Constrained PRFs for Unbounded Inputs

Hamza Abusalah[*] Georg Fuchsbauer[*] Krzysztof Pietrzak[*]

IST Austria

{habusalah, gfuchsbauer, pietrzak}@ist.ac.at

Abstract

A constrained pseudorandom function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ for family of subsets of $\mathcal{X}$ is a function where for any key $k \in \mathcal{K}$ and set S from the family one can efficiently compute a short constrained key k_S which allows to evaluate $F(k, \cdot)$ on all inputs $x \in S$, while given this key, the outputs on all inputs $x \notin S$ look random.

Constrained PRFs have been constructed for several families of sets, the most general being the circuit-constrained PRF by Boneh and Waters [Asiacrypt'13]. Their construction allows for constrained keys k_C, where C is a boolean circuit that defines the set $S = \{x \in \mathcal{X} \mid C(x) = 1\}$. In their construction the input length and the size of the circuits C for which constrained keys can be computed must be fixed a priori during key generation.

In this paper we construct a constrained PRF that has an unbounded input length and constrained keys can be defined for any set that can be decided by a polynomial-time Turing machine. The only a priori bound we make is on the size of the Turing machines. We discuss applications of our CPRF, such as broadcast encryption where the number of potential receivers need not be fixed at setup (in particular, the length of the keys is independent of the number of parties), and identity-based non-interactive key-agreement between sets of users where again there is no bound on the number of parties that can agree on a shared key.

Our CPRF is simply defined as $F(k, H(x))$ where F is a puncturable PRF (e.g. the GGM construction) and H is a collision-resistant hash function. A constrained key for a Turing machine M is a signature on M. At setup we also publish an obfuscated circuit, which on input M, a signature σ, a value h and a short non-interactive argument of knowledge π outputs $F(k, h)$ if (1) σ is a valid signature on M and (2) π proves knowledge of some x s.t. $H(x) = h$ and $M(x) = 1$. For our security proof, we assume extractability obfuscation for the particular circuit just explained.

Keywords: Constrained PRFs, broadcast encryption, identity-based NIKE.

[*]Supported by the European Research Council, ERC Starting Grant (259668-PSPC)

1 Introduction

Constrained PRFs. A pseudorandom function (PRF) [GGM86] is a keyed function $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ for which no efficient adversary, given access to an oracle $\mathcal{O}(\cdot)$, can distinguish the case where $\mathcal{O}(\cdot)$ is $\mathsf{F}(k, \cdot)$ with a random key $k \in \mathcal{K}$ from the case where $\mathcal{O}(\cdot)$ is a uniformly random function $\mathcal{X} \to \mathcal{Y}$.

Three papers [BW13,BGI14,KPTZ13] independently introduced the concept of a *constrained* PRF. Let $\mathcal{S}$ be a set of subsets of $\mathcal{X}$, and for $S \in \mathcal{S}$ let p_S denote the predicate where $p_S(x) = 1$ iff $x \in S$. A constrained PRF for a predicate family $\mathcal{P}$ is a PRF F with an additional constrain algorithm $k_S \leftarrow \mathsf{F.Constr}(k, S)$ that on input a key $k \in \mathcal{K}$ and a set $S \in \mathcal{S}$ outputs a (short) constrained key k_S that can be used to compute $\mathsf{F}(k, x)$ for all $x \in S$, while, given this key, all values $\mathsf{F}(k, x)$ for $x \notin S$ still look pseudorandom.

Constrained PRFs have been constructed for several interesting predicates. All three papers show that the classical GGM construction [GGM86] of a PRF with input domain $\{0,1\}^n$ gives a *prefix-fixing* constrained PRF. This means that for any $v \in \{0,1\}^{\leq n}$, one can derive a key k_v which allows to compute $\mathsf{F}(k, x)$ for all $x \in \{0,1\}^n$ of the form $x = v \| x'$. Assuming (leveled) multilinear maps [GGH13a, CLT13, LSS14], Boneh and Waters [BW13] construct constrained PRFs for much more general set systems. They present a bit-fixing PRF, for which the sets are defined by vectors $v \in \{0, 1, ?\}^n$, and $p_v(x) = 1$ if x agrees with v on all indices different from '?', i.e., for all $i = 1, \ldots, n$, either $v[i] = ?$ or $v[i] = x[i]$. They moreover construct a circuit-constrained PRF, where the predicate is an arbitrary circuit $p\colon \{0,1\}^n \to \{0,1\}$ of some fixed depth.

Constrained PRFs (CPRF) have already found many interesting applications. From a prefix-fixing CPRF, one can construct a so-called puncturable PRF, which is a constrained PRF for the sets $\mathcal{S} = \{S_x = \{0,1\}^n \setminus \{x\} \mid x \in \{0,1\}^n\}$; a key k_{S_x} allows thus evaluation of $\mathsf{F}(k, \cdot)$ on all inputs except x. Puncturable PRFs play a crucial role in the security proof of most of the recent constructions based on indistinguishability obfuscation, and we will also use them in this paper.

The more general bit-fixing and circuit-constrained PRFs can be used directly to construct a variety of sophisticated cryptographic tools, of which we will just mention two, broadcast encryption (BE) and identity-based non-interactive key-exchange (ID-NIKE).

Broadcast encryption. In a broadcast encryption scheme [FN93,YFDL04,BGW05,PPS11,BH08, BWZ14] there is a set of n users, and we want to be able encrypt a message for any given subset $S \subseteq \{1, \ldots, n\}$ of them. This can be achieved using a bit-fixing PRF with domain $\{0,1\}^n$: Sample a random key k, and give a constrained key k_{v_i} to user i where $v_i[i] = 1$ and $v_i[j] = ?$ for any $j \neq i$. To broadcast a message m to a set S of users, we simply send a symmetric encryption of m under the key $\mathsf{F}(k, v_S)$, where $v_S[i] = 1$ if $i \in S$ and $v[i] = 0$ otherwise. Note that user i can compute $\mathsf{F}(k, v_S)$ (and thus decrypt) iff her key k_{v_i} satisfies $v_i[i] = v_S[i]$, which by construction holds iff $i \in S$.

Non-interactive key exchange. In an identity-based non-interactive key-exchange (ID-NIKE) [SOK00, FHPS13, BW13] scheme we have a potentially huge number of parties, each having some identity $id \in \{0,1\}^\ell$. We want that any set S of at most n parties can locally compute a key K_S, whereas for every party outside of S this key is indistinguishable from random. For this, let F be a bit-fixing PRF with domain $\{0,1\}^{n \cdot \ell}$. We sample a key k for F and give to party $id \in \{0,1\}^\ell$ a set of n constrained keys $k_{id}^{(1)}, \ldots, k_{id}^{(n)}$, where $k_{id}^{(i)}$ is a key for the set $?^{(i-1)\ell} \| id \| ?^{(n-i)\ell}$. Now, parties $id_1, \ldots, id_n$ (which we assume are in lexicographic order) can compute a joint key $\mathsf{F}(k, id_1 \| id_2 \| \ldots \| id_n)$.

CPRFs with unbounded input length. The disadvantage of the BE and ID-NIKE constructions just outlined is that the number n of possible recipients (for BE) or parties agreeing on a key (for

ID-NIKE) must be fixed when setting up the system. Moreover, the length of the constrained keys given to every user is at least linear in n.

In this paper we construct a constrained PRF where the input length can be unbounded. The constraints on keys are given by Turing machines (TM), that is, given a key k and a TM M, we can derive a constrained key k_M that allows to compute $\mathsf{F}(k, x)$ for any input x where $M(x) = 1$. The only thing that must be a priori bounded is the *size* of TMs for constrained keys we want to tolerate. The constrained key for M will simply be a signature on M together with an obfuscated circuit, which is however universal in the sense that it is identical for all constrained keys and need not be kept secret.

In Section 4 we show how such a CPRF yields BE and ID-NIKE for an unbounded number of parties.

Adaptive vs. selective security. We prove *selective* security of our constrained PRF, that is, we assume the adversary commits to the input x^* for which it wants to distinguish $\mathsf{F}(k, x^*)$ from random at the beginning of the security game (that is, before it can query constrained keys for sets $S \not\ni x^*$). From a selectively secure CPRF we can get an *adaptively* secure CPRF (where the adversary can decide on x^* after its key queries) via "complexity leveraging"—but this reduction loses a factor that is exponential in the input length. Proving adaptive security for CPRFs without an exponential security loss is generally hard and Fuchsbauer et al. [FKPR14] show that for the bit-fixing CPRF from [BW13] any "simple" security reduction must lose an exponential factor.

Adaptive security of CPRFs was shown for the simple GGM-based prefix-constrained PRF [BW13, BGI14, KPTZ13] in [FKPR14], whose proof only loses a quasi-polynomial (rather than an exponential) factor. Moreover, Hohenberger et al. [HKW14] construct an adaptively secure puncturable PRF with only polynomial security loss, but they use heavier tools including indistinguishability obfuscation ($i\mathcal{O}$) [GGH+13b, SW14, PST14]. Hofheinz [Hof14] constructs an adaptively secure bit-fixing PRF, also using $i\mathcal{O}$, and additionally relying on the random-oracle model. We leave the construction of adaptively secure constrained unbounded-length PRFs (for any interesting set of constraints) as a challenging open problem.

Our construction. An obvious approach to construct a constrained PRF is to start with any standard PRF F. Given a key k and a set S, we can now define a constrained key as a program P_S which on input x checks if $x \in S$, and if so, outputs $\mathsf{F}(k, x)$. Of course we cannot just use a normal program P_S, as an adversary could extract the key k from P_S, and thus $\mathsf{F}(k, \cdot)$ does not look random on $x \notin S$ given P_S.

A CIRCUIT-CONSTRAINED PRF. To avoid the above issue, we must *obfuscate* P_S before outputting it. The strongest notion of obfuscation is *virtual black-box obfuscation*, which requires that an obfuscated program leaks nothing about the program apart from its input/output behavior. Unfortunately, such a strong notion does not exist for general functionalities [BGI+12]. We therefore use *indistinguishability obfuscation* ($i\mathcal{O}$), which only guarantees that obfuscations of two circuits (of the same size) which output the same on any input are indistinguishable. A candidate $i\mathcal{O}$ scheme has recently been proposed by Garg et al. [GGH+13b]. Although the notion seems very weak, it has proven to be surprisingly useful.

A useful trick in the $i\mathcal{O}$ literature is the use of a puncturable PRF [SW14], for which, given a key k and some input x^*, one can compute a punctured key k_{x^*} that lets one evaluate $\mathsf{F}(k, x)$ on all $x \neq x^*$. Given k_{x^*}, the value $\mathsf{F}(k, x^*)$ is indistinguishable from random. The GGM construction [GGM86] of a PRF from a length-doubling pseudorandom generator is a puncturable PRF (where the length of punctured keys is linear in the input length of the PRF).

Consider a circuit-constrained PRF derived from a PRF F where a constrained key k_S is computed as an $i\mathcal{O}$ obfuscation of the circuit P_S (which outputs $\mathsf{F}(k,x)$ if $x \in S$ and $\perp$ otherwise). If F is a puncturable PRF then we can reduce selectively security of this CPRF to selective security of F as follows. In the selective-security game for CPRFs, an adversary $\mathcal{A}$ chooses some input x^*, can then ask for constrained keys for any sets S with $x^* \notin S$ and must distinguish $\mathsf{F}(k,x^*)$ from random. We first define a modified game where $\mathcal{A}$, when asking for a constrained key for a set S, gets an $i\mathcal{O}$ obfuscation of a circuit P'_S that outputs $\mathsf{F}(k_{x^*},x)$ if $x \in S$ and $\perp$ otherwise. (The difference of P_S and P'_S is that in the latter F is evaluated using a key k_{x^*} that is punctured at x^*.)

Recall that the adversary can only submit sets S with $x^* \notin S$ to its oracle. We thus have $P_S(x^*) = P'_S(x^*) = \perp$. Moreover, on any other input x, P_S and P'_S also return the same output (namely $\mathsf{F}(k,x)$ if $x \in S$ and $\perp$ otherwise.) By security of $i\mathcal{O}$, obfuscations of P_S and P'_S are thus indistinguishable, which means that the modified game is indistinguishable from the original game. From an $\mathcal{A}$ winning the modified game, we easily obtain an adversary $\mathcal{B}$ that breaks the puncturable PRF F: When $\mathcal{A}$ commits to x^*, $\mathcal{B}$ does the same and asks for a punctured key k_{x^*}. This key allows $\mathcal{B}$ to answer $\mathcal{A}$'s constrained-key queries in the modified game. If $\mathcal{A}$ distinguishes $\mathsf{F}(k,x^*)$ from random then so does $\mathcal{B}$.

The drawback with this construction is that $i\mathcal{O}$ has only been constructed for circuits, which means that the above construction only works for an a priori bounded input length.

A Turing-machine-constrained PRF. To overcome this problem and allow for unbounded input lengths, as a first step we use a collision-resistant hash function H to map long inputs to inputs of fixed length. Concretely, we define our CPRF F as $\mathsf{F}(k,x) := \mathsf{PF}(k,H(x))$, where PF is a puncturable PRF, like the GGM PRF.

Now how do we define a constrained key for S? Defining a circuit that takes x, checks whether $x \in S$ and if so outputs $\mathsf{PF}(k,H(x))$ is not possible, since there is no bound on x, so it cannot be decided by a circuit. We "outsource" the verification of whether $x \in S$ and use a succinct non-interactive argument of knowledge (SNARK). A SNARK system allows to give a non-interactive argument (that is, a proof which is only computationally sound) of an NP statement, whose size is independent of the size of the witness used to compute it. In particular, we use a SNARK system for the NP language $L := \{(H,S,h) \mid \exists\, x : x \in S \wedge H(x) = h\}$. We then define a circuit P_S that takes input (h,π) and outputs $\mathsf{PF}(k,h)$ if π is a valid SNARK for (H,S,h). This approach solves the problem of checking the legitimacy of an input (that is, $x \in S$) within a circuit. Moreover, our sets S can now be described by Turing machines instead of circuits.

Again, a constrained key k_S is an obfuscation of the program P_S. In order to give a reduction of security to the puncturable PRF PF, we would, as before, replace the obfuscation of P_S by one of P'_S, which uses $k_{H(x^*)}$ instead of k. Unfortunately, indistinguishability of this replacement is not guaranteed by indistinguishability obfuscation, as P_S and P'_S are not functionally equivalent, which can be seen as follows. There exist values x with $x \neq x^*$ and $H(x) = H(x^*)$ and the adversary is allowed to query a constrained key for a set S containing such an x (provided it does not contain x^*). It could then compute a SNARK π for $(H,S,H(x)) \in L$ and run its constrained key on $(H(x),\pi)$. Whereas P_S would output $\mathsf{PF}(k,H(x)) = \mathsf{PF}(k,H(x^*))$, the modified circuit P'_S would output $\perp$, since its key is punctured at $H(x^*)$.

Intuitively an adversary can only distinguish P_S from P'_S if it finds such an x, which together with x^* constitutes a collision for H, and should therefore be hard to find. To make this formal in our security proof, we must construct an adversary that finds such a collision. Instead of $i\mathcal{O}$, we therefore resort to a stronger form of obfuscation, called differing-input obfuscation or extractability obfuscation ($e\mathcal{O}$) [BGI+12,BCP14]. Whereas $i\mathcal{O}$ provides indistinguishability of obfuscations of equivalent circuits, $e\mathcal{O}$ guarantees that from an adversary that distinguishes obfuscations of two circuits, one can extract an input on which they differ. From an adversary distinguishing P_S and P'_S we can then extract a collision for H.

Short constrained keys. In the construction just sketched the master key is simply a key for the puncturable PRF PF, and evaluating x only consists of hashing x and evaluating PF on the hash. The expensive operations are issuing constrained keys (which involves obfuscating a circuit) and evaluating the PRF with a constrained key (which runs an obfuscated circuit). Moreover, being obfuscated circuits, the constrained keys are long. We modify the construction and reduce the complexity of the constraining algorithm and the size of keys drastically (whereas evaluation will still be expensive).

When setting up the PRF, we construct one single circuit P which we obfuscate and publish as public parameters. A constrained key for a set S decided by a Turing machine M is then simply a signature σ on M (that verifies w.r.t. a verification key contained in the parameters). Given a constrained key (M, σ), the PRF is evaluated on input x as follows:

- define $h := H(x)$ and compute a SNARK π for the statement (H, M, h) showing that for some x: $M(x) = 1$ and $H(x) = h$;
- run the (obfuscated) circuit P from the public parameters on input (M, h, π, σ),

where $P(M, h, \pi, \sigma)$ does the following: if σ is valid on M and π is valid on (H, M, h), it outputs $\mathsf{PF}(k, h)$, otherwise it outputs $\perp$. Let us mention that instead of normal signatures, we must use a functional signatures in order to prove security of the above construction (this is similar to the construction of functional encryption from $e\mathcal{O}$ in [BCP14]).

Assuming a puncturable PRF PF, a collision-resistant hash function H, a SNARK system for the language $L_{legit} := \{(H, M, h) \mid \exists\, x : M(x) = 1 \wedge H(x) = h\}$, extractability obfuscation for circuits, and functional signatures, we prove that the above construction is a Turing-machine-constrained PRF for inputs of unbounded length.

2 Preliminaries

2.1 Notations and Conventions

Let $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}} = \{\mathsf{F}\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of keyed functions with key space $\mathcal{K}_\lambda$, domain $\mathcal{X}_\lambda$ and range $\mathcal{Y}_\lambda$. (We will often drop the security parameter when it is clear from the context.) A family of circuits $\mathcal{C}_\lambda$ is of polynomial size if for every $C \in \mathcal{C}_\lambda$ the description size of C is polynomial in λ, i.e., $|C| = \mathsf{poly}(\lambda)$. The same holds for Turing Machine (TM) families.

Let $\mathcal{X}$ be a finite set, then $x \leftarrow \mathcal{X}$ denotes the process of sampling x uniformly at random from $\mathcal{X}$. Let $\mathcal{A}$ be a probabilistic polynomial-time (PPT) algorithm, then $\Pr[y \leftarrow \mathcal{A}(x)]$ denotes the probability that $\mathcal{A}(x)$ outputs y when run on uniformly sampled coins and $\Pr\big[x_1 \leftarrow \mathcal{X}_1; x_2 \leftarrow \mathcal{X}_2; \ldots : \varphi(x_1, x_2, \ldots) = 1\big]$ denotes the probability that the predicate φ evaluated on $(x_1, x_2, \ldots)$ is true after the ordered execution of $x_1 \leftarrow \mathcal{X}_1$, $x_2 \leftarrow \mathcal{X}_2$, etc.

A function $\nu : \mathbb{N} \to \mathbb{R}$ is called *negligible*, if for every positive polynomial $p(\cdot)$, and all sufficiently large $n \in \mathbb{N}$, it holds that $\nu(n) \leq \frac{1}{p(n)}$. We use $\mathsf{negl}(\cdot)$ to denote that there exists a negligible function.

2.2 Constrained and Puncturable PRFs

Definition 1 (Constrained Functions). *A family of keyed functions $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ over a key space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$, is* efficiently computable *if there exist a PPT sampler* $\mathsf{F.Smp}$, *and a deterministic polynomial-time (PT) evaluator,* $\mathsf{F.Eval}$ *such that*

- $k \leftarrow \mathsf{F.Smp}(1^\lambda)$*: On input a security parameter λ,* $\mathsf{F.Smp}$ *outputs a key $k \in \mathcal{K}$.*

```
$\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}(\lambda)$ :
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$; $C, E := \emptyset$
$(x^*, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(1^\lambda)$
If $x^* \in E$, then abort
If $b = 1$, $y := \mathsf{F.Eval}(k, x^*)$, else $y \leftarrow \mathcal{Y}$
$C := C \cup \{x^*\}$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(st, y)$; Return $b'$
```

```
Oracle constr($S$) :
If $S \notin \mathcal{S}_\lambda \vee S \cap C \neq \emptyset$
    Return $\perp$
$E := E \cup S$
$k_S \leftarrow \mathsf{F.Constr}(k, S)$
Return $k_S$
```

```
Oracle eval($x$) :
If $x \notin \mathcal{X} \vee x \in C$
    Return $\perp$
$E := E \cup \{x\}$
$y = \mathsf{F.Eval}(k, x)$
Return $y$
```

Figure 1: $\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}(\lambda)$: The security game for constrained PRFs.

- $\mathsf{F.Eval}(k, x) = \mathsf{F}(k, x)$: *On input a key* $k \in \mathcal{K}$, *and* $x \in \mathcal{X}$, $\mathsf{F.Eval}$ *outputs* $\mathsf{F}(k, x)$.

We say $\mathcal{F}_\lambda$ *is* constrained *w.r.t. a family* $\mathcal{S}_\lambda$ *of subsets of* $\mathcal{X}$, *with constrained key space* $\mathcal{K}_\mathcal{S}$ *such that* $\mathcal{K}_\mathcal{S} \cap \mathcal{K} = \emptyset$, *if* $\mathsf{F.Eval}$ *accepts inputs from* $(\mathcal{K} \cup \mathcal{K}_\mathcal{S}) \times \mathcal{X}$ *and there exists the following PPT algorithm:*

- $k_S \leftarrow \mathsf{F.Constr}(k, S)$: *On input a key* $k \in \mathcal{K}$, *and a description*[1] *of a set* $S \in \mathcal{S}_\lambda$, $\mathsf{F.Constr}$ *outputs a constrained key* $k_S \in \mathcal{K}_\mathcal{S}$ *such that*

$$\mathsf{F.Eval}(k_S, x) = \begin{cases} \mathsf{F}(k, x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

Definition 2 (Security of constrained PRFs)**.** *A family of (efficiently computable) constrained functions* $\mathcal{F}_\lambda = \{\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ *is selectively* pseudorandom, *if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in* $\mathbf{Exp}^{\mathcal{O},0}_{\mathcal{F},\mathcal{A}}$ *(Figure 1) with* $\mathcal{O}_1 = \emptyset$ *and* $\mathcal{O}_2 = \{\mathit{constr}(\cdot), \mathit{eval}(\cdot)\}$, *it holds that*

$$\mathsf{Adv}^{\mathcal{O}}_{\mathcal{F},\mathcal{A}}(\lambda) := \left| \Pr[\mathbf{Exp}^{\mathcal{O},0}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathcal{O},1}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] \right| \leq \mathsf{negl}(\lambda) \ .$$

The family $\mathcal{F}_\lambda$ *is* adaptively pseudorandom *if the same holds for* $\mathcal{O}_1 = \mathcal{O}_2 = \{\mathit{constr}(\cdot), \mathit{eval}(\cdot)\}$.

Puncturable PRFs [SW14] are a simple type of constrained PRFs, whose domain is $\{0,1\}^n$ for some n, and constrained keys can only be derived for "punctured" sets S, that is, $S = \{0,1\}^n \setminus T$ for some T of polynomial size. Moreover, we only require pseudorandomness to hold against selective adversaries that only make one key query. See Appendix B.1 for a formal definition.

In this work we only require selectively secure PRFs for which keys for sets of the form $\{0,1\}^n \setminus \{x\}$ for some $x \in \{0,1\}^n$ can be derived. Puncturable PRFs are easily obtained from (selectively secure) prefix-constrained PRFs, which were constructed from the GGM pseudorandom function [GGM86] for input space $\{0,1\}^n$ in [BW13, BGI14, KPTZ13].

2.3 Collision-Resistant Hash Functions

A family of hash functions is collision-resistant if given a uniformly sampled hash function, it is hard to find two inputs on which the function collides, i.e., returns the same hash value.

Definition 3 (Collision-Resistant Hash Functions)**.** *A family of (efficiently computable) functions* $\mathcal{H}_\lambda = \{H \colon \{0,1\}^\ell \to \{0,1\}^n\}$, *for which* $\mathsf{H.Smp}$ *samples a random function, is a family of hash functions if* $\ell(\cdot) > n(\cdot)$, *i.e.,* H *is compressing. The family is collision-resistant if for every PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[H \leftarrow \mathsf{H.Smp}(1^\lambda); (x_1, x_2) \leftarrow \mathcal{A}(1^\lambda, H) : \ x_1 \neq x_2 \ \wedge \ H(x_1) = H(x_2)\right] \leq \mathsf{negl}(\lambda) \ .$$

[1]We assume that all sets in $\mathcal{S}$ have short descriptions.

2.4 Indistinguishability and Extractability Obfuscation

As a consequence of their impossibility results for virtual black-box obfuscation, Barak et al. [BGI$^+$12], proposed two relaxations: indistinguishability obfuscation ($i\mathcal{O}$), and *differing-input obfuscation*, a.k.a. extractability obfuscation ($e\mathcal{O}$). Both $i\mathcal{O}$ and $e\mathcal{O}$ provide means to obfuscate families of circuits. Security of $i\mathcal{O}$ guarantees that obfuscations of equivalent circuits are computationally indistinguishable. Extractability obfuscation $e\mathcal{O}$ strengthens this security guarantee by requiring that for any efficient adversary that distinguishes obfuscations of two circuits, there exists an efficient *extractor* that extracts a point on which the circuits differ.

Definition 4 (Indistinguishability Obfuscation [GGH$^+$13b]). *A uniform PPT algorithm $i\mathcal{O}$ is an indistinguishability obfuscator for a family of polynomial-size circuits $\mathcal{C}_\lambda$, if the following hold:*

- *For all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and x:* $\Pr\left[\widetilde{C} \leftarrow i\mathcal{O}(1^\lambda, C) :\ C(x) = \widetilde{C}(x)\right] = 1$.
- *For every PPT adversary $\mathcal{A}$ and all $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$ for all x:*

$$\left| \Pr\left[\mathcal{A}\big(i\mathcal{O}(1^\lambda, C_0)\big) = 1\right] - \Pr\left[\mathcal{A}\big(i\mathcal{O}(1^\lambda, C_1)\big) = 1\right] \right| \leq \mathsf{negl}(\lambda) \ .$$

Definition 5 (Extractability Obfuscation [BCP14]). *A uniform PPT algorithm $e\mathcal{O}$ is an extractability obfuscator for a family of polynomial-size circuits $\mathcal{C}_\lambda$ and a polynomial-time sampler* $\mathsf{Sampler}$*, if the following hold:*

- *For all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and x:* $\Pr\left[\widetilde{C} \leftarrow e\mathcal{O}(1^\lambda, C) :\ C(x) = \widetilde{C}(x)\right] \geq 1 - \mathsf{negl}(\lambda)$.
- *For every PPT adversary $\mathcal{A}$ and every polynomial $q(\cdot)$, there exists a PPT extractor $\mathcal{E}_\mathcal{A}$ and a polynomial $p(\cdot)$, such that for every $\lambda \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l} (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda); \\ b \leftarrow \{0,1\};\ \widetilde{C}_b \leftarrow e\mathcal{O}(1^\lambda, C_b) \end{array} \ :\ \mathcal{A}(1^\lambda, C_0, C_1, \widetilde{C}_b, \mathsf{aux}) = b\right] \geq \frac{1}{2} + \frac{1}{q(\lambda)}$$
$$\Rightarrow\ \Pr\left[x \leftarrow \mathcal{E}_\mathcal{A}(1^\lambda, C_0, C_1, \mathsf{aux}) :\ C_0(x) \neq C_1(x)\right] \geq \frac{1}{p(\lambda)} \ . \quad (1)$$

A candidate $i\mathcal{O}$ for functionalities implementable by NC^1 circuits was constructed based on a simplified variant of multi-linear maps, and proven secure in an idealized model [GGH$^+$13b]. The same candidate was conjectured to be an $e\mathcal{O}$ for NC^1 [BCP14]. In both [GGH$^+$13b] and [BCP14], the obfuscators were boosted to functionalities implementable by polynomial-size circuits by using fully-homomorphic encryption [Gen09].

We mention that Garg et al. [GGHW14] presented an implausibility result for $e\mathcal{O}$ for general distributions. Their counterexample used very contrived general auxiliary inputs, and thus seems to have no implications for the particular auxiliary inputs we use (cf. Proposition 2).

2.5 Succinct Non-interactive Arguments of Knowledge

Succinct non-interactive arguments of knowledge (SNARK) were defined by Bitansky at al. [BCCT13, BCC$^+$14]. They provide constructions of SNARKs based on knowledge-of-exponent assumptions [BCCT13] and extractable collision-resistant hash-functions [BCC$^+$14]. These are both non-falsifiable assumptions, but Gentry and Wichs [GW11] prove that SNARKs cannot be built from falsifiable assumptions via black-box reductions.

Definition 6 (The NP language L_{legit})**.** *We define the following language L_{legit} which consists of triples of hash-function descriptions, Turing-machine descriptions and hash values defined by the following* NP *relation:*

$$R_{legit}\big((H, M, h), x\big) = 1 \Leftrightarrow M(x) = 1 \wedge H(x) = h \ .$$

In our construction, we will require a SNARK proof system for the language just defined.

Definition 7 (SNARK)**.** *A triple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$*, where* Verify *is deterministic, is a succinct non-interactive argument of knowledge (SNARK) for the language L_{legit} (Definition 6), if the following hold:*

1. *Completeness: For every $\big(\eta = (H, M, h), x\big) \in R_{legit}$:*

$$\Pr\big[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda);\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \eta, x) :\ \mathsf{Verify}(\mathsf{crs}, \eta, \pi) = 1\big] = 1 \ .$$

2. *(Adaptive) Soundness: For every PPT adversary $\mathcal{A}$:*

$$\Pr\big[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda);\ (\eta, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) :\ \mathsf{Verify}(\mathsf{crs}, \eta, \pi) = 1 \wedge \eta \notin L_{legit}\big] \leq \mathsf{negl}(\lambda) \ .$$

3. *(Adaptive) Proof of Knowledge: For every PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that*

$$\Pr\left[\begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda); \\ (\eta, \pi) \leftarrow \mathcal{A}(\mathsf{crs});\ x \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}) \end{array} :\ \mathsf{Verify}(\mathsf{crs}, \eta, \pi) = 1 \wedge (\eta, x) \notin R_{legit}\right] \leq \mathsf{negl}(\lambda) \ .$$

4. *Succinctness: The length of an honestly generated proof $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \eta, x)$ and the running time of $\mathsf{Verify}(\mathsf{crs}, \eta, \pi)$ are both bounded by $p(\lambda + |H| + |M| + |h| + \log T(R_{legit}))$, where $T(R_{legit})$ is the running time of R_{legit}, and $p(\cdot)$ is a universal polynomial, i.e., it does not depend on R_{legit}.*

2.6 Functional Signatures

Functional signatures were introduced by Boyle et al. [BGI14]. They generalize the concept of digital signatures by letting the holder of a secret key sk derive keys sk_f for functions f.[2] Such a key sk_f enables signing of (and only of) messages in the range of f: running $\mathsf{Sign}(f, \mathsf{sk}_f, w)$ produces a signature on $f(w)$. *Function privacy* requires that signatures under different signing keys be indistinguishable and *succinctness* requires that the signature length be independent of w and the size of f.

Definition 8 (Functional Signatures [BGI14])**.** *A functional signature scheme $\mathcal{FS}$ for message space $\mathcal{M}$ and function family $\mathcal{F} = \{f \colon \mathcal{D}_f \to \mathcal{R}_f\}$ with $\mathcal{R}_f \subseteq \mathcal{M} \cup \{\bot\}$ consists of the following PPT algorithms:*

$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda)$*: On input a security parameter λ,* Setup *outputs a pair of master signing and verification key* $(\mathsf{msk}, \mathsf{mvk})$.

$\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$*: On input a master secret key* msk*, and a function $f \in \mathcal{F}$,* KeyGen *outputs a signing key sk_f for f.*

$\sigma \leftarrow \mathsf{Sign}(f, \mathsf{sk}_f, w)$*: On input a function $f \in \mathcal{F}$, a signing key sk_f for f, and $w \in \mathcal{D}_f$,* Sign *outputs a signature σ on $f(w)$ if $f(w) \neq \bot$, and $\bot$ otherwise.*

[2] In the original definition [BGI14], f is given as a circuit, but in their construction of functional encryption, Boyle et al. [BCP14] allow f to be a Turing machine. In this work we adopt the latter definition.

$\mathsf{Verify}(\mathsf{mvk}, m, \sigma) \in \{0,1\}$*: On input a master verification key* mvk*, a message* $m \in \mathcal{M}$*, and a signature* σ*,* Verify *outputs* $b \in \{0,1\}$*.*

Correctness states that correctly generated signatures verify. *Unforgeability* is formalized via a game in which an adversary is given the verification key and is allowed queries to a key-generation oracle, $\mathsf{key}(f, i)$, and a signing oracle, $\mathsf{sign}(f, i, m)$, that work as follows:

- $\mathsf{key}(f, i)$: if a signing key for (f, i) has already been generated, return the recorded key; otherwise generate and return a fresh signing key $\mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$ and record $\big((f, i), \mathsf{sk}_f\big)$.
- $\mathsf{sign}(f, i, w)$: check if there is a record $\big((f, i), \mathsf{sk}^i_f\big)$ for some sk^i_f, if not generate sk^i_f for (f, i) and record it. In both cases, return a signature on $f(w)$ as $\sigma \leftarrow \mathsf{Sign}(f, \mathsf{sk}^i_f, w)$.

Function privacy is formalized via a game in which an adversary is given signing keys for two functions f_0, f_1 (of equal description size) of its choice, then outputs (w_0, w_1) (with $|w_0| = |w_1|$), is given the output of $\mathsf{Sign}(f_b, \mathsf{sk}_{f_b}, w_b)$ for some $b \in \{0,1\}$, which it has to guess. Finally, *succinctness* requires that the size of a signature is independent of $|w|$ and $|f|$, the description size of f.

Boyle et al. [BGI14] present a construction based on succinct non-interactive arguments of knowledge (SNARKs).

3 Constrained PRFs for Unbounded Inputs

In this section we construct a family of constrained PRFs with unbounded input length. As a warm-up, we first construct a family of constrained PRFs w.r.t. polynomial-size circuits, whose inputs are of some fixed length.

3.1 A Circuit-Constrained PRF

Our circuit-constrained PRF F uses a puncturable PRF PF with input space $\mathcal{X} = \{0,1\}^n$. The output of $\mathsf{F}(k, x)$ is simply $\mathsf{PF}(k, x)$. To constrain F w.r.t. a circuit C, we construct a circuit $P_{k,C}$, which, on input x, runs C on x and outputs $\mathsf{PF}(k, x)$ if $C(x) = 1$, and $\perp$ otherwise. A constrained key k_C for C is then an indistinguishability obfuscation of $P_{k,C}$, i.e., $k_C \leftarrow i\mathcal{O}(1^\lambda, P_{k,C})$.

Construction 1 (Circuit-Constrained PRF)**.** *Let* $\mathcal{C}_\lambda = \{C : \{0,1\}^n \to \{0,1\}\}$ *be a family of polynomial-size circuits,* $\mathcal{PF}_\lambda = \{\mathsf{PF} \colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ *a family of puncturable PRFs, and* $i\mathcal{O}$ *an indistinguishability obfuscator for a family of polynomial-size circuits* $\mathcal{P}_\lambda$*, which contains all circuits defined in* (2) *for all* $C \in \mathcal{C}_\lambda$*. We construct a family of PRFs* $\mathcal{F}_\lambda = \{\mathsf{F} \colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ *constrained w.r.t.* $\mathcal{C}_\lambda$ *with a constrained-key space* $\mathcal{K}_\mathcal{C}$ *such that* $\mathcal{K}_\mathcal{C} \cap \mathcal{K} = \emptyset$*.*[3] *Following is a description of* $\mathcal{F} = (\mathsf{F.Smp}, \mathsf{F.Eval}, \mathsf{F.Constr})$*:*

$\underline{k \leftarrow \mathsf{F.Smp}(1^\lambda)}$*: On input a security parameter* λ*, output a secret key* $k \in \mathcal{K}$ *as* $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$*.*

$\underline{k_C \leftarrow \mathsf{F.Constr}(k, C)}$*: On input a secret key* $k \in \mathcal{K}$ *and a description of a circuit* $C \in \mathcal{C}_\lambda$*, output* $k_C \in \mathcal{K}_\mathcal{C}$ *as* $k_C \leftarrow i\mathcal{O}(1^\lambda, P_{k,C})$*, i.e., compute an indistinguishability obfuscation of the circuit* $P_{k,C} \in \mathcal{P}_\lambda$ *defined as*

$$P_{k,C}(x) := \begin{cases} \mathsf{PF}(k, x) & \textit{if } |x| = n \wedge C(x) = 1 \\ \perp & \textit{otherwise} \end{cases} \tag{2}$$

[3] W.l.o.g. we assume from now on that we have $\mathcal{K} \cap \mathcal{K}_\mathcal{C} = \emptyset$, as this can always be achieved by simply prepending a '0' to elements from $\mathcal{K}$ and a '1' to elements from $\mathcal{K}_\mathcal{C}$.

<u>$\mathsf{F.Eval}(\kappa, x)$</u>: *On input a key* $\kappa \in \mathcal{K} \cup \mathcal{K}_{\mathcal{C}}$ *and* $x \in \{0,1\}^n$, *do the following:*

- *If* $\kappa \in \mathcal{K}$, *output* $\mathsf{PF.Eval}(\kappa, x)$.
- *If* $\kappa \in \mathcal{K}_{\mathcal{C}}$, *output* $\kappa(x)$, *interpreting* κ *as a circuit.*

The proof of selective security of $\mathcal{F}$, as just constructed, is relatively straightforward. Recall that in the selective-security game, the adversary $\mathcal{A}$ outputs x^*, then the challenger chooses $k \leftarrow \mathsf{F.Smp}$ and gives $\mathcal{A}$ access to a constrained-key oracle constr, which can be queried on any C with $C(x^*) = 0$. $\mathcal{A}$ must then distinguish $\mathsf{F}(k, x^*)$ from random. We modify this game by deriving from k a key k_{x^*} which is punctured at x^* and computing constrained keys as obfuscations of $P_{k_{x^*},C}$ (defined like $P_{k,C}$ but using k_{x^*} instead of k). Since $\mathsf{PF}(k, x) = \mathsf{PF}(k_{x^*}, x)$ for all $x \neq x^*$, and since for any circuit C that the adversary can query we have $P_{k,C}(x^*) = P_{k_{x^*},C}(x^*) = \perp$, the circuits $P_{k_{x^*},C}$ and $P_{k,C}$ are functionally equivalent, and thus by $i\mathcal{O}$ the two games are indistinguishable.

An adversary $\mathcal{A}$ winning the modified game can then be translated into an adversary $\mathcal{B}$ against $\mathcal{PF}$. In the security game for $\mathcal{PF}$ (Figure 3, p. 19), $\mathcal{B}$ runs $(x^*, st) \leftarrow \mathcal{A}$ and outputs $(x^*, \{x^*\}, st)$. Given k_{x^*} and y, $\mathcal{B}$ can now simulate the modified game and output whatever $\mathcal{A}$ outputs. $\mathcal{B}$'s probability of breaking the security of $\mathcal{PF}$ is the same as that of $\mathcal{A}$ winning the modified game.

3.2 A Turing-Machine-Constrained PRF

In this section we construct a family of constrained PRFs for unbounded inputs, whose keys can be constrained for sets decided by Turing machines. We start by observing that in the circuit-constrained PRFs (Construction 1) the size of a constrained key k_C for a circuit C depends on the running time of C. This is so, because k_C is an indistinguishability obfuscation of the circuit $P_{k,C}$ that runs C to check whether the input is legitimate, i.e., whether $C(x) = 1$, and if so, evaluates PF. Towards constructing constrained PRFs w.r.t. Turing machines, and avoiding translating running time into key size, we look at a progression of modifications to the circuit-constrained PRFs.

At first attempt, replacing C in $P_{k,C}$ with a TM M, we get a TM $P_{k,M}$, and therefore we cannot use obfuscation, as current constructions of $i\mathcal{O}$ and $e\mathcal{O}$ only exist for circuits. Towards making $P_{k,M}$ a circuit, one could outsource the check of input legitimacy outside the circuit to be obfuscated, by using succinct non-interactive arguments (SNARG). However, legitimate inputs are still unbounded, and hence we are back to obfuscating a TM. It is thus necessary to compress the unbounded input to a fixed length in order to obtain a circuit, which in the end we can obfuscate.

We achieve this by applying a collision-resistant hash function H to the unbounded inputs, that is, we evaluate the PRF on hashed inputs. In order to guarantee input legitimacy, we use a SNARK to prove that a given hash is the hash value of a legitimate input. We define a circuit $P_{k,M}$ that is given a hash value and a SNARK proof and evaluates the PRF on the hash if the proof verifies. The secret key is then an $e\mathcal{O}$ obfuscation of $P_{k,M}$.

Let us justify the use of $e\mathcal{O}$ and SNARKs. As in the case of circuit-constrained PRFs, we want to reduce the selective security of the TM-constrained PRF F to the selective security of the underlying puncturable PRF PF. In a first game hop we replace $P_{k,M}$ with $P_{k_{h^*},M}$, which is identical to $P_{k,M}$ except that the key k is replaced with a key k_{h^*} that punctures out $h^* := H(x^*)$. Unfortunately, the use of the hash function makes the two circuits, $P_{k,M}$ and $P_{k_{h^*},M}$, inequivalent: there exists $x \neq x^*$ such that $H(x) = H(x^*)$, and on input $H(x)$, $P_{k,M}$ outputs $\mathsf{PF}(k, H(x)) = \mathsf{PF}(k, h^*)$ and $P_{k_{h^*},M}$ outputs $\perp$, which means we cannot use $i\mathcal{O}$, and hence we use $e\mathcal{O}$ instead.

Hash-function collisions are also the reason we need to use SNARKs rather than SNARGs: if an adversary can distinguish obfuscations of $P_{k,M}$ and $P_{k_{h^*},M}$ by finding a collision for H then we need to extract this collision in the security proof. Therefore, we use SNARKs (arguments of *knowledge*).

In this construction, a constrained key k_M for a TM M is now an $e\mathcal{O}$ obfuscation of a circuit $P_{k,M}$ which is given (h, π) and checks whether π proves knowledge of an x such that $H(x) = h$ and $M(x) = 1$, and if so, evaluates PF on h. The size of a constrained key k_M depends on the size of the description of M, but no longer on its running time.

We further enhance this construction by using functional signatures to reduce both the running time of the key-constraining algorithm and the size of the effective constrained keys (by effective we mean the part of the key that needs to be kept secret). Instead of obfuscating a circuit for each TM M, we obfuscate a *single* circuit C that works for all TMs. A constrained key for a TM M is now simply a signature σ on M. The circuit C is given σ in addition to (M, h, π), verifies the signature σ on M in addition to verifying π; and if all checks pass, it evaluates PF on h.

The reason for using functional signatures is the following: in the proof of Proposition 2, we will use an adversary against F to build a distinguisher between $P_{k,M}$ and $P_{k_{h^*},M}$, who will have to sign TMs to answer the adversary's constraining queries. By $e\mathcal{O}$ we know that there exists an extractor $\mathcal{E}$ that extracts a differing input. We then need to argue unforgeability of signatures; however, we don't know how $\mathcal{E}$ answers the adversary's queries. Thus instead of providing $\mathcal{E}$ with a signing oracle, we give it a functional signing key that allows it to produce all necessary signatures.

Construction 2 (TM-constrained PRF)**.** *Let $\mathcal{PF}_\lambda = \{\mathsf{PF}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ be a puncturable PRF with fixed input length, $\mathcal{H}_\lambda = \{H\colon \{0,1\}^* \to \{0,1\}^n\}_\lambda$ a family of collision-resistant hash functions, $\mathcal{FS}$ a functional signature scheme,* SNARK *a SNARK system for the language L_{legit} (Definition 6) and $e\mathcal{O}$ an extractability obfuscator for a family of polynomial-size circuits $\mathcal{P}_\lambda$.*

We construct a family of PRFs $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \{0,1\}^* \to \mathcal{Y}\}$ constrained w.r.t. to a polynomial-size family of Turing machines $\mathcal{M}_\lambda$. Following is a description of $\mathcal{F} = (\mathsf{F.Smp}, \mathsf{F.Eval}, \mathsf{F.Constr})$.

$\underline{\mathsf{K} \leftarrow \mathsf{F.Smp}(1^\lambda)}$*: On input a security parameter λ, do the following:*

$H \leftarrow \mathsf{H.Smp}(1^\lambda)$*, i.e., sample a collision-resistant hash function.*

$\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$*, i.e., sample a common reference string for the SNARK system.*

$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$*, i,e., sample a pair of master signing and verification key for the functional signature scheme. Let $f_I : \mathcal{M}_\lambda \to \mathcal{M}_\lambda$ be the identity function, i.e., $f_I(M) = M$. Compute a signing key for f_I as* $\mathsf{sk}_{f_I} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_I)$.

$k \leftarrow \mathsf{PF.Smp}(1^\lambda)$*, i.e., sample a secret key for the puncturable PRF.*

$\widetilde{P} \leftarrow e\mathcal{O}(1^\lambda, P)$*, i.e, compute an extractability obfuscation for the following circuit $P \in \mathcal{P}_\lambda$:*

$$P(M, h, \pi, \sigma) := \begin{cases} \mathsf{PF.Eval}(k, h) & \textit{if } \mathsf{SNARK.Verify}\big(\mathsf{crs}, (H, M, h), \pi\big) = 1 \\ & \qquad\qquad\qquad \wedge\ \mathsf{FS.Verify}(\mathsf{mvk}, M, \sigma) = 1 \\ \perp & \textit{otherwise} \end{cases}$$

where $(H, \mathsf{crs}, \mathsf{mvk}, k)$ is hard-coded in P.

Set $\mathsf{pp} = (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P})$ *and output* $\mathsf{K} := (k, \mathsf{sk}_{f_I}, \mathsf{pp})$.

$\underline{k_M \leftarrow \mathsf{F.Constr}(\mathsf{K}, M)}$*: On input a secret key K, and a TM $M \in \mathcal{M}_\lambda$, compute a signature on M as* $\sigma \leftarrow \mathsf{FS.Sign}(I, \mathsf{sk}_{f_I}, M)$*, and output* $k_M := (M, \sigma, \mathsf{pp})$.

$\underline{\mathsf{F.Eval}(\kappa, x)}$*: On input a key $\kappa \in \mathcal{K} \cup \mathcal{K}_\mathcal{M}$, and an $x \in \{0,1\}^*$, do the following:*

- *Case $\kappa \in \mathcal{K}$, $\kappa = (k, \mathsf{sk}_{f_I}, \mathsf{pp} = (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P}))$: output* $\mathsf{PF.Eval}(k, H(x))$.

- *Case* $\kappa \in \mathcal{K}_{\mathcal{M}}$, $\kappa = (M, \sigma, \mathsf{pp} = (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P}))$*: if* $M(x) = 1$*, set* $h := H(x)$ *(thus* $((H, M, h), x) \in R_{legit}$*), generate a SNARK proof* $\pi \leftarrow \mathsf{SNARK.Prove}(\mathsf{crs}, (H, M, h), x)$*, and output* $\widetilde{P}(M, h, \pi, \sigma)$*.*

Remark 1. *Although* pp *and* $\widetilde{P}$ *are computed once and for all, and in fact serve as public parameters for the constrained PRF, we include them in the constrained key* k_M *for notational simplicity.*

$P \in \mathcal{P}_\lambda$ *with the succinct proof* π *being the dominating factor in its input length. However,* $|\pi|$ *is polynomially bounded by some universal polynomial independent of* L_{legit}*, even for a super-polynomial witness length* $|x|$*.*

Theorem 1. $\mathcal{F}_\lambda$ *of Construction 2 is a selectively secure family of constrained PRFs with input space* $\{0,1\}^*$ *for which constrained keys can be derived for any set that can be decided by a Turing machine with polynomial description size, if* $\mathcal{PF}_\lambda$ *is a selectively secure family of puncturable PRFs,* $\mathcal{H}_\lambda$ *is a family of collision-resistant hash functions,* $e\mathcal{O}$ *is a secure extractability obfuscator for a polynomial-size family of circuits* $\mathcal{P}_\lambda$*,* SNARK *is a SNARK system for* L_{legit} *from Definition 6, and* $\mathcal{FS}$ *is a secure functional signature scheme.*

Proof. Let $\mathcal{A}$ be an arbitrary PPT adversary for the game $\mathbf{Exp}^{(\emptyset,\{\mathsf{constr},\mathsf{eval}\}),b}_{\mathcal{F},\mathcal{A}}(\lambda)$, as defined in Figure 1, which we abbreviate as $\mathbf{Exp}^b$ for simplicity. We need to show that $\mathbf{Exp}^0$ and $\mathbf{Exp}^1$ are indistinguishable. Our proof will be by game hopping and we define a series of hybrid games $\mathbf{Exp}^{b,(0)} := \mathbf{Exp}^b$, $\mathbf{Exp}^{b,(1)}$, $\mathbf{Exp}^{b,(2)}$, $\mathbf{Exp}^{b,(3)}$ and show that for $b = 0, 1$ and $c = 0, 1, 2$ the games $\mathbf{Exp}^{b,(c)}$ and $\mathbf{Exp}^{b,(c+1)}$ are indistinguishable. Finally we show that $\mathbf{Exp}^{0,(3)}$ and $\mathbf{Exp}^{1,(3)}$ are also indistinguishable, which concludes the proof. All games are defined in Figure 2, using the following definitions:

$$f_I(M) := M \tag{3}$$

$$f_{x^*}(M) := \begin{cases} M & \text{if } M(x^*) = 0 \\ \perp & \text{otherwise} \end{cases} \tag{4}$$

$$P_{H,\mathsf{crs},\mathsf{mvk},k}(M, h, \pi, \sigma) := \begin{cases} \mathsf{PF.Eval}(k, h) & \text{if } \ \mathsf{SNARK.Verify}\big(\mathsf{crs}, (H, M, h), \pi\big) = 1 \\ & \qquad \wedge\ \mathsf{FS.Verify}(\mathsf{mvk}, M, \sigma) = 1 \\ \perp & \text{otherwise} \end{cases} \tag{5}$$

$\mathbf{Exp}^{b,(0)}$ is the original game $\mathbf{Exp}^{b,(\emptyset,\{\mathsf{constr},\mathsf{eval}\})}_{\mathcal{F},\mathcal{A}}(\lambda)$ for Construction 2.

$\mathbf{Exp}^{b,(1)}$ differs from $\mathbf{Exp}^{b,(0)}$ by replacing the signing key sk_{f_I} with $\mathsf{sk}_{f_{x^*}}$, which only allows to sign machines M with $M(x^*) = 0$.

$\mathbf{Exp}^{b,(2)}$ differs from $\mathbf{Exp}^{b,(1)}$ by replacing the full key of the puncturable PRF PF, with one that is punctured at $H(x^*)$ in the definition of P.

$\mathbf{Exp}^{b,(3)}$ differs from $\mathbf{Exp}^{b,(2)}$ by answering eval queries using the punctured key k_{h^*} and aborting whenever the adversary queries its eval oracle on a collision with x^*.

Intuitively, $\mathbf{Exp}^{b,(0)}(\lambda)$ and $\mathbf{Exp}^{b,(1)}(\lambda)$ are computationally indistinguishable as the only difference between them is the use of the signing key sk_{f_I} and $\mathsf{sk}_{f_{x^*}}$, respectively, in answering constraining queries. By the definition of the selective-security game, a signature is computed only on a TM M such that $M(x^*) = 0$. Therefore, f_{x^*} coincides with f_I on all such legitimate queries. By function privacy of $\mathcal{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$, signatures generated under f_{x^*} and f_I are computationally indistinguishable. See Appendix C.1 for the proof of the following.

```
$\underline{\mathbf{Exp}^{(\emptyset,\{\mathsf{constr},\mathsf{eval}\}),b}_{\mathcal{F},\mathcal{A}}(\lambda)}$
$(x^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$\mathsf{K} \leftarrow \mathsf{F.Smp}(1^\lambda)$
If $b = 1$
    $y^* := \mathsf{F.Eval}(\mathsf{K}, x^*)$
Else
    $y^* \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{constr}(\cdot),\mathsf{eval}(\cdot)}(st, y^*)$
Return $b'$

$\underline{\textbf{Oracle } \mathsf{constr}(M)}$
If $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$
    Return $\perp$
$k_M \leftarrow \mathsf{F.Constr}(\mathsf{K}, M)$
Return $k_M$

$\underline{\textbf{Oracle } \mathsf{eval}(x)}$
If $x = x^*$
    Return $\perp$
$y = \mathsf{F.Eval}(\mathsf{K}, x)$
Return $y$
```

```
$\underline{\mathbf{Exp}^{b,(c)}_{\mathcal{F},\mathcal{A}}(\lambda)}$   // $c \in \{0,1,2,3\}$
$(x^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$H \leftarrow \mathsf{H.Smp}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$
$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$
$\mathsf{sk}_{f_I} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_I)$ with $f_I$ defined in Eq. (3)
$\mathsf{sk}_{f_{x^*}} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_{x^*})$ with $f_{x^*}$ defined in Eq. (4)
$k \leftarrow \mathsf{PF.Smp}(1^\lambda)$
$k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{H(x^*)\})$
$\underline{\text{If } c \geq 2}$ then
    $P := P_{H,\mathsf{crs},\mathsf{mvk},k}$ as defined Eq. (5)
Else
    $P := P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ as defined Eq. (5)
$\widetilde{P} \leftarrow e\mathcal{O}(1^\lambda, P)$
Set $\mathsf{pp} = (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P})$
If $b = 1$, $y^* := \mathsf{PF.Eval}(k, H(x^*))$, else $y^* \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{constr}(\cdot),\mathsf{eval}(\cdot)}(st, y^*)$
Return $b'$

$\underline{\textbf{Oracle } \mathsf{constr}(M)}$
If $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$
    Return $\perp$
$\underline{\text{If } c \geq 1}$
    $\sigma \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_{f_{x^*}}, M)$
Else
    $\sigma \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_{f_I}, M)$
Return $k_M := (M, \sigma, \mathsf{pp})$

$\underline{\textbf{Oracle } \mathsf{eval}(x)}$
If $x = x^*$
    Return $\perp$
$\underline{\text{If } c = 3}$
    If $H(x) = H(x^*)$ then abort
    Else $y := \mathsf{PF.Eval}(k_{h^*}, H(x))$
Else
    $y := \mathsf{PF.Eval}(k, H(x))$
Return $y$
```

Figure 2: Original security game and hybrids used in the proof of Theorem 1.

Proposition 1. *Games* $\mathbf{Exp}^{b,(0)}$ *and* $\mathbf{Exp}^{b,(1)}$ *are computationally indistinguishable for* $b = 0, 1$ *if* $\mathcal{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$ *is a correct functional signature scheme satisfying function privacy and succinctness.*

The only difference between $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ is the definition of the circuit P that is obfuscated. In $\mathbf{Exp}^{b,(1)}$ the circuit P is defined as in (5), with $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$. In $\mathbf{Exp}^{b,(2)}$, the key k is replaced by $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{H(x^*)\})$, a key that punctures out $H(x^*)$. An adversary that distinguishes $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ distinguishes $e\mathcal{O}$ obfuscations of P_k and $P_{k_{h^*}}$. There exists thus an $e\mathcal{O}$ extractor that extracts an input on which P_k and $P_{k_{h^*}}$ differ.

By correctness of the puncturable PRF, the circuits only differ on inputs $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$, where

$$\hat{h} = H(x^*) \ , \tag{6}$$

as that is where the punctured key behaves differently. Moreover, the signature σ must be valid on M, as otherwise both circuits output $\perp$. By unforgeability of the functional signature scheme we must have

$$\hat{M}(x^*) = 0 \ , \tag{7}$$

as the adversary only obtains signatures via its constrain queries, when it submits machines satisfying (7).

Finally the extracted proof $\hat{\pi}$ must be valid for $(H, \hat{M}, \hat{h})$, as otherwise both circuits output $\perp$. By SNARK extractability, we can therefore extract a witness $\hat{x}$ for $(H, \hat{M}, \hat{h}) \in L_{legit}$, that is, (i) $\hat{M}(\hat{x}) = 1$ and (ii) $H(\hat{x}) = \hat{h}$. Now (i) and (7) imply $\hat{x} \neq x^*$ and (ii) and (6) imply $H(\hat{x}) = H(x^*)$. Together, this means $(\hat{x}, x^*)$ is a collision for H. We make this argument formal in the following proposition, which is proved in Appendix C.2.

Proposition 2. *$\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ are computationally indistinguishable for $b = 0, 1$, if $e\mathcal{O}$ is a secure extractability obfuscator, $\mathcal{FS}$ is unforgeable and $\mathcal{H}$ is collision-resistant.*

For the game hop from games $\mathbf{Exp}^{b,(2)}$ to $\mathbf{Exp}^{b,(3)}$, indistinguishability follows directly from collision resistance of $\mathcal{H}$, as the only difference is that $\mathbf{Exp}^{b,(3)}$ aborts when $\mathcal{A}$ finds a collision.

Proposition 3. *Games $\mathbf{Exp}^{b,(2)}$ and $\mathbf{Exp}^{b,(3)}$ are computationally indistinguishable for $b = 0, 1$, if $\mathcal{H}$ is collision-resistant.*

See Appendix C.3 for the proof. We have now reached a game, $\mathbf{Exp}^{b,(3)}$, in which the key k is only used to create a punctured key k_{h^*}. The experiment can thus be simulated by an adversary $\mathcal{B}$ against selective security of $\mathcal{PF}$, who first asks for a key for the set $\{0,1\}^n \setminus \{H(x^*)\}$ and then uses $\mathcal{A}$ to distinguish $y^* = \mathsf{PF.Eval}(k, H(x^*))$ from random.

Proposition 4. *Games $\mathbf{Exp}^{0,(3)}$ and $\mathbf{Exp}^{1,(3)}$ are indistinguishable if $\mathcal{PF}$ is a selectively secure family of puncturable PRFs.*

See Appendix C.4 for the proof. Theorem 1 now follows from Propositions 1, 2, 3 and 4. $\square$

4 Applications

Our first application of constrained PRFs with unbounded input length is broadcast encryption (BE). We construct a scheme where during setup the number of potential receivers need not be known. Users can be dynamically added to the system and are assigned consecutive numbers $i \in \mathbb{N}$.

Our scheme is set up by computing a PRF key k, which is used to broadcast and to derive user keys. In order to broadcast a message to the set $S \subseteq \mathbb{N}$, let $x \in \{0,1\}^*$ be the characteristic vector of S, i.e., $x_i = 1$ iff $i \in S$. Using a symmetric encryption scheme, the message is encrypted under the key $K := \mathsf{F}(k, x)$. User i is given a key $sk_i \leftarrow \mathsf{F.Constr}(k, S_i)$ where $S_i \subseteq \{0,1\}^*$ is the set of strings $x \in \{0,1\}^*$ with $x_i = 1$. User i can therefore compute all keys K for sets to which she belongs. Due to space constraints, details are deferred to Appendix A.

4.1 ID-Based Non-interactive Key Exchange for Unbounded Groups

In this section we present a construction of identity-based non-interactive key exchange (ID-NIKE) [SOK00]. This allows users to compute shared keys without any interaction—it suffices to know the identity of the users one wants to share a key with. In our construction, a user can compute a shared key for any group of users and there is no a priori bound on the size of these groups. We generalize the construction of [BW13, Hof14], where identities are elements from $\{0,1\}^\ell$ and the system is set up by creating a secret key msk for a constrained PRF. A key for a group of users $\{id_1, \ldots, id_n\}$ is defined as $\mathsf{F.Eval}(msk, x)$, where $x = id_1 \| \ldots \| id_n$ and we assume identities are always ordered lexicographically.

Since in the previous constructions the CPRF is set up for a fixed input length m there is an a-prior-fixed maximum number of users which can share a key, namely m/ℓ. As a user's id could appear in any position of the string x, the owner of id is given constrained keys for the sets $(id\|?^{(n-1)\ell}) :=$

$\{id \| z \mid z \in \{0,1\}^{(n-1)\cdot\ell}\}$, $(?^{\ell} \| id \| ?^{(n-2)\ell})$, $\ldots$, $(?^{(n-1)}\ell \| id)$. These keys thus allow the user to compute the key for any set which she is part of.

We generalize this to sets of users of unbounded size. Again, a key for a set $\{id_1, \ldots, id_n\}$ is defined as $\mathsf{F.Eval}(msk, id_1 \| \ldots \| id_n)$, but now n can be arbitrary and is not fixed in advance. In order to let a user with identity id compute the keys of the sets which she is part of—but not anything else—, she is given a constrained key for the following Turing machine M_{id}: on input $x \in \{0,1\}^*$, machine M_{id} outputs 1 if and only if id is a substring of x, which starts at position $i \cdot \ell + 1$, for some $i \geq 0$, that is, at position 1 or $\ell + 1$ or $2\ell + 1$, etc.

ID-NIKE. An (unbounded) ID-NIKE scheme consists of three algorithms:

$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: on input λ, output public parameters pp and a master secret key msk;

$sk_{id} \leftarrow \mathsf{Extract}(msk, id)$: in input the master key and $id \in \{0,1\}^\ell$, output a secret key sk_{id}.

$k_{\mathcal{I}} \leftarrow \mathsf{KeyGen}(pp, sk_{id}, \mathcal{I})$: on input pp, a key sk_{id} for id and a list $\mathcal{I} \subseteq \{0,1\}^\ell$ of n (for arbitrary n) users with $id \in \mathcal{I}$, output a shared key $k_{\mathcal{I}}$.

Correctness is defined as follows: for all $id, id' \in \{0,1\}^\ell$, all $\mathcal{I} \subseteq \{0,1\}^\ell$ with $id, id' \in \mathcal{I}$, all $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$ and $sk_{id'} \leftarrow \mathsf{Extract}(msk, id')$, it holds that $\mathsf{KeyGen}(pp, sk_{id}, \mathcal{I}) = \mathsf{KeyGen}(pp, sk_{id'}, \mathcal{I})$.

Following [PS09] we define security via a game where an adversary can obtain secret keys sk_{id} for identities of his choice and can query secret keys $k_{\mathcal{I}}$ for sets $\mathcal{I}$ of his choice. The scheme is secure if the adversary cannot distinguish a key $k_{\mathcal{I}^*}$ for a set $\mathcal{I}^*$ of his choice from random, where we must have $id \notin \mathcal{I}^*$ for all id for which the adversary queried key extraction, and $\mathcal{I}^* \neq \mathcal{I}$ for all $\mathcal{I}$ for which the adversary queried a shared key. We prove that our scheme satisfies the selective variant of this definition, where the adversary must output $\mathcal{I}^*$ before getting access to its oracles.

ID-NIKE from constrained PRFs for unbounded inputs. Our unbounded ID-NIKE is obtained from a constrained PRF with unbounded input length $(\mathsf{F.Smp}, \mathsf{F.Constr}, \mathsf{F.Eval})$ as follows.

- $\mathsf{Setup}(1^\lambda)$: Return $msk \leftarrow \mathsf{F.Smp}(1^\lambda)$.
- $\mathsf{Extract}(msk, id)$: On input $id \in \{0,1\}^\ell$ do the following: define a Turing machine M_{id} that on input a string $x \in \{0,1\}^*$ outputs 1 iff x is of the form $x' \| id \| x''$ with $x' \in \{0,1\}^{n' \cdot \ell}$ and $x'' \in \{0,1\}^{n'' \cdot \ell}$ for some $n', n'' \in \mathbb{N}$; return $sk_{id} \leftarrow \mathsf{F.Constr}(msk, M_{id})$.
- $\mathsf{KeyGen}(pp, sk_{id}, \mathcal{I})$: If $\mathcal{I} = \{id_1, \ldots, id_n\} \subseteq \{0,1\}^\ell$ for some n and $id \in \mathcal{I}$ then define $x := id_{i_1} \| \ldots \| id_{i_n}$, with $id_{i_j} < id_{i_j+1}$ for all j, and output $k_{\mathcal{I}} := \mathsf{F.Eval}(sk_{id}, x)$; else output $\perp$.

Correctness of our scheme follows from correctness of the underlying constrained PRF. Selective security of the ID-NIKE follows from selective security of the CPRF (Definition 2). Given an adversary $\mathcal{A}$ against the ID-NIKE, we construct an adversary $\mathcal{B}$ against the CPRF. First $\mathcal{B}$ runs $\mathcal{A}$ to obtain $\mathcal{I}^*$ and sends x^*, the concatenation of the lexicographically ordered elements of $\mathcal{I}^*$, to its challenger.

$\mathcal{B}$ answers $\mathcal{A}$'s queries as follows: When $\mathcal{A}$ queries a secret key for $id \in \mathcal{I}^*$ or the shared key for $\mathcal{I}^*$ then reply with $\perp$. On a legal secret-key query for id, construct a Turing machine M_{id} as in the definition of $\mathsf{Extract}$, query the constr oracle on M_{id} and forward the reply to $\mathcal{A}$. When $\mathcal{A}$ queries a shared key for a set $\mathcal{I} \neq \mathcal{I}^*$, construct x as in KeyGen, query eval on x and forward the reply.

Note that $\mathcal{B}$ makes no illegal queries (any queried M evaluates x^* to 0 and x^* is never queried to eval) and perfectly simulates the game for $\mathcal{A}$. When $\mathcal{B}$ receives a value y which is either $\mathsf{F.Eval}(msk, x^*)$ or random, it forwards y as the challenge key $k_{\mathcal{I}^*}$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ does. $\mathcal{B}$ thus breaks the CPRF with the same probability as $\mathcal{A}$ breaking the ID-NIKE, which concludes the proof.

References

[BCC$^+$14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, February 2014.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 520–537. Springer, March 2014.

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, March 2014.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, August 2005.

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 455–470. Springer, December 2008.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, December 2013.

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 206–223. Springer, August 2014.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, August 2013.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 513–530. Springer, August 2013.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 82–101. Springer, 2014. Available at `http://eprint.iacr.org/2014/416`.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, August 1993.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, May 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 FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, August 2014.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33:792–807, 1986.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[HKW14] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. Cryptology ePrint Archive, Report 2014/521, 2014. `http://eprint.iacr.org/2014/521`.

[Hof14] Dennis Hofheinz. Fully secure constrained pseudorandom functions using random oracles. Cryptology ePrint Archive, Report 2014/372, 2014. `http://eprint.iacr.org/2014/372`.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, May 2014.

[PPS11] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Security notions for broadcast encryption. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 377–394. Springer, June 2011.

[PS09] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography*, 52(2):219–241, 2009.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517. Springer, August 2014.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[YFDL04] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04*, pages 354–363. ACM Press, October 2004.

A Application: Broadcast Encryption to Unbounded Number of Users

We now show how a constrained PRF for unbounded input lengths can be used to construct broadcast encryption (BE) [FN93] where there is no limit on the number of receivers. We start with defining dynamic BE, where users can join the system after it is set up. Each user is identified by a consecutive number i.

A broadcast encryption scheme $\mathcal{BE}$ for a symmetric-key encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with key space $\mathcal{K}_{\text{sym}}$, consists of the following four PPT algorithms:

$(bk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: On input a security parameter λ, output a broadcast key bk and a master secret key msk, used to enroll new members in the system.

$sk_i \leftarrow \mathsf{KeyGen}(msk, i)$: On input a master key msk and a member id i, output sk_i, a secret key for member i.

$(hdr, K) \leftarrow \mathsf{Encrypt}(bk, S)$: On input a set $S \subseteq \mathbb{N}$ and a broadcast key bk, output a header hdr and a key $K \in \mathcal{K}_{\text{sym}}$. (A message m is then broadcast as $(S, hdr, \mathsf{enc}(K, m))$.)

$K \leftarrow \mathsf{Decrypt}(i, sk_i, S, hdr)$: On input a member id i and an associated secret key sk_i, a set $S \subseteq \mathbb{N}$ and a header hdr, if $i \in S$ then output a symmetric key $K \in \mathcal{K}_{\text{sym}}$. (Given a broadcast (S, hdr, C), compute $m \leftarrow \mathsf{dec}(K, C)$.)

Like Boneh and Waters [BW13], whose construction we build on, we will construct a *secret-key* BE scheme, where bk must only be known to the broadcaster.

Correctness of a BE scheme is defined as follows: for all $S \subseteq \mathbb{N}$, $i \in S$, all $(bk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_i \leftarrow \mathsf{KeyGen}(msk, i)$ and $(hdr, K) \leftarrow \mathsf{Encrypt}(bk, S)$, we have $K \leftarrow \mathsf{Decrypt}(i, sk_i, S, hdr)$.

Selective security is defined via the following game $\mathbf{Exp}^{\text{BE-}b}$ for an adversary $\mathcal{A}$:

```
$\underline{\mathbf{Exp}^{\text{BE-}b}_{\mathcal{BE},\mathcal{A}}(\lambda)}$
$(bk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$
$(S^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$(hdr^*, K^*) \leftarrow \mathsf{Encrypt}(bk, S^*)$
If $b = 0$ then $K^* \leftarrow \mathcal{K}_{\text{sym}}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{key}(\cdot),\mathsf{encrypt}(\cdot)}(st, (hdr^*, K^*))$; Return $b'$
```

```
$\underline{\textbf{Oracle } \mathsf{key}(i):}$
If $i \in S^*$
  Return $\perp$
$sk_i \leftarrow \mathsf{KeyGen}(msk, i)$
Return $sk_i$
```

```
$\underline{\textbf{Oracle } \mathsf{encrypt}(S):}$
If $S = S^*$
  Return $\perp$
$(hdr, K) \leftarrow \mathsf{Encrypt}(bk, S)$
Return $(hdr, K)$
```

We say that $\mathcal{BE}$ is secure if $\mathsf{Adv}^{\text{BE}}_{\mathcal{BE},\mathcal{A}}(\lambda) := \big|\Pr[\mathbf{Exp}^{\text{BE-0}}_{\mathcal{BE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{BE-1}}_{\mathcal{BE},\mathcal{A}}(\lambda) = 1]\big| \leq \mathsf{negl}(\lambda)$.

BE from constrained PRFs for unbounded inputs. Let $(\mathsf{enc}, \mathsf{dec})$ be a symmetric encryption scheme with key space $\mathcal{K}_{\text{sym}}$. Let $\mathcal{F} = \{\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ be a constrained PRF with input space $\mathcal{X} = \{0,1\}^*$ and range $\mathcal{Y} = \mathcal{K}_{\text{sym}}$, for which constrained keys k_i for the following set can be computed:

$$S_i := \{x \in \{0,1\}^* \mid x_i = 1\} \ . \tag{8}$$

(As S_i can be decided by a polynomial-time Turing machine, our construction from Section 3.2 can be used.) Then we define a broadcast encryption scheme $\mathcal{BE}$ with optimal ciphertext length (that is, the header is empty: $hdr = \emptyset$) as follows:

- $\mathsf{Setup}(1^\lambda)$: Generate $k \leftarrow \mathsf{F.Smp}(1^\lambda)$ and return $bk := k$, $msk := k$.
- $\mathsf{KeyGen}(msk, i)$: Return $k_i \leftarrow \mathsf{F.Constr}(msk, S_i)$ with S_i as in (8).
- $\mathsf{Encrypt}(bk, S)$: Let $x_S \in \{0,1\}^*$ be the characteristic vector of S, compute $K \leftarrow \mathsf{F.Eval}(bk, x_S)$ and output $(\emptyset, K)$
- $\mathsf{Decrypt}(i, sk_i, S, hdr)$: With x_S as above, output $K \leftarrow \mathsf{F.Eval}(sk_i, x_S)$.

Correctness of $\mathcal{BE}$ follows from correctness of $\mathcal{F}$; security follows by reduction to selective pseudorandomness of $\mathcal{F}$. Let $\mathcal{A}$ be a PPT adversary that breaks security of $\mathcal{BE}$; then we construct a PPT algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks $\mathcal{F}$ with the same probability:

```
$\underline{\mathcal{B}_1(1^\lambda)}$
  – $(S^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
  – Let $x^*$ be the charac-
    teristic string of $S^*$.
  – Return $(x^*, st_{\mathcal{A}})$.
```

```
$\underline{\mathcal{B}_2^{\mathsf{constr}(\cdot),\mathsf{eval}(\cdot)}(st, K^*)}$
  – $b' \leftarrow \mathcal{A}_2^{\mathsf{key}(\cdot),\mathsf{encrypt}(\cdot)}(st, (\emptyset, K^*))$;
    – simulate $\mathsf{key}(i)$: define $S_i$ as in (8), query $k_i \leftarrow \mathsf{constr}(S_i)$; reply $k_i$;
    – simulate $\mathsf{encrypt}(S)$: define $x_S \in \{0,1\}^*$ as the characteristic vector
      of $S$; query $K \leftarrow \mathsf{eval}(x_S)$; reply $(\emptyset, K)$.
  – Return $b'$.
```

By construction, we have $\mathbf{Exp}^{(\emptyset,\{\mathsf{constr},\mathsf{eval}\}),b}_{\mathcal{F},\mathcal{B}} = \mathbf{Exp}^{\text{BE-}b}_{\mathcal{BE},\mathcal{A}}$, which proves the claim.

B Complementary Definitions of Used Primitives

B.1 Puncturable PRFs

Definition 9 (Puncturable PRFs [SW14])**.** *A family of PRFs $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ is called puncturable if it is constrainable for sets $\{0,1\}^n \setminus T$, where $T \subseteq \{0,1\}^n$ is of polynomial size. $\mathcal{F}_\lambda$ is (selectively) pseudorandom if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in $\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{F},\mathcal{A}}(\lambda)$, defined in Figure 3, we have*

$$\mathsf{Adv}^{PCT}_{\mathcal{F},\mathcal{A}}(\lambda) := \big|\Pr[\mathbf{Exp}^{PCT\text{-}0}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{PCT\text{-}1}_{\mathcal{F},\mathcal{A}}(\lambda) = 1]\big| \leq \mathsf{negl}(\lambda) \ .$$

```
$\underline{\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{F},\mathcal{A}}(\lambda):}$

$(x^*, T, st) \leftarrow \mathcal{A}_1(1^\lambda)$
If $x^* \notin T$, then abort
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$
$k_{\overline{T}} \leftarrow \mathsf{F.Constr}(k, \{0,1\}^n \setminus T)$
If $b = 1$, $y := \mathsf{F.Eval}(k, x^*)$, else $y \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{eval}(\cdot)}(st, k_{\overline{T}}, y)$
Return $b'$
```

```
$\underline{\mathbf{Oracle}\ \mathsf{eval}(x):}$

If $x = x^*$, return $\perp$
Return $\mathsf{F.Eval}(k, x)$
```

Figure 3: $\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{F},\mathcal{A}}(\lambda)$: The selective-security game for puncturable PRFs.

B.2 Security of Functional Signatures

A functional signature scheme, as defined in Definition 8 is secure if it has the following properties, where we formalize unforgeability following [BF14], who introduce a similar primitive.

1. Correctness: For all $\lambda \in \mathbb{N}$, all $f \in \mathcal{F}$, $w \in \mathcal{D}_f$, $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\sigma \leftarrow \mathsf{Sign}(f, \mathsf{sk}_f, w)$, we have $\mathsf{Verify}(\mathsf{mvk}, f(w), \sigma) = 1$.

2. Unforgeability: For every PPT adversary $\mathcal{A}$, with $\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{A}}(\lambda)$ defined in Figure 4, we have:
$$\Pr[\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{A}}(\lambda) = 1] \le \mathsf{negl}(\lambda) \ .$$

3. Function Privacy: For every PPT adversary $\mathcal{A}$, with $\mathbf{Exp}^{\mathsf{priv}\text{-}b}_{\mathcal{A}}(\lambda)$ defined in Figure 5, we have:
$$\big|\Pr[\mathbf{Exp}^{\mathsf{priv}\text{-}0}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{priv}\text{-}1}_{\mathcal{A}}(\lambda) = 1]\big| \le \mathsf{negl}(\lambda) \ .$$

4. Succinctness: There exists a polynomial $s(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$, $w \in \mathcal{D}_f$, $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\sigma \leftarrow \mathsf{Sign}(f, \mathsf{sk}_f, w)$, we have $|\sigma| \le s(\lambda, |f(w)|)$. (Thus the signature size is independent of $|w|$ and $|f|$, the length of the description of f.

```
$\underline{\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{FS},\mathcal{A}}(\lambda):}$

$\ell := 0$; $K := \emptyset$
    //$K[j][1]$ holds $(f, i)$; $K[j][2]$ holds $\mathsf{sk}^i_f$
    //$K[j][3]$ holds signed $m$'s
    //$K[j][4] = 1$ if $\mathcal{A}$ obtained $\mathsf{sk}^i_f$
$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{key}(\cdot,\cdot),\mathsf{sign}(\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{mvk})$
If $\mathsf{Verify}(\mathsf{mvk}, m^*, \sigma^*) = 0$, return 0.
For $j = 1, \ldots, \ell$ do
  If $m^* \in K[j][3]$, return 0
  $(f, i) := K[v][1]$
  If $K[v][4] = 1$ and $m^* \in \mathcal{R}_f$, return 0
Return 1
```

```
$\underline{\mathbf{Oracle}\ \mathsf{key}(f, i):}$

For $j = 1, \ldots, \ell$ do
  If $K[j][1] = (f, i)$
      $K[j][4] := 1$
      Return $K[j][2]$
$\mathsf{sk}^i_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$
$\ell := \ell + 1$
$K[\ell][1] := (f, i)$
$K[\ell][2] := \mathsf{sk}^i_f$
$K[\ell][4] := 1$
Return $\mathsf{sk}^i_f$
```

```
$\underline{\mathbf{Oracle}\ \mathsf{sign}(f, i, w):}$

$\mathsf{found} := 0$; $j := 0$
While $\mathsf{found} = 0 \wedge j < \ell$ do
  $j := j + 1$
  If $K[j][1] = (f, i)$
      $\mathsf{sk}^i_f := K[j][2]$; $\mathsf{found} := 1$
If $\mathsf{found} = 0$
  $\mathsf{sk}^i_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$
  $\ell := \ell + 1$; $j := \ell$
  $K[j][1] := (f, i)$; $K[j][2] := \mathsf{sk}^i_f$
$K[j][3] := K[j][3] \cup \{f(w)\}$
Return $\mathsf{Sign}(f, \mathsf{sk}^i_f, w)$
```

Figure 4: $\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{FS},\mathcal{A}}(\lambda)$: The unforgeability game for functional signatures.

```
$\underline{\mathbf{Exp}^{\mathsf{priv}\text{-}b}_{\mathcal{FS},\mathcal{A}}(\lambda)}$

$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(f_0, st_1) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{msk}, \mathsf{mvk})$
$\mathsf{sk}_{f_0} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_0)$
$(f_1, st_2) \leftarrow \mathcal{A}_2(st_1, \mathsf{sk}_{f_0})$
If $|f_0| \neq |f_1|$, return 0
$\mathsf{sk}_{f_1} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_1)$
$(w_0, w_1, st_3) \leftarrow \mathcal{A}_3(st_2, \mathsf{sk}_{f_1})$
If $|w_0| \neq |w_1| \vee f_0(w_0) \neq f_1(w_1)$
    Return 0
$\sigma_b \leftarrow \mathsf{Sign}(f_b, \mathsf{sk}_{f_b}, w_b)$ // A signature on $f_b(w_b)$
$b' \leftarrow \mathcal{A}_4(st_3, \sigma_b)$
Return $b'$
```

Figure 5: $\mathbf{Exp}^{\mathsf{priv}\text{-}b}_{\mathcal{FS},\mathcal{A}}(\lambda)$: The function privacy game for functional signatures.

C Proofs

C.1 Proof of Proposition 1

Assume towards contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that distinguishes $\mathbf{Exp}^{b,(0)}(\lambda)$ and $\mathbf{Exp}^{b,(1)}(\lambda)$ with non-negligible probability, i.e., there exists a polynomial $p(\cdot)$ such that for infinitely many λ:

$$\big|\Pr[\mathbf{Exp}^{b,(0)}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{b,(1)}_{\mathcal{A}}(\lambda) = 1]\big| \geq \frac{1}{p(\lambda)} \ .$$

Then we use $\mathcal{A}$ to construct a series of PPT adversaries $\mathcal{D}^{(i)}$, $i = 1, \ldots, q$, one of which breaks function privacy of $\mathcal{FS}$ with non-negligible probability. We construct a series of hybrids between $\mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(1)}$ as follows. Let $q = q(\lambda)$ be a polynomial upper bound on the total number of constraining queries $\mathcal{A}$ makes. Define the i-th hybrid $\mathbf{Exp}^{b,(0,i)}$ like $\mathbf{Exp}^{b,(0)}$, except that the first i constraining queries are answered by using the signing key $\mathsf{sk}_{f_{x^*}}$, and all remaining queries are answered by using the signing key sk_{f_I}. By construction, we have $\mathbf{Exp}^{b,(0,0)} = \mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(0,q)} = \mathbf{Exp}^{b,(1)}$.

We use $\mathcal{A}$ to construct a PPT adversary $\mathcal{D}^{(i)}$ which runs in the function-privacy game $\mathbf{Exp}^{\mathsf{priv}\text{-}d}_{\mathcal{FS},\mathcal{D}^{(i)}}(\lambda)$ of $\mathcal{FS}$ (cf. Figure 5) and simulates $\mathbf{Exp}^{b,(0,i-1)}$ if $\mathcal{D}^{(i)}$'s challenger's bit $d = 0$ and $\mathbf{Exp}^{b,(0,i)}$ if $d = 1$.

$\underline{\mathcal{D}^{(i)}_1(\lambda, \mathsf{msk}, \mathsf{mvk})}$

- $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $H \leftarrow \mathsf{H.Smp}(1^\lambda)$.
- $\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$.
- $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.
- $\widetilde{P} \leftarrow e\mathcal{O}(1^\lambda, P_{H,\mathsf{crs},\mathsf{mvk},k})$ with $P_{H,\mathsf{crs},\mathsf{mvk},k}$ defined in (5).
- Let f_I and f_{x^*} as defined in (3) and (4).
- Set $\mathsf{pp} := (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P})$, $st := (x^*, st_{\mathcal{A}}, \mathsf{pp}, k, f_I, f_{x^*})$.
- Return $(f_0 := f_I, st)$, where f_I is padded to be of length $|f_{x^*}|$.

$\underline{\mathcal{D}^{(i)}_2(st, \mathsf{sk}_{f_I})}$

- Return $(f_1 := f_{x^*}, st' = (st, \mathsf{sk}_{f_I}))$.

```
$\underline{\mathcal{D}_3^{(i)}(st, \mathsf{sk}_{f_{x^*}})}$
    – If $b = 1$ then $y^* := \mathsf{PF.Eval}(k, H(x^*))$; otherwise $y^* \leftarrow \mathcal{Y}$.
    – $b' \leftarrow \mathcal{A}_2^{\mathsf{constr}(\cdot),\mathsf{eval}(\cdot)}(st_{\mathcal{A}}, y^*)$;
        – simulate $\mathsf{eval}(x)$:
            if $x = x^*$, reply $\perp$; else reply $y := \mathsf{PF.Eval}(k, H(x))$;
        – simulate $\mathsf{constr}(M)$:
            if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$; else do the following:
            – first $i-1$ queries: compute $\sigma \leftarrow \mathsf{FS.Sign}(f_{x^*}, \mathsf{sk}_{f_{x^*}}, M)$; reply $k_M := (M, \sigma, \mathsf{pp})$.
            – $i$-th query $M$: return $(m_0, m_1) = (M, M)$ to own challenger.

$\underline{\mathcal{D}_4^{(i)}(st, \sigma_c)}$   // $\sigma_c$ is either a signature under $\mathsf{sk}_{f_I}$ or under $\mathsf{sk}_{f_{x^*}}$
    – Finish the constr query reply for $\mathcal{A}_2$ with $(M, \sigma_c, \mathsf{pp})$.
    – Simulate eval queries like $\mathcal{D}_3^{(i)}$.
    – Simulate further constr queries:
        if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$; else $\sigma \leftarrow \mathsf{FS.Sign}(I, \mathsf{sk}_{f_I}, M)$; reply $k_M := (M, \sigma, \mathsf{pp})$.
    – Output $b'$.
```

If σ_c was generated using the signing key sk_{f_I} then $\mathcal{D}^{(i)}$ simulates $\mathbf{Exp}^{b,(0,i-1)}$ and if $\mathsf{sk}_{f_{x^*}}$ was used then $\mathcal{D}^{(i)}$ simulates $\mathbf{Exp}^{b,(0,i)}$. The only difference between $\mathcal{D}^{(i)}$'s simulation and the actual game is that $\mathcal{D}^{(i)}$ pads the function f_I to match the length of f_{x^*}. This is however oblivious to $\mathcal{A}$, since all $\mathcal{A}$ gets to see are signatures computed using f_I, which, by succinctness of $\mathcal{FS}$, are independent of $|f_I|$. We therefore have

$$\Pr[\mathbf{Exp}^{\mathsf{priv}\text{-}d}_{\mathcal{FS},\mathcal{D}^{(i)}}(\lambda) = 1] = \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0,i-1+d)} = 1] \ . \tag{9}$$

We assumed that

$$\tfrac{1}{p(\lambda)} \leq \big|\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0)}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(1)}(\lambda) = 1]\big| \leq \sum_{i=1}^{q} \big|\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0,i-1)}(\lambda) = 1] - [\mathbf{Exp}_{\mathcal{A}}^{b,(0,i)}(\lambda) = 1]\big| \ .$$

There must thus exist an $i \in \{1, \ldots q\}$ such that for infinitely many λ's:

$$\tfrac{1}{q(\lambda)\cdot p(\lambda)} \leq \big|\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0,i-1)}(\lambda) = 1] - [\mathbf{Exp}_{\mathcal{A}}^{b,(0,i)}(\lambda) = 1]\big| \overset{(9)}{=} \big|\Pr[\mathbf{Exp}^{\mathsf{priv}\text{-}0}_{\mathcal{D}^{(i)}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{priv}\text{-}1}_{\mathcal{D}^{(i)}}(\lambda) = 1]\big| \ .$$

This contradicts function privacy of the functional-signature scheme, and we conclude that

$$\big|\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0)}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(1)}(\lambda) = 1]\big| \leq \mathsf{negl}(\lambda) \ .$$

C.2 Proof of Proposition 2

Proof. Assume towards contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that distinguishes $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ with non-negligible probability, i.e., there exists a polynomial $q(\cdot)$ such that for infinitely many λ,

$$\big|\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(1)}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(2)}(\lambda) = 1]\big| \geq \frac{1}{q(\lambda)} \ . \tag{10}$$

Then we construct $\mathcal{B}$, that distinguishes $e\mathcal{O}$-obfuscations with auxiliary input distributed according to a PPT sampler $(P_k, P_{k_{x^*}}, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda)$, defined as follows:

```
Sampler(1^λ)
  – $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
  – $H \leftarrow \mathsf{H.Smp}(1^\lambda)$, and set $h^* = H(x^*)$.
  – $\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$.
  – $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$.
  – $\mathsf{sk}_{f_{x^*}} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_{x^*})$, with $f_{x^*}$ as defined in (4).
  – $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.
  – $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\})$.
  – Construct $P_0 := P_{H,\mathsf{crs},\mathsf{mvk},k}$ and $P_1 := P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ as defined in (5).
  – Set $\mathsf{aux} = (\mathsf{mvk}, x^*, st_{\mathcal{A}}, H, \mathsf{crs}, \mathsf{sk}_{f_{x^*}}, k)$.
  – Return $(P_0, P_1, \mathsf{aux})$.
```

We then define an algorithm $\mathcal{B}$, which is run on the output of Sampler, that can distinguish obfuscations of P_0 and P_1.

```
B(1^λ, P̃_c, P_0, P_1, aux)
  – $\mathsf{pp} := (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P}_c)$.
  – If $b = 1$ then $y^* := \mathsf{PF.Eval}(k, H(x^*))$; otherwise $y^* \leftarrow \mathcal{Y}$.
  – $b' \leftarrow \mathcal{A}_2^{\mathsf{constr}(\cdot),\mathsf{eval}(\cdot)}(st_{\mathcal{A}}, y^*)$;
      – simulate $\mathsf{constr}(M)$:
          if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$;
          else compute $\sigma \leftarrow \mathsf{FS.Sign}(f_{x^*}, \mathsf{sk}_{f_{x^*}}, M)$; reply $k_M := (M, \sigma, \mathsf{pp})$;
      – simulate $\mathsf{eval}(x)$:
          if $x = x^*$, reply $\perp$;  else reply $y := \mathsf{PF.Eval}(k, H(x))$.
  – Output $b'$.
```

If $\widetilde{P}_c$ is an obfuscation of P_0 then Sampler and $\mathcal{B}$ together simulate $\mathbf{Exp}^{b,(1)}$ for $\mathcal{A}$, if it is an obfuscation of P_1 then they simulate $\mathbf{Exp}^{b,(2)}$ for $\mathcal{A}$. We thus have for $c = 0, 1$ and all $\lambda \in \mathbb{N}$:

$$\Pr\left[(P_0, P_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda); \widetilde{P}_c \leftarrow e\mathcal{O}(1^\lambda, P_c) : \mathcal{B}(1^\lambda, P_0, P_1, \widetilde{P}_c, \mathsf{aux}) = 1\right] \\ = \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(c+1)}(\lambda) = 1] \ .$$

Thus,

$$\begin{aligned}
&\Pr\left[(P_0, P_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda); c \leftarrow \{0,1\}; \widetilde{P}_c \leftarrow e\mathcal{O}(1^\lambda, P_c) : \mathcal{B}(1^\lambda, P_0, P_1, \widetilde{P}_c, \mathsf{aux}) = c\right] \\
&\quad = \frac{1}{2}\Big(1 - \Pr\left[(P_0, P_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda); \widetilde{P}_0 \leftarrow e\mathcal{O}(1^\lambda, P_0) : \mathcal{B}(1^\lambda, P_0, P_1, \widetilde{P}_0, \mathsf{aux}) = 1\right] + \\
&\qquad\qquad \Pr\left[(P_0, P_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda); \widetilde{P}_1 \leftarrow e\mathcal{O}(1^\lambda, P_1) : \mathcal{B}(1^\lambda, P_0, P_1, \widetilde{P}_1, \mathsf{aux}) = 1\right]\Big) \\
&\quad = \frac{1}{2} + \frac{1}{2}\Big(\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(1)}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(2)}(\lambda) = 1]\Big) \ \geq \ \frac{1}{2} + \frac{q(\lambda)}{2}
\end{aligned}$$

for infinitely many λ's, by Eq. (10). (The last inequality only holds of the difference of probabilities in that line is positive. This is however w.l.o.g.: if $\mathcal{A}$ was such that the difference was negative in (10) then we would define $\mathcal{B}$ to output $1 - b'$.)

By security of $e\mathcal{O}$ (cf. Eq. (1) in Definition 5), there exists a PPT extractor $\mathcal{E}_{\mathcal{B}}$, which when given (P_0, P_1, aux) computed by Sampler finds a differing input $\chi := (M, h, \pi, \sigma)$. That is, for some polynomial $p(\cdot)$, we have for infinitely many λ:

$$\Pr\left[\chi \leftarrow \mathcal{E}_{\mathcal{B}}\big(1^\lambda, P_0, P_1, \mathsf{aux} = (\mathsf{mvk}, x^*, st_{\mathcal{A}}, H, \mathsf{crs}, \mathsf{sk}_{f_{x^*}}, k)\big) : P_0(\chi) \neq P_1(\chi)\right] \geq \frac{1}{p(\lambda)} \ . \qquad (11)$$

Let $\hat{\chi} = (\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$ be a differing input output by $\mathcal{E}_{\mathcal{B}}$. Recall that $\hat{\sigma}$ is a signature on a TM $\hat{M}$, and $\hat{\pi}$ is a short proof of $\hat{\eta} = (H, \hat{M}, \hat{h}) \in L_{legit}$, i.e., a short proof of knowledge of a witness x such that $\hat{M}(x) = 1$ and $H(x) = \hat{h}$. By the definitions of $P_0 := P_{H,\mathsf{crs},\mathsf{mvk},k}$ and $P_1 := P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ (cf. Eq. (5)), the following two conditions must hold.

$\mathsf{condition}(1)$: Both $\mathsf{SNARK.Verify}(\mathsf{crs}, (H, \hat{M}, \hat{h}), \hat{\pi}) = 1$ and $\mathsf{FS.Verify}(\mathsf{mvk}, \hat{M}, \hat{\sigma}) = 1$ hold, for otherwise both P_0 and P_1 output $\perp$, and

$\mathsf{condition}(2)$: $\hat{h} = h^* = H(x^*)$, for otherwise P_0 outputs $\mathsf{PF.Eval}(k, \hat{h})$ and P_1 outputs $\mathsf{PF.Eval}(k_{h^*}, \hat{h})$, which are equal by the correctness of puncturing.

Next we will show that moreover any such output must satisfy $\hat{M}(x^*) = 0$. Intuitively, this is the case because $\mathcal{E}_{\mathcal{B}}$ gets a signing key $\mathsf{sk}_{f_{x^*}}$, with which it can only sign machines M with $M(x^*) = 0$. So if it outputs $\hat{M}$ with $\hat{M}(x^*) = 1$ then $(\hat{M}, \hat{\sigma})$, which by $\mathsf{condition}(1)$ is a valid signature, is a forgery. We make this formal in the following claim.

Claim 1. *Let* $\mathsf{Sampler}$ *be as defined above and* $\mathcal{E}_{\mathcal{B}}$ *be the* $e\mathcal{O}$ *extractor guaranteed by Eq.* (11) *and* $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$ *its output. If* $\mathcal{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$ *is a secure functional signature scheme then* $\hat{M}(x^*) = 0$.

Proof. Formally, assume towards a contradiction that $\hat{M}(x^*) = 1$. Then we construct a PPT adversary $\mathcal{A}_{\mathsf{forg}}$ against $\mathcal{FS}$, such that

$$\Pr[\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{FS},\mathcal{A}_{\mathsf{forg}}}(\lambda) = 1] \geq \frac{1}{p(\lambda)} \ .$$

with $\mathbf{Exp}^{\mathsf{unforg}}(\lambda)$ defined in Figure 4. $\mathcal{A}_{\mathsf{forg}}$ behaves like $\mathsf{Sampler}$ but uses its input mvk and obtains the key $\mathsf{sk}_{f_{x^*}}$ from its key oracle, and then runs $\mathcal{E}_{\mathcal{B}}$. (Note that the oracle is called on inputs (f, i); we arbitrarily set $i := 1$.)

```
$\underline{\mathcal{A}_{\mathsf{forg}}^{\mathsf{key}(\cdot,\cdot),\mathsf{sign}(\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{mvk})}$
  – $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
  – $H \leftarrow \mathsf{H.Smp}(1^\lambda)$, and set $h^* = H(x^*)$.
  – $\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$.
  – **Query** $\mathsf{key}(\cdot,\cdot)$ on $(f_{x^*}, 1)$ to obtain $\mathsf{sk}_{f_{x^*}}$ for $f_{x^*}$ as defined in (4).
  – $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.
  – $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\})$.
  – Construct $P_0 := P_{H,\mathsf{crs},\mathsf{mvk},k}$ and $P_1 := P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ as defined in (5).
  – Set $\mathsf{aux} = (\mathsf{mvk}, x^*, st_{\mathcal{A}}, H, \mathsf{crs}, \mathsf{sk}_{f_{x^*}}, k)$.
  – $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma}) \leftarrow \mathcal{E}_{\mathcal{B}}(1^\lambda, P_0, P_1, \mathsf{aux})$.
  – Output $(\hat{M}, \hat{\sigma})$.
```

By $\mathsf{condition}(1)$, $(\hat{M}, \hat{\sigma})$ satisfies $\mathsf{FS.Verify}(\mathsf{mvk}, \hat{M}, \hat{\sigma}) = 1$. Furthermore, $\mathcal{A}_{\mathsf{forg}}$ asked for a single signing key $\mathsf{sk}_{f_{x^*}}$, and no signing queries. So, if $\hat{M}(x^*) = 1$, then by definition of f_{x^*}, $\hat{M} \notin \mathcal{R}_{f_{x^*}}$, i.e., not in the range of f_{x^*}, and hence $\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{A}_{\mathsf{forg}}}(\lambda) = 1$. Consequently, $\Pr[\mathbf{Exp}^{\mathsf{unforg}}_{\mathcal{A}_{\mathsf{forg}}}(\lambda) = 1] \geq \frac{1}{p(\lambda)}$, a contradiction to the unforgeability of functional signatures, and therefore $\hat{M}(x^*) = 0$. $\square$

Since the SNARK $\hat{\pi}$ extracted by $\mathcal{E}_{\mathcal{B}}$ is a proof of knowledge, we can extract a witness $\hat{x}$ for it. In order to formally apply item 3. of Definition 7, we first construct a machine $\mathcal{A}_{\mathsf{snrk}}$ that outputs $\hat{\pi}$

together with the statement. $\mathcal{A}_{\mathsf{snrk}}$ simply runs $\mathsf{Sampler}$ and $\mathcal{E}_{\mathcal{B}}$ as defined above, except that it uses crs from its input.

```
$\underline{\mathcal{A}_{\mathsf{snrk}}(\mathsf{crs})}$
    – $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
    – $H \leftarrow \mathsf{H.Smp}(1^\lambda)$, and set $h^* = H(x^*)$.
    – $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$.
    – $\mathsf{sk}_{f_{x^*}} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_{x^*})$ with $f_{x^*}$ defined in (4).
    – $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.
    – $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\})$.
    – Construct $P_0 := P_{H,\mathsf{crs},\mathsf{mvk},k}$ and $P_1 := P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ as defined in (5).
    – Set $\mathsf{aux} = (\mathsf{mvk}, x^*, st_{\mathcal{A}}, H, \mathsf{crs}, \mathsf{sk}_{f_{x^*}}, k)$.
    – $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma}) \leftarrow \mathcal{E}_{\mathcal{B}}(1^\lambda, P_0, P_1, \mathsf{aux})$.
    – Output $(\eta := (H, \hat{M}, \hat{h}), \hat{\pi})$.
```

By the construction of $\mathcal{A}_{\mathsf{snrk}}$, Eq. (11) and $\mathsf{condition}(1)$ we have that

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda);\\ ((H,\hat{M},\hat{h}),\hat{\pi}) \leftarrow \mathcal{A}_{\mathsf{snrk}}(\mathsf{crs})\end{array} : \mathsf{Verify}(\mathsf{crs},(H,\hat{M},\hat{h}),\pi) = 1\right] \geq \frac{1}{p(\lambda)} \quad . \tag{12}$$

Further, since SNARK is an adaptive proof of knowledge, there exists $\mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}$ which extracts a witness, that is:

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda);\\ ((H,\hat{M},\hat{h}),\hat{\pi}) \leftarrow \mathcal{A}_{\mathsf{snrk}}(\mathsf{crs});\ \hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(\mathsf{crs})\end{array} : \begin{array}{c}\mathsf{Verify}(\mathsf{crs},(H,\hat{M},\hat{h}),\pi) = 1\\ \wedge\ ((H,\hat{M},\hat{h}),\hat{x}) \notin R_{legit}\end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

which together with (12) yields:

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda);\\ ((H,\hat{M},\hat{h}),\hat{\pi}) \leftarrow \mathcal{A}_{\mathsf{snrk}}(\mathsf{crs});\ \hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(\mathsf{crs})\end{array} : ((H,\hat{M},\hat{h}),\hat{x}) \in R_{legit}\right] \geq \frac{1}{p(\lambda)} - \mathsf{negl}(\lambda) \ . \tag{13}$$

We now construct an adversary $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ against $\mathcal{H}$ that on input λ, and a uniform H outputs a collision for H: $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ generates a CRS for SNARKs, then runs $\mathcal{A}_{\mathsf{snrk}}(\mathsf{crs})$, but using the hash function H from its input, (the steps marked with '∘') and then runs $\mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}$ to extract a collision:

```
$\underline{\mathcal{A}_{\mathsf{cll\text{-}fnd}}(1^\lambda, H)}$
    – $\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$.
    ∘ $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$; set $h^* = H(x^*)$.
    ∘ $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$.
    ∘ $\mathsf{sk}_{f_{x^*}} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_{x^*})$ with $f_{x^*}$ defined in (4).
    ∘ $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.
    ∘ $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\})$.
    ∘ Construct $P_0 := P_{H,\mathsf{crs},\mathsf{mvk},k}$ and $P_1 := P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ as defined in (5).
    ∘ Set $\mathsf{aux} = (\mathsf{mvk}, x^*, st_{\mathcal{A}}, H, \mathsf{crs}, \mathsf{sk}_{f_{x^*}}, k)$.
    ∘ $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma}) \leftarrow \mathcal{E}_{\mathcal{B}}(1^\lambda, P_0, P_1, \mathsf{aux})$.
    – $\hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(\mathsf{crs})$.
    – Output $(\hat{x}, x^*)$ as a collision pair for $H$.
```

By Eq. (13), with non-negligible probability, the values $\hat{M}, \hat{h}, \hat{\pi}$ computed during the execution of $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ satisfy $((H,\hat{M},\hat{h}),\hat{x}) \in R_{legit}$, that is, $\hat{M}(\hat{x}) = 1$ and $\hat{h} = H(\hat{x})$.

By Claim 1, $\hat{M}(x^*) = 0$, and hence $\hat{x} \neq x^*$. By $\mathsf{condition}(2)$, $\hat{h} = H(x^*)$, and hence (x, x^*) is a collision. In particular, the following is non-negligible:

$$\Pr\left[H \leftarrow \mathsf{H.Smp}(1^\lambda); (x_1, x_2) \leftarrow \mathcal{A}_{\mathsf{cll\text{-}fnd}}(1^\lambda, H) :\ x_1 \neq x_2 \wedge H(x_1) = H(x_2)\right] .$$

Therefore we have reached a contradiction to collision resistance of $\mathcal{H}$, and it must be that $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ are computationally indistinguishable, i.e.,

$$\left|\Pr[\mathbf{Exp}^{b,(1)}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{b,(2)}_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda) .$$ □

C.3 Proof of Proposition 3

Proof. The only difference between games $\mathbf{Exp}^{b,(2)}$ and $\mathbf{Exp}^{b,(3)}$ is when $\mathcal{A}$ queries $\mathsf{eval}(x)$ with $H(x) = H(x^*)$. Then $\mathbf{Exp}^{b,(3)}$ aborts, while on any other query the oracle eval behaves equivalently in both games, since $H(x) \neq H(x^*)$ implies $\mathsf{PF.Eval}(k_{h^*}, H(x)) = \mathsf{PF.Eval}(k, H(x))$.

We can therefore build an adversary $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ against the hash function family $\mathcal{H}$ that on input $(1^\lambda, H)$ simulates $\mathbf{Exp}^{b,(3)}$ (except that it uses H instead of sampling one) until in an oracle query $\mathsf{eval}(x)$ the game would abort. $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ then outputs (x^*, x), which is a collision precisely when the game would have aborted. □

C.4 Proof of Proposition 4

Proof. Assume towards a contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a polynomial $p(\cdot)$ such that for infinitely many λ,

$$\left|\Pr[\mathbf{Exp}^{0,(3)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{1,(3)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1]\right| \geq \frac{1}{p(\lambda)} .$$

Then we construct a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ playing $\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{PF},\mathcal{B}}(\lambda)$, the selective-security game of $\mathcal{PF}$ (cf. Figure 3, p. 19) as follows. (Note that $\mathcal{B}_2$ does not use its $\mathsf{eval}(\cdot)$ oracle.)

```
$\underline{\mathcal{B}_1(1^\lambda)}$
    – $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
    – $H \leftarrow \mathsf{H.Smp}(1^\lambda)$, and set $h^* = H(x^*)$.
    – Return $(h^*, T := \{h^*\}, st := (H, x^*, st_{\mathcal{A}}))$.

$\underline{\mathcal{B}_2^{\mathsf{eval}(\cdot)}(st, k_{h^*}, y^*)}$    // $y^*$ is either $\mathsf{PF.Eval}(k, H(x^*))$ or random
    – $\mathsf{crs} \leftarrow \mathsf{SNARK.Gen}(1^\lambda)$.
    – $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$.
    – $\mathsf{sk}_{f_{x^*}} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_{x^*})$ with $f_{x^*}$ defined in (4).
    – $\widetilde{P} \leftarrow e\mathcal{O}(1^\lambda, P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}})$ with $P_{H,\mathsf{crs},\mathsf{mvk},k_{h^*}}$ defined in (5).
    – Set $\mathsf{pp} := (H, \mathsf{crs}, \mathsf{mvk}, \widetilde{P})$.
    – $b' \leftarrow \mathcal{A}_2^{\mathsf{constr}(\cdot),\mathsf{eval}(\cdot)}(st_{\mathcal{A}}, y^*)$.
        – simulate $\mathsf{constr}(M)$:
            if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$;
            else compute $\sigma \leftarrow \mathsf{FS.Sign}(f_{x^*}, \mathsf{sk}_{f_{x^*}}, M)$; reply $k_M := (M, \sigma, \mathsf{pp})$;
        – simulate $\mathsf{eval}(x)$:
            if $x = x^*$, reply $\perp$; if $H(x) = H(x^*)$ then abort;
            else reply $y := \mathsf{PF.Eval}(k_{h^*}, H(x))$.
    – Output $b'$.
```

By construction $\Pr[\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{PF},\mathcal{B}}(\lambda) = 1] = \Pr[\mathbf{Exp}^{b,(3)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1]$, and therefore

$$\left|\Pr[\mathbf{Exp}^{\text{PCT-0}}_{\mathcal{PF},\mathcal{B}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{PCT-1}}_{\mathcal{PF},\mathcal{B}}(\lambda) = 1]\right| \geq \frac{1}{p(\lambda)} \tag{14}$$

for infinitely many λ. This contradicts the selective security of PF, and we conclude that

$$\left|\Pr[\mathbf{Exp}^{0,(3)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{1,(3)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda) \ . \qquad \square$$