Functional Signatures and Pseudorandom Functions

Elette Boyle[1,⋆], Shafi Goldwasser[2,3,⋆⋆, ⋆⋆⋆], and Ioana Ivan[2]

[1] Technion – Israel Institute of Technology
eboyle@alum.mit.edu
[2] MIT CSAIL
shafi@theory.csail.mit.edu, ioanai@mit.edu
[3] Weizmann Institute of Science

Abstract. We introduce two new cryptographic primitives: *functional digital signatures* and *functional pseudorandom functions*.
In a functional signature scheme, in addition to a master signing key that can be used to sign any message, there are *signing keys for a function* f, which allow one to sign any message in the range of f. As a special case, this implies the ability to generate keys for predicates P, which allow one to sign any message m for which $P(m) = 1$.
We show applications of functional signatures to constructing succinct non-interactive arguments and delegation schemes. We give several general constructions for this primitive based on different computational hardness assumptions, and describe the trade-offs between them in terms of the assumptions they require and the size of the signatures.
In a functional pseudorandom function, in addition to a master secret key that can be used to evaluate the pseudorandom function F on any point in the domain, there are additional *secret keys for a function* f, which allow one to evaluate F on any y for which there exists an x such that $f(x) = y$. As a special case, this implies *pseudorandom functions with selective access*, where one can delegate the ability to evaluate the pseudorandom function on inputs y for which a predicate $P(y) = 1$ holds. We define and provide a sample construction of a functional pseudorandom function family for prefix-fixing functions. This construction yields, in particular, *punctured pseudorandom functions*, which have proven an invaluable tool in recent advances in obfuscation (Sahai and Waters ePrint 2013).

⋆ The research of the first author has received funding from the European Union's Tenth Framework Programme (FP10/ 2010-2016) under grant agreement no. 259426 ERC-CaC. This work was primarily completed while the first author was a student at MIT.
⋆⋆ This work was supported in part by Trustworthy Computing: NSF CCF-1018064.
⋆⋆⋆ This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

1 Introduction

We introduce new cryptographic primitives with a variety of accompanying constructions: *functional digital signatures (FDS)*, *functional pseudorandom functions (F-PRF)*, and *psuedorandom functions with selective access (PRF-SA)*.[4]

Functional Signatures

In digital signature schemes, as defined by Diffie and Hellman [11], a signature on a message provides information which enables the receiver to verify that the message has been created by a proclaimed sender. The sender has a secret *signing key*, used in the signing process, and there is a corresponding verification key, which is public and can be used by anyone to verify that a signature is valid. Following Goldwasser, Micali and Rackoff [20], the standard security requirement for signature schemes is unforgeability against chosen-message attack: an adversary that runs in probabilistic polynomial time and is allowed to request signatures for a polynomial number of messages of his choice, cannot produce a signature of any new message with non-negligible probability.

In this work, we extend the classical digital signature notion to what we call *functional signatures*. In a functional signature scheme, in addition to a *master signing key* that can be used to sign any message, there are secondary *signing keys for functions* f (called sk_f), which allow one to sign any message in the range of f. These additional keys are derived from the master signing key. The notion of security we require such a signature scheme to satisfy is that any probabilistic polynomial time (PPT) adversary, who can request signing keys for functions $f_1 \dots f_l$ of his choice, and signatures for messages $m_1, \dots m_q$ of his choice, can only produce a signature of a message m with non-negligible probability, if m is equal to one of the queried messages $m_1, \dots m_q$, or if m is in the range of one of the queried functions $f_1 \dots f_l$.

An immediate application of a functional signature scheme is the ability to delegate the signing process from a master authority to another party. Suppose someone wants to allow their assistant to sign on their behalf only those messages with a certain tag, such as "signed by the assistant". Let P be a predicate that outputs 1 on messages with the proper tag, and 0 on all other messages. In order to delegate the signing of this restricted set of messages, one would give the assistant a signing key for the following function:

$$f(m) := \begin{cases} m \text{ if } P(m) = 1 \\ \perp \text{ otherwise} \end{cases} .$$

[4] We note that independently the notion of pseudorandom functions with selective access was studied by Boneh-Waters under the name of *constrained pseudorandom functions* [9] and by Kiayias, Papadopoulos, Triandopoulos and Zacharias under the name *delegatable pseudorandom functions* [23]. Subsequent to our posting of an earlier manuscript of this work, [4] and [2] have additionally posted similar results on functional signatures.

P could also be a predicate that checks if the message does not contain a given phrase, if it is related to a certain subject, or if it satisfies a more complex policy.

Another application of functional signatures is to certify that only allowable computations were performed on data. For example, imagine the setting of a digital camera that produces signed photos (i.e the original photos produced by the camera can be certified). In this case, one may want to allow photo-processing software to perform minor touch-ups of the photos, such as changing the color scale or removing red-eyes, but not allow more significant changes such as merging two photos or cropping a picture. Functional signatures can naturally address this problem by providing the photo processing software with keys which enable it to sign only the allowable modifications of an original photograph. Generalizing, we think of a client and a server (e.g. photo-processing software), where the client provides the server with data (e.g. signed original photos, text documents, medical data) which he wants to be processed in a restricted fashion.A functional signature of the processed data provides proof of allowable processing.

Functional signatures can also be used to construct a delegation scheme. In this setting, there is a client who wants to allow a more powerful server to compute a function f on inputs chosen by the client, and wants to be able to verify that the result returned by the server is correct. The verification process should be more efficient than for the client to compute f himself. The client can give the server a key for the function $f'(x) = (f(x)|x)$. To prove that $y = f(x)$, the prover gives the client a signature of $y|x$, which he could only have obtained if $y|x$ is in the range of f'; that is, if $y = f(x)$.

A desirable property of a functional signature scheme is *function privacy*: the signature should reveal neither the function f corresponding to the key used in the signing process, nor the message m that f was applied to. In the example with the signed photos, one might not wish to reveal the original image, just that the final photographs were obtained by running one of the allowed functions on some image taken with the camera.

An additional desirable property is *succinctness*: the size of the signature should only depend on the size of the output $f(m)$ and the security parameter (or just the security parameter), rather than the size of the circuit for computing f.

Functional Pseudorandomness

Pseudorandom functions (PRFs), introduced by Goldreich, Goldwasser, and Micali [14], are a family of indexed functions $F = \{F_s\}$ such that: (1) given the index s, F_s can be efficiently evaluated on all inputs, and (2) no probabilistic polynomial-time algorithm *without* s can distinguish evaluations $F_s(x_i)$ for inputs x_i of its choice from random values. Pseudorandom functions are useful for numerous symmetric-key cryptographic applications, including generating passwords, identify-friend-or-foe systems, and symmetric-key encryption schemes secure against chosen-ciphertext attacks.

In this work, we extend pseudorandom functions to a primitive which we call *functional pseudorandom functions (F-PRF)*. The idea is that in addition to a

master secret key (that can be used to evaluate the pseudorandom function F_s on any point in the domain), there are additional *secret keys* sk_f *per function* f, which allow one to evaluate F_s on any y for which there exists x such that $f(x) = y$ (i.e $y \in Range(f)$). An immediate application of such a construct is to specify succinctly the randomness to be used by parties in a randomized distributed protocol with potentially faulty players, so as to force honest behavior. A centralized authority holds an index s of a pseudorandom function F_s. One may think of this authority as providing a service which dispenses pseudorandomness (alternatively, the secret s can be shared among players in an MPC). The authority provides each party id with a secret key s_{id} which enables party id to (1) evaluate $F_s(y)$ whenever $y =$ "$id\|h$", where h corresponds to say the public history of communication, and (2) use $F_s(y)$ as her next sequence of coins in the protocol. To prove that the appropriate randomness was used, id can utilize NIZK proofs. An interesting open question is how to achieve a *verifiable* F-PRF, where there is additional information vk_s that can be used to verify that a given pair $(x, F_s(x))$ is valid, without assuming the existence of an honestly generated common reference string, as in the NIZK setting. Note that in this example the function $f(x) = y$ is simply the function which appends the string prefix id to x. We note that there are many other ways to force the use of proper randomness in MPC protocols by dishonest parties, starting with the classical paradigm [19, 15] where parties interact to execute a "coin flip in the well" protocol forcing players to use the results of these coins, but we find the use of F-PRF appealing in its simplicity, lack of interaction and potential efficiency.

The notion of functional pseudorandom functions has many variations. One natural variant that immediately follows is *PRFs with selective access*, in which secondary keys sk_P can be produced per predicate P to enable computing $F_s(x)$ on inputs x for which $P(x) = 1$. This is a special case of F-PRF, as we can take the secret key for predicate P to be sk_f where $f(x) = x$ if $P(x) = 1$ and $\perp$ otherwise. The special case of *punctured PRFs*, in which secondary keys allow computing $F_s(x)$ on all inputs except one, is similarly implied and has recently been shown to have important applications (e.g., [29, 22]). Another variant is *hierarchical PRFs*, with an additional property that parties with functional keys sk_f may also generate subordinate keys sk_g for functions g of the form $g = f \circ f'$ (i.e., first evaluate f', then evaluate f). Note that the range of such composition g is necessarily contained within the range of f.

1.1 Our Results on Functional Signatures and Their Applications

We provide a construction of functional signatures achieving function privacy and succinctness, assuming the existence of succinct non-interactive arguments of knowledge (SNARKS) and (standard) non-interactive zero-knowledge arguments of knowledge (NIZKAoKs) for NP languages.

As a building block, we first give a construction of a functional signature scheme that is not succinct or function private, based on a much weaker assumption: the existence of one-way functions.

Theorem 1 (Informal). *Based on any one-way function, there exists a functional signature scheme that supports signing keys for any function f computable by a polynomial-sized circuit. This scheme satisfies the unforgeability requirement for functional signatures, but not function privacy or succinctness.*

Overview of the construction: The master signing and verification keys for the functional signature scheme will correspond to a key pair, $(\mathsf{msk}, \mathsf{mvk})$, in an underlying (standard) signature scheme. To generate a signing key for a function f, we sample a fresh signing and verification key pair $(\mathsf{sk}', \mathsf{vk}')$ in the underlying signature scheme, and sign the concatenation $f|\mathsf{vk}'$ using msk. The signing key for f consists of this signature together with sk'. Given this signing key, a user can sign any message $m^* = f(m)$ by signing m using sk', and outputting this signature, together with the signature of $f|\mathsf{vk}'$ given as part of sk_f.

We then now show how to use SNARKs, together with this initial construction, to construct a *succinct*, *function-private* functional signature scheme.

A SNARK system for an NP language L with corresponding relation R is an extractable proof system where the size of a proof is sublinear in the size of the witness corresponding to an instance. SNARKs have been constructed under various non-falsifiable [26] assumptions. Bitansky et al. [6] construct zero-knowledge SNARKs where the length of the proof and the verifier's running time are bounded by a polynomial in the security parameter, and the *logarithm* of running time of the corresponding relation $R(x, w)$, assuming the existence of collision-resistant hash functions and a knowledge-of-exponent assumption.[5] (More details are given in the full version).

Theorem 2 (Informal). *Assuming the existence of SNARK and NIZKAoK for NP, and a functional signature scheme that is not necessarily function-private or succinct, there exists a succinct, function-private functional signature scheme that supports signing keys for the class of polynomial-sized circuits.*

Overview of the construction: In the setup algorithm for our functional signature scheme, we sample a key pair $(\mathsf{msk}, \mathsf{mvk})$ for the underlying (non-succinct, non-function-pivate) functional signature scheme $\mathsf{FS1}$, and a common reference string crs for the SNARK system. We use msk as the new master singing key and $(\mathsf{mvk}, \mathsf{crs})$ as the new master verification key. The sk_f key generation algorithm is the same as in the underlying functional signature scheme $\mathsf{FS1}$. To sign a message m^* using a resulting key sk_f, we generate a zero-knowledge SNARK for the following statement: $\exists\sigma$ such that σ is a valid signature of m^* under mvk in the functional signature scheme $\mathsf{FS1}$. To verify the signature, we run the verification algorithm for the SNARK argument system.

Resorting to non-falsifiable assumptions, albeit strong, seems necessary to obtain succinctness for functional signatures. We show that, given a functional signature scheme with short signatures, we can construct a SNARG system.

[5] In [5], Bitansky et al. also show that any SNARK + NIZKAoK directly yield zero-knowledge (ZK)-SNARK with analogous parameters.

Theorem 3 (Informal). *If there exists a functional signature scheme supporting keys for all polynomial-sized circuits f, with short signatures (i.e., of size $poly(k) \cdot (|f(m)| + |m|)^{o(1)}$ for security parameter k), then there exists a SNARG scheme with preprocessing for any language $L \in NP$ with proof size $poly(k) \cdot (|w| + |x|)^{o(1)}$, where w is the witness and x is the instance.*

The main idea in the SNARG construction is for the verifier (CRS generator) to give out a single signing key sk_f for a function whose range consists of exactly those strings that are in the language L. Then, with sk_f, the prover will be able to sign only those messages x that are in L, and thus can use this (short) signature as his proof.

Gentry and Wichs showed in [13] that SNARG schemes with proof size $poly(k) \cdot (|w| + |x|)^{o(1)}$ cannot be obtained using black-box reductions to falsifiable assumptions. We can thus conclude that in order to obtain a functional signature scheme with signature size $poly(k) \cdot (|f(m)| + |m|)^{o(1)}$ we must either rely on non-falsifiable assumptions (as in our SNARK construction) or make use of non black-box techniques.

Finally, we can construct a scheme which satisfies unforgeability and functional privacy but not succinctness, based on the weaker assumption of non-interactive zero-knowledge arguments of knowledge (NIZKAoK) for NP.

Theorem 4 (Informal). *Assuming the existence of non-interactive zero-knowledge arguments of knowledge (NIZKAoK) for NP, there exists a functional signature scheme that supports signing keys for any function f computable by a polynomial-sized circuit. This scheme satisfies function privacy, but not succinctness: the size of the signature is dependent on the size of f and m.*

Overview of the construction: The construction is analogous to the SNARK-based construction above, with the SNARK replaced with NIZKAoK. Namely, a signature will be a NIZK Argument of Knowledge for the following statement: $\exists \sigma$ such that σ is a valid signature of m^* under mvk, in an underlying non-succinct, non-function-private functional signature scheme, as before (recall such a scheme exists based on OWF). The signature size is now polynomial in the size of σ, which, if $m^* = f(m)$, and sigma was generated using sk_f, is itself polynomial in the security parameter, $|m|$, and $|f|$.

Relation to Delegation: Functional signatures are highly related to delegation schemes. A delegation scheme allows a client to outsource evaluation of a function f to a server, allowing the client to verify the correctness of the computation more efficiently than evaluating f himself. We show that given *any* functional signature scheme supporting a class of functions $\mathcal{F}$, we can obtain a delegation scheme in the preprocessing model for functions in $\mathcal{F}$, with related parameters.

Theorem 5 (Informal). *If there exists a functional signature scheme for function class $\mathcal{F}$, with signature size $s(k)$, and verification time $t(k)$, then there exists a one-round delegation scheme for functions in $\mathcal{F}$, with server message size $s(k)$ and client verification time $t(k)$.*

Overview of the construction: The client gives the server a key $\mathsf{sk}_{f'}$ for the function $f'(x) = (f(x)|x)$. To prove that $y = f(x)$, the prover gives the client a signature of $y|x$, which he could only have obtained if $y|x$ is in the range of f'; that is, if $y = f(x)$. The length of a proof is equal to the length of a signature in the functional signature scheme, $s(k)$, and the verification time for the delegation scheme is equal to the verification time of the functional signature scheme.

1.2 Summary of our Results on Functional Pseudorandom Functions and Selective Pseudorandom Functions

We present formal definitions and constructions of functional pseudorandom functions (F-PRF) and pseudorandom functions with selective access (PRF-SA). In particular, we present a construction based on one-way functions of an F-PRF supporting the class of *prefix-fixing functions*. Our construction is based on the Goldreich-Goldwasser-Micali (GGM) tree-based PRF construction [GGM86].

Theorem 6 (Informal). *Assuming the existence of OWF, there exists an F-PRF supporting keys for the class of prefix-matching functions:* $\mathcal{F}_{\mathsf{pre}} = \{f_z | z \in \{0,1\}^m, m \leq n\}$, *where* $f_z(x) = x$ *if* z *is a prefix of* x, *and* $\perp$ *otherwise. The pseudorandomness property holds against a selective adversary, who declares the functions he will query before seeing the public parameters.*

We remark that one can directly obtain a *fully* secure F-PRF for $\mathcal{F}_{\mathsf{pre}}$, in which security holds against an adversary who adaptively requests key queries, from our selectively secure construction, with a loss of 2^{-n} in security for each functional secret key sk_{f_z} queried by the adversary, via standard complexity leveraging. For appropriate choices of the input length n, security of the underlying OWF, and number of key queries, this still provides desirable security.

Overview of the construction. We show that the original Goldreich-Goldwasser-Micali (GGM) tree-based PRF construction [14] provides the desired functionality, where the functional key sk_f corresponding to a prefix-fixing function $f_z(x) = z_1 z_2 \cdots z_i x_{i+1} \cdots x_n$ will be given by the partial evaluation of the PRF down the tree, at the node corresponding to prefix $z_1 z_2 \cdots z_i$.

This partial evaluation clearly enables a user to compute all possible continuations in the evaluation tree, corresponding to the output of the PRF on any input possessing prefix z. Intuitively, security holds since the other partial evaluations at this level i in the tree still appear random given the evaluation sk_f (indeed, this corresponds to a truncated i-bit input GGM construction).

Punctured pseudorandom functions. Punctured pseudorandom functions [29] are a special case of functional PRFs, where one can generate keys for the function family $\mathcal{F} = \{f_x(y) = y$ if $y \neq x$, and $\perp$ otherwise$\}$. Namely, a key for function f_x allows one to compute the pseudorandom function on any input except for x. Punctured PRFs have recently proven useful as one of the main techniques used in proving the security of various cryptographic primitives based on the existence of indistinguishability obfuscation. Some examples include a construction

of public-key encryption from symmetric-key encryption and the construction of deniable encryption given by Sahai and Waters in [29], as well as an instantiation of random oracles with a concrete hash function for full-domain hash applications by Hohenberger et al. in [22].

We note that the existence of a functional PRF for the prefix-fixing function family gives a construction of punctured PRFs. A key that allows one to compute the PRF on all inputs except $x = x_1 \dots x_n$ consists of n functional keys for the prefix-fixing function family for prefixes: $\bar{x}_1$, $x_1\bar{x}_2$, $x_1x_2\bar{x}_3, \dots, x_1x_2 \cdots x_{n-1}\bar{x}_n$.

Corollary 1 (Informal). *Assuming the existence of OWF, there exists a (selectively secure) punctured PRF for any desired poly-size input length.*

Our construction has the additional beneficial property of *hierarchical key generation*: i.e., a party with a functional key sk_{f_z} for a prefix z may generate valid "subordinate" functional keys $\mathsf{sk}_{f_{z'}}$ for any prefix $z' = z|*$. That is, we prove the following additional statement.

Corollary 2 (Informal). *Assuming the existence of OWF, there exists a (selectively secure)* hierarchical *functional PRF for the class of functions* F_{pre}.

1.3 Other Related Work

Functional Encryption. This work is inspired by recent results on the problem of functional encryption, introduced by Sahai and Waters [28], and formalized by Boneh et al. [8]. In the past few years there has been significant progress on constructing functional encryption schemes for general classes of functions (e.g., [21, 17, 18]). In this setting, a party with access to a master secret key can generate secret keys sk_f for functions f, which allow a third party with sk_f and an encryption of a message m to learn $f(m)$, but nothing else about m. In [17], Goldwasser et al. construct a functional encryption scheme supporting general functions, and secure according to a simulation-based definition, as long as a single key is given out. In [1], Agrawal et al. show that constructing functional encryption schemes achieving this notion of security in the presence of an unbounded number of secret keys is impossible for general functions. In contrast, no such impossibility results are known in the setting of functional signatures.

Connections to Obfuscation. The goal of program obfuscation is to construct a compiler O that takes as input a program P and outputs a program $O(P)$ that preserves the functionality of P, but hides all other information about the original program. Following [3], this is often formalized by requiring that the single-bit output of an efficient adversary given access to an obfuscation of P can be simulated given only black-box access to P. However, Barak et al. [3] show that this definition is unachievable for general functions. Furthermore, in [16], Goldwasser and Kalai give evidence that several natural cryptographic algorithms, including the signing algorithm of any unforgeable signature scheme, are not obfuscatable with respect to this strong definition.

Consider the function $\mathsf{Sign} \circ f$, where Sign is the signing algorithm of an unforgeable signature scheme, f is an arbitrary function and $\circ$ denotes function composition. Based on the results in [16] we would expect this function not to be obfuscatable according to the black-box simulation definition. A meaningful relaxation of the definition is that, while having access to an obfuscation of this function might not hide all information about the signing algorithm, it does not completely reveal the secret key, and does not allow one to sign messages that are not in the range of f. In our function signature scheme, the signing key corresponding to a function f achieves exactly this definition of security, and we can think of it as an obfuscation of $\mathsf{Sign} \circ f$ according to this relaxed definition. Indeed it has recently come to our attention that Barak in an unpublished manuscript has considered *delegatable signatures*, a highly related concept.

Homomorphic Signatures. In a homomorphic signature scheme, a third party is able to perform computations over *already-signed* data, and obtain a new signature that authenticates the resulting message with respect to this computation. In [12], Gennaro and Wichs construct homomorphic (privately verifiable) message authentication codes. For homomorphic signature schemes with public verification, the most general construction of Boneh and Freeman [7] only allows the evaluation of multivariate polynomials on signed data. Constructing homomorphic signature schemes for general functions remains an open problem.

Signatures of correct computation. Papamanthou, Shi and Tamassia consider a notion of functional signatures under the name "signatures of correct computation" [27]. They give constructions for schemes that support operations over multivariate polynomials, such as polynomial evaluation and differentiation. Their schemes are secure in the random oracle model and allow efficient updates to the signing keys: the keys can be updated in time proportional to the number of updated coefficients. In contrast, our constructions that support signing keys for general functions, in the plain model, assuming the existence of succinct non-interactive arguments of knowledge.

Independent work. Finally, as mentioned earlier, related notions to functional PRFs appear in the concurrent and independent works [9, 23]. Based on the Multilinear Decisional Diffie-Hellman assumption (a recently coined assumption related to existence of secure multilinear maps), [9] show that PRFs with Selective Access can be constructed for all predicates describable as polynomial-sized circuits. We remark that this is not equivalent to functional PRFs for polynomial-sized circuits, which additionally captures NP relations (i.e., the predicate $y \in \mathit{Range}(f)$ may not be efficiently testable directly). Subsequent to our posting of an earlier manuscript of this work, [4] and [2] have additionally posted similar results on functional signatures.

1.4 Overview of the paper

In Section 2, we give a formal definition of functional signature schemes, and present three constructions satisfying the definition. In Section 3, we show how to

construct delegation schemes and succinct non-interactive arguments (SNARGs) from functional signatures schemes. In Section 4, we give a formal definition of functional pseudorandom functions and pseudorandom functions with selective access, and present a sample construction for the prefix-fixing function family. In Section 5, we discuss open problems. Due to space constraints, we defer the preliminaries and proofs of theorem statements to the full version of the paper [10].

2 Functional Signatures: Definition and Constructions

We now give a formal definition of a functional signature scheme, specifying the desired unforgeability, function-privacy, and succinctness properties.

Definition 1. *A functional signature scheme for a message space $\mathcal{M}$, and function family $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ consists of algorithms (*$\mathsf{FS.Setup}$*,* $\mathsf{FS.KeyGen}$*,* $\mathsf{FS.Sign}$*,* $\mathsf{FS.Verify}$*):*

- $\mathsf{FS.Setup}(1^k) \to (\mathsf{msk}, \mathsf{mvk})$*: the setup algorithm takes as input the security parameter and outputs the master signing key and master verification key.*
- $\mathsf{FS.KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$*: the key generation algorithm takes as input the master signing key and a function $f \in \mathcal{F}$ (represented as a circuit), and outputs a signing key for f.*
- $\mathsf{FS.Sign}(f, \mathsf{sk}_f, m) \to (f(m), \sigma)$*: the signing algorithm takes as input the signing key for a function $f \in \mathcal{F}$ and an input $m \in \mathcal{D}_f$, and outputs $f(m)$ and a signature of $f(m)$.*
- $\mathsf{FS.Verify}(\mathsf{mvk}, m^*, \sigma) \to \{0, 1\}$*: the verification algorithm takes as input the master verification key* mvk*, a message m and a signature σ, and outputs 1 if the signature is valid.*

We require the following conditions to hold:

***Correctness**:*
$\forall f \in \mathcal{F}, \forall m \in \mathcal{D}_f, (\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^k), \mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$,
$(m^*, \sigma) \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_f, m)$, *it holds that* $\mathsf{FS.Verify}(\mathsf{mvk}, m^*, \sigma) = 1$.

***Unforgeability**:*
The scheme is unforgeable if the advantage of any PPT algorithm A *in the following game is negligible:*

- *The challenger generates* $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^k)$*, and gives* mvk *to* A.
- *The adversary is allowed to query a key generation oracle* $\mathsf{O}_{\mathsf{key}}$*, and a signing oracle* $\mathsf{O}_{\mathsf{sign}}$*, that share a dictionary indexed by tuples $(f, i) \in \mathcal{F} \times \mathbb{N}$, whose entries are signing keys: $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$. This dictionary keeps track of the keys that have been previously generated during the unforgeability game. The oracles are defined as follows :*
  - $\mathsf{O}_{\mathsf{key}}(f, i)$:
    - *if there exists an entry for the key (f, i) in the dictionary, then output the corresponding value, sk_f^i.*
    - *otherwise, sample a fresh key $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, add an entry $(f, i) \to \mathsf{sk}_f^i$ to the dictionary, and output sk_f^i*

- $\mathsf{O}_{\mathsf{sign}}(f, i, m)$:
  - *if there exists an entry for the key (f, i) in the dictionary, then generate a signature on $f(m)$ using this key: $\sigma \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_f^i, m)$.*
  - *otherwise, sample a fresh key $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, add an entry $(f, i) \rightarrow \mathsf{sk}_f^i$ to the dictionary, and generate a signature on $f(m)$ using this key: $\sigma \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_f^i, m)$.*

– *The adversary wins if it can produce (m^*, σ) such that*
- $\mathsf{FS.Verify}(\mathsf{mvk}, m^*, \sigma) = 1$.
- *there does not exist m such that $m^* = f(m)$ for any f which was sent as a query to the $\mathsf{O}_{\mathsf{key}}$ oracle.*
- *there does not exist a (f, m) pair such that (f, m) was a query to the $\mathsf{O}_{\mathsf{sign}}$ oracle and $m^* = f(m)$.*

***Function privacy**:*
Intuitively, we require the distribution of signatures on a message m' generated via different keys sk_f to be computationally indistinguishable, even given the secret keys and master signing key*. Namely, the advantage of any PPT adversary in the following game is negligible:*

– *The challenger honestly generates a key pair $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^k)$ and gives both values to the adversary.*
– *The adversary chooses a function f_0 and receives an (honestly generated) secret key $\mathsf{sk}_{f_0} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_0)$.*
– *The adversary chooses a second function f_1 for which $|f_0| = |f_1|$ (where padding can be used if there is a known upper bound) and receives an (honestly generated) secret key $\mathsf{sk}_{f_1} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_1)$.*
– *The adversary chooses a pair of values m_0, m_1 for which $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.*
– *The challenger selects a random bit $b \leftarrow \{0, 1\}$ and generates a signature on the image message $m' = f_0(m_0) = f_1(m_1)$* using secret key sk_{f_b}*, and gives the resulting signature $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_{f_b}, m_b)$ to the adversary.*
– *The adversary outputs a bit b', and wins the game if $b' = b$.*

***Succinctness**:*
There exists a polynomial $s(\cdot)$ such that for every $k \in \mathbb{N}, f \in \mathcal{F}, m \in \mathcal{D}_f$, it holds with probability 1 over $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^k), \mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, $(f(m), \sigma) \leftarrow \mathsf{FS.Sign}(f, \mathsf{sk}_f, m)$ that the resulting signature on $f(m)$ has size $|\sigma| \leq s(k, |f(m)|)$. In particular, the signature size is independent of the size $|m|$ of the input to the function, and of the size $|f|$ of a description of the function f.

Constructions. In the full version of the paper, we give three constructions of functional signature schemes, and describe the trade-offs between them in terms of the assumptions they require and the function privacy and succinctness properties of the functional signature scheme.

Theorem 7. *The following three implications hold:*

1. *Assuming the existence of one-way functions, there exists a functional signature scheme for the class $\mathcal{F}$ of polynomial-size circuits that satisfies the unforgeability requirement described above.*
2. *Assuming the existence of Non-Interactive Zero Knowledge Arguments of Knowledge for* NP *and one-way functions, there exists a function-private (but not necessarily succinct) functional signature scheme for the class $\mathcal{F}$ of polynomial-size circuits.*
3. *Assuming the existence of an unforgeable (but not necessarily succinct or function-private) functional signature scheme supporting the class of functions $\mathcal{F}$, and an adaptive zero-knowledge Succinct Non-Interactive Argument of Knowledge (SNARK) system for* NP*, there exists* succinct, function-private *functional signatures for $\mathcal{F}$.*

As a corollary, it follows that succinct, function-private functional signatures for the class of polynomial-size circuits can be based on SNARKs for NP and OWFs.

3 Applications of Functional Signatures

In this section we discuss applications of functional signatures to other cryptographic problems, such as constructing delegation schemes and succinct non-interactive arguments (SNARGs).

3.1 SNARGs from Functional Signatures

Recall that in a SNARG system for a language L, there is a verifier V, and a prover P who wishes to convince the verifier that an input x is in L. To achieve succinctness, proofs produced by the prover must be sublinear in the size of the input plus the size of the witness.

We show how to use a functional signature scheme supporting keys for functions f describable as polynomial-size circuits, and which has short signatures (i.e of size $r(k) \cdot (|f(m)| + |m|)^{o(1)}$ for a polynomial $r(\cdot)$) to construct a SNARG scheme with preprocessing for any language $L \in NP$ with proof size bounded by $r(k) \cdot (|w| + |x|)^{o(1)}$, where w is the witness and x is the instance. We note that this is the proof size used in the lower bound of [13].

Let L be an NP-complete language, and R the corresponding relation. The main idea in the construction is for the verifier (or CRS setup) to give out a single signing key for a function whose range consists of exactly those strings that are in L. Note that this can be efficiently described by use of the relation R (where the function also takes as input a witness). Then, with sk_f for this appropriate function f, the prover will be able to sign *only* those messages that are in the language L, and hence can use a signature on x as a convincing argument that $x \in L$. The resulting argument is succinct and publicly verifiable.

More explicitly, let $\mathsf{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$ be a succinct functional signature scheme (as in Definition 1) supporting the class $\mathcal{F}$ of polynomial-size circuits. We construct the desired SNARG system $\Pi = (\Pi.\mathsf{Gen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for NP language L with relation R, as follows:

– $\Pi.\mathsf{Gen}(1^k)$:
  • run the functional signature scheme setup: $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^k)$.
  • generate a signing key $\mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$ for the function $f(x|w) := x$ if $R(x, w) = 1, \perp$ otherwise, and output $\mathsf{crs} = (\mathsf{mvk}, \mathsf{sk}_f)$.
– $\Pi.\mathsf{Prove}(x, w, \mathsf{crs})$: output $\mathsf{FS.Sign}(f, \mathsf{sk}_f, x|w)$.
– $\Pi.\mathsf{Verify}(\mathsf{crs}, x, \pi)$: output $\mathsf{FS.Verify}(\mathsf{mvk}, x, \pi)$.

Theorem 8. *If* FS *is a functional signature scheme supporting the class* $\mathcal{F}$ *of polynomial-sized circuits, then* Π *is a succinct non-interactive argument (SNARG) for NP language* L*.*

We defer the proof of Theorem 8 to the full version.

Remark 1 (Functional PRFs as Functional MACs). Note that functional pseudorandom functions directly imply a notion of *functional message authentication codes (MACs)*, where the master PRF seed s serves as the (shared) master secret MAC key, and a functional PRF subkey sk_f enables one to both MAC and verify messages $f(m)$. Using the transformation above with such a functional MAC in the place of functional signatures yields a *privately verifiable* SNARG system.

Remark 2 (Lower bound of [13]). Gentry and Wichs showed in [13] that SNARG schemes for NP, with proof size $r(k) \cdot (|x| + |w|)^{o(1)}$ for polynomial $r(\cdot)$ cannot be obtained using black-box reductions to falsifiable assumptions [26]. Therefore, combined with Theorem 8, it follows that in order to obtain a functional signature scheme with signature size $r(k) \cdot (|f(m)| + |m|)^{o(1)}$ we must either rely on non-falsifiable assumptions (as in our SNARK-based construction) or make use of non black-box techniques.

In the full version of this paper, we demonstrate a similar implication of functional signatures on the existence of efficient delegation schemes.

4 Functional Pseudorandom Functions

In a standard pseudorandom function family, the ability to evaluate the chosen function is all-or-nothing: a party who holds the secret seed s can compute $F_s(x)$ on all inputs x, whereas a party without knowledge of s cannot distinguish evaluations $F_s(x)$ on requested inputs x from random. We propose the notion of a *functional pseudorandom function (F-PRF)* family, which partly fills this gap between evaluation powers. The idea is that, in addition to a master secret key that can be used to evaluate the pseudorandom function F on any point in the domain, there are additional *secret keys per function* f, which allow one to evaluate F on y for any y for which there exists an x such that $f(x) = y$ (i.e., y is in the range of f).

Definition 2 (Functional PRF). *We say that a PRF family* $\{F_s : D \rightarrow R\}_{s \in S}$ *is a* functional pseudorandom function (F-PRF) *with respect to a class of functions* $\mathcal{F} = \{f : A_f \rightarrow D\}$ *if there exist additional algorithms*

$\mathsf{KeyGen}(s, f)$ *: On input a seed* $s \in S$ *and function description* $f \in \mathcal{F}$*, the algorithm* KeyGen *outputs a key* sk_f*.*

$\mathsf{Eval}(\mathsf{sk}_f, f, x)$ *: On input key* sk_f*, function* $f \in \mathcal{F}$*, and input* $x \in A_f$*, then* Eval *outputs the PRF evaluation* $F_s(f(x))$*.*

which satisfy the following properties:

- **Correctness:** *For every* $f \in \mathcal{F}$*,* $\forall x \in A_f$*, it holds that* $\forall s \leftarrow S,\ \forall \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(s, f),\ \mathsf{Eval}(\mathsf{sk}_f, f, x) = F_s(f(x))$*.*
- **Pseudorandomness:** *Given a set of keys* $\mathsf{sk}_{f_1} \dots \mathsf{sk}_{f_\ell}$ *for functions* $f_1 \dots f_\ell$*, the evaluation of* $F_s(y)$ *should remain pseudorandom on all inputs* y *that are not in the range of any of the functions* $f_1 \dots f_\ell$*. That is, for any PPT adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in distinguishing between the following two experiments is negligible (for any polynomial* $\ell = \ell(k)$*):*

Experiment Rand	**Experiment PRand**
Key query Phase	*Key query Phase*
$(\mathsf{pp}, s) \leftarrow \mathsf{Gen}(1^k)$	$(\mathsf{pp}, s) \leftarrow \mathsf{Gen}(1^k)$
$f_1 \leftarrow \mathcal{A}(\mathsf{pp})$	$f_1 \leftarrow \mathcal{A}(\mathsf{pp})$
$\mathsf{sk}_{f_1} \leftarrow \mathsf{KeyGen}(s, f_1)$	$\mathsf{sk}_{f_1} \leftarrow \mathsf{KeyGen}(s, f_1)$
$\vdots$	$\vdots$
$f_\ell \leftarrow \mathcal{A}(\mathsf{pp}, f_1, sk_{f_1}, \dots, f_{l-1}, sk_{f_{l-1}})$	$f_\ell \leftarrow \mathcal{A}(\mathsf{pp}, f_1, sk_{f_1}, \dots, f_{l-1}, sk_{f_{l-1}})$
$\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(s, f_\ell)$	$\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(s, f_\ell)$
Challenge Phase	*Challenge Phase*
$H \leftarrow \mathbb{F}_{D \to R}$ *a random function*	
$b \leftarrow \mathcal{A}^{\mathcal{O}_{s,H}^{\{f_i\}}(\cdot)}(f_1, \mathsf{sk}_{f_1}, \dots, f_\ell, \mathsf{sk}_{f_\ell})$	$b \leftarrow \mathcal{A}^{F_s(\cdot)}(f_1, \mathsf{sk}_{f_1}, \dots, f_\ell, \mathsf{sk}_{f_\ell})$

$$\text{where } \mathcal{O}_{s,H}^{\{f_i\}}(y) := \begin{cases} F_s(y) & \text{if } \exists i \in [l] \text{ and } x \text{ s.t. } f_i(x) = y \\ H(y) & \text{otherwise} \end{cases}.$$

Note that, as defined, the oracle $\mathcal{O}_{s,H}^{\{f_i\}}(y)$ need not be efficiently computable. This inefficiency stems both from sampling a truly random function H, and from testing whether the adversary's evaluation queries y are contained within the range of one of his previously queried functions f_i. However, within particular applications, the system can be set up so that this oracle is efficiently simulatable: For example, evaluations of a truly random function can be simulated by choosing each queried evaluation one at a time; Further, the range of the relevant functions f_i may be efficiently testable given trapdoor information (e.g., determining the range of $f : r \mapsto \mathsf{Enc}(\mathsf{pk}, 0; r)$ for a public-key encryption scheme is infeasible given only pk but efficiently testable given the secret key).

We also consider a weaker security definition, where the adversary has to reveal which functions he will request keys for before seeing the public parameters or any of the keys. Namely, the key query phase takes place as follows:

Selective Key query Phase
$(\mathsf{pp}, s) \leftarrow \mathsf{Gen}(1^k)$
$(f_1, \ldots, f_\ell) \leftarrow \mathcal{A}(\mathsf{pp})$
For $i \in [\ell]$, $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(s, f_i)$

We refer to this as a *selectively secure F-PRF*.

A special case of functional PRFs arises when access control is to be determined by predicates. (Indeed, fitting within the F-PRF framework, one can emulate predicate policies by considering the corresponding functions $f_P(x) = x$ if $P(x) = 1$ and $= \perp$ if $P(x) = 0$). We refer to this as *PRFs with selective access*.

Finally, we consider *hierarchical* F-PRFs, where a party holding key sk_f for function $f : B \to D$ can generate subsidiary keys $\mathsf{sk}_{f \circ g}$ for functions $g : A \to B$.

We present formal definitions of these notions in the full version of this paper.

4.1 Construction Based on OWF

We now construct a functional pseudorandom function family $F_s : \{0,1\}^n \to \{0,1\}^n$ supporting the class of prefix-fixing functions, building upon the Goldreich-Goldwasser-Micali (GGM) tree-based PRF construction [14]. More precisely, our construction supports the function class

$$\mathcal{F}_{\mathsf{pre}} = \left\{ f_z(x) : \{0,1\}^n \to \{0,1\}^n \;\middle|\; z \in \{0,1\}^m \text{ for } m \leq n \right\},$$

$$\text{where } f_z(x) := \begin{cases} x & \text{if } (x_1 = z_1) \wedge \cdots \wedge (x_m = z_m) \\ \perp & \text{otherwise} \end{cases}.$$

Recall that the GGM construction makes use of a length-doubling pseudorandom generator $G : \{0,1\}^k \to \{0,1\}^{2k}$ (which can be constructed from any one-way function). Denoting the two halves of the output of G as $G(y) = G_0(y)G_1(y)$, the PRF with seed s is defined as $F_s(y) = G_{y_k}(\cdots G_{y_2}(G_{y_1}(s)))$.

We show that we can obtain a functional PRF for $\mathcal{F}_{\mathsf{pre}}$ by adding the following two algorithms on top of the GGM PRF construction. Intuitively, in these algorithms the functional secret key sk_{f_z} corresponding to a queried function $f_z \in \mathcal{F}_{\mathsf{pre}}$ will be the partial evaluation of the GGM prefix corresponding to prefix z: i.e., the label of the node corresponding to node z in the GGM evaluation tree. Given this partial evaluation, a party will be able to compute the completion for any input x which has z as a prefix. However, as we will argue, the evaluation on all other inputs will remain pseudorandom.

$\mathsf{KeyGen}(s, f_z)$: output $G_{z_m}(\cdots G_{z_2}(G_{z_1}(s)))$, where $m = |z|$

$\mathsf{Eval}(\mathsf{sk}_{f_z}, y)$: output $\begin{cases} G_{y_n}(\cdots G_{y_{m+2}}(G_{y_{m+1}}(\mathsf{sk}_{f_z}))) & \text{if } y_1 = z_1 \wedge \cdots \wedge y_m = z_m \\ \perp & \text{otherwise} \end{cases}$

Theorem 9. *Based on the existence of one-way functions, the GGM pseudorandom function family together with algorithms* KeyGen *and* Eval *defined as above, yields a selectively secure functional PRF for the class of functions* $\mathcal{F}_{\mathsf{pre}}$.

We remark that one can directly obtain a *fully* secure F-PRF for $\mathcal{F}_{\mathsf{pre}}$ (as in Definition 2) from our selectively secure construction, with a loss of 2^{-n} in security for each secret key sk_{f_z} queried by the adversary. This is achieved simply by guessing the adversary's query $f_z \in \mathcal{F}_{\mathsf{pre}}$. For appropriate choices of input size n and security parameter k, this can still provide useful security.

As an immediate corollary of Theorem 9, we obtain a (selectively secure) *PRF with selective access* for the class of equivalent prefix-matching predicates $\mathcal{P}_{\mathsf{pre}} = \{P_z : \{0,1\}^n \to \{0,1\} | z \in \{0,1\}^m \text{ for } m \leq n\}$, where $P_z(x) := 1$ if $(x_1 = z_1) \wedge \cdots \wedge (x_m = z_m)$ and 0 otherwise.

Our F-PRF construction has the additional benefit of being *hierarchical*. Given a secret key sk_{f_z} for a prefix $z \in \{0,1\}^m$, a party can generate subordinate secret keys $\mathsf{sk}_{f_{z'}}$ for any $z' \in \{0,1\}^{m'}$, $m' > m$ that aligns with z on its first m bits. This secondary key generation process is accomplished simply by applying the PRGs to sk_{f_z}, traversing the GGM tree according to the additional bits of z'.

Punctured Pseudorandom Functions Punctured PRFs, formalized by [29], are a special case of functional PRFs where one can generate keys for the function family $\mathcal{F} = \{f_x(y) = y \text{ if } y \neq x, \text{ and } \perp \text{ otherwise}\}$. Such PRFs have recently been shown to have important applications, including use as a primary technique in proving security of various cryptographic primitives based on the existence of indistinguishability obfuscation (see, e.g., [29, 22]).

The existence of a functional PRF for the prefix-fixing function family gives a construction of punctured PRFs. Namely, a punctured key sk_x allowing one to compute the PRF on all inputs except $x = x_1 \ldots x_n$ consists of n functional keys for the prefix-fixing function family for prefixes:

$$(\bar{x}_1),\ (x_1\bar{x}_2),\ (x_1x_2\bar{x}_3), \ldots, (x_1x_2 \ldots x_{n-1}\bar{x}_n).$$

Our GGM-based construction in the previous section thus directly yields a selectively secure punctured PRF based on OWFs.

Corollary 3 (Selectively-Secure Punctured PRFs). *Assuming the existence of OWF, there exists a selectively secure punctured PRF for any desired poly-size input length.*

We remark that full security can be achieved with a security loss of 2^{-n} (as the reduction needs only to guess which of the 2^n query sets will be made by the adversary, corresponding to the 2^n possible point puncturings).

5 Open Problems

The size of the signatures in our SNARK-based functional signature scheme is dependent only on the security parameter (as one would desire), but the construction is based on non-falsifiable assumptions. In Section 3, we show that, for any sufficiently expressive functional signature scheme (supporting a function class $\mathcal{F}$ that contains any NP-complete relation), a functional signature for $y =$

$f(x)$ cannot be sublinear in the size of y or x, unless the construction is either proven secure under a non-falsifiable assumption or makes use of non-black-box techniques. However, no lower bound exists that relates the size of the signature to the description of f (which may have short inputs/outputs x, y but a large description). Constructing functional signatures with short (sublinear in the size of the functions supported) signatures and verification time under falsifiable assumptions remains an open problem.

An interesting problem left open by this work is to construct a functional PRF that is also *verifiable*. A verifiable PRF, introduced by Micali, Rabin and Vadhan in [25] has the property that, in addition to the secret seed s of the PRF, there is a corresponding public key pk_s and a way to generate a proof π_x given the secret seed, such that given pk_s, x, y and π_x, one can check that y is indeed the consistent output of the PRF on x. (The challenge arises in guaranteeing soundness even though the public key is produced by the potentially malicious party. This, for example, rules out direct application of non-interactive zero-knowledge proofs, which require an honestly generated common reference string.) The public parameters and proofs π_x should not allow an adversary to distinguish the outputs of the PRF from random on any point x' for which the adversary has not received a proof. A construction of standard verifiable PRFs was given by Lysyanskaya based on the many-DH assumption in bilinear groups in [24].

One may extend the notion of verifiable PRFs to the setting of functional PRFs by enabling a user with functional key sk_f to also generate verifiable proofs π_x of correctness for evaluations of the PRF on inputs x for which his key allows. We note that such a verifiable functional pseudorandom function family supporting keys for a function class $\mathcal{F}$, implies a functional signature scheme that supports signing keys for the same function class, so the lower bound mentioned for functional signatures applies also to the proofs output in the verifiable functional PRF context.

References

1. S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*, 2013.
2. M. Backes, S. Meiser, and D. Schrder. Delegatable functional signatures. Cryptology ePrint Archive, Report 2013/408, 2013.
3. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.
4. M. Bellare and G. Fuchsbauer. Policy-based signatures. Cryptology ePrint Archive, Report 2013/413, 2013.
5. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.
6. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.
7. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*, pages 149–168, 2011.

8. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.
9. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2013/352, 2013.
10. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401, 2013.
11. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
12. R. Gennaro and D. Wichs. Fully homomorphic message authenticators. *IACR Cryptology ePrint Archive*, 2012:290, 2012.
13. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.
14. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
15. O. Goldreich, S. Micali, and A. Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO*, pages 171–185, 1986.
16. S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.
17. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. *IACR Cryptology ePrint Archive*, 2012:733, 2012.
18. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Overcoming the worst-case curse for cryptographic constructions. In *CRYPTO*, 2013.
19. S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.
20. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
21. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.
22. S. Hohenberger, A. Sahai, and B. Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/509, 2013.
23. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. Cryptology ePrint Archive, Report 2013/379, 2013.
24. A. Lysyanskaya. Unique signatures and verifiable random functions from the dh-ddh separation. In *CRYPTO*, pages 597–612, 2002.
25. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
26. M. Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.
27. C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In *TCC*, pages 222–242, 2013.
28. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
29. A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013.