Constrained PRFs for Unbounded Inputs with Short Keys

Hamza Abusalah [1,*] Georg Fuchsbauer [2]

[1] IST Austria
habusalah@ist.ac.at

[2] ENS, CNRS, INRIA and PSL Research University, Paris, France
georg.fuchsbauer@ens.fr

Abstract

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

Abusalah et al. recently constructed the first constrained PRF for inputs of arbitrary length whose sets S are decided by Turing machines. They use their CPRF to build broadcast encryption and the first ID-based non-interactive key exchange for an unbounded number of users. Their constrained keys are obfuscated circuits and are therefore large.

In this work we drastically reduce the key size and define a constrained key for a Turing machine M as a short signature on M. For this, we introduce a new signature primitive with constrained signing keys that let one only sign certain messages, while forging a signature on others is hard even when knowing the coins for key generation.

Keywords: Constrained PRFs, unbounded inputs

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 decide whether $\mathcal{O}(\cdot)$ is $\mathsf{F}(k,\cdot)$ with a random key $k\in\mathcal{K}$, or whether $\mathcal{O}(\cdot)$ is a uniformly random function $\mathcal{X}\to\mathcal{Y}$. A PRF F is called *constrained* [BW13, KPTZ13, BGI14] for a predicate family $\mathcal{P}$ if additionally there exists a PPT constraining algorithm $k_p\leftarrow\mathsf{F}.\mathsf{Constr}(k,p)$ that, on input a key k and a predicate $p\colon\mathcal{X}\to\{0,1\}$ specifying a subset $S_p=\{x\in\mathcal{X}\mid p(x)=1\}$ of potentially exponential size, derives a constrained key k_p. The latter allows computing $\mathsf{F}(k,x)$ on all $x\in S_p$, while even given keys for $p_1,\dots,p_\ell$, values $\mathsf{F}(k,x)$ for $x\notin\bigcup_i S_{p_i}$ still look random. Note that if all sets S_p are polynomial-size, a simple solution would be to set $k_p:=\{\mathsf{F}(k,x)\,|\,x\in S_p\}$, which would achieve the desired security. The challenge is to have short keys for potentially big sets.

The simplest type of constrained PRFs (CPRF) are puncturable PRFs [SW14], where for any input $x\in\{0,1\}^*$ one can derive a key k_{x^*} that allows evaluation everywhere except on x^*, whose image is pseudorandom even given k_{x^*}. The most general CPRF is one that is constrained w.r.t. a Turing-machine (TM) predicate family $\mathcal{M}$, where $M\in\mathcal{M}$ defines a subset of inputs of arbitrary

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

length $S_M = \{x \in \{0,1\}^* \mid M(x) = 1\}$. In a TM-constrained PRF a constrained key k_M can be derived for any set S_M defined by a TM M.

Abusalah et al. (AFP) [AFP16] construct a (selectively secure) TM-constrained PRF and show how to use it to construct broadcast encryption (BE) [FN94, BWZ14] where there is no a priori bound on the number of possible recipients and the first identity-based non-interactive key-exchange scheme [SOK00, FHPS13, BW13] with no a priori bound on the number of parties that agree on a key.

The main shortcoming of their construction is that a constrained key k_M for a TM M is an obfuscated circuit and is therefore not short, but typically huge. This translates to huge user keys in the BE and ID-NIKE schemes built from their CPRF. In this paper we overcome this and reduce the key size drastically defining a constrained key k_M for M as simply a signature on M.

TM-constrained PRFs with short keys. The AFP TM-constrained PRF in [AFP16] is built from puncturable PRFs, succinct non-interactive arguments of knowledge (SNARKs), which let one prove knowledge of an NP witness via a short proof, collision-resistant hashing and public-coin differing-input obfuscation ($di\mathcal{O}$) [IPS15]. The latter lets one *obfuscate* programs, so that one can only distinguish obfuscations of two equal-size programs if one knows an input on which those programs' outputs differ. Moreover, if for two programs it is hard to find such a differing input, *even when knowing the coins used to construct the programs*, then their obfuscations are indistinguishable.

Relying on essentially the same assumptions, we enhance the AFP construction, making the the constrained keys short. Let us look at their CPRF F first, which is defined as $\mathsf{F}(k,x) := \mathsf{PF}(k, H(x))$, where PF is a puncturable PRF, and H is a hash function (this way they map unbounded inputs to constant-size inputs for a puncturable PRF). A constrained key for a TM M is a $di\mathcal{O}$ obfuscation of the circuit P_M that on input (h,π) outputs $\mathsf{PF}(k,h)$ iff π is a valid SNARK proving the following statement: $(*)$ $\exists\, x : h = H(x) \wedge M(x) = 1$. So P_M only outputs the PRF value if the evaluator knows an input x which satisfies the constraint defined by M.

We also define our TM-CPRF as $\mathsf{F}(k,x) := \mathsf{PF}(k, H(x))$. However at setup, we publish as a public parameter once and for all a $di\mathcal{O}$-obfuscated circuit P that on input (h,π,M,σ) outputs $\mathsf{PF}(k,h)$ iff π is a valid SNARK for $(*)$ and additionally σ is a valid signature on M. A constrained key k_M for a TM M is a signature on M and a party holding $k_M := \sigma$ can generate a SNARK π, as in the AFP construction, and additionally use M, σ to run P to evaluate F.

The intuition behind the construction is simple: in order to evaluate F on x, one needs a signature on a machine M with $M(x) = 1$. And unforgeability of such signatures should guarantee that without a key for such an M the PRF value of x should be pseudorandom. However, actually proving this turns out quite tricky.

In the selective security game for CPRFs, the adversary first announces an input x^* and can then query keys for any sets that do not contain x^*, that is, sets described by TMs M with $M(x^*) = 0$. The adversary then needs to distinguish the PRF image of x^* from random. To argue that $\mathsf{F}(k,x^*)$ is pseudorandom, we replace the circuit P by P^* for which F looks random on x^*, because it uses a key that is punctured at $H(x^*)$. Intuitively, since P is obfuscated, an adversary should not notice the difference. However, to formally prove this we need to construct a sampler that constructs P and P^* and argue that it is hard to find a differing input (h,π,M,σ) even when given the coins to construct the circuits.

One such differing input would be one containing a signature $\hat{\sigma}$ on a machine $\hat{M}$ with $\hat{M}(x^*) = 1$. Since $\hat{\sigma}$ is a key for a set containing x^*, P outputs the PRF value, while P^* does not, as its key is punctured. As the adversary only obtains signatures for M's with $M(x^*) = 0$, $\hat{\sigma}$ intuitively is a forgery. But the sampler that computes P and P^* also computed the signature verification key. So how can it be hard to construct a differing input containing $\hat{\sigma}$ for someone knowing the coins that also define the secret key?

We overcome this seeming contradiction by introducing a new primitive called *functional signatures with obliviously samplable keys* (FSwOSK). To produce the circuits P, P^*, the sampler needs to answer the adversary's key queries, that is, compute signatures on M's with $M(x^*) = 0$. FSwOSK lets the sampler create a verification and signing key, of which the latter only allows to sign such machines M; and security for FSwOSK guarantees that *even when knowing the coins used to set up the keys*, it is hard to create a signature on a message $\hat{M}$ with $\hat{M}(x^*) = 1$.

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 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 $\Pr\left[x_1 \leftarrow \mathcal{X}_1; x_2 \leftarrow \mathcal{X}_2; \ldots : \varphi(x_1, x_2, \ldots) = 1\right]$ 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 probabilistic polynomial-time (PPT) sampler* $\mathsf{F.Smp}$ *and a deterministic PT evaluator* $\mathsf{F.Eval}$ *as follows:*

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

The family $\mathcal{F}_\lambda$ 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 (short) description 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 $\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}$, defined in Fig. 1, with $\mathcal{O}_1 = \emptyset$ and $\mathcal{O}_2 = \{\textsc{Constr}(\cdot), \textsc{Eval}(\cdot)\}$, it holds that*

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

Furthermore, $\mathcal{F}_\lambda$ is adaptively pseudorandom *if the same holds for $\mathcal{O}_1 = \mathcal{O}_2 = \{\textsc{Constr}(\cdot), \textsc{Eval}(\cdot)\}$.*

$\underline{\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}(\lambda):}$

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

$\underline{\textbf{Oracle } \textsc{Constr}(S):}$

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

$\underline{\textbf{Oracle } \textsc{Eval}(x):}$

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: The security game for constrained PRFs

Remark 1. We require $\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}$ of Fig. 1 to be efficient. Thus when sets are described by Turing machines then every machine M queried to Constr must terminate on x^* within a polynomial number of steps T (as the oracle must check whether $S \cap \{x^*\} \neq \emptyset$, that is, $M(x^*) = 1$).

Puncturable PRFs [SW14] are simple constrained PRFs whose domain is $\{0,1\}^n$ for some n and whose constrained keys are for sets $\{\{0,1\}^n \setminus \{x_1, \ldots, x_m\} \mid x_1, \ldots, x_m \in \{0,1\}^n, m = \mathsf{poly}(\lambda)\}$, i.e., a punctured key can evaluate the PRF on all except polynomially many inputs.

Definition 3 (Puncturable PRFs [SW14])**.** *A family $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ of PRFs is called* puncturable *if it is constrainable for sets $\{0,1\}^n \setminus T$, where T 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 Fig. 2, we have*

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

Selectively secure puncturable PRFs are easily obtained from selectively secure prefix-constrained PRFs, which were constructed from the GGM PRF [GGM86] in [BW13, BGI14, KPTZ13]. In this work we only require selectively secure puncturable PRFs.

2.3 Public-Coin Differing-Input Obfuscation

Public-coin differing-input (di) obfuscation guarantees that if for pairs of *publicly* sampled programs it is hard to find an input on which they differ then their obfuscations are computationally indistinguishable. We follow [IPS15] by first defining public-coin di samplers that output programs whose obfuscations are indistinguishable.

Definition 4 (Public-Coin DI Sampler [IPS15])**.** *A non-uniform PPT sampler Samp is a public-coin differing-input sampler for a family of polynomial-size circuits $\mathcal{C}_\lambda$ if the output of Samp is in $\mathcal{C}_\lambda \times \mathcal{C}_\lambda$ and for every non-uniform PPT extractor $\mathcal{E}$ it holds that*

$$\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ (C_0, C_1) := \mathsf{Samp}(1^\lambda, r);\ x \leftarrow \mathcal{E}(1^\lambda, r):\ C_0(x) \neq C_1(x)\right] = \mathsf{negl}(\lambda) \ . \tag{2}$$

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

$(x^*, T, st) \leftarrow \mathcal{A}_1(1^\lambda)$; 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 $y \leftarrow \mathcal{Y}$
Return $b' \leftarrow \mathcal{A}_2^{\textsc{Eval}(\cdot)}(st, k_{\overline{T}}, y)$

$\underline{\textbf{Oracle } \textsc{Eval}(x):}$

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

Figure 2: The selective-security game for puncturable PRFs

Definition 5 (Public-Coin diO [IPS15]). *A uniform PPT algorithm* diO *is a public-coin differing-input 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: $\Pr\left[\widetilde{C} \leftarrow \mathsf{diO}(1^\lambda, C) : \ C(x) = \widetilde{C}(x)\right] = 1$.
- *For every public-coin di sampler* Samp *for a family of poly-size circuits* $\mathcal{C}_\lambda$*, every non-uniform PPT distinguisher* $\mathcal{D}$ *and every* $\lambda \in \mathbb{N}$:

$$\begin{aligned}\big|\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ (C_0, C_1) := \mathsf{Samp}(1^\lambda, r);\ \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C_0) :\ 1 \leftarrow \mathcal{D}(r, \tilde{C})\right] - \\ \Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ (C_0, C_1) := \mathsf{Samp}(1^\lambda, r);\ \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C_1) :\ 1 \leftarrow \mathcal{D}(r, \tilde{C})\right]\big| = \mathsf{negl}(\lambda)\ .\end{aligned} \tag{3}$$

Ishai et al. [IPS15] conjecture that Garg et al.'s [GGH+13] $i\mathcal{O}$ construction satisfies their notion of public-coin $di\mathcal{O}$.

2.4 Non-interactive Proof Systems

An efficient non-interactive proof system in the *common-random-string* (CRS) model for a language $L \in \mathsf{NP}$ consists of PPT prover P and verifier V sharing a uniformly random string *crs*. On input a statement and a witness, P outputs a proof; V, on input a statement and a proof outputs 0 or 1. We require proof systems to be complete (honestly generated proofs verify) and sound (no adversary can produce a a valid proof of a false statement).

A non-interactive proof system is *zero-knowledge* if proofs of true statements reveal nothing beyond their validity. This is formalized by requiring the existence of a PPT simulator $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$ that on input a true statement produces a CRS and a proof that are computationally indistinguishable from real ones.

Definition 6 (NIZK). *A tuple of PPT algorithms* $\mathsf{NIZK} = (\mathsf{G}, \mathsf{P}, \mathsf{V}, \mathsf{S})$ *is a statistically sound non-interactive zero-knowledge (NIZK) proof system in the common-random-string model for* $L \in \mathsf{NP}$ *with witness relation* R *if we have:*

1. *Perfect completeness: For every* (η, w) *such that* $R(\eta, w) = 1$*, it holds that*

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ \pi \leftarrow \mathsf{P}(crs, \eta, w) : \mathsf{V}(crs, \eta, \pi) = 1\right] = 1\ .$$

2. *Statistical soundness:*

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} :\ \exists\,(\eta, \pi)\ \textit{s.t.}\ \eta \notin L\ \wedge\ \mathsf{V}(crs, \eta, \pi) = 1\ \right] = \mathsf{negl}(\lambda)\ . \tag{4}$$

3. *Computational zero-knowledge: For every* (η, w) *such that* $R(\eta, w) = 1$*, and non-uniform PPT adversary* $\mathcal{A}$*, it holds that*

$$\begin{aligned}\big|\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ \pi \leftarrow \mathsf{P}(crs, \eta, w) : \mathcal{A}(crs, \eta, \pi) = 1\right] - \\ \Pr\left[(crs, \tau) \leftarrow \mathsf{S}_1(1^\lambda, \eta);\ \pi \leftarrow \mathsf{S}_2(crs, \tau, \eta) : \mathcal{A}(crs, \eta, \pi) = 1\right]\big| = \mathsf{negl}(\lambda)\ .\end{aligned} \tag{5}$$

A succinct non-interactive argument of knowledge (SNARK) is a computationally sound NI proof-of-knowledge system with universally succinct proofs. A proof for a statement η is *succinct* if its length and verification time are bounded by a fixed polynomial in the statement length $|\eta|$. We define SNARK systems in the common-random-string model following Bitansky et al. [BCCT13, BCC+14, IPS15].

Definition 7 (The Universal Relation $\mathcal{R}_\mathcal{U}$ [BG08]). *The universal relation* $\mathcal{R}_\mathcal{U}$ *is the set of instance-witness pairs of the form* $((M, m, t), w)$ *where* M *is a TM accepting an input-witness pair* (m, w) *within* t *steps. In particular* $|w| \leq t$*.*

Definition 8 (SNARK)**.** *A pair of PPT algorithms* (P, V)*, where* V *is deterministic, is a succinct non-interactive argument of knowledge (SNARK) in the common-random-string model for a language* $\mathcal{L}$ *with witness relation* $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ *if there exist polynomials* p, ℓ, q independent *of* $\mathcal{R}$ *such that the following hold:*

1. *Completeness: For every* $(\eta = (M, m, t), w) \in \mathcal{R}$*, it holds that*

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ \pi \leftarrow \mathsf{P}(crs, \eta, w) :\ \mathsf{V}(crs, \eta, \pi) = 1\right] = 1\ .$$

   Moreover, P *runs in time* $q(\lambda, |\eta|, t)$*.*

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

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ (\eta, \pi) \leftarrow \mathcal{A}(crs) :\ \eta \notin \mathcal{L}\ \wedge\ \mathsf{V}(crs, \eta, \pi) = 1\right] = \mathsf{negl}(\lambda)\ .$$

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

$$\Pr\left[\begin{array}{l} crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} \\ (\eta, \pi) := \mathcal{A}(crs; r);\ w \leftarrow \mathcal{E}_{\mathcal{A}}(1^{\lambda}, crs, r) \end{array} :\ \mathsf{V}(crs, \eta, \pi) = 1\ \wedge\ (\eta, w) \notin \mathcal{R}\right] = \mathsf{negl}(\lambda)\ .$$

4. *Succinctness: For all* $(crs, \eta, w) \in \{0,1\}^{\mathsf{poly}(\lambda)} \times \mathcal{R}$*, the length of an honestly generated proof* $\pi \leftarrow \mathsf{P}(crs, \eta, w)$ *is bounded by* $\ell(\lambda, \log t)$ *and the running time of* $\mathsf{V}(crs, \eta, \pi)$ *is bounded by* $p(\lambda + |\eta|) = p(\lambda + |M| + |m| + \log t)$*.*

Bitansky et al. [BCC+14] provide a construction of SNARKs for $\mathcal{R}_c \subset \mathcal{R}_{\mathcal{U}}$ where $t = |m|^c$ and c is a constant, based on knowledge-of-exponent assumptions [BCCT13] and extractable collision-resistant hash functions (ECRHF) [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. Relying on exponentially hard one-way functions and ECRHF, [BCC+14] provide a SNARK construction for $\mathcal{R}_{\mathcal{U}}$.

2.5 Commitment Schemes

A commitment scheme CS for a message space $\mathcal{M} \not\ni \perp$ consists of the following PPT algorithms: On input 1^{λ}, Setup outputs a commitment key ck; on input ck and a message $m \in \mathcal{M}$, Com outputs a commitment c and an opening d; on input ck, c and d, Open opens c to a message m or $\perp$. Besides correctness (commitments open to the committed message), we require *computational hiding* (no PPT adversary can distinguish commitments to messages of his choice) and *statistical binding* (no unbounded adversary can find some c and two openings d, d', which open c to two different messages, except with negligible probability over the choice of ck).

Definition 9 (Commitment Schemes)**.** *A commitment scheme for a message space* $\mathcal{M} \not\ni \perp$ *is a tuple of PPT algorithms* $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open})$ *where:*

- $ck \leftarrow \mathsf{Setup}(1^{\lambda})$*: On input a security parameter* 1^{λ}*,* Setup *outputs a commitment key* ck*.*
- $(c, d) \leftarrow \mathsf{Com}(ck, m)$*: On input* ck *and* $m \in \mathcal{M}$*,* Com *outputs commitment* c *and opening* d*.*
- $\{m \cup \perp\} \leftarrow \mathsf{Open}(ck, c, d)$*: On input* ck*,* c *and opening* d*,* Open *opens* c *to* $m \in \mathcal{M}$ *or* $\perp$*.*

We require correctness and security:

1. *Correctness:* CS *is correct if for all* $m \in \mathcal{M}$*:*

$$\Pr\left[ck \leftarrow \mathsf{Setup}(1^\lambda);\ (c,d) \leftarrow \mathsf{Com}(ck,m);\ m' \leftarrow \mathsf{Open}(ck,c,d) : m = m'\right] = 1 - \mathsf{negl}(\lambda). \tag{6}$$

2. *Computational hiding:* CS *is computationally hiding if for any PPT algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*:*

$$\left|\Pr\left[ck \leftarrow \mathsf{Setup}(1^\lambda);\ (st, m_0, m_1) \leftarrow \mathcal{A}_1(ck);\ (c_0,d_0) \leftarrow \mathsf{Com}(ck,m_0) :\ \mathcal{A}_2(st,c_0) = 1\right] - \right. \tag{7}$$
$$\left.\Pr\left[ck \leftarrow \mathsf{Setup}(1^\lambda);\ (st, m_0, m_1) \leftarrow \mathcal{A}_1(ck);\ (c_1,d_1) \leftarrow \mathsf{Com}(ck,m_1) :\ \mathcal{A}_2(st,c_1) = 1\right]\right| = \mathsf{negl}(\lambda).$$

3. *Perfect binding:* CS *is perfectly binding if for all any (unbounded)* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Setup}(1^\lambda);\ (c,d,d') \leftarrow \mathcal{A}(ck); \\ m \leftarrow \mathsf{Open}(ck,c,d);\ m' \leftarrow \mathsf{Open}(ck,c,d') \end{array} : \ m \neq m' \wedge m, m' \neq \bot\right] = \mathsf{negl}(\lambda)\ . \tag{8}$$

2.6 Collision-Resistant Hash Functions

A family of hash functions is collision-resistant (CR) if for a uniformly sampled function H it is hard to find two values that map to the same image under H. It is *public-coin* CR if this is hard even when given the coins used to sample H.

Definition 10 (Public-Coin CR Hash Functions [HR04])**.** *A family of (efficiently computable) functions* $\mathcal{H}_\lambda = \{H\colon \{0,1\}^* \to \{0,1\}^n\}$ *with a sampler* Smp*, is public-coin collision-resistant if for every PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; H := \mathsf{Smp}(1^\lambda, r); (x_1,x_2) \leftarrow \mathcal{A}(1^\lambda, r) :\ x_1 \neq x_2\ \wedge\ H(x_1) = H(x_2)\right] = \mathsf{negl}(\lambda)\ .$$

2.7 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.[1] Such a key sk_f enables signing (only) messages in the range of f: running $\mathsf{Sign}(f, sk_f, w)$ produces a signature on $f(w)$.

Definition 11 (Functional Signatures [BGI14])**.** *A functional signature scheme for a message space* $\mathcal{M} \not\ni \bot$ *and a function family* $\mathcal{F}_\lambda = \{f\colon \mathcal{D}_f \to \mathcal{R}_f\}_\lambda$ *with* $\mathcal{R}_f \subseteq \mathcal{M}$ *is a tuple of PPT algorithms* $\mathsf{FS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *where:*

- $(msk, mvk) \leftarrow \mathsf{Setup}(1^\lambda)$*: On input a security parameter* 1^λ*,* Setup *outputs a master signing and verification key.*
- $sk_f \leftarrow \mathsf{KeyGen}(msk, f)$*: On input* msk *and a function* $f \in \mathcal{F}_\lambda$*,* KeyGen *outputs a signing key* sk_f*.*
- $(f(w), \sigma) \leftarrow \mathsf{Sign}(f, sk_f, w)$*: On input* $f \in \mathcal{F}_\lambda$*, a signing key* sk_f *for* f*, and* $w \in \mathcal{D}_f$*,* Sign *outputs a signature on* $f(w) \in \mathcal{M}$*.*
- $b = \mathsf{Verify}(mvk, m, \sigma)$*: On input a master verification key* mvk*, a message* $m \in \mathcal{M}$*, and signature* σ*,* Verify *outputs* $b \in \{0,1\}$*.*

We start with an informal definition of security. A functional signature is *correct* if correctly generated signatures verify; it should additionally satisfy *unforgeability*, *function privacy*, and *succinctness*. *Unforgeability* is formalized via a game in which an adversary is given the verification key and can make queries to a key-generation oracle, $\textsc{Key}(f, i)$, and a signing oracle, $\textsc{Sign}(f, i, m)$, which work as follows:

[1] In [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.

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

$\ell := 0$; $K := \emptyset$
//$K[j][1]$ holds (f, i)
//$K[j][2]$ holds sk^i_f
//$K[j][3]$ holds signed m's
//$K[j][4] = 1$ if $\mathcal{A}$ obtained sk^i_f
$(msk, mvk) \leftarrow \mathsf{Setup}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{KEY}(\cdot,\cdot),\text{SIGN}(\cdot,\cdot,\cdot)}(1^\lambda, mvk)$
If $\mathsf{Verify}(mvk, m^*, \sigma^*) = 0$
  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

$\underline{\textbf{Oracle } \text{KEY}(f, i):}$

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

$\underline{\textbf{Oracle } \text{SIGN}(f, i, w):}$

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

Figure 3: The unforgeability game for functional signatures

- $\text{KEY}(f, i)$: if a signing key for (f, i) has already been generated, return the recorded key; otherwise generate and return a fresh signing key $sk_f \leftarrow \mathsf{FS.KeyGen}(msk, f)$ and record $((f, i), sk_f)$.
- $\text{SIGN}(f, i, w)$: check if there is a record $((f, i), sk^i_f)$ 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 $(f(w), \sigma) \leftarrow \mathsf{Sign}(f, sk^i_f, w)$.

The adversary wins the unforgeability game if it returns a signature on some m^* that was not returned by SIGN and for all f queried to KEY we have $m^* \notin \mathcal{R}_f$.

Function privacy states that signatures neither reveal the function associated to the secret key nor the preimage w used. It is formalized via a game in which an adversary $\mathcal{A}$ is given the master signing key as well as signing keys for two functions f_0, f_1 (of equal description size) of its choice; $\mathcal{A}$ outputs (w_0, w_1) (with $|w_0| = |w_1|$ and $f_0(w_0) = f_1(w_1)$) and receives $\mathsf{Sign}(f_b, sk_{f_b}, w_b)$ for $b \leftarrow \{0, 1\}$, which it has to guess. *Succinctness* requires that the size of a signature is independent of $|w|$ and $|f|$.

Definition 12 (Security of Functional Signatures)**.** *A functional signature scheme* FS*, as defined in Def. 11, is secure if it satisfies the following:*

1. *Correctness: For all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$, $w \in \mathcal{D}_f$, $(msk, mvk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_f \leftarrow \mathsf{KeyGen}(msk, f)$, and $(f(w), \sigma) \leftarrow \mathsf{Sign}(f, sk_f, w)$: $\mathsf{Verify}(mvk, f(w), \sigma) = 1$.*
2. *Unforgeability: For every PPT adversary $\mathcal{A}$ in $\mathbf{Exp}^{\text{unforg}}_{\mathcal{A}}(\lambda)$ defined in Fig. 3, it holds that*
$$\Pr\left[\mathbf{Exp}^{\text{unforg}}_{\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda) \ .$$
3. *Function privacy: For every PPT adversary $\mathcal{A}$ in $\mathbf{Exp}^{\text{priv-}b}_{\mathcal{A}}(\lambda)$ defined in Fig. 4, it holds that*
$$\left|\Pr\left[\mathbf{Exp}^{\text{priv-0}}_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{priv-1}}_{\mathcal{A}}(\lambda) = 1\right]\right| = \mathsf{negl}(\lambda) \ .$$
4. *Succinctness: There exists a polynomial $s(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$, $w \in \mathcal{D}_f$, $(msk, mvk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_f \leftarrow \mathsf{KeyGen}(msk, f)$, $(f(w), \sigma) \leftarrow \mathsf{Sign}(f, sk_f, w)$, we have $|\sigma| \leq s(\lambda, |f(w)|)$. (Thus the signature size is independent of $|w|$ and $|f|$).*

Boyle et al. [BGI14] construct functional signatures based on zero-knowledge SNARKs.

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

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

Figure 4: The function-privacy game for functional signatures

3 Functional Signatures with Obliviously Samplable Keys

We now introduce and construct a new primitive we call *functional signatures with obliviously samplable keys* (FSwOSK), which will be our central tool in order to achieve short keys for CPRF with unbounded inputs. We first extend a (standard) signature scheme by an extra functionality that given a message m allows one to sample a verification key together with a signature on m in an oblivious way. This means that, while the key and the signature look like regularly generated ones, no one can forge a signature on a different message under this key, even when given the coins used to sample the key/signature pair. We call this primitive *signatures with obliviously samplable signatures* (SwOSS) and construct it from one-way functions and NIZK by adapting a signature scheme due to Bellare and Goldwasser [BG90]. We then combine this scheme with SNARKs in order to construct our FSwOSK following the construction of a (standard) functional signature scheme of Boyle et al. [BGI14].

3.1 Signature Schemes with Obliviously Samplable Signatures

Definition 13 (SwOSS)**.** *Let* $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a (standard) signature scheme that is existentially unforgeable under chosen-message attacks (EUF-CMA) with message space* $\mathcal{M} \not\ni \bot$*. We say* S *has obliviously samplable signatures if there exists a PPT algorithm* OSmp *such that:*

- $(\mathsf{vk}, \sigma) \leftarrow \mathsf{OSmp}(1^\lambda, m)$*: On input security parameter* 1^λ *and a message* $m \in \mathcal{M}$*,* OSmp *outputs a verification key* vk *and a signature* σ *on* m*.*

SwOSS S *is secure if it satisfies:*

1. *Indistinguishability: For every PPT algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in* $\mathbf{Exp}^{\text{ind-}b}_{\mathsf{S},\mathcal{A}}(\lambda)$ *defined in Fig. 5, it holds that*
$$\left| \Pr\left[\mathbf{Exp}^{\text{ind-}0}_{\mathsf{S},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-}1}_{\mathsf{S},\mathcal{A}}(\lambda) = 1\right] \right| = \mathsf{negl}(\lambda) \ . \tag{9}$$
2. *Oblivious unforgeability: For every PPT algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in* $\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda)$ *defined in Fig. 6, it holds that*
$$\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda) \ . \tag{10}$$

We now construct a standard signature scheme with obliviously samplable signatures.

Construction 1 (Signatures with Obliviously Samplable Signatures)**.** *Let* $\mathcal{F}_\lambda = \{\mathsf{F} \colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ *be a family of PRFs,* $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open})$ *a perfectly binding computationally hiding commitment scheme for message space* $\mathcal{M}$*, and* $\mathsf{NIZK} = (\mathsf{G}, \mathsf{P}, \mathsf{V}, \mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2))$ *a statistically sound NIZK scheme for*
$$L_\eta := \left\{ (ck, c_0, c_1, y, m) \;\middle|\; \begin{array}{c} \exists\, (k, r) : \big(c_0 = \mathsf{CS.Com}_1(ck, k; r) \wedge y = \mathsf{F}(k, m)\big) \\ \vee\ c_1 = \mathsf{CS.Com}_1(ck, m; r) \end{array} \right\} \tag{11}$$

$\underline{\mathbf{Exp}^{\text{ind-}b}_{\mathsf{S},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
If $b = 0$
  $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$; $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
Else $(\mathsf{vk}, \sigma) \leftarrow \mathsf{OSmp}(1^\lambda, m)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

Figure 5: The oblivious-indistinguishability game

$\underline{\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$; $(\mathsf{vk}, \sigma) \leftarrow \mathsf{OSmp}(1^\lambda, m; r)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}_2(st, r)$
Return 1 iff $m^* \neq m$
  $\wedge\, \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$

Figure 6: The oblivious-unforgeability game

(where Com_1 *denotes the first output of* Com*). Let* $\top \in \mathcal{M}$ *be such that* $\top \notin \mathcal{K}$ *and* $\top \notin \{0,1\}^n$*. Our signatures-with-obliviously-samplable-signatures scheme* $\mathsf{OS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{OSmp})$ *is defined as follows:*

$\underline{(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}$*: On input a security parameter* 1^λ*, compute*

- $k \leftarrow \mathsf{F.Smp}(1^\lambda)$; $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$; $ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$;
- $r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$;
- $(c_0, d_0) := \mathsf{CS.Com}(ck, k; r_0)$; $(c_1, d_1) := \mathsf{CS.Com}(ck, \top; r_1)$;

return $\mathsf{sk} := (k, r_0), \mathsf{vk} := (crs, ck, c_0, c_1)$

$\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)}$*: On input* $\mathsf{sk} = (k, r_0)$ *and* $m \in \mathcal{M}$ *compute*

- $y := \mathsf{F}(k, m)$;
- $\pi \leftarrow \mathsf{NIZK.P}(crs, \eta := (ck, c_0, c_1, y, m), (k, r_0))$*, where* $\eta \in L_\eta$ *from* (11);

return $\sigma := (y, \pi)$.

$\underline{b := \mathsf{Verify}(\mathsf{vk}, m, \sigma)}$*: On input* $\mathsf{vk} = (crs, ck, c_0, c_1)$*,* m *and* $\sigma = (y, \pi)$*,*
return $b := \mathsf{NIZK.V}(crs, \eta = (ck, c_0, c_1, y, m), \pi)$.

$\underline{(\mathsf{vk}, \sigma) \leftarrow \mathsf{OSmp}(1^\lambda, m)}$*: On input* 1^λ *and* $m \in \mathcal{M}$*, compute*

- $r := r_0 \| r_1 \| r_y \| r_{\mathsf{Setup}} \| crs \| r_{\mathsf{P}} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$,
- $y \leftarrow_{r_y} \mathcal{Y}$ *// r_y is used to sample y from $\mathcal{Y}$,*
- $ck := \mathsf{CS.Setup}(1^\lambda; r_{\mathsf{Setup}})$,
- $(c_0, d_0) := \mathsf{CS.Com}(ck, \top; r_0)$; $(c_1, d_1) := \mathsf{CS.Com}(ck, m; r_1)$,
- $\pi := \mathsf{NIZK.P}(crs, \eta := (ck, c_0, c_1, y, m), w := (m, r_1); r_{\mathsf{P}})$;

return $\mathsf{vk} := (crs, ck, c_0, c_1)$ *and* $\sigma := (y, \pi)$.

Theorem 1. *Scheme* OS *in Construction 1 is an EUF-CMA-secure signature scheme with obliviously samplable signatures.*

Proof. We need to show that $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is (standard) EUF-CMA-secure and prove indistinguishability (9) and oblivious unforgeability (10). The proof of EUF-CMA is analogous to that of Bellare and Goldwasser's [BG90] (noting that the second clause in (11) is always false) and is therefore omitted.

Indistinguishability: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary that non-negligibly distinguishes honestly generated ($\mathbf{Exp}^{\text{ind-0}}_{\mathsf{OS},\mathcal{A}}(\lambda)$) and obliviously sampled verification key-signature pairs ($\mathbf{Exp}^{\text{ind-1}}_{\mathsf{OS},\mathcal{A}}(\lambda)$). Our

$\underline{\mathbf{Exp}^{(0)}_{\mathsf{OS},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$
$y := \mathsf{F}(k, m)$
$ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$
$r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$c_0 := \mathsf{CS.Com}_1(ck, k; r_0)$
$c_1 := \mathsf{CS.Com}_1(ck, \top; r_1)$
$\eta := (ck, c_0, c_1, y, m)$
$w := (k, r_0)$
$crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$\pi \leftarrow \mathsf{NIZK.P}(crs, \eta, w)$
$\mathsf{vk} := (crs, ck, c_0, c_1)$
$\sigma := (m, y, \pi)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

$\underline{\mathbf{Exp}^{(1)}_{\mathsf{OS},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$
$y := \mathsf{F}(k, m)$
$ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$
$r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$c_0 := \mathsf{CS.Com}_1(ck, k; r_0)$
$c_1 := \mathsf{CS.Com}_1(ck, \top; r_1)$
$\eta := (ck, c_0, c_1, y, m)$
$(crs, \tau) \leftarrow \mathsf{NIZK.S}_1(1^\lambda, \eta)$
$\pi \leftarrow \mathsf{NIZK.S}_2(crs, \tau, \eta)$
$\mathsf{vk} := (crs, ck, c_0, c_1)$
$\sigma := (y, \pi)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

$\underline{\mathbf{Exp}^{(2)}_{\mathsf{OS},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$
$y := \mathsf{F}(k, m)$
$ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$
$r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$c_0 := \mathsf{CS.Com}_1(ck, \top; r_0)$
$c_1 := \mathsf{CS.Com}_1(ck, \top; r_1)$
$\eta := (ck, c_0, c_1, y, m)$
$(crs, \tau) \leftarrow \mathsf{NIZK.S}_1(1^\lambda, \eta)$
$\pi \leftarrow \mathsf{NIZK.S}_2(crs, \tau, \eta)$
$\mathsf{vk} := (crs, ck, c_0, c_1)$
$\sigma := (y, \pi)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

$\underline{\mathbf{Exp}^{(3)}_{\mathsf{OS},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$
$y := \mathsf{F}(k, m)$
$ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$
$r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$c_0 := \mathsf{CS.Com}_1(ck, \top; r_0)$
$c_1 := \mathsf{CS.Com}_1(ck, m; r_1)$
$\eta := (ck, c_0, c_1, y, m)$
$(crs, \tau) \leftarrow \mathsf{NIZK.S}_1(1^\lambda, \eta)$
$\pi \leftarrow \mathsf{NIZK.S}_2(crs, \tau, \eta)$
$\mathsf{vk} := (crs, ck, c_0, c_1)$
$\sigma := (y, \pi)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

$\underline{\mathbf{Exp}^{(4)}_{\mathsf{OS},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$y \leftarrow \mathcal{Y}$
$ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$
$r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$c_0 := \mathsf{CS.Com}_1(ck, \top; r_0)$
$c_1 := \mathsf{CS.Com}_1(ck, m; r_1)$
$\eta := (ck, c_0, c_1, y, m)$
$(crs, \tau) \leftarrow \mathsf{NIZK.S}_1(1^\lambda, \eta)$
$\pi \leftarrow \mathsf{NIZK.S}_2(crs, \tau, \eta)$
$\mathsf{vk} := (crs, ck, c_0, c_1)$
$\sigma := (y, \pi)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

$\underline{\mathbf{Exp}^{(5)}_{\mathsf{OS},\mathcal{A}}(\lambda)}$

$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$
$y \leftarrow \mathcal{Y}$
$ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$
$r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$c_0 := \mathsf{CS.Com}_1(ck, \top; r_0)$
$c_1 := \mathsf{CS.Com}_1(ck, m; r_1)$
$\eta := (ck, c_0, c_1, y, m)$
$w := (m, r_1)$
$crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$\pi \leftarrow \mathsf{NIZK.P}(crs, \eta, w)$
$\mathsf{vk} := (crs, ck, c_0, c_1)$
$\sigma := (y, \pi)$
Return $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$

Figure 7: The hybrids used in the proof of Theorem 1

proof will be by game hopping and we define a series of games $\mathbf{Exp}^{(0)} := \mathbf{Exp}^{\text{ind-0}}_{\mathsf{OS},\mathcal{A}}(\lambda)$, $\mathbf{Exp}^{(1)}, \ldots,$ $\mathbf{Exp}^{(5)} := \mathbf{Exp}^{\text{ind-1}}_{\mathsf{OS},\mathcal{A}}(\lambda)$ and show that for $c = 0, \ldots, 4$, $\mathbf{Exp}^{(c)}$ and $\mathbf{Exp}^{(c+1)}$ are computationally indistinguishable. All games are given in Fig. 7. In $\mathbf{Exp}^{(0)}$ the adversary obtains vk output by KeyGen and σ output by Sign as defined in Construction 1.

$\mathbf{Exp}^{(1)}$ differs from $\mathbf{Exp}^{(0)}$ in that the CRS for the NIZK and the proof π are simulated. By zero knowledge of NIZK the game is indistinguishable from $\mathbf{Exp}^{(0)}$.

$\mathbf{Exp}^{(2)}$ differs from $\mathbf{Exp}^{(1)}$ in that c_0 commits to $\top$ rather than a PRF key k. By computational hiding of CS, this is indistinguishable for PPT adversaries (note that r_0 is not used elsewhere in the game).

$\mathbf{Exp}^{(3)}$ differs from $\mathbf{Exp}^{(2)}$ in that c_1 commits to m rather than $\top$. Again, by hiding of CS (and since r_1 is not used anywhere), this is indistinguishable.

$\mathbf{Exp}^{(4)}$ differs from $\mathbf{Exp}^{(3)}$ in that $y \leftarrow \mathcal{Y}$ is random rather than $y := \mathsf{F}(k, m)$. Pseudorandomness of F guarantees this change is indistinguishable to PPT adversaries (note that k is not used anywhere else in the game).

$\mathbf{Exp}^{(5)}$ differs from $\mathbf{Exp}^{(4)}$ in that the CRS crs for the NIZK is chosen at random (rather than simulated) and π is computed by $\mathsf{NIZK.P}$. Again, this is indistinguishable by zero knowledge of NIZK.

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

$(st, f) \leftarrow \mathcal{A}_1(1^\lambda)$
If $b = 0$
  $(msk, mvk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $sk_f \leftarrow \mathsf{KeyGen}(msk, f)$
Else $(mvk, sk_f) \leftarrow \mathsf{OSmp}(1^\lambda, f)$
Return $b' \leftarrow \mathcal{A}_2(st, mvk, sk_f)$

Figure 8: The oblivious-indist. game

$\underline{\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS},\mathcal{A}}(\lambda)}$

$(st, f) \leftarrow \mathcal{A}_1(1^\lambda)$
$r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$(mvk, sk_f) \leftarrow \mathsf{OSmp}(1^\lambda, f; r)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}_2(st, r)$
Return 1 iff $m^* \notin \mathcal{R}_f$
  $\wedge\, \mathsf{Verify}(mvk, m^*, \sigma^*) = 1$

Figure 9: The oblivious-unforgeability game

Oblivious unforgeability. This follows from soundness of NIZK and the binding property of CS. OSmp sets c_0 to a commitment of $\top$ and c_1 to a commitment of m. If $\mathcal{A}$ manages to output a signature (y^*, π^*) that is valid on message $m^* \neq m$, i.e., $\mathsf{NIZK.V}(crs, (ck, c_0, c_1, y^*, m^*), \pi^*) = 1$, then by soundness of NIZK, $(ck, c_0, c_1, y^*, m^*) \in L_\eta$ (11), meaning that either c_0 is a commitment to a valid PRF key or c_1 is a commitment to m^*. Either case would contradict the binding property of the commitment scheme.

This proves Theorem 1. A formal proof can be found in Appendix A.1. □

3.2 Functional Signature Schemes with Obliviously Samplable Keys

Definition 14 (FSwOSK)**.** *Let* $\mathsf{FS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a functional signature scheme (Def. 11). We say* FS has obliviously samplable keys *if there exists a PPT algorithm* OSmp *such that:*

$(mvk, sk_f) \leftarrow \mathsf{OSmp}(1^\lambda, f)$*: On input a security parameter* 1^λ *and a function* $f \in \mathcal{F}_\lambda$*,* OSmp *outputs a master verification key* mvk *and a functional signing key* sk_f *for* f*.*

FSwOSK FS *is secure if in addition to Def. 12 the following hold:*

1. *Indistinguishability: For every PPT algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in* $\mathbf{Exp}^{\text{ind-}b}_{\mathsf{FS},\mathcal{A}}(\lambda)$ *defined in Fig. 8, it holds that*
$$|\Pr\left[\mathbf{Exp}^{\text{ind-0}}_{\mathsf{FS},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-1}}_{\mathsf{FS},\mathcal{A}}(\lambda) = 1\right]| = \mathsf{negl}(\lambda) \ . \tag{12}$$
2. *Oblivious unforgeability: For every PPT algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in* $\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS},\mathcal{A}}(\lambda)$ *defined in Fig. 9, it holds that*
$$\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda) \ . \tag{13}$$

We next show that if in the construction of functional signatures of Boyle et al. [BGI14] we replace the signature scheme by a SwOSS (Def. 13) then we obtain a FSwOSK. As a first step Boyle et al. [BGI14, Theorem 3.3] construct $\mathsf{FS}' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$, which does not satisfy function privacy nor succinctness, but which is unforgeable if the underlying signature scheme is EUF-CMA. Relying on adaptive zero-knowledge SNARKs for NP, they then transform FS' into a scheme FS satisfying all notions from Def. 12 [BGI14, Theorem 3.4].

We first enhance their scheme FS' by an oblivious sampler OSmp' so it also satisfies indistinguishability and oblivious unforgeability, as defined in Def. 14. Then we show that the transformation of [BGI14] also applies to our FS', yielding a secure FSwOSK.

Construction 2. *Let* $\mathsf{OS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{OSmp})$ *be a secure SwOSS and* SS *an EUF-CMA-secure signature scheme. For a message space* $\mathcal{M} \not\ni \bot$ *and a function family* $\mathcal{F}_\lambda = \{f\colon \mathcal{D}_f \to \mathcal{R}_f \subseteq \mathcal{M}\}_\lambda$*, we construct* $\mathsf{FS}' = (\mathsf{FS}'.\mathsf{Setup}, \mathsf{FS}'.\mathsf{KeyGen}, \mathsf{FS}'.\mathsf{Sign}, \mathsf{FS}'.\mathsf{Verify}, \mathsf{FS}'.\mathsf{OSmp})$*, a scheme with obliviously samplable keys that is correct, UF, indistinguishable and obl. UF but neither function-private nor succinct.*

$\underline{(msk, mvk) \leftarrow \mathsf{FS}'.\mathsf{Setup}(1^\lambda)}$: *Return* $(msk, mvk) \leftarrow \mathsf{OS.KeyGen}(1^\lambda)$.

$\underline{sk_f \leftarrow \mathsf{FS}'.\mathsf{KeyGen}(msk, f)}$: *On input* msk *and* $f \in \mathcal{F}_\lambda$, *compute*

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SS.KeyGen}(1^\lambda)$,
- $\sigma_{f\|\mathsf{vk}} \leftarrow \mathsf{OS.Sign}(msk, f\|\mathsf{vk})$,

and return $sk_f := (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, \mathsf{sk})$.

$\underline{(f(w), \sigma) \leftarrow \mathsf{FS}'.\mathsf{Sign}(f, sk_f, w)}$: *On input* $f \in \mathcal{F}_\lambda$, *key* $sk_f = (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, sk)$ *for* f *and* $w \in \mathcal{D}_f$, *compute* $\sigma_w \leftarrow \mathsf{SS.Sign}(\mathsf{sk}, w)$; *return* $\sigma := (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, w, \sigma_w)$.

$\underline{\mathsf{FS}'.\mathsf{Verify}(mvk, m, \sigma) \in \{0,1\}}$: *On input* mvk, $m \in \{0,1\}^*$ *and* $\sigma = (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, w, \sigma_w)$, *return 1 if* $\mathsf{OS.Verify}(mvk, f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}) = \mathsf{SS.Verify}(\mathsf{vk}, w, \sigma_w) = 1$ *and* $m = f(w)$; *else return 0.*

$\underline{(mvk, sk_f) \leftarrow \mathsf{FS}'.\mathsf{OSmp}(1^\lambda, f)}$: *On input* 1^λ *and* $f \in \mathcal{F}_\lambda$, *compute*

- $r_\mathsf{G}, r_\mathsf{O} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$,
- $(\mathsf{sk}, \mathsf{vk}) := \mathsf{SS.KeyGen}(1^\lambda; r_\mathsf{G})$,
- $(mvk, \sigma_{f\|\mathsf{vk}}) := \mathsf{OS.OSmp}(1^\lambda, f\|\mathsf{vk}; r_\mathsf{O})$,

and return mvk *and* $sk_f := (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, \mathsf{sk})$.

Theorem 2. FS' *of Construction 2 is a FSwOSK that satisfies correctness, unforgeability, indistinguishability and oblivious unforgeability (but neither function privacy nor succinctness).*

Theorem 2 is formally proved in Appendix A.2 and we give some proof intuition here. Theorem 3.3 in [BGI14] proves that $(\mathsf{FS}'.\mathsf{Setup}, \mathsf{FS}'.\mathsf{KeyGen}, \mathsf{FS}'.\mathsf{Sign}, \mathsf{FS}'.\mathsf{Verify})$ is a functional signature scheme that is correct and unforgeable. What remains then is to show that $\mathsf{FS}'.\mathsf{OSmp}$ satisfies both indistinguishability (12) and oblivious unforgeability (13).

Note that a FSwOSK master verification key is a SwOSS verification key, and a FswOSK functional signing key is a SwOSS signature; thus an obliviously samplable pair for FSwOSK translates to a pair for SwOSS; indistinguishability for FSwOSK reduces thus to indistinguishability for SwOSS. Similarly, oblivious unforgeability for FSwOSK reduces to oblivious unforgeability of SwOSS (note that in this game the adversary cannot ask for functional signatures, so EUF-CMA of the regular signature scheme is not needed).

Next we show that the transformation of [BGI14] applies to our scheme FS', and therefore the transformed FS is a FSwOSK satisfying Def. 14.

Theorem 3. *Assuming an adaptive zero-knowledge SNARK system for* NP, FS' *from Construction 2 can be transformed into a secure FSwOSK scheme* FS.

Proof sketch. The construction and proof of the theorem are exactly the same as those of Theorem 3.4 of [BGI14], and therefore we only give an intuitive argument and refer the reader to [BGI14] for more details.

First observe that in FS' a signature $\sigma := (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, w, \sigma_w)$ on $f(w)$ contains both f and w in the clear and is therefore neither function-private nor succinct. In the new scheme FS a signature on m is instead a zero-knowledge SNARK proof π of knowledge of the following: f, vk, a signature $\sigma_{f\|vk}$ on $f\|vk$ that verifies under mvk, an element w such that $f(w) = m$, and a signature σ on w, valid under vk. Now function privacy reduces to zero knowledge and succinctness of signatures reduces to succinctness of the underlying SNARK. □

4 Constrained PRFs for Unbounded Inputs

In this section we construct a family of constrained PRFs for unbounded inputs such that a constrained key is simply a (functional) signature on the constraining TM M. As a warm-up, we review the construction of [AFP16] where a constrained key is a $di\mathcal{O}$ obfuscation of a circuit that depends on the size of the constraining TM M. In particular, the circuit verifies a SNARK for the following relation.

Definition 15 (R_{legit})**.** *We define the relation $R_{legit} \subset \mathcal{R}_{\mathcal{U}}$ (with $\mathcal{R}_{\mathcal{U}}$ from Def. 7) to be the set of instance-witness pairs $(((H, M), h, t), x)$ such that M and H are descriptions of a TM and a hash function, $M(x) = 1$ and $H(x) = h$ within t steps. We let L_{legit} be the language corresponding to R_{legit}. For notational convenience, abusing notation, we write $((H, M, h), x) \in R_{legit}$ to mean $(((H, M), h, t), x) \in R_{legit}$ while implicitly setting $t = 2^{\lambda}$.*

Remark 2. Let $t = 2^{\lambda}$ in the definition of R_{legit}; then by succinctness of SNARKs (Def. 8), the length of a SNARK proof is bounded by $\ell(\lambda)$ and its verification time is bounded by $p(\lambda + |M| + |H| + |h|)$, where p, ℓ are a priori fixed polynomials that do not depend on R_{legit}.

Construction 3. *[AFP16] Let $\mathcal{PF}_\lambda = \{\mathsf{PF}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ be a selectively secure puncturable PRF, $\mathcal{H}_\lambda = \{H\colon \{0,1\}^* \to \{0,1\}^n\}_\lambda$ a family of public-coin CR hash functions, diO a public-coin $di\mathcal{O}$ obfuscator for a family of polynomial-size circuits $\mathcal{P}_\lambda$, and SNARK a SNARK system for R_{legit} (Def. 15). A family of selectively secure PRFs $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \{0,1\}^* \to \mathcal{Y}\}$ constrained w.r.t. to any polynomial-size family of TMs $\mathcal{M}_\lambda$ is defined as follows:*

$K \leftarrow \mathsf{F.Smp}(1^\lambda)$: Sample $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$, $H \leftarrow \mathsf{H.Smp}(1^\lambda)$, $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$; return $K := (k, H, crs)$.

$k_M \leftarrow \mathsf{F.Constr}(K, M)$: On input $K = (k, H, crs)$ and $M \in \mathcal{M}_\lambda$, define

$$P_{M,H,crs,k}(h, \pi) := \begin{cases} \mathsf{PF.Eval}(k, h) & \text{if } \mathsf{SNARK.V}(crs, (H, M, h), \pi) = 1 \\ \bot & \text{otherwise} \end{cases} \tag{14}$$

compute $\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P_{M,H,crs,k})$ and output $k_M := (M, \widetilde{P}, H, crs)$.

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

- *If $\kappa \in \mathcal{K}$, $\kappa = (k, H, crs)$: return $\mathsf{PF.Eval}(k, H(x))$.*
- *If $\kappa = (M, \widetilde{P}, H, crs) \in \mathcal{K}_{\mathcal{M}}$: if $M(x) = 1$, let $h := H(x)$ (thus $(H, M, h) \in L_{legit}$), $\pi \leftarrow \mathsf{SNARK.P}(crs, (H, M, h), x)$ and return $y := \widetilde{P}(h, \pi)$.*

The drawback of Construction 3 is that a constrained key for a set decided by a TM M is a $di\mathcal{O}$-obfuscated circuit and is therefore large. In our construction below we use FSwOSK to define a constrained key simply as a functional signature k_M on M. As in Construction 3, our constrained PRF F is defined as as $\mathsf{F}(k, x) = \mathsf{PF}(k, H(x))$, where PF is a puncturable PRF and H is a collision-resistant hash function. To enable evaluating F given a constrained key k_M, in the setup we output as a public parameter a $di\mathcal{O}$-obfuscation of a circuit P (defined in (15) below) that on input (M, h, π, σ) outputs $\mathsf{PF}(k, H(x))$ if π is a valid SNARK proving knowledge of some x such that $M(x) = 1$ and $H(x) = h$, and moreover σ is a valid functional signature on M; and outputs $\bot$ otherwise.

To evaluate F on x with a constrained key k_M, first set $h := H(x)$ and produce a SNARK π proving knowledge of x with $M(x) = 1$ and $h = H(x)$. Then run the public circuit P on (M, h, π, k_M), which will return $\mathsf{PF}(k, h) = \mathsf{F}(k, H(x))$.

Construction 4 (TM-constrained PRF with short keys)**.** *Let $\mathcal{PF}_\lambda = \{\mathsf{PF}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ be a selectively secure puncturable PRF, $\mathcal{H}_\lambda = \{H\colon \{0,1\}^* \to \{0,1\}^n\}_\lambda$ a family of public-coin collision-resistant hash functions, $\mathsf{FS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{OSmp})$ a FSwOSK scheme, diO a public-coin differing-input obfuscator for a family of poly-size circuits $\mathcal{P}_\lambda$, and* SNARK *a SNARK system in the common-random-string model for R_{legit} (cf. Def. 15).*

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 $\mathcal{F} = (\mathsf{F.Smp}, \mathsf{F.Constr}, \mathsf{F.Eval})$.

$K \leftarrow \mathsf{F.Smp}(1^\lambda)$:

- *$H \leftarrow \mathsf{H.Smp}(1^\lambda)$.*
- *$crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.*
- *$(msk, mvk) \leftarrow \mathsf{FS.Setup}(1^\lambda)$.*
- *$sk_{f_I} \leftarrow \mathsf{FS.KeyGen}(msk, f_I)$ where f_I is the identity function, i.e., $f_I\colon M \mapsto M$.*
- *$k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.*
- *$\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P)$ where $P = P_{H,crs,mvk,k} \in \mathcal{P}_\lambda$ is defined as:*

$$P(M,h,\pi,\sigma) := \begin{cases} \mathsf{PF.Eval}(k,h) & \text{if } \mathsf{SNARK.V}\big(crs,(H,M,h),\pi\big) = 1 \\ & \quad \wedge\ \mathsf{FS.Verify}(mvk,M,\sigma) = 1 \\ \bot & \text{otherwise} \end{cases} \tag{15}$$

- *Set $pp = (H, crs, mvk, \widetilde{P})$ and return $K := (k, sk_{f_I}, pp)$.*

$k_M \leftarrow \mathsf{F.Constr}(K, M)$: On input $K = (k, sk_{f_I}, pp)$ and $M \in \mathcal{M}_\lambda$, compute $(M,\sigma) \leftarrow \mathsf{FS.Sign}(I, sk_{f_I}, M)$ and return $k_M := (M, \sigma, pp)$.

$y := \mathsf{F.Eval}(\kappa, x)$: On input $\kappa \in \mathcal{K} \cup \mathcal{K}_\mathcal{M}$ and $x \in \{0,1\}^*$:

- *If $\kappa \in \mathcal{K}$, $\kappa = (k, sk_{f_I}, pp = (H, crs, mvk, \widetilde{P}))$: set $y := \mathsf{PF.Eval}(k, H(x))$.*
- *If $\kappa \in \mathcal{K}_\mathcal{M}$, $\kappa = (M, \sigma, (H, crs, mvk, \widetilde{P}))$: if $M(x) = 1$, set $h := H(x)$ (thus $(H, M, h) \in L_{legit}$), compute $\pi \leftarrow \mathsf{SNARK.P}(crs, (H, M, h), x)$, and return $y := \widetilde{P}(M, h, \pi, \sigma)$.*

Remark 3. The values in pp, including $\widetilde{P}$, are computed once and for all. As in the model for constrained PRFs there are no public parameters, we formally include them in the constrained key k_M. Note that $\mathcal{P}_\lambda$ is in fact a family of circuits with an input length $|M| + n + |\pi| + |\sigma|$ where $|\pi|$ is upper bounded by $\ell(\lambda)$ even for an exponentially long x (cf. Remark 2).

Let us now argue why we need functional signatures with obliviously samplable keys in order to prove our construction secure.

If we could replace the PRF key k by a punctured one $k^* := k_{H(x^*)}$ then $\mathsf{F}(k, x^*)$ would look random, as required for selective security of F. The obfuscated circuit P would thus use k^* instead of k. But obfuscations of P_k and P_{k^*} are only indistinguishable if it is hard to find an input on which they differ. And, since we use public-coin diO, this should be even hard when given all coins used to produce P_k and P_{k^*}.

In the security experiment the adversary can query keys for machines M with $M(x^*) = 0$ and when fed to P_k and P_{k^*}, both output the same. However, if the adversary manages to forge a signature on some $\hat{M}$ with $\hat{M}(x^*) = 1$ then P_k outputs $\mathsf{F}(k, x^*)$, but P_{k^*}, using a punctured key, outputs $\bot$.

The tricky part is to break some unforgeability notion when this happens. The differing-input sampler that computes P_k and P_{k^*} must simulate the experiment for $\mathcal{A}$ and thus create signatures to answer key queries. This is why we need functional signatures, as then the sampler can use a signing key sk_{f^*}, which only allows signing of machines with $M(x^*) = 0$, to answer key queries. FS unforgeability guarantees that even given such a key it is hard to compute a signature on some $\hat{M}$ with $\hat{M}(x^*) = 1$.

The next problem is that finding a differing input (and thus a forgery on $\hat{M}$) should be hard *even when given all coins*, so in particular the coins to create the signature verification key mvk contained in P_k and P_{k^*}; thus it would be easy to "forge a signature". This is why we need FSwOSK, as they allow to sample a verification key together with sk_{f^*} and even when given the coins, forgeries should be hard.

Theorem 4. *$\mathcal{F}_\lambda$ of Construction 4 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 polynomial-size Turing machine.*

Proof. Let $\mathcal{A}$ be a PPT adversary for the game $\mathbf{Exp}^{(\emptyset,\{\textsc{Constr},\textsc{Eval}\}),b}_{\mathcal{F},\mathcal{A}}(\lambda)$, as defined in Fig. 10, which we abbreviate as $\mathbf{Exp}^b$. 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 $\mathbf{Exp}^{b,(0)} := \mathbf{Exp}^b$, $\mathbf{Exp}^{b,(1)}$,$\mathbf{Exp}^{b,(2)}$, $\mathbf{Exp}^{b,(3)}$, $\mathbf{Exp}^{b,(4)}$ and show that for $b = 0,1$ and $c = 0,1,2,3$ the games $\mathbf{Exp}^{b,(c)}$ and $\mathbf{Exp}^{b,(c+1)}$ are indistinguishable. Finally we show that $\mathbf{Exp}^{0,(4)}$ and $\mathbf{Exp}^{1,(4)}$ are also indistinguishable, which concludes the proof. All games are defined in Fig. 10, using the following definitions:

$$f_I \colon M \mapsto M, \qquad f_{x^*} \colon M \mapsto \begin{cases} M & \text{if } M(x^*) = 0 \\ \bot & \text{otherwise} \end{cases} \tag{16}$$

- $\mathbf{Exp}^{b,(0)}$ is the original game $\mathbf{Exp}^{b,(\emptyset,\{\textsc{Constr},\textsc{Eval}\})}_{\mathcal{F},\mathcal{A}}(\lambda)$ for Construction 4. (Note that we padded f_I but, by succinctness, functional signatures (returned by Constr) are independent of the length of f.)
- $\mathbf{Exp}^{b,(1)}$ differs from $\mathbf{Exp}^{b,(0)}$ by replacing the signing key sk_{f_I} with $sk_{f_{x^*}}$, which only allows to sign machines M with $M(x^*) = 0$.
- $\mathbf{Exp}^{b,(2)}$ differs from $\mathbf{Exp}^{b,(1)}$ by replacing the verification/signing key pair $(mvk, sk_{f_{x^*}})$ with an obliviously sampled one.
- $\mathbf{Exp}^{b,(3)}$ differs from $\mathbf{Exp}^{b,(2)}$ by replacing the full key of the puncturable PRF PF with one that is punctured at $H(x^*)$ in the definition of P.
- $\mathbf{Exp}^{b,(4)}$ differs from $\mathbf{Exp}^{b,(3)}$ by answering Eval queries using the punctured key k_{h^*} and aborting whenever the adversary queries Eval on a value that collides with x^* under H.

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 $sk_{f_{x^*}}$, respectively, in answering constraining queries. The Constr oracle only computes signatures on TMs M with $M(x^*) = 0$. Therefore, f_{x^*} coincides with f_I on all such legitimate queries. By function privacy of FS (Def. 12), signatures generated with f_{x^*} and f_I are computationally indistinguishable.

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

$\underline{\mathbf{Exp}^{(\emptyset,\{\textsc{Constr},\textsc{Eval}\}),b}_{\mathcal{F},\mathcal{A}}(\lambda)}$

$(x^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$K \leftarrow \mathsf{F.Smp}(1^\lambda)$
If $b = 1$
  $y^* := \mathsf{F.Eval}(K, x^*)$
Else
  $y^* \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st, y^*)$
Return b'

Oracle $\textsc{Constr}(M)$

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

Oracle $\textsc{Eval}(x)$

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

$\underline{\mathbf{Exp}^{b,(c)}_{\mathcal{F},\mathcal{A}}(\lambda)}$ // $c \in \{0,1,2,3,4\}$

$(x^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$H \leftarrow \mathsf{H.Smp}(1^\lambda)$
$crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
If $c \le 1$
  $(msk, mvk) \leftarrow \mathsf{FS.Setup}(1^\lambda)$
  Define f_I and f_{x^*} as in (16) and pad them to the same length.
  $sk_{f_I} \leftarrow \mathsf{FS.KeyGen}(msk, f_I)$
  $sk_{f_{x^*}} \leftarrow \mathsf{FS.KeyGen}(msk, f_{x^*})$
Else
  $(mvk, sk_{f_{x^*}}) \leftarrow \mathsf{FS.OSmp}(1^\lambda, f_{x^*})$
$k \leftarrow \mathsf{PF.Smp}(1^\lambda)$
$k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{H(x^*)\})$
If $c \le 2$ then
  $P := P_{H,crs,mvk,k}$ as defined in (15)
Else
  $P := P_{H,crs,mvk,k_{h^*}}$ as defined in (15)
$\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P)$
$pp := (H, crs, mvk, \widetilde{P})$
If $b = 1$, $y^* := \mathsf{PF.Eval}(k, H(x^*))$, else $y^* \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st, y^*)$
Return b'

Oracle $\textsc{Constr}(M)$

If $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$
  Return $\perp$
If $c = 0$
  $(M, \sigma) \leftarrow \mathsf{FS.Sign}(f_I, sk_{f_I}, M)$
Else
  $(M, \sigma) \leftarrow \mathsf{FS.Sign}(f_{x^*}, sk_{f_{x^*}}, M)$
Return $k_M := (M, \sigma, pp)$

Oracle $\textsc{Eval}(x)$

If $x = x^*$
  Return $\perp$
If $c \le 3$
  $y := \mathsf{PF.Eval}(k, H(x))$
Else
  If $H(x) = H(x^*)$, abort
  Else $y := \mathsf{PF.Eval}(k_{h^*}, H(x))$
Return y

Figure 10: The original security game and hybrids used in the proof of Theorem 4

The only difference between $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ is in how mvk and $sk_{f_{x^*}}$ are computed. In $\mathbf{Exp}^{b,(1)}$ the keys mvk (used to define P) and $sk_{f_{x^*}}$ (used to answer $\textsc{Constr}$ queries) are generated by $\mathsf{FS.Setup}$ and $\mathsf{FS.KeyGen}$, resp., whereas in $\mathbf{Exp}^{b,(2)}$ they are obliviously sampled together. Indistinguishability of honestly generated and obliviously sampled pairs (Def. 14) of verification/signing key pairs guarantees that this change is indistinguishable to PPT adversaries.

Proposition 2. $\mathbf{Exp}^{b,(1)}$ *and* $\mathbf{Exp}^{b,(2)}$ *are computationally indistinguishable for* $b = 0, 1$ *if* FS *is a functional signature scheme with obliviously samplable keys.*

It is in the next step that we use the full power of our new primitive FSwOSK. The only difference between $\mathbf{Exp}^{b,(2)}$ and $\mathbf{Exp}^{b,(3)}$ is in the definition of the circuit P that is obfuscated. In $\mathbf{Exp}^{b,(2)}$ the circuit $P =: P^{(2)}$ is defined as in (15), with $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$. In $\mathbf{Exp}^{b,(3)}$, the key k in circuit

$P =: P^{(3)}$ is replaced by a punctured key $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{H(x^*)\})$.

The two games differ thus in whether $\tilde{P}$ is an obfuscation of $P^{(2)}$ or $P^{(3)}$. By public-coin diO, these are indistinguishable, if for a sampler Samp that outputs $P^{(2)}$ and $P^{(3)}$, no extractor, even when given the coins used by Samp, can find a differing input $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$.

Suppose there exists an extractor $\mathcal{E}$ outputs such a tuple. By correctness of PF, $P^{(2)}$ and $P^{(3)}$ only differ on inputs $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$, where

$$\hat{h} = H(x^*) \ , \tag{17}$$

as that is where the punctured key behaves differently. Moreover, the signature $\hat{\sigma}$ must be valid on $\hat{M}$, as otherwise both circuits output $\perp$. Intuitively, unforgeability of functional signatures should guarantee that

$$\hat{M}(x^*) = 0 \ , \tag{18}$$

as the adversary only obtains a signature from its Constr oracle when it submits machines satisfying (18), so a valid $\hat{\sigma}$ on $\hat{M}$ with $\hat{M}(x^*) = 1$ would be a forgery.

To construct $P^{(2)}$ and $P^{(3)}$, Samp must simulate the experiment for $\mathcal{A}$, during which it needs to answer $\mathcal{A}$'s Constr queries and thus create signatures. This shows the need for a functional signature scheme: we need to enable Samp to create signatures on M's with $M(x^*) = 0$ (by giving it $sk_{f_{x^*}}$) while still arguing that it is hard to find a signature on $\hat{M}$ with $\hat{M}(x^*) = 1$.

Finally, if we used standard functional signatures then we would need to embed a master verification key (under which the forgery will be) into Samp, but this would require diO *with auxiliary inputs*. We avoid this using FSwOSK, which let Samp create mvk (together with sk_{f^*}) itself, and which ensure that for $\mathcal{E}$, even given Samp's coins, it is hard to find a forgery $\hat{\sigma}$. It follows that (18) must hold with overwhelming probability.

Finally the 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 (18) imply $\hat{x} \neq x^*$ and (ii) and (17) imply $H(\hat{x}) = H(x^*)$. Together, this means $(\hat{x}, x^*)$ is a collision for H.

Overall, we showed that an extractor can only find a differing input for $P^{(2)}$ and $P^{(3)}$ with negligible probability. By security of diO (Def. 5), we thus have that obfuscations of $P^{(2)}$ and $P^{(3)}$ are indistinguishable.

Proposition 3. $\mathbf{Exp}^{b,(2)}$ *and* $\mathbf{Exp}^{b,(3)}$ *are computationally indistinguishable for* $b = 0, 1$*, if* diO *is a public-coin differing-input obfuscator,* FS *a FSwOSK satisfying oblivious unforgeability and* $\mathcal{H}$ *is public-coin collision-resistant.*

For the game hop from games $\mathbf{Exp}^{b,(3)}$ to $\mathbf{Exp}^{b,(4)}$, indistinguishability follows directly from collision resistance of $\mathcal{H}$, as the only difference is that $\mathbf{Exp}^{b,(4)}$ aborts when $\mathcal{A}$ finds a collision.

Proposition 4. $\mathbf{Exp}^{b,(3)}$ *and* $\mathbf{Exp}^{b,(4)}$ *are computationally indistinguishable for* $b = 0, 1$*, if* $\mathcal{H}$ *is CR.*

We have now reached a game, $\mathbf{Exp}^{b,(4)}$, 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 5. $\mathbf{Exp}^{0,(4)}$ *and* $\mathbf{Exp}^{1,(4)}$ *are indistinguishable if* $\mathcal{PF}$ *is a selectively secure family of puncturable PRFs.*

Theorem 4 now follows from Propositions 1, 2, 3, 4 and 5, which we prove in Appendix A.3. □

References

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Constrained PRFs for unbounded inputs. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 413–428. Springer, 2016. Available at http://eprint.iacr.org/2014/840.

[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, Heidelberg, February 2014.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[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, Heidelberg, March 2014.

[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, Heidelberg, 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, Heidelberg, August 2014.

[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, Heidelberg, August 2013.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Heidelberg, August 1994.

[GGH+13] 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.

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

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Heidelberg, March 2015.

[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.

[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.

A Proofs

A.1 Proof of Theorem 1

In the sequel, $\mathbf{Exp}^{(i)}$ in Proposition 6–10 abbreviates $\mathbf{Exp}^{(i)}_{\mathsf{S},\mathcal{A}}$ in Fig. 7.

Proposition 6. $\mathbf{Exp}^{(0)}(\lambda)$ *and* $\mathbf{Exp}^{(1)}(\lambda)$ *are computationally indistinguishable if* NIZK *is zero-knowledge.*

Proof. Assume there exists a polynomial $p(\cdot)$ such that for infinitely many λ, it holds that

$$\epsilon(\lambda) := |\Pr[\mathbf{Exp}^{(0)}(\lambda) = 1] - \Pr[\mathbf{Exp}^{(1)}(\lambda) = 1]| \geq \tfrac{1}{p(\lambda)} \ .$$

We use $\mathcal{A}$ to construct a PPT adversary $\mathcal{B}$ against the zero-knowledge security of NIZK (cf. (5)):

$\underline{\mathcal{B}(1^\lambda)}$:

- $(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $k \leftarrow \mathsf{F.Smp}(1^\lambda)$.
- $y := \mathsf{F}(k, m)$.
- $ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$.
- $r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $c_0 := \mathsf{CS.Com}_1(ck, k; r_0)$; $c_1 := \mathsf{CS.Com}_1(ck, \top; r_1)$.
- $\eta := (ck, c_0, c_1, y, m)$ and $w := (k, r_0)$ and note that $R_\eta(\eta, w) = 1$.
- Submit (η, w) to the zero-knowledge game of (5) to obtain either
  - An honest (crs, π): $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ and $\pi \leftarrow \mathsf{NIZK.P}(crs, \eta, w)$, or
  - A simulated (crs, π): $(crs, \tau) \leftarrow \mathsf{NIZK.S}_1(1^\lambda, \eta)$, $\pi \leftarrow \mathsf{NIZK.S}_2(crs, \tau, \eta)$.
- $\mathsf{vk} := (crs, ck, c_0, c_1)$ and $\sigma := (y, \pi)$.
- Output $b' \leftarrow \mathcal{A}_2(st, \mathsf{vk}, \sigma)$.

By construction, if (crs, π) is generated honestly then $\mathcal{B}$ simulates $\mathbf{Exp}^{(0)}$, and if (crs, π) is simulated then $\mathcal{B}$ simulates $\mathbf{Exp}^{(0)}$. Therefore, for infinitely many λ:

$$\begin{aligned}\tfrac{1}{p(\lambda)} &\leq \left|\Pr\left[\mathbf{Exp}^{(0)}(\lambda)=1\right]-\Pr\left[\mathbf{Exp}^{(1)}(\lambda)=1\right]\right| \\ &= \Big|\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ \pi \leftarrow \mathsf{NIZK.P}(crs,\eta,w):\ \mathcal{B}(crs,\eta,\pi)=1\right]- \\ &\Pr\left[(crs,\tau)\leftarrow \mathsf{NIZK.S_1}(1^\lambda,\eta);\ \pi\leftarrow \mathsf{NIZK.S_2}(crs,\tau,\eta):\ \mathcal{B}(crs,\eta,\pi)=1\right]\Big|\ .\end{aligned}$$

A contradiction to the zero-knowledge security, and therefore $\epsilon(\lambda)=\mathsf{negl}(\lambda)$. □

Proposition 7. $\mathbf{Exp}^{(1)}(\lambda)$ *and* $\mathbf{Exp}^{(2)}(\lambda)$ *are computationally indistinguishable if* CS *is computationally hiding.*

Proof. Assume towards contradiction that there exists a polynomial $p(\cdot)$ such that for infinitely many λ it holds that

$$\epsilon(\lambda) := \left|\Pr\left[\mathbf{Exp}^{(1)}(\lambda)=1\right]-\Pr\left[\mathbf{Exp}^{(2)}(\lambda)=1\right]\right| \geq \tfrac{1}{p(\lambda)}\ .$$

We use $\mathcal{A}$ to construct a PPT adversary $\mathcal{B}=(\mathcal{B}_1,\mathcal{B}_2)$ against (7) the computational hiding of the commitment scheme CS as follows:

$\underline{\mathcal{B}_1(ck)}$:

- $(st_{\mathcal{A}}, m) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $k \leftarrow \mathsf{F.Smp}(1^\lambda)$.
- $y := \mathsf{F}(k, m)$.
- $c_1 \leftarrow \mathsf{CS.Com}_1(ck, \top)$.
- $st := (st_{\mathcal{A}}, c_1, y, m)$.
- Return $(st, m_0^* := k, m_1^* := \top)$.

$\underline{\mathcal{B}_2(st, c_b^*)}$: // with $c_b^* \leftarrow \mathsf{CS.Com}_1(ck, m_b^*)$.

- $c_0 := c_b^*$.
- $\eta := (ck, c_0, c_1, y, m)$.
- $(crs, \tau) \leftarrow \mathsf{NIZK.S_1}(1^\lambda, \eta)$.
- $\pi \leftarrow \mathsf{NIZK.S_2}(crs, \tau, \eta)$.
- $\mathsf{vk} := (crs, ck, c_0, c_1)$ and $\sigma := (y, \pi)$.
- Output $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, \mathsf{vk}, \sigma)$.

By construction, if c_b^* commits to k then $\mathcal{B}$ simulates $\mathbf{Exp}^{(1)}$, and if c_b^* commits to $\top$ then $\mathcal{B}$ simulates $\mathbf{Exp}^{(2)}$. Therefore for infinitely many λ:

$$\begin{aligned}&\frac{1}{p(\lambda)} \leq \left|\Pr[\mathbf{Exp}^{(1)}(\lambda)=1]-\Pr[\mathbf{Exp}^{(2)}(\lambda)=1]\right| = \\ &\Big|\Pr\left[\ ck\leftarrow \mathsf{CS.Setup}(1^\lambda);\ (st,m_0^*,m_1^*)\leftarrow \mathcal{B}_1(ck);\ c_0^*\leftarrow \mathsf{CS.Com}_1(ck,m_0^*)\ :\ \mathcal{B}_2(st,c_0^*)=1\right]- \\ &\quad\Pr\left[\ ck\leftarrow \mathsf{CS.Setup}(1^\lambda);\ (st,m_0^*,m_1^*)\leftarrow \mathcal{B}_1(ck);\ c_1^*\leftarrow \mathsf{CS.Com}_1(ck,m_1^*)\ :\ \mathcal{B}_2(st,c_1^*)=1\right]\Big|\ .\end{aligned}$$

We therefore reach a contradiction to the computational-hiding security of CS, (7), and conclude that $\epsilon(\lambda)=\mathsf{negl}(\lambda)$. □

Proposition 8. $\mathbf{Exp}^{(2)}(\lambda)$ *and* $\mathbf{Exp}^{(3)}(\lambda)$ *are computationally indistinguishable if* CS *is computationally hiding.*

The proof of Proposition 8 is similar to that of Proposition 7, and is omitted.

Proposition 9. $\mathbf{Exp}^{(3)}(\lambda)$ *and* $\mathbf{Exp}^{(4)}(\lambda)$ *are computationally indistinguishable if* F *is a selectively secure PRF.*

Proof. Assume towards contradiction that there exists a polynomial $p(\cdot)$ such that for infinitely many λ it holds that

$$\epsilon(\lambda) := \left| \Pr\left[\mathbf{Exp}^{(3)}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{(4)}(\lambda) = 1\right] \right| \geq \tfrac{1}{p(\lambda)} \ .$$

We use $\mathcal{A}$ to construct a PPT distinguisher $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against pseudorandomness of F as follows:

$\underline{\mathcal{B}_1(1^\lambda)}$:
- $(st_{\mathcal{A}}, m) \leftarrow \mathcal{A}_1(1^\lambda)$.
- Return $(m, st := st_{\mathcal{A}})$.

$\underline{\mathcal{B}_2(st, y)}$: // y is either $\mathsf{F}(k, m)$, or $y \leftarrow \mathcal{Y}$.
- $ck \leftarrow \mathsf{CS.Setup}(1^\lambda)$.
- $r_0, r_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $c_0 := \mathsf{CS.Com}_1(ck, \top; r_0)$.
- $c_1 := \mathsf{CS.Com}_1(ck, m; r_1)$.
- $\eta := (ck, c_0, c_1, y, m)$.
- $(crs, \tau) \leftarrow \mathsf{NIZK.S}_1(1^\lambda, \eta)$.
- $\pi \leftarrow \mathsf{NIZK.S}_2(crs, \tau, \eta)$.
- $\mathsf{vk} := (crs, ck, c_0, c_1)$ and $\sigma := (y, \pi)$.
- Output $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, \mathsf{vk}, \sigma)$.

By construction, if $y = \mathsf{F}(k, m)$ then $\mathcal{B}$ simulates $\mathbf{Exp}^{(3)}$, and if $y \leftarrow \mathcal{Y}$ then $\mathcal{B}$ simulates $\mathbf{Exp}^{(4)}$. Therefore for infinitely many λ it holds that

$$\tfrac{1}{p(\lambda)} \leq \left| \Pr\left[\mathbf{Exp}^{(3)}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{(4)}(\lambda) = 1\right] \right| = \mathsf{Adv}^{\emptyset}_{\mathsf{F},\mathcal{B}}(\lambda) \ ,$$

where $\mathsf{Adv}^{\mathcal{O}}_{\mathsf{F},\mathcal{B}}(\lambda)$ is defined in (1). (Note that $\mathcal{F}$ is a selectively secure, respectively an adaptively secure, PRF if (1) holds with $\mathcal{O}_1 = \emptyset, \mathcal{O}_2 = \textsc{Eval}(\cdot)$, respectively with $\mathcal{O}_1 = \mathcal{O}_2 = \textsc{Eval}(\cdot)$. Here we use even restricted selective security with $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$.) We reach a contradiction to F's security, and therefore $\epsilon(\lambda) = \mathsf{negl}(\lambda)$. □

Proposition 10. $\mathbf{Exp}^{(4)}(\lambda)$ *and* $\mathbf{Exp}^{(5)}(\lambda)$ *are computationally indistinguishable if* NIZK *is zero-knowledge.*

The proof of Proposition 10 is similar to that of Proposition 6, and is omitted.

Proposition 11. *For every PPT adversary* $\mathcal{A}$*:* $\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$.

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

$$\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda) = 1\right] \geq \tfrac{1}{p(\lambda)} \ ,$$

where $\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda)$ is defined in Fig. 6 and instantiated by OSmp from Construction 1, that is:

$\underline{\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda)}$
$(st, m) \leftarrow \mathcal{A}_1(1^\lambda)$.
$r := r_0 \| r_1 \| r_y \| r_{\mathsf{Setup}} \| r_{\mathsf{G}} \| r_{\mathsf{P}} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
$y \leftarrow_{r_y} \mathcal{Y}$.
$ck := \mathsf{CS.Setup}(1^\lambda; r_{\mathsf{Setup}})$.
$(c_0, d_0) := \mathsf{CS.Com}(ck, \top; r_0)$.
$(c_1, d_1) := \mathsf{CS.Com}(ck, m; r_1)$
$crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
$\pi \leftarrow \mathsf{NIZK.P}(crs, (ck, c_0, c_1, y, m), (m, r_1); r_{\mathsf{P}})$.
$(m^*, (y^*, \pi^*)) \leftarrow \mathcal{A}_2(st, r)$.
Return 1 iff $m^* \neq m \wedge \mathsf{NIZK.V}(crs, (ck, c_0, c_1, y^*, m^*), \pi^*) = 1$

We have that $\eta^* := (ck, c_0, c_1, y^*, m^*)$ must satisfy one of the following:

Case 1: $\eta^* \notin L_\eta$. Since $\mathsf{NIZK.V}(crs, \eta^*, \pi^*) = 1$, this implies that $\mathcal{A}$ broke statistical soundness (4) of NIZK, which only happens with negligible probability (Note that $\mathcal{A}$ having access to r_P does not give it any advantage as, with overwhelming probability over the choice of crs, there simply exists no wrong statement with a proof that verifies.)

Case 2: $\eta^* \in L_\eta$ with a witness $w = (\hat{k}, \hat{r})$ satisfying the first clause in (11); that is, for some $\hat{d}$ we have $(c_0, \hat{d}) = \mathsf{CS.Com}(ck, \hat{k}; \hat{r}) \wedge y^* = \mathsf{F}(\hat{k}, m^*)$. Since $\top \notin \mathcal{K}$, but $\mathsf{F}(\hat{k}, m^*) \in \mathcal{R}$, we have $\hat{k} \neq \top$. On the other hand we have $(c_0, d_0) = \mathsf{CS.Com}(ck, \top; r_0)$. By correctness of CS (6), we have with overwhelming probability:

$$\hat{k} \leftarrow \mathsf{Open}(ck, c_0, \hat{d}) \qquad \top \leftarrow \mathsf{Open}(ck, c_0, d_0) \qquad \text{and } \hat{k} \neq \top \ ,$$

meaning that $\mathcal{A}$ broke the binding property of CS (8).

Case 3: $\eta^* \in L_\eta$ with a witness $w = (\hat{k}, \hat{r})$ satisfying the second clause in (11); that is, for some $\hat{d}$ we have $(c_1, \hat{d}) = \mathsf{CS.Com}(ck, m^*; \hat{r})$. Since we also have $(c_1, d_1) = \mathsf{CS.Com}(ck, m; r_1)$ and moreover $m^* \neq m$, this again means that $\mathcal{A}$ broke the binding property of CS.

We therefore conclude that $\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{S},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$. □

A.2 Proof of Theorem 2

Boyle et al. [BGI14, Thm 3.3] prove that $(\mathsf{FS.Setup}', \mathsf{FS.KeyGen}', \mathsf{FS.Sign}', \mathsf{FS.Verify}')$ is a functional signature scheme that is both correct and unforgeable but neither function private nor succinct. We now prove that $\mathsf{FS.OSmp}'$ satisfies both indistinguishability (Fig. 8) and oblivious unforgeability (Fig. 9) of Def. 14, and therefore conclude that FS' has furthermore obliviously samplable keys.

Informally, as a master verification key is simply a verification key for a signature scheme with obliviously samplable signatures, and a functional signing key is simply a standard signature, an obliviously samplable pair of a (standard) verification key and a signature, translates to a pair of a master verification key and a functional signing key. Therefore indistinguishability of honestly generated and obliviously sampled pairs of master verification keys and functional signing keys, reduces to the indistinguishability of honestly generated and obliviously sampled pairs of standard verification keys and signatures. Similarly, functional oblivious unforgeability reduces to standard oblivious unforgeability. Formal details follow.

We first prove oblivious indistinguishability of honestly generated and obliviously sampled pairs of verification keys and functionals keys. Assume towards contradiction that there exists a polynomial $p(\cdot)$ such that for infinitely many λ, it holds that

$$\epsilon(\lambda) := \left|\Pr\left[\mathbf{Exp}^{\text{ind-0}}_{\mathsf{FS}',\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-1}}_{\mathsf{FS}',\mathcal{A}}(\lambda) = 1\right]\right| \geq \tfrac{1}{p(\lambda)} \ ,$$

where $\mathbf{Exp}^{\text{ind-}b}_{\mathsf{FS}',\mathcal{A}}(\lambda)$ defined in Fig. 8 is instantiated by FS' of Construction 2.

We use $\mathcal{A}$ to construct a PPT distinguisher $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the oblivious indistinguishability $\mathbf{Exp}^{\text{ind-}b}_{\mathsf{OS},\mathcal{B}}(\lambda)$ (Fig. 5) of OS and breaks (9):

$\underline{\mathcal{B}_1(1^\lambda)}$:
- $(st_{\mathcal{A}}, f) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SS.KeyGen}(1^\lambda)$.
- $st := (st_{\mathcal{A}}, f, \mathsf{sk}, \mathsf{vk})$.
- Return $(st, f\|\mathsf{vk})$.
- $st := (st_{\mathcal{A}}, f, mvk, sk_f)$.

$\underline{\mathcal{B}_2(st, mvk, sk_f)}$: // $(mvk, \sigma_{f\|\mathsf{vk}})$ is either honestly generated or obliviously sampled
- $sk_f := (f\|\mathsf{vk}, \sigma_{f\|\mathsf{vk}}, \mathsf{sk})$.
- Output $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, mvk, sk_f)$.

By construction, if $(mvk, \sigma_{f\|\mathsf{vk}})$ is honestly generated then $\mathcal{B}$ simulates $\mathbf{Exp}^{\text{ind-0}}_{\mathsf{OS},\mathcal{A}}$, and if it is obliviously sampled then $\mathcal{B}$ simulates $\mathbf{Exp}^{\text{ind-1}}_{\mathsf{OS},\mathcal{A}}$. Therefore, for infinitely many λ it holds that

$$\begin{aligned}\tfrac{1}{p(\lambda)} &\leq \left|\Pr\left[\mathbf{Exp}^{\text{ind-0}}_{\mathsf{FS}',\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-1}}_{\mathsf{FS}',\mathcal{A}}(\lambda)=1\right]\right| \\ &= \left|\Pr\left[\mathbf{Exp}^{\text{ind-0}}_{\mathsf{OS},\mathcal{B}}(\lambda)=1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-1}}_{\mathsf{OS},\mathcal{B}}(\lambda)=1\right]\right| \ .\end{aligned}$$

Therefore we reach a contradiction to (9) the oblivious indistinguishability of OS, and hence conclude that $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.

We now prove oblivious unforgeability. Assume towards contradiction that there exist a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a polynomial $p(\cdot)$ such that for infinitely many λ, it holds that

$$\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}',\mathcal{A}}(\lambda)=1\right] \geq \tfrac{1}{p(\lambda)} \ ,$$

where $\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}',\mathcal{A}}(\lambda)$ defined in Fig. 9 is instantiated by OSmp of Construction 2.

We use $\mathcal{A}$ to construct a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ to the oblivious-unforgeability game $\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{OS},\mathcal{A}}(\lambda)$ (Fig. 6) of OS and breaks (10) as follows.

$\underline{\mathcal{B}_1(1^\lambda)}$:
- $(st_{\mathcal{A}}, f) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $r_{\mathsf{G}} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $(\mathsf{sk}, \mathsf{vk}) := \mathsf{SS.KeyGen}(1^\lambda; r_{\mathsf{G}})$.
- $st := (st_{\mathcal{A}}, f, \mathsf{sk}, \mathsf{vk})$.
- Return $(st, f\|\mathsf{vk})$.

$\underline{\mathcal{B}_2(st, r_{\mathsf{O}})}$: // r_{O} is s.t. $(mvk, \sigma_{f\|\mathsf{vk}}) := \mathsf{OS.OSmp}(1^\lambda, f\|\mathsf{vk}; r_{\mathsf{O}})$.
- $(m^*, (f^*\|\mathsf{vk}^*, \sigma^*_{fv}, w^*, \sigma^*_w)) \leftarrow \mathcal{A}_2(st, r_{\mathsf{G}}\|r_{\mathsf{O}})$.
- Output $(f^*\|\mathsf{vk}^*, \sigma^*_{fv})$.

By assumption $(m^*, \sigma^* := (f^*\|\mathsf{vk}^*, \sigma^*_{fv}, w^*, \sigma^*_w))$ is a valid forgery and thus satisfies

$$m^* \notin \mathcal{R}_f \ \wedge \ \mathsf{FS.Verify}'(mvk, m^*, \sigma^*) = 1, \ \text{i.e.,}$$
$$\mathsf{OS.Verify}(mvk, f^*\|\mathsf{vk}^*, \sigma^*_{fv}) = 1 = \mathsf{SSVerify}(\mathsf{vk}^*, w^*, \sigma^*_w) \ \wedge \ m^* = f^*(w^*) \ . \quad (19)$$

From (19), together with $m^* \notin \mathcal{R}_f$, we have $f^* \neq f$, thus $f^*\|vk^* \neq f\|vk$ and thus $(f^*\|\mathsf{vk}^*, \sigma^*_{fv})$ is a forgery to OS and therefore

$$\tfrac{1}{p(\lambda)} \leq \Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}',\mathcal{A}}(\lambda)=1\right] = \Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{OS},\mathcal{B}}(\lambda)=1\right] \ .$$

A contradiction to the oblivious unforgeability (10) of OS, and hence $\Pr\left[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}',\mathcal{A}}(\lambda)=1\right] = \mathsf{negl}(\lambda)$, which concludes the proof of Theorem 2.

A.3 Proof of Theorem 4

Proof of Proposition 1. Assume towards 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, it holds that

$$\epsilon(\lambda) := \left| \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0)}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(1)}(\lambda) = 1] \right| \geq \tfrac{1}{p(\lambda)} \ .$$

Then we use $\mathcal{A}$ to construct a series of PPT adversaries $\mathcal{D}^{(i)} = \left(\mathcal{D}_1^{(i)}, \mathcal{D}_2^{(i)}, \mathcal{D}_3^{(i)}, \mathcal{D}_4^{(i)}\right)$, $i = 1, \ldots, q$, one of which breaks function privacy of FS.

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 $sk_{f_{x^*}}$, 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 $\mathbf{Exp}^{\text{priv-}d}_{\mathsf{FS},\mathcal{D}^{(i)}}(\lambda)$ of FS (cf. Fig. 4) and simulates $\mathbf{Exp}^{b,(0,i-1)}$ if $\mathcal{D}^{(i)}$'s challenger bit $d = 0$ and $\mathbf{Exp}^{b,(0,i)}$ if $d = 1$.

$\underline{\mathcal{D}_1^{(i)}(\lambda, msk, mvk)}$
- $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $H \leftarrow \mathsf{H.Smp}(1^\lambda)$.
- $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.
- $\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P_{H,crs,mvk,k})$ with $P_{H,crs,mvk,k}$ defined in (15).
- Let f_I and f_{x^*} be defined as in (16).
- $pp := (H, crs, mvk, \widetilde{P})$.
- $st_1 := (x^*, st_{\mathcal{A}}, pp, k, f_I, f_{x^*})$.
- Return $(f_0 := f_I, st_1)$, where f_I is padded to be of length $|f_{x^*}|$.

$\underline{\mathcal{D}_2^{(i)}(st_1, sk_{f_I})}$
- Return $(f_1 := f_{x^*}, st_2 := (st_1, sk_{f_I}))$.

$\underline{\mathcal{D}_3^{(i)}(st_2, sk_{f_{x^*}})}$
- If $b = 1$ then $y^* := \mathsf{PF.Eval}(k, H(x^*))$; otherwise $y^* \leftarrow \mathcal{Y}$.
- $b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st_{\mathcal{A}}, y^*)$;
  - simulate $\textsc{Eval}(x)$:
    if $x = x^*$, reply $\perp$; else reply $y := \mathsf{PF.Eval}(k, H(x))$;
  - simulate $\textsc{Constr}(M)$:
    if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$; else do the following:
    - first $i-1$ queries: compute $(M, \sigma) \leftarrow \mathsf{FS.Sign}(f_{x^*}, sk_{f_{x^*}}, M)$; reply $k_M := (M, \sigma, pp)$.
    - i-th query M: set $st_3 := (st_2, sk_{f_{x^*}})$ and return $(m_0, m_1) := (M, M)$ to own challenger.

$\underline{\mathcal{D}_4^{(i)}(st_3, (M, \sigma_c))}$ // σ_c is either a signature on M under sk_{f_I} or under $sk_{f_{x^*}}$.
- Finish the $\textsc{Constr}$ query reply for $\mathcal{A}_2$ with $k_M := (M, \sigma_c, pp)$.
- Simulate $\textsc{Eval}$ queries like $\mathcal{D}_3^{(i)}$.
- Simulate further $\textsc{Constr}$ queries:
  if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$;
  else $(M, \sigma) \leftarrow \mathsf{FS.Sign}(I, sk_{f_I}, M)$; reply $k_M := (M, \sigma, pp)$.
- Output b'.

If σ_c was generated using sk_{f_I} then $\mathcal{D}^{(i)}$ simulates $\mathbf{Exp}^{b,(0,i-1)}$ and if $sk_{f_{x^*}}$ 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 FS, are independent of $|f_I|$. We therefore have

$$\Pr[\mathbf{Exp}^{\text{priv-}d}_{\mathsf{FS},\mathcal{D}^{(i)}}(\lambda)=1]=\Pr[\mathbf{Exp}^{b,(0,i-1+d)}_{\mathcal{A}}=1]\ . \tag{20}$$

We assumed that

$$\begin{aligned}\tfrac{1}{p(\lambda)} &\le \left|\Pr[\mathbf{Exp}^{b,(0)}_{\mathcal{A}}(\lambda)=1]-\Pr[\mathbf{Exp}^{b,(1)}_{\mathcal{A}}(\lambda)=1]\right| \\ &\le \textstyle\sum_{i=1}^{q}\left|\Pr[\mathbf{Exp}^{b,(0,i-1)}_{\mathcal{A}}(\lambda)=1]-[\mathbf{Exp}^{b,(0,i)}_{\mathcal{A}}(\lambda)=1]\right|\ .\end{aligned}$$

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

$$\begin{aligned}\tfrac{1}{q(\lambda)\cdot p(\lambda)} &\le \left|\Pr[\mathbf{Exp}^{b,(0,i-1)}_{\mathcal{A}}(\lambda)=1]-[\mathbf{Exp}^{b,(0,i)}_{\mathcal{A}}(\lambda)=1]\right| \\ &\overset{(20)}{=} \left|\Pr[\mathbf{Exp}^{\text{priv-0}}_{\mathcal{D}^{(i)}}(\lambda)=1]-\Pr[\mathbf{Exp}^{\text{priv-1}}_{\mathcal{D}^{(i)}}(\lambda)=1]\right|\ .\end{aligned}$$

A contradiction to the function privacy of FS, and therefore $\epsilon(\lambda)=\mathsf{negl}(\lambda)$.

Proof of Proposition 2. Assume towards 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 it holds that

$$\epsilon(\lambda):=\left|\Pr[\mathbf{Exp}^{b,(1)}_{\mathcal{A}}(\lambda)=1]-\Pr[\mathbf{Exp}^{b,(2)}_{\mathcal{A}}(\lambda)=1]\right|\ge\tfrac{1}{p(\lambda)}\ .$$

Then we use $\mathcal{A}$ to construct a PPT adversary $\mathcal{B}=(\mathcal{B}_1,\mathcal{B}_2)$ which breaks the oblivious-indistinguishability $\mathbf{Exp}^{\text{ind-}b}_{\mathsf{FS},\mathcal{B}}(\lambda)$ of FS (cf. Fig. 8) as follows.

$\underline{\mathcal{B}_1(1^\lambda)}$
- $(x^*,st_{\mathcal{A}})\leftarrow\mathcal{A}_1(1^\lambda)$.
- $H\leftarrow\mathsf{H.Smp}(1^\lambda)$.
- $crs\leftarrow\{0,1\}^{\mathsf{poly}(\lambda)}$.
- $k\leftarrow\mathsf{PF.Smp}(1^\lambda)$.
- $st:=(x^*,st_{\mathcal{A}},H,crs,k)$.
- Return (st_1,f_{x^*}).

$\underline{\mathcal{B}_2(st_1,mvk,sk_{f_{x^*}})}$ //$(mvk,sk_{f_{x^*}})$ is either honestly generated or obliviously sampled.
- $\widetilde{P}\leftarrow\mathsf{diO}(1^\lambda,P_{H,crs,mvk,k})$.
- $pp:=(H,crs,mvk,\widetilde{P})$.
- If $b=1$, $y^*:=\mathsf{PF.Eval}(k,H(x^*))$; else $y^*\leftarrow\mathcal{Y}$.
- $b'\leftarrow\mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st_{\mathcal{A}},y^*)$;
  - simulate $\textsc{Eval}(x)$: if $x=x^*$, reply $\perp$; else reply $y:=\mathsf{PF.Eval}(k,H(x))$;
  - simulate $\textsc{Constr}(M)$: if $M\notin\mathcal{M}_\lambda\vee M(x^*)=1$, reply $\perp$; else $(M,\sigma)\leftarrow\mathsf{FS.Sign}(f_{x^*},sk_{f_{x^*}},M)$ and reply $k_M:=(M,\sigma,pp)$.
- Output b'.

By construction if $(mvk,sk_{f_{x^*}})$ is honestly generated, then $\mathcal{B}$ simulates $\mathbf{Exp}^{b,(1)}$ and if $(mvk,sk_{f_{x^*}})$ is obliviously sampled, then $\mathcal{B}$ simulates $\mathbf{Exp}^{b,(2)}$. Therefore for infinitely many λ, it holds that

$$\tfrac{1}{p(\lambda)}\le\left|\Pr[\mathbf{Exp}^{b,(1)}_{\mathcal{A}}(\lambda)=1]-\Pr[\mathbf{Exp}^{b,(2)}_{\mathcal{A}}(\lambda)=1]\right|=\left|\mathbf{Exp}^{\text{ind-0}}_{\mathsf{FS},\mathcal{B}}(\lambda)-\mathbf{Exp}^{\text{ind-1}}_{\mathsf{FS},\mathcal{B}}(\lambda)\right|\ .$$

We therefore reach a contradiction to the oblivious indistinguishability of FS (cf. Fig. 8) and hence conclude that $\epsilon(\lambda)=\mathsf{negl}(\lambda)$.

Proof of Proposition 3. Assume towards 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, it holds that

$$\epsilon(\lambda) := \left| \Pr[\mathbf{Exp}^{b,(2)}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{b,(3)}_{\mathcal{A}}(\lambda) = 1] \right| \geq \tfrac{1}{p(\lambda)} \ .$$

We use $\mathcal{A}$ to construct a PPT adversary which distinguishes public-coin $di\mathcal{O}$ obfuscations. We first define a public-coin sampler Samp:

$\underline{\mathsf{Samp}(1^\lambda, r := r_{\mathcal{A}} \| r_H \| r_S \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y)}$ $\quad //r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.

- $(x^*, st_{\mathcal{A}}) := \mathcal{A}_1(1^\lambda; r_{\mathcal{A}})$.
- $H := \mathsf{H.Samp}(1^\lambda; r_H)$; $h^* := H(x^*)$.
- $crs := r_S$.
- $(mvk, sk_{f_{x^*}}) := \mathsf{FS.OSmp}(1^\lambda, f_{x^*}; r_{\mathsf{FS}})$ with f_{x^*} as defined in (16).
- $k := \mathsf{PF.Smp}(1^\lambda; r_k)$; $k_{h^*} := \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\}; r_c)$.
- Construct $P_0 := P_{H,crs,mvk,k}$ and $P_1 := P_{H,crs,mvk,k_{h^*}}$ as defined in (15).
- Return (P_0, P_1).

Then we define a public-coin $di\mathcal{O}$ distinguisher $\mathcal{D}$ as follows:

$\underline{\mathcal{D}(r, \widetilde{P}_u)}$ $\quad //\widetilde{P}_u$ is either $\widetilde{P}_0 \leftarrow \mathsf{diO}(1^\lambda, P_0)$ or $\widetilde{P}_1 \leftarrow \mathsf{diO}(1^\lambda, P_1)$.

- Use r to generate $pp := (H, crs, mvk, \widetilde{P}_u)$, x^*, $st_{\mathcal{A}}$, $sk_{f_{x^*}}$ as Samp does.
- $y^* := \mathsf{PF.Eval}(k, H(x^*))$ if $b = 1$; otherwise $y^* \leftarrow \mathcal{Y}$ using randomness r_y.
- $b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot), \textsc{Eval}(\cdot)}(st_{\mathcal{A}}, y^*)$;
  - simulate $\textsc{Constr}(M)$:
    if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$;
    else $(M, \sigma) \leftarrow \mathsf{FS.Sign}(f_{x^*}, sk_{f_{x^*}}, M)$ and reply $k_M := (M, \sigma, pp)$;
  - simulate $\textsc{Eval}(x)$:
    if $x = x^*$, reply $\perp$; else reply $y := \mathsf{PF.Eval}(k, H(x))$.
- Output b'.

By construction, if $\widetilde{P}_u$ is an obfuscation of P_0 then Samp and $\mathcal{D}$ together simulate $\mathbf{Exp}^{b,(2)}$ for $\mathcal{A}$, if it is an obfuscation of P_1 then they simulate $\mathbf{Exp}^{b,(3)}$ for $\mathcal{A}$. Thus for infinitely many λ, it holds that

$$\begin{aligned}
\frac{1}{p(\lambda)} &\leq \left| \Pr[\mathbf{Exp}^{b,(2)}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{b,(3)}_{\mathcal{A}}(\lambda) = 1] \right| = \\
&= \Big| \Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (P_0, P_1) \leftarrow \mathsf{Samp}(1^\lambda, r); \widetilde{P}_0 \leftarrow \mathsf{diO}(1^\lambda, P_0) : \ 1 \leftarrow \mathcal{D}(r, \widetilde{P}_0)\right] - \\
&\quad \Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (P_0, P_1) \leftarrow \mathsf{Samp}(1^\lambda, r); \widetilde{P}_1 \leftarrow \mathsf{diO}(1^\lambda, P_1) : \ 1 \leftarrow \mathcal{D}(r, \widetilde{P}_1)\right] \Big| \ .
\end{aligned}$$

By security of $di\mathcal{O}$ (Def. 5), this means that Samp cannot satisfy Def. 4. Thus, there exists a non-uniform PPT extractor $\mathcal{E}$ that on input r finds an input $\chi := (M, h, \pi, \sigma)$ on which P_0 and P_1 differ. That is, for some polynomial $\ell(\cdot)$ and for infinitely many λ, it holds that

$$\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (P_0, P_1) \leftarrow \mathsf{Samp}(1^\lambda, r); \chi \leftarrow \mathcal{E}(1^\lambda, r) : \ P_0(\chi) \neq P_1(\chi)\right] \geq \frac{1}{\ell(\lambda)} \ . \tag{21}$$

Let $\hat{\chi} := (\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$ be a differing input output by $\mathcal{E}$. Recall that $\hat{\sigma}$ is a signature on the description of 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 $H(x) = \hat{h}$. Since $\hat{\chi}$ is a differing input, by the definitions of $P_0 := P_{H,crs,mvk,k}$ and $P_1 := P_{H,crs,mvk,k_{h^*}}$ (cf. (15)), the following two conditions must hold.

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

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 $\mathcal{E}$ gets a signing key $sk_{f_{x^*}}$, with which it can only sign machines M with $M(x^*) = 0$. So if it outputs $\hat{M}$ with $\hat{M}(x^*) = 1$ then $(\hat{M}, \hat{\sigma})$ is a valid signature by condition(1) and hence a forgery.

Claim 1. *Let* Samp *be as defined above and* $\mathcal{E}$ *be the diO extractor guaranteed by* (21) *and* $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma})$ *its output. If* FS *is an unforgeable with obliviously samplable keys, then* $\hat{M}(x^*) = 0$ *with overwhelming probability.*

Proof. Assume towards contradiction that $\hat{M}(x^*) = 1$, then we construct a PPT adversary $\mathcal{A}^{\text{forg}}$ against the oblivious unforgeability of FS $\mathbf{Exp}^{\text{obl-uf}}(\lambda)$ (Fig. 9) such that $\Pr[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}, \mathcal{A}^{\text{forg}}}(\lambda) = 1] \geq \frac{1}{\ell(\lambda)}$. $\mathcal{A}^{\text{forg}}$, defined below, behaves exactly like Samp and obtains its forgery by running $\mathcal{E}$. Here we make an essential use of the obliviously samplable signing keys functionality guaranteed by the PPT algorithm FS.OSmp as defined in Def. 14.

$\underline{\mathcal{A}^{\text{forg}}_1(1^\lambda)}$
- $r_{\mathcal{A}} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $(x^*, st_{\mathcal{A}}) := \mathcal{A}_1(1^\lambda; r_{\mathcal{A}})$.
- $st := (x^*, st_{\mathcal{A}}, r_{\mathcal{A}})$.
- Return (st, f_{x^*}) with f_{x^*} defined in (16).

$\underline{\mathcal{A}^{\text{forg}}_2(st, r_{\mathsf{FS}})}$
- $r_H, r_S, r_k, r_c, r_y \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $r := r_{\mathcal{A}} \| r_H \| r_S \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y$.
- $H := \mathsf{H.Smp}(1^\lambda; r_H)$ and $h^* := H(x^*)$.
- $crs := r_S$.
- $(mvk, sk_{f_{x^*}}) := \mathsf{FS.OSmp}(1^\lambda, f_{x^*}; r_{\mathsf{FS}})$
- $k := \mathsf{PF.Smp}(1^\lambda; r_k)$.
- $k_{h^*} := \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\}; r_c)$.
- $P_0 := P_{H, crs, mvk, k}$ and $P_1 := P_{H, crs, mvk, k_{h^*}}$ defined in (15).
- $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma}) \leftarrow \mathcal{E}(1^\lambda, r)$.
- Output $(\hat{M}, \hat{\sigma})$.

By condition(1), $(\hat{M}, \hat{\sigma})$ satisfies $\mathsf{FS.Verify}(mvk, \hat{M}, \hat{\sigma}) = 1$. 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 $\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}, \mathcal{A}^{\text{forg}}}(\lambda) = 1$.

Consequently, $\Pr[\mathbf{Exp}^{\text{obl-uf}}_{\mathsf{FS}, \mathcal{A}^{\text{forg}}}(\lambda) = 1] \geq \frac{1}{\ell(\lambda)}$, a contradiction to the unforgeability of functional signatures with respect to obliviously samplable signing keys, and therefore $\hat{M}(x^*) = 0$. □

Since the SNARK $\hat{\pi}$ extracted by $\mathcal{E}$ is a proof of knowledge, we can extract a witness $\hat{x}$ for it. In order to formally apply item 3 of Def. 8, we first construct a PPT algorithm $\mathcal{A}_{\mathsf{snrk}}$ that outputs $\hat{\pi}$ together with the statement. $\mathcal{A}_{\mathsf{snrk}}$ simply runs Samp and $\mathcal{E}$ as defined above, except that it embeds the *crs* from its input into the randomness.

$\underline{\mathcal{A}_{\mathsf{snrk}}(crs; r)}$ // r is $\mathcal{A}_{\mathsf{snrk}}$'s randomness.
- Parse $r_{\mathcal{A}} \| r_H \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y := r$.
- Simulate $\mathsf{Samp}(1^\lambda; r_{\mathcal{A}} \| r_H \| crs \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y)$; let H be the sampled hash function.
- $(\hat{M}, \hat{h}, \hat{\pi}, \hat{\sigma}) \leftarrow \mathcal{E}(1^\lambda, r_{\mathcal{A}} \| r_H \| crs \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y)$.
- Output $(\eta := (H, \hat{M}, \hat{h}), \hat{\pi})$.

By the construction of $\mathcal{A}_{\mathsf{snrk}}$, (21) and $\mathsf{condition}(1)$ we have that

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; ((H,\hat{M},\hat{h}),\hat{\pi}) \leftarrow \mathcal{A}_{\mathsf{snrk}}(crs) : \mathsf{V}(crs,(H,\hat{M},\hat{h}),\pi)=1\right] \geq \frac{1}{\ell(\lambda)} \ . \tag{22}$$

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

$$\Pr\left[\begin{array}{l} crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; \\ ((H,\hat{M},\hat{h}),\hat{\pi}) := \mathcal{A}_{\mathsf{snrk}}(crs;r); \\ \hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(crs,r) \end{array} : \begin{array}{l} \mathsf{V}(crs,(H,\hat{M},\hat{h}),\hat{\pi})=1 \\ \wedge\ ((H,\hat{M},\hat{h}),\hat{x}) \notin R_{legit} \end{array}\right] = \mathsf{negl}(\lambda),$$

which together with (22) yields:

$$\Pr\left[\begin{array}{l} crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)};\ r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; \\ ((H,\hat{M},\hat{h}),\hat{\pi}) := \mathcal{A}_{\mathsf{snrk}}(crs;r); \\ \hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(crs,r) \end{array} : ((H,\hat{M},\hat{h}),\hat{x}) \in R_{legit}\right] \geq \frac{1}{\ell(\lambda)} - \mathsf{negl}(\lambda) \ . \tag{23}$$

We now construct an adversary $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ against $\mathcal{H}$ that on input λ, and coins r_H used to sample a function H outputs a collision for H: Given r_H, $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ generates a CRS for SNARKs, then runs $\mathcal{A}_{\mathsf{snrk}}(crs)$, but using the randomness r_H from its input, and then runs $\mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}$ to extract a collision:

$\underline{\mathcal{A}_{\mathsf{cll\text{-}fnd}}(1^\lambda, r_H)}$

- Sample $r_{\mathcal{A}}, crs, r_k, r_{\mathsf{FS}}, r_c, r_y \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- Simulate $\mathsf{Samp}(1^\lambda; r_{\mathcal{A}} \| r_H \| crs \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y)$;
  let x^* be the challenge of $\mathcal{A}$.
  let H be the sampled hash function.
- $(\hat{M},\hat{h},\hat{\pi},\hat{\sigma}) \leftarrow \mathcal{E}(1^\lambda, r_{\mathcal{A}} \| r_H \| crs \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y)$.
- $\hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(crs, r_{\mathcal{A}} \| r_H \| r_k \| r_{\mathsf{FS}} \| r_c \| r_y)$.
- Output $(\hat{x}, x^*)$ as a collision pair for H.

By (23), 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:

$$\Pr\left[\begin{array}{l} r_H \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; H := \mathsf{H.Smp}(1^\lambda, r_H); \\ (\hat{x},x^*) \leftarrow \mathcal{A}_{\mathsf{cll\text{-}fnd}}(1^\lambda, r_H) \end{array} : \hat{x} \neq x^* \wedge H(\hat{x}) = H(x^*) = \hat{h}\right] \ .$$

Therefore we reach a contradiction to collision resistance of $\mathcal{H}$, and it must be that $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.

Proof of Proposition 4. The only difference between games $\mathbf{Exp}^{b,(3)}$ and $\mathbf{Exp}^{b,(4)}$ is when $\mathcal{A}$ queries $\textsc{Eval}$ on x with $H(x)=H(x^*)$. Then $\mathbf{Exp}^{b,(4)}$ aborts, while on any other query the oracle $\textsc{Eval}$ 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 $(1^\lambda, r_H)$ simulates $\mathbf{Exp}^{b,(4)}$ (except that it uses $H := \mathsf{H.Smp}(1^\lambda; r_H)$ instead of sampling H) until in an oracle query $\textsc{Eval}(x)$ 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.

Proof of Proposition 5. Assume towards 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 it holds that

$$\epsilon(\lambda) := \left| \Pr[\mathbf{Exp}^{0,(4)}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{1,(4)}_{\mathcal{A}}(\lambda) = 1] \right| \geq \tfrac{1}{p(\lambda)} \ .$$

Then we construct a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ playing $\mathbf{Exp}^{\text{pct-}b}_{\mathcal{B}}(\lambda)$, the selective-security game of $\mathcal{PF}$ (cf. Fig. 2) as follows. (Note $\mathcal{B}_2$ does not use $\textsc{Eval}(\cdot)$)

$\underline{\mathcal{B}_1(1^\lambda)}$

- $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $H \leftarrow \mathsf{H.Smp}(1^\lambda)$.
- $st := (H, x^*, st_{\mathcal{A}})$.
- $h^* := H(x^*)$.
- $T := \{h^*\}$.
- Return (h^*, T, st).

$\underline{\mathcal{B}_2^{\textsc{Eval}(\cdot)}(st, k_{h^*}, y^*)}$ // y^* is $\mathsf{PF.Eval}(k, H(x^*))$ or random.

- $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $(mvk, sk_{f_{x^*}}) \leftarrow \mathsf{FS.OSmp}(1^\lambda, f_{x^*})$.
- $\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P_{H,crs,mvk,k_{h^*}})$.
- $pp := (H, crs, mvk, \widetilde{P})$.
- $b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st_{\mathcal{A}}, y^*)$.
  - simulate $\textsc{Constr}(M)$:
    if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\perp$;
    else $(M, \sigma) \leftarrow \mathsf{FS.Sign}(f_{x^*}, sk_{f_{x^*}}, M)$ and reply $k_M := (M, \sigma, pp)$;
  - simulate $\textsc{Eval}(x)$:
    if $x = x^*$, reply $\perp$; if $H(x) = H(x^*)$ abort;
    else reply $y := \mathsf{PF.Eval}(k_{h^*}, H(x))$.
- Output b'.

By construction $\Pr[\mathbf{Exp}^{\text{pct-}b}_{\mathcal{B}}(\lambda) = 1] = \Pr[\mathbf{Exp}^{b,(4)}_{\mathcal{A}}(\lambda) = 1]$, and therefore for infinitely many λ it holds $\left| \Pr[\mathbf{Exp}^{\text{pct-0}}_{\mathcal{B}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{pct-1}}_{\mathcal{B}}(\lambda) = 1] \right| \geq \frac{1}{p(\lambda)}$. This contradicts the selective security of PF, and we conclude that $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.