute signature $\sigma \leftarrow \mathsf{F}_{sk}(\mathsf{H}(m))$ via 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 the algorithm PEVF.$\mathsf{PubEval}$ is not used in the above construction, but will prove useful in security argument.

Theorem 11.1. *The above signature is strongly unforgeable under adaptive chosen-message attack in the random oracle model if the underlying PEVFs are hard to compute on average. Suppose* H *is a random oracle, for any adversary* $\mathcal{A}$ *breaking the signature with advantage* $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ *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* $\mathsf{Adv}_{\mathcal{A}}(\lambda)/Q_h$*.*

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 $\mathcal{A}$: 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 $r_i \xleftarrow{\text{R}} R$, runs PEVF.$\mathsf{SampRel}(r_i)$ 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 11.2, 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, because there is no corresponding efficiently computable trapdoor function/permutation.

Replacing random oracle: Hohenberger, Sahai, and Waters [HSW14] utilized $i\mathcal{O}$ to give a way to instantiate the random oracle with a concrete hash function in FDH applications. We extend their techniques to replace 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 described in Figure 5.

H

Constants: Punctured PRF key k.
Input: message m.

1. compute $r \leftarrow \mathsf{PPRF}(k, m)$, output $x \leftarrow \mathsf{SampLan}(r)$;

Figure 5: Hash H

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" as described in Figure 6, where m^* is the message that $\mathcal{A}$ commits to attack before seeing the vk, and $x^* = \mathsf{SampLan}(\mathsf{PPRF}(k, m^*))$.

H^*

Constants: Punctured PRF key $k(\{m^*\})$, m^*, x^*.
Input: message m.

1. if $m \neq m^*$, then compute $r \leftarrow \mathsf{PPRF}(k, m)$, output $x \leftarrow \mathsf{SampLan}(r)$;
2. if $m = m^*$, then output x^*.

Figure 6: Hash H^*

Since the input/output behaviors of the two programs are identical, Game 0 and Game 1 are computationally indistinguishable by the security of $i\mathcal{O}$. In Game 2, we replace $x^* = \mathsf{SampLan}(\mathsf{PPRF}(k, m^*))$ with a random element chosen from L in the program of H^*. Due to the selective pseudorandomness 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. After $\mathcal{B}$ receiving PEVF challenge (pk, x^*) and $\mathcal{A}$'s target message m^*, $\mathcal{B}$ creates $k(\{m^*\})$, then uses it together with the information of x^* and m^* to build the program H^*, then sends $vk = (pk, i\mathcal{O}(\mathsf{H}^*))$ to $\mathcal{A}$. During Game 2, $\mathcal{B}$ is able handle signing queries for any $m \neq m^*$ without knowing sk as follows: 1) compute $r \leftarrow \mathsf{PPRF}(k(\{m^*\}), m)$; 2) compute $(x, w) \leftarrow \mathsf{SampRel}(r)$; 2) output $\sigma \leftarrow \mathrm{PEVF.PubEval}(pk, x, w)$. 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 11.2. We remark that our security proofs are 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 random coins is added to the image. We believe RPEVFs are suitable for admitting more applications, such as probabilistic "hash-and-sign" signatures.

11.2 Instantiations of PEVFs

Here we construct PEVFs based on the RSA assumption (c.f. definition in A.5) and the CDH assumption in bilinear groups, respectively.

PEVFs based on the RSA assumption

- $\mathsf{Setup}(1^\lambda)$: run $\mathsf{GenModulus}(1^\lambda) \to (N, p, q)$, set $W = Y = \mathbb{Z}_N^*$, set $PK = SK = \mathbb{Z}_{\phi(N)}^*$, set $L = \{L_{pk}\}_{pk \in PK}$, where $L_{pk} = \{x : \exists w \in W \text{ s.t. } w^{pk} \equiv x \bmod N\}$. The corresponding sampling algorithm $\mathsf{SampRel}$ on input pk and random coins r, picks $w \xleftarrow{R} \mathbb{Z}_N^*$ and computes $x \leftarrow w^{pk} \bmod N$, then outputs $(x, w) \in \mathsf{R}_L$.
- $\mathsf{KeyGen}(pp)$: pick $sk \xleftarrow{R} SK$, compute $pk \leftarrow sk^{-1} \bmod \phi(N)$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, output $x^{sk} \bmod N$. This algorithm defines F_{sk}.
- $\mathsf{PubEval}(pk, x, w)$: on input pk, x and w, output w.
- $\mathsf{PubVefy}(pk, x, y)$: on input pk, x and y, output true if $x \equiv y^p k \bmod N$ and false if not.

PEVFs based on the CDH assumption in bilinear groups

- $\mathsf{Setup}(1^\lambda)$: run bilinear group generator $\mathsf{BLGroupGen}(1^\lambda)$ to generate $(e, g, p, \mathbb{G}, \mathbb{G}_T)$, set $L = Y = PK = \mathbb{G}$, set $SK = W = \mathbb{Z}_p$, set $L = \{x : \exists w \in W \text{ s.t. } g^w = x\}$. The corresponding sampling algorithm $\mathsf{SampRel}$ on input random coins r, picks $w \xleftarrow{R} \mathbb{Z}_p$ and computes $x \leftarrow g^w$, then outputs $(x, w) \in \mathsf{R}_L$.
- $\mathsf{KeyGen}(pp)$: pick $sk \xleftarrow{R} \mathbb{Z}_p$, compute $pk = g^{sk}$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and x, output $y = x^{sk}$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk, x and w, output $y = pk^w$.
- $\mathsf{PubVefy}(pk, x, y)$: on input pk, x and y, output true if $e(pk, x) = e(g, y)$ and false if not.

Acknowledgments

We are grateful to Yi Deng, Qiong Huang, and Dennis Hofheinz for insightful discussions and advice. We thank the reviewers of Journal of Computer Security for many helpful comments.

Yu Chen is supported by the National Natural Science Foundation of China (Grant No. 61303257, No. 61379141), the IIE's Cryptography Research Project (Grant No. Y4Z0061B02), the Strategic Priority Research Program of CAS (Grant No. XDA06010701), and the State Key Laboratory of Cryptology's Open Project (Grant No. MMKFKT201511). Zongyang Zhang is an International Research Fellow of JSPS and supported by the National Natural Science Foundation of China (Grant No. 61303201).

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.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658, 2014.

[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.

[BFMLS08] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless kems. *Journal of Cryptology*, 21(2):178–199, 2008.

[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.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. 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.

[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.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *LNCS*, pages 480–499. Springer, 2014.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *LNCS*, pages 247–266. Springer, 2015.

[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.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

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

[CLR15] Jung Hee Cheon, Changmin Lee, and Hansol Ryu. Cryptanalysis of the new CLT multi-

linear maps. *IACR Cryptology ePrint Archive*, 2015:934, 2015.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013*, volume 8042 of *LNCS*, pages 476–493. Springer, 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *LNCS*, pages 267–286. Springer, 2015.

[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.

[DH76] Whitefield Diffie and Martin E. Hellman. New directions in cryptograpgy. *IEEE Transactions on Infomation Theory*, 22(6):644–654, 1976.

[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.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[GGH+13c] 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.

[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. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*, volume 8772 of *LNCS*, pages 197–219. Springer, 2014.

[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.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *LNCS*, pages 500–517. Springer, 2014.

[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.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity based encryption. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[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.

[Zha16] Mark Zhandry. How to avoid obfuscation using witness prfs. In *Theory of Cryptography - 13th International Conference, TCC 2016-A*, volume 9563 of *LNCS*, pages 421–448. Springer, 2016.

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:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters pp. We assume that pp 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 Encaps and Decaps.
- $\mathsf{KeyGen}(pp)$: on input pp, output a public/secret key pair (pk, sk).
- $\mathsf{Encaps}(pk)$: on input public key pk, output a ciphertext c and a DEM key $k \in K$.
- $\mathsf{Decaps}(sk, c)$: 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: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$ and any $(c, k) \leftarrow \mathsf{Encaps}(pk)$, we have $\Pr[\mathsf{Decaps}(sk, c) = k] = 1$.

Security: Let $\mathcal{A}$ be an adversary against KEM and define its advantage as:

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

where $\mathcal{O}_{\mathsf{dec}}(c) = \mathsf{Decaps}(sk, c)$, and $\mathcal{A}$ 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 λ. 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 Attribute-Based Key Encapsulation Mechanism

We now give a formal definition of AB-KEM for circuits, which essentially follows the definition of ABE for circuits [GGH+13c]. An AB-KEM consists of five polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, output public parameters pp. We assume that pp include descriptions of index set I (an index usually represents an assignment of attributes), a class of circuits $\mathcal{F}$, ciphertext space C, and DEM (data encapsulation mechanism) key space K. pp will be used as an implicit input for algorithms KeyGen, Encaps, and Decaps.
- $\mathsf{KeyGen}(pp)$: on input pp, output a master public/secret key pair (mpk, msk).
- $\mathsf{Extract}(msk, f)$: on input msk and $f \in \mathcal{F}$, output a secret key sk_f for f.
- $\mathsf{Encaps}(ind; r)$: on input $ind \in I$, output a ciphertext c and a DEM key $k \in K$.
- $\mathsf{Decaps}(sk_f, c)$: on input a secret key sk_f and ciphertext $c \in C$, output a DEM key $k \in K$ or a reject symbol $\perp$ indicating c is invalid.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $(mpk, msk) \leftarrow \mathsf{KeyGen}(pp)$, any $ind \in I$, any $(c, k) \leftarrow \mathsf{Encaps}(ind)$ and any $sk_f \leftarrow \mathsf{Extract}(msk, f)$ satisfying $f(ind) = 1$, we have $\mathsf{Decaps}(sk_f, c) = k$.

Security: The security of AB-KEM intuitively guarantees that a ciphertext encapsulated under index ind hides all information about the underlying DEM key k unless one is in possession of a secret key giving the explicit ability to decapsulate, i.e., if an adversary $\mathcal{A}$ holds keys $sk_{f_1}, \ldots, sk_{f_l}$, then $\mathcal{A}$ learns nothing about the DEM key if $f_1(ind) = \cdots = f_l(ind) = 0$. Formally, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against AB-KEM and define its advantage in the following experiment:

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

where $\mathcal{O}_{\mathsf{ext}}(f) = \mathsf{Extract}(msk, f)$, and throughout the experiment $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\mathsf{ext}}(\cdot)$ with the circuit f such that $f(ind^*) = 1$; $\mathcal{O}_{\mathsf{dec}}(ind, c) = \mathsf{Decaps}(sk_f, c)$ where $f(ind) = 1$, and in the second phase $\mathcal{A}_2$ is not allowed to query $\mathcal{O}_{\mathsf{dec}}(\cdot,\cdot)$ with the challenge (ind^*, c^*). An AB-KEM is said to be IND-CCA secure if for any PPT adversary $\mathcal{A}$, its advantage defined as above is negligible in λ. The IND-CPA security for AB-KEM can be defined using the same experiment except that the adversary is not allowed to access $\mathcal{O}_{\mathsf{dec}}(\cdot,\cdot)$.

A.3 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.[11]

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.4 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 let $\mathsf{MLGroupGen}$ denote the multilinear maps parameter generator which on input 1^λ 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 $\mathsf{MLGroupGen}(1^\lambda, k)$. For any PPT adversary $\mathcal{A}$ it holds that:

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

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 $\mathsf{MLGroupGen}(\lambda, k)$ and picking $g \xleftarrow{R} \mathbb{G}_1$, $c_i \xleftarrow{R} \mathbb{Z}_p$, $b \xleftarrow{R} \{0,1\}$.

Remark A.1. We currently do not have candidates for multilinear maps between groups with cryptographically hard problems. However, Garg, Gentry, and Halevi [GGH13a] (henceforth GGH) showed the next best thing. They proposed a concrete candidate for "approximate multilinear maps", named *graded encoding systems*. After this breakthrough, Coron, Lepoint, and Tibouchi [CLT13, CLT15] (henceforth CLT) proposed another two candidates that works over integers. Unfortunately, all these known candidates are subject to a number of attacks [CGH+15, CHL+15, CLR15].

[11] 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.5 RSA Assumption

Assumption A.2 (RSA Assumption)**.** Let $\mathsf{GenModulus}(1^\lambda)$ be an algorithm which on input 1^λ, outputs (N, p, q) where n is the product of two λ-bit, distinct odd primes p, q. The RSA assumption states that for any PPT adversary $\mathcal{A}$ it holds that:

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

where the probability is defined over the random coins of $\mathsf{GenModulus}(1^\lambda)$ and the random choices of $x \xleftarrow{R} \mathbb{Z}_N^*$ and $e \xleftarrow{R} \mathbb{Z}_{\phi(N)}^*$.

A.6 Quadratic Residuosity (QR) Assumption

Assumption A.3 (QR Assumption)**.** Let $\mathsf{Jacobi}_p(x)$ denote the Jacobi symbol of x modulo p. Let N be a composite number generated by $\mathsf{GenModulus}(1^\lambda)$, QR_N be the set of quadratic residues modulo N, and QNR_N^{+1} be the set of quadratic non-residues modulo N having Jacobi symbol $+1$. Let SampQR and $\mathsf{SampQNR}$ be two algorithms which on input N, output a random element from QR_N and QNR_N^{+1} respectively. The QR assumption states that for any PPT adversary $\mathcal{A}$ it holds that:

$$|\Pr[\mathcal{A}(N, qr) = 1] - \Pr[\mathcal{A}(N, qnr) = 1]| \leq \mathsf{negl}(\lambda)$$

where the probability is defined over the random coins of $\mathsf{GenModulus}(1^\lambda)$ and $qr \leftarrow \mathsf{SampQR}(N)$ and $qnr \leftarrow \mathsf{SampQNR}(N)$.

A.7 Decisional Diffie-Hellman (DDH) Assumption

Assumption A.4 (DDH Assumption)**.** Let $\mathsf{GroupGen}(1^\lambda)$ be a group generator which on input 1^λ, outputs $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a group with λ-bit prime order p, g is a random generator of $\mathbb{G}$. The DDH assumption states that for any PPT adversary $\mathcal{A}$ it holds that:

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1] \leq \mathsf{negl}(\lambda)$$

where the probability is defined over the random coins of $\mathsf{GroupGen}(1^\lambda)$ and the random choices of $a, b, c \xleftarrow{R} \mathbb{Z}_p$.