Robust Distributed Pseudorandom Functions for mNP Access Structures

Bei Liang[1] and Aikaterini Mitrokotsa[1]

[1] Chalmers University of Technology, Gothenburg, Sweden
{lbei,aikmitr}@chalmers.se

Abstract. *Distributed pseudorandom functions* (DPRFs) formally defined by Naor *et al.* (EUROCRYPT'99) provide the properties of regular PRFs as well as the ability to distribute the evaluation of the PRF function; rendering them useful against single point of failures in multiple settings (*e.g.,* key distribution centres). To avoid the corruption of the partial PRF values computed by distributed servers, Naor *et al.* proposed the notion of *robust distributed PRFs*, which not only allows the evaluation of the PRF value by a set of distributed servers, but also allows to verify if the partial evaluation values are computed correctly.
In this paper, we investigate different approaches to build non-interactive robust distributed PRFs for a general class of access structures, going beyond the existing threshold and monotone span programs (MSP). More precisely, our contributions are two fold: *(i)* we first adapt the notion of single round robust distributed PRFs for threshold access structures to one for any mNP access structure (monotone functions in NP), and *(ii)* we provide a provably secure general construction of robust distributed PRFs by employing puncturable PRFs, a non-interactive witness indistinguishable proof (NIWI) and indistinguishable obfuscation. We compare our robust DPRF with existing DPRFs in terms of security guarantees, underlying assumptions and required primitives.

Keywords: robust distributed PRFs, threshold access structures, monotone functions, puncturable PRFs.

1 Introduction

Distributed pseudorandom functions (DPRFs), defined by Naor *et al.* [26], provide the properties of regular PRFs (*i.e.,* indistinguishability from random functions) and the capability to evaluate the function f (approximate of a random function) among a set of distributed servers. More precisely, Naor *et al.* [26] considered the setting where the PRF secret key is split among N servers and at least t servers are needed to evaluate the PRF. The user who wishes to get the PRF value on input x, sends x to each server, and receives partial function values from at least t out of N servers. It then combines the t partial values into a PRF value on x. There is no other interaction in the system, namely the servers do not need to communicate during the protocol. Non-interactive distributed PRFs in this setting are known as *distributed PRFs* for threshold access structures.

The main feature of distributed PRFs is that in order to evaluate the PRF, it is not required to reconstruct the key at a single location (*e.g.,* client).

Distributed PRFs provide strong guarantees against single point of failures and can be employed in key distribution centres [26] as well as to provide reliable byzantine agreement protocols [27]. However, standard distributed PRFs do not provide a guarantee about the correct computation of the pseudorandom function value, which is rather important in secure multi-party settings. For instance, corrupted servers controlled by adversaries could send to a client incorrect partial evaluations of the PRF, thus, preventing the user (client) from computing the correct (combined) PRF value. To avoid such cases, Naor *et al.* [26] proposed the *robustness* property for the distributed PRFs, which requires an additional procedure to verify that the partial values received from the servers on some input are computed correctly.

Although robust distributed PRFs have received some attention all existing solutions [26,23,5,4] require either threshold structures or monotone access programs (MSP). In this paper, we address the following question: *Is it possible to build non-interactive robust distributed PRFs for a general class of access mechanisms (e.g., that can be described by an access predicate) going beyond existing threshold and monotone span programs (MSP) access structures?* More precisely, we investigate whether it is possible to transform a PRF into a *robust distributed PRF* for an even more general class of access structures.

The answer is affirmative and our contributions are two-fold: *(i)* we adapt the notion of single round robust distributed PRFs for threshold access structures to one for any mNP access structure (monotone functions in NP), and *(ii)* we provide a provably secure general construction of robust distributed PRFs by employing puncturable PRFs, a non-interactive witness indistinguishable proof (NIWI) and indistinguishable obfuscation.

An immediate application of distributed PRFs is to construct a distributed key distribution centre (DKDC), where the task of generating a key is distributed among N servers, such that any t of them together can obtain the key (which corresponds to the PRF value). With our proposed robust distributed PRF for mNP access structures, it is feasible to construct a DKDC in such a way that only properly authorized servers – *i.e.,* who satisfy an access predicate (*e.g.,* the servers are within some predetermined distance) – can learn the key. Furthermore, our proposed robust DPRF is secure against malicious servers and can detect corrupted servers that broadcast incorrect values by using a proof to verify that the servers' partial computations are carried out correctly. For example as shown in Figure 1, imagine the scenario, where many users in a conference (associated with a public value H_C) need a key to get access to the online resources of this conference. There are N servers that can be accessed and distributed around the whole conference centre. When a user asks for the key, he contacts the servers that are located within a distance δ away from the user's electronic device, and sends them H_C as the input for the distributed PRF evaluation. Then, each server within distance δ, sends the partial value y_i and the corresponding proof π_i of correctness for the evaluation of y_i on H_C.

Once having received the response from the contacted servers, the user checks the proofs of correctness π_i of the partial values y_i. If the proofs received in the previous phase are verified, by employing the Combine algorithm, the user can compute the PRF value y on the input H_C, where y is the key to access the online resources. Note that all users in the conference learn the same PRF value y from different subsets of servers on the same input H_C, even though the servers located around different users (who sit in different areas within distance δ) usually are not the same.

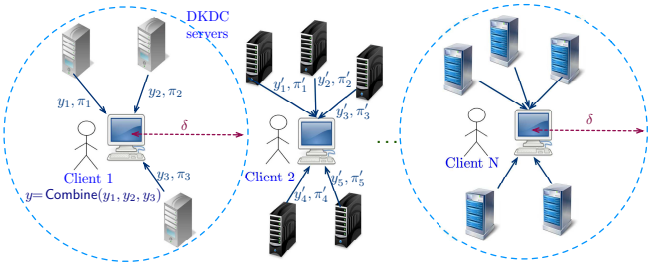

Fig. 1: An application of robust distributed PRFs for mNP to DKDC

In Table 1, we provide a comparison of our robust distributed PRF for mNP access structures to existing distributed PRFs. We compare them in terms of used access structure classes, the assumptions and building blocks they are based on, as well as the security that they can achieve. It is important to note that existing robust DPRFs do not provide the flexibility of general access structures that ours does and thus, could not be employed in the example described above, where the distance of the servers from the user (client) is used as an access predicate.

Naor *et al.* [26] pointed out that the robustness property can be achieved either via error-correcting properties of Reed-Solomon codes [24] w.r.t. their threshold construction of an ℓ-wise independent function, or via a zero-knowledge variant of Schnorr's proof for the value of the Diffie-Hellman function [30] w.r.t. their DDH-based distributed PRF scheme. Such proof techniques, however, require interaction[1], are tailored to the DDH-type of PRF schemes, and none of them is known to be compatible with existing PRFs based on other types of assumptions (such as one-way functions). Libert *et al.* [23] recently showed how to use fully homomorphic signatures [14] that are context-hiding to generally compile distributed PRFs (more precisely, Libert *et al.* [23] construct key-homomorphic PRFs) into robust distributed PRFs against malicious servers. Boneh, Lewi, Montgomery and Raghunathan [5] proposed a generic construction of threshold distributed PRFs from key-homomorphic PRFs, which is in-

[1] Except of the case where a non-interactive proof is computed in the random oracle model.

stantiated for the first time in the standard model under the LWE assumption. Boneh *et al.* [4] suggested another generic robust distributed PRF construction from a general "universal thresholdizer" tool, which is a natural generalization of a threshold fully homomorphic encryption scheme with robustness. Boneh *et al.* [4] also showed that the robustness of a universal thresholdizer can be enforced either by using NIZK with pre-processing ,which can be constructed from one-way functions [10], or homomorphic signatures [14]. However, contrary to our proposed distributed PRF none of the existing ones can be employed for general access structures *e.g.,* structures that satisfy an access predicate (for instance being within some predetermined distance as described in the example above).

Table 1: Comparison of existing distributed PRF schemes.

Distributed PRF	Access Structure	Robustness	Pseudorandomness	Tool	Assumption
Naor *et al.* [26]	threshold & monotone span programs	✓*	message-adaptive & selective-corruption	Reed-Solomon codes & Schnorr's proof	ℓ-wise independence or DDH
Libert *et al.* [23]	threshold	✓	message-adaptive & adaptive-corruption	homomorphic signatures	LWE
Boneh *et al.* [5]	threshold	no	message-adaptive & selective-corruption	key-homomorphic PRFs	LWE
Boneh *et al.* [4]	monotone boolean formulas	✓	message-adaptive & selective-corruption	universal thresholdizer	LWE
Our proposal	mNP	✓	message-selective & adaptive-corruption	NIWI & Commitment & $i\mathcal{O}$	$i\mathcal{O}$ & one-way permutation†

* Naor *et al.* [26] mentioned that their constructions can be amended to be robust based either on error-correcting mechanisms or on proof techniques, but no concrete construction is given.

† In order not to bring in more assumptions, here we use the NIWI construction given by Bitansky and Paneth [2] assuming the existence of $i\mathcal{O}$ and one-way permutations. NIWI proofs also can be constructed based on the decisional linear assumption [17]. The commitment scheme used in our robust DPRF is a perfectly binding non-interactive commitment, without trusted setup, which can be constructed from certifiably injective one way functions. Goyal *et al.* [15] also proposed alternative constructions of perfectly binding non-interactive commitment (without trusted setup) under the learning parity with low noise (LPLN) assumption and the learning with errors (LWE) assumption.

Our contributions In this work, we consider single round robust DPRFs for monotone functions in NP, also known as mNP that was initially studied by Komargodski *et al.* [20]. We also show that robust DPRFs for an mNP access structure can be constructed generally from the ingredients of puncturable PRFs [6,19,9], a non-interactive witness indistinguishable proof (NIWI) and indistinguishable obfuscation [11].

Specifically, a single round robust DPRF for an mNP access structure is defined as follows: given an access structure, the setup algorithm outputs a public parameter PP and a secret key msk which defines a PRF. On input the

PRF key msk, there is an algorithm that "splits" it into different pieces (shares) sk_i and then distributes the shares of the key to a collection of servers. Each server, using its own secret share sk_i, is allowed to compute a partial function value y_i and its proof π_i on input x. Using the proof π_i, anyone can locally verify the correctness of the computation for the partial function value y_i on point x with a local verification algorithm. For the "qualified" subsets, there is a witness attesting to this fact and given the witness along with the (correct) partial values y_i of these "qualified" servers, it should be possible to reconstruct the evaluation of the PRF on an input x. On the other hand, for the "unqualified" subsets there is no witness, and so it should be impossible to reconstruct the PRF on the input x. Let us consider the application scenario described in Figure 1 as an example. For client 1, the "qualified" set of servers refers to a subset of N servers that consists of all the servers whose distance from client 1 is not bigger than δ. Let us denote them as S_1, S_2 and S_3. The witness corresponding to such a qualified set $T = \{S_1, S_2, S_3\}$ is the set of distances of S_1, S_2 and S_3 to client 1, *i.e.*, $(\delta_1, \delta_2, \delta_3)$. The set $T' = \{S_1, S_2, S_4, S_5\}$ can be recognized as an "unqualified" set, since it includes the servers S_4 and S_5, whose distance from client 1 is beyond δ as depicted in Figure 2. Furthermore, except of the correctness, verifiability, and pseudorandomness properties, robust DPRFs should satisfy the *robustness* property, which requires that it should be infeasible for a corrupted server holding the secret share sk_j to come up with an incorrect partial value y_j, for some input x', that can be still verified.

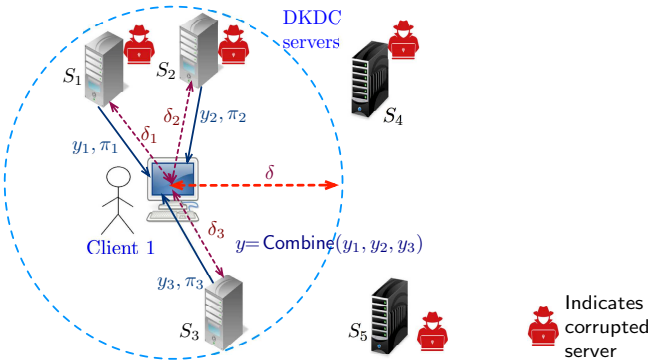

Fig. 2: An example of qualified/unqualified servers in DKDC

The main challenge is how to provide a non-interactive proof for the correct computation of the partial evaluation, while allowing the reconstruction for the final function value. To address this problem, Libert *et al.* [23] proposed one solution that employs homomorphic signatures in a black-box manner so that each server can produce a valid signature for the partial evaluation of the PRF on any input x using its own share sk_i and the corresponding signature. The robustness

guarantee is provided by the unforgeability property of the homomorphic signatures. However, Libert *et al.*'s approach requires homomorphic signatures that are simultaneously context-hiding and adaptively unforgeable, which, to the best of our knowledge, are not known to be instantiated yet.

In this work, we employ Goyal *et al.*'s verifiable random function (VRF) scheme [15] to provide the NIWI proof for the partial evaluation, which is computed from a new independent PRF with the secret key sk_i on an input x. The robustness is guaranteed by the uniqueness of Goyal *et al.*'s VRF [15], which in turn relies on the soundness of the NIWI proof system. In order to reconstruct the final PRF value from the totally independent (partial) PRF values, we use program obfuscation which outputs the correct final PRF value if each partial PRF value is verified by the specific NIWI proof system.

Overview of Our Techniques We now give a high level overview of our technical approach. Our basic scheme is rather easy to describe. Let $\mathbb{A} \in \mathsf{mNP}$ be an access structure on N servers $S_1, \ldots, S_N$. Given the verification procedure $V_{\mathbb{A}}$ for an mNP access structure $\mathbb{A}$, a trusted third party samples the PRF key K as well as N independent PRF keys $K_1, \ldots, K_N$. Each server receives a key K_i for $i \in \{1, \ldots, N\}$. For an input x, each server computes the partial evaluation $y_i = \mathsf{PRF}(K_i, x)$ and the corresponding proof π_i for certifying that the server's output y_i is the evaluation of $\mathsf{PRF}(K_i, x)$. We employ the basic idea of Goyal *et al.*'s VRF [15] scheme, which is to create a NIWI proof π_i for the statement that at least two out of three commitments (c_{i1}, c_{i2}, c_{i3}) (in the verification key) open to keys K_{i1}, K_{i2} such that $y_i = \mathsf{PRF}(K_{ij}, x) = \mathsf{PRF}(K_{ij'}, x)$ where $j, j' \in \{1, 2, 3\}$ and $j \neq j'$, using $(j = i1, j' = i2, K_i, K_i, r_{i1}, r_{i2})$ as witness. In order to reconstruct the function value on input x from a set of shares of qualified servers Γ, with witness w, the client runs the public obfuscated program $i\mathcal{O}(\mathsf{Prog})$, which is setup as part of the public parameters. The program Prog is hardwired with a PRF key K, the verification $V_{\mathbb{A}}$ and three independent commitments (c_{i1}, c_{i2}, c_{i3}) for each server's key K_i. Prog takes as input the valid witness w of the set of qualified servers $\Gamma \subseteq S_1, \ldots, S_N$, $\{y_i, \pi_i\}_{i \in \Gamma}$ and x and checks if the condition $V_{\mathbb{A}}(\Gamma, w) = 1$ and (x, y_i, π_i) is verified by the NIWI verification procedure for every $i \in \Gamma$. If the condition holds, the program Prog outputs $\mathsf{PRF}(K, x)$; otherwise it outputs $\perp$. We show that the robustness property is implied by the soundness of the NIWI proof system. We also show that the resulting robust DPRF remains selectively pseudorandom even when a set of unqualified servers $T \subseteq \{S_1, \ldots, S_N\}$, namely $T \notin \mathbb{A}$, are corrupted and the adversary is given the share of the servers that are uncorrupted on the inputs of its choice. For instance in our application scenario described above, we show that the pseudorandomness of our DPRF (on some input that has never been queried) is still maintained even if the adversary corrupts a subset of qualified and a subset of unqualified servers, *e.g.,* if the servers S_1, S_2, S_4 and S_5 are corrupted by the adversary as depicted in Figure 2.

Related work Function secret sharing is closely related to distributed PRFs. The notion of function secret sharing (FSS) was introduced by Boyle *et al.* [7]

as a natural generalization of distributed point functions (DPF). More precisely, a *function secret sharing* scheme for a class $\mathcal{F}$, which is a class of efficiently computable and succinctly described functions $f : \{0,1\}^n \to \mathbb{G}$, allows to split an arbitrary $f \in \mathcal{F}$ into p functions $f_1, \ldots, f_p$ such that: (1) $f(x) = \Sigma_{i=1}^{p} f_i(x)$ (on every input x), (2) each f_i is described by a short key k_i that enables its efficient evaluation, yet (3) any strict subset of the keys completely hides f.

Boyle *et al.* [7] have also initiated a study of FSS for general polynomial-time computable functions that can be obtained either from *virtual black-box* (VBB) obfuscation or from *probabilistic iO* (*piO*). Indeed, function secret sharing for pseudorandom functions is a special case of our distributed PRFs for any mNP access structure, where the access structure $\mathbb{A}$ is set to be the N-out-of-N threshold access structure.

Following the work of FSS [7], Boyle *et al.* [8] introduced the notion of verifiable FSS, where on the one hand, a function f can be split into functions $f_1, \ldots, f_m$, described by the corresponding keys $k_1, \ldots, k_m$, such that for any input x we have that $f(x) = f_1(x) + \ldots + f_m(x)$ and every strict subset of the keys hides f; on the other hand, there is an additional m-parties interactive protocol Ver for verifying that the keys $(k_1^*, \ldots, k_m^*)$, generated by a potentially malicious client, are consistent with some f. We should note that Boyle *et al.*'s VFSS can be employed to verify the validity of the function shared by the authority. For instance, the key shares could be generated by a potentially untrusted authority, but the evaluation algorithm should be performed by semi-honest servers. However, our robust DPRF can be employed even if the servers are malicious and can be used to verify that the partial values that a user receives from the servers on some input are computed correctly.

Boneh *et al.* [4,18] proposed an alternative method to build non-interactive robust distributed PRFs for any threshold access structure, using a universal thresholdizer (UT), which is a natural generalization of a threshold fully homomorphic encryption (TFHE) scheme with robustness. Boneh *et al.* [4] also showed that the robustness of a universal thresholdizer can be enforced either by using NIZK with pre-processing, which can be constructed from one-way functions [10], or homomorphic signatures [14]. Using their approach, it is possible to construct distributed PRFs that support arbitrary access structures such as monotone boolean formulas assuming a secure TFHE scheme exists for the access structures. Boneh *et al.* [4] also gave an FSS construction for any access structure from a universal thresholdizer, which relies on homomorphic signatures and allows the verification of the correctness of the servers' computations.

Another primitive related to distributed PRFs is the one-round t-out-of-N threshold signatures. One-round (t,N) threshold signatures are digital signatures that allow to share a signing key among N servers and any subset of t servers out of N are able to generate a valid signature, but that disallows the generation of a valid signature if fewer than t servers participate in the combination algorithm. The one-round threshold signature requires that for a user who wishes to sign a message m, he sends m to all N servers, and receives replies from t of them. The user then combines the t replies, and obtains the final signature on m. There is no

other interaction between them. In particular, the servers may not communicate with one another, or interact further with the user. The unforgeability property must hold even if the adversary is allowed to corrupt any subset of less than t servers. Many threshold signature schemes have been proposed, such as the threshold variations based on RSA signatures [13,31], Schnorr signatures [32], (EC)DSA signatures [12], BLS signatures [3].

The robust distributed PRF bears resemblance with one-round threshold signatures on the sharing phase of the secret key and the combination of partial values to produce the PRF value (correspondingly signature) on some input (correspondingly message). However, their difference lies on how the verification procedure works. That is, robust distributed PRFs verify the proof of correctness of the partial evaluation, while in threshold signatures there is no partial verification for partial signatures, instead there is a single verification algorithm to check whether the combined signature generated from the t servers' replies is a valid signature or not.

In this paper, we explore new methods in order to achieve robustness for distributed PRFs and establish the connection of robust DPRFs with other non-interactive witness indistinguishable proofs (NIWI) and indistinguishable obfuscation primitives; going beyond the existing DPRFs based on universal thresholdizers and homomorphic signatures. Contrary to existing work [4,18] our robust distributed PRFs can be employed for general access structures (*e.g.,* structures that satisfy a general access predicate), away from existing threshold access structures and monotone span programs.

Organization. The paper is organised as follows. In Section 2, we provide the definitions and building blocks used throughout the paper. In Section 3, we present the syntax and security notion of robust distributed pseudorandom functions (DPRFs) for any mNP access structure. In Section 4, we provide our construction of robust distributed PRFs and its security proof. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Monotone-NP and Access Structures

We recall the essential background knowledge of access structures and monotone-NP (mNP) as defined by Komargodski, Naor and Yogev in [20] as well as by Komargodski and Zhandry in [21]. The definitions below are taken from [20,21].

A function $f : 2^{[n]} \to \{0, 1\}$ is said to be monotone if for every $\Gamma \subseteq [n]$, such that $f(\Gamma) = 1$, it also holds that $\forall \Gamma' \subseteq [n]$ such that $\Gamma \subseteq \Gamma'$, then it holds that $f(\Gamma') = 1$. A monotone boolean circuit is a boolean circuit with AND and OR gates (without negations). A non-deterministic circuit is a Boolean circuit whose inputs are divided into two parts: standard inputs and non-deterministic inputs. A non-deterministic circuit accepts a standard input if and only if there is some setting of the non-deterministic input that causes the circuit to evaluate to 1. A monotone non-deterministic circuit is a non-deterministic circuit, where

the monotonicity requirement applies only to the standard inputs, that is, every path from a standard input wire to the output wire does not have a negation gate.

Definition 1 ([16,20,21]). *We say that a function L is in mNP if there exists a uniform family of polynomial-size monotone non-deterministic circuit that computes L.*

Lemma 1 ([16,20,21]). $\mathsf{mNP} = \mathsf{NP} \bigcap \mathsf{mono}$, *where* mono *is the set of all monotone functions.*

Definition 2 (Access structure [20,21]). *A monotone access structure $\mathbb{A}$ on $\mathcal{S}$ is a monotone set of subsets of $\mathcal{S}$. That is, for all $\Gamma \in \mathbb{A}$ it holds that $\Gamma \subseteq \mathcal{S}$ and for all $\Gamma \in \mathbb{A}$ and Γ' such that $\Gamma \subseteq \Gamma' \subseteq \mathcal{S}$ it holds that $\Gamma' \in \mathbb{A}$.*

Following the idea of Komargodski, Naor and Yogev [20] in defining access structures in secret sharing schemes, here we consider an access structure $\mathbb{A}$ as a characteristic function $\mathbb{A} : 2^{\mathcal{P}} \to \{0,1\}$ that outputs 1 given as input $\Gamma \subseteq \mathcal{S}$ if and only if Γ is in the access structure, namely $\Gamma \in \mathbb{A}$. We assume $\mathbb{A} : 2^{\mathcal{P}} \to \{0,1\}$ to be an access structure corresponding to a language $L \in \mathsf{mNP}$ that can be recognized by a polynomial-sized verification circuit $V_{\mathbb{A}}$.

2.2 Non-Interactive Witness Indistinguishable Proofs

Definition 3 (NIWI). *A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is a NIWI for a language $\mathcal{L} \in \mathsf{NP}$ with witness relation $\mathcal{R}$, if it satisfies the following conditions:*

- *(Perfect Completeness) For all (x, w) such that $\mathcal{R}(x, w) = 1$,*

$$Pr[\mathcal{V}(x, \pi) = 1 : \pi \leftarrow \mathcal{P}(x, w)] = 1.$$

- *(Statistical Soundness) For every $x \notin \mathcal{L}$ and $\pi \in \{0,1\}^*$,*

$$\mathbf{Pr}[\mathcal{V}(x, \pi) = 1] \leq 2^{-\Omega(|x|)}.$$

- *(Witness Indistinguishability) For any sequence $\mathcal{I} = \{(x, w_1, w_2) : \mathcal{R}(x, w_1) = 1 \wedge \mathcal{R}(x, w_2) = 1\}$, it holds:*

$$\{\pi_1 : \pi_1 \leftarrow \mathcal{P}(x, w_1)\}_{(x,w_1,w_2)\in\mathcal{I}} \approx_c \{\pi_2 : \pi_2 \leftarrow \mathcal{P}(x, w_2)\}_{(x,w_1,w_2)\in\mathcal{I}}.$$

Barak, Ong, and Vadhan [1] provided constructions for NIWIs based on Nisan-Wigderson type pseudorandom generators [28] and ZAPs (two-message public-coin witness indistinguishable proofs). Groth, Ostrovsky, and Sahai [17] then provided the first NIWI construction based on standard assumptions on bilinear maps such as the Decision Linear Assumption, the Symmetric External Diffie Hellman assumption, or the Subgroup Decision Assumption. Bitansky and Paneth [2] have recently provided NIWI proofs based on iO indistinguishability obfuscation and one-way permutations.

2.3 Perfectly Binding Commitments (with no setup assumptions)

A commitment scheme with message space $\{\mathcal{M}_\lambda\}_\lambda$, randomness space $\{\mathcal{R}_\lambda\}_\lambda$ and commitment space $\{\mathcal{C}_\lambda\}_\lambda$ consists of a pair of polynomial time algorithms (Commit, Verify) with the following syntax:

- $\mathsf{Commit}(1^\lambda, m \in \mathcal{M}_\lambda; r \in \mathcal{R}_\lambda)$: The commit algorithm is a randomized algorithm that takes as input the security parameter λ, message m to be committed and random coins r. It outputs a commitment/opening pair (c, d).
- $\mathsf{Verify}(m \in \mathcal{M}_\lambda, c \in \mathcal{C}_\lambda, d \in \mathcal{R}_\lambda)$: The verification algorithm takes as input the message m, commitment c and an opening d. It outputs either 0 or 1.

Definition 4. *[15] A pair of polynomial time algorithms (Commit, Verify) is a perfectly binding computationally hiding commitment scheme if it satisfies the following conditions:*

- *(Perfect Correctness) For all security parameters $\lambda \in \mathbb{N}$, message $m \in \mathcal{M}_\lambda$ and randomness $r \in \mathcal{R}_\lambda$, if $(c, r) \leftarrow \mathsf{Commit}(1^\lambda, m; r)$, then $\mathsf{Verify}(m, c, r) = 1$.*
- *(Perfect Binding) For every (c, m_1, r_1, m_2, r_2) such that $m_1 \neq m_2$, the following holds for at least one $i \in \{1, 2\}$:*

$$\mathbf{Pr}[\mathsf{Verify}(m_i, c_i, r_i) = 1] = 0.$$

- *(Computationally Hiding) For all security parameters $\lambda \in \mathbb{N}$, two messages $m_1, m_2 \in \mathcal{M}_\lambda$, it holds*

$$\{c_1 : (c_1, r_1) \leftarrow \mathsf{Commit}(1^\lambda, m_1; r_1); r_1 \leftarrow \mathcal{R}_\lambda\} \approx_c \{c_2 : (c_2, r_2) \leftarrow \mathsf{Commit}(1^\lambda, m_2; r_2); r_2 \leftarrow \mathcal{R}_\lambda\}.$$

Perfectly binding commitments (without trusted setup) can be constructed from certifiably injective one way functions. Goyal *et al.* [15] proposed alternative constructions of perfectly binding non-interactive commitment (without trusted setup) under the learning parity with low noise (LPLN) assumption and learning with errors (LWE) assumption.

In the rest of the paper, we also employ the notions of indistinguishability obfuscation (iO) [11] and puncturable PRFs [29]. Due to space constraints, we provide these definitions in the appendix A.1.

3 Robust Distributed Pseudorandom Functions for mNP

In this section, we formally define the syntax and security notion of robust distributed pseudorandom functions (DPRFs) for any mNP access structure $\mathbb{A}$ on N servers. On the one hand, it is a natural generalization of the definition of robust DPRFs for threshold access structures, which was initially proposed by Naor, Pinkas and Reingold [26] and formally defined by Libert, Stehlé, and Radu

Titiu [23]. On the other hand, it can be seen as a robustness augmentation of the DPRFs for any mNP access structure proposed by Liang and Mitrokotsa [22].

Recall that a PRF [25] is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm, where $\mathcal{K}$ is a secret key space, $\mathcal{X}$ is a domain, and $\mathcal{Y}$ is a range (and these sets may be parameterized by the security parameter λ). We define a robust distributed PRF for any mNP access structure $\mathbb{A}$ over N servers $\mathcal{S} = \mathcal{S}_N = \{S_1, \ldots, S_N\}$, and each server S_i can be identified with its index i.

For $N \in \mathbb{N}$, let $\mathbb{A}$ be an mNP access structure on N servers $S_1, \ldots, S_N$. A robust distributed PRF for $\mathbb{A}$ is a tuple of polynomial time algorithms $\Pi = (\mathsf{Setup}, \mathsf{GEval}, \mathsf{Share}, \mathsf{PEval}, \mathsf{PVerify}, \mathsf{Comb})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, N, V_\mathbb{A})$: On input the security parameter λ, the number N of servers and the verification procedure $V_\mathbb{A}$ for an mNP access structure $\mathbb{A}$ on N servers, the setup algorithm outputs the public parameters PP and a master secret key α.
- $\mathsf{GEval}(\alpha, x)$: On input the master secret key α and an input $x \in \mathcal{X}$, the global function evaluation algorithm executes as in an ordinary PRF and outputs a function value $y \in \mathcal{Y}$.
- $\mathsf{Share}(\alpha)$: On input the master secret key α, the key sharing algorithm outputs a tuple of N shares, $(\alpha_1, \ldots, \alpha_N)$.
- $\mathsf{PEval}(i, \alpha_i, x)$: On input a server index i, a key share α_i and input $x \in \mathcal{X}$, the partial evaluation algorithm outputs a tuple (x, y_i, π_i), where $y_i \in \mathcal{Z}_i$ is the server S_i's share of the function value $\mathsf{GEval}(\alpha, x)$ and π_i is the corresponding proof for the correct computation of y_i.
- $\mathsf{PVerify}(\mathsf{PP}, i, x, y_i, \pi_i)$: On input the public parameters PP, the index $i \in [N]$, an input x and a partial evaluation y_i on behalf of the server S_i, together with a corresponding proof π_i, the verification algorithm outputs a bit; 1 if (x, y_i, π_i) is deemed valid and 0 if it does not.
- $\mathsf{Comb}\big(\mathsf{PP}, V_\mathbb{A}, w, x, \{(y_i, \pi_i)\}_{i \in \Gamma}\big)$: On input the public parameters PP, the verification procedure $V_\mathbb{A}$ for an mNP language $\mathbb{A}$, a witness w, and a set of shares $\{y_i\}_{i \in \Gamma}$ for a set of servers $\Gamma \subseteq \{S_1, \ldots, S_N\}$ (where we recall that we identify a server S_i with its index i), the combining algorithm outputs a value $y \in \mathcal{Y} \cup \perp$;

and satisfying the following requirements:

Verifiability: For any $(\mathsf{PP}, \alpha) \leftarrow \mathsf{Setup}(1^\lambda, N, V_\mathbb{A}), (\alpha_1, \ldots, \alpha_N) \leftarrow \mathsf{Share}(\alpha)$, any index $i \in [N]$, any input $x \in \mathcal{X}$, and any $(x, y_i, \pi_i) \leftarrow \mathsf{PEval}(i, \alpha_i, x)$, we have $\mathsf{PVerify}(\mathsf{PP}, i, x, y_i, \pi_i) = 1$.

Robustness: Informally, the robustness property guarantees that it should be impossible for any corrupted server S_j holding the secret share α_j to generate an incorrect partial evaluation y_j, for some input x, that can be successfully verified, namely, $\mathsf{PVerify}(\mathsf{PP}, j, x, y_j, \pi_j) = 1$. More precisely, for any $j \in [N]$ and any PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such

that:

$$\Pr\left[\begin{array}{c} y''_j \neq y'_j \wedge \\ \mathsf{PVerify}(\mathsf{PP}, j, x', y''_j, \pi''_j) = 1 \end{array} \middle| \begin{array}{l} (\mathsf{PP}, \alpha) \leftarrow \mathsf{Setup}(1^\lambda, N, V_{\mathbb{A}}), \\ (\alpha_1, \ldots, \alpha_N) \leftarrow \mathsf{Share}(\alpha), \\ (x', y''_j, \pi''_j) \leftarrow \mathcal{A}(\mathsf{PP}, j, \alpha_j), \\ (y'_j, \pi'_j) \leftarrow \mathsf{PEval}(j, \alpha_j, x'), \end{array}\right] \leq negl(\lambda)$$

where the probability is taken over the random choices of Setup, Share and PEval, and the coin tosses of $\mathcal{A}$.

Correctness: If for all $\lambda, N, t \in \mathbb{N}$, any mNP access structure $\mathbb{A}$, any $x \in \mathcal{X}$, and any set of qualified servers $\Gamma = \{i_1, \ldots, i_t\} \subseteq \{S_1, \ldots, S_N\}$ with valid witness w (i.e., $V_{\mathbb{A}}(\Gamma, w) = 1$), it holds that:

$$\Pr\left[\begin{array}{r} \mathsf{Comb}\big(\mathsf{PP}, V_{\mathbb{A}}, w, x, \{(y_{i_j}, \pi_{i_j})\}_{j\in[t]}\big) \\ = \mathsf{GEval}(\alpha, x) \end{array} \middle| \begin{array}{l} (\mathsf{PP}, \alpha) \leftarrow \mathsf{Setup}(1^\lambda, N, V_{\mathbb{A}}), \\ (\alpha_1, \ldots, \alpha_N) \leftarrow \mathsf{Share}(\alpha), \\ (x, y_{i_j}, \pi_{i_j}) \leftarrow \mathsf{PEval}(i_j, \alpha_{i_j}, x) \text{ for } \forall j \in [t], \end{array}\right] = 1$$

Selective Pseudorandomness: Consider the following indistinguishability challenge experiment for corrupted servers $T \subseteq [N]$:

1. On input the security parameter 1^λ and number N, the adversary $\mathcal{A}$ outputs the challenge input x^*, an access structure $\mathbb{A} \in \mathsf{mNP}$ and an unqualified set $T \subseteq [N]$ (that is, $T \notin \mathbb{A}$).
2. The challenger runs $(\mathsf{PP}, \alpha) \leftarrow \mathsf{Setup}(1^\lambda, N, V_{\mathbb{A}})$ and $(\alpha_1, \ldots, \alpha_N) \leftarrow \mathsf{Share}(\alpha)$, and publishes the public parameters PP to the adversary $\mathcal{A}$.
3. The challenger sends the corresponding keys $\{\alpha_i\}_{i\in T}$ to $\mathcal{A}$.
4. The adversary (adaptively) sends queries $x_1, \ldots, x_Q \in \mathcal{X}$ to the challenger such that $x^* \notin \{x_1, \ldots, x_Q\}$, and for each query x_j the challenger responds with $\mathsf{PEval}(i, \alpha_i, x_j)$ for all $i \in [N]\backslash T$.
5. The challenger chooses a random bit $b \leftarrow \{0, 1\}$. If $b = 0$, the challenger returns a uniformly random $y^* \in \mathcal{Y}$ to the adversary. If $b = 1$, the challenger responds with $y^* = \mathsf{GEval}(\alpha, x^*)$.
6. The adversary $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

Let us denote by $\mathsf{Adv}^{\mathsf{pseudo}}_{\Pi,\mathcal{A}} := \Pr[b' = b] - 1/2$ the advantage of the adversary $\mathcal{A}$ in guessing b in the above experiment, where the probability is taken over the randomness of the challenger and of $\mathcal{A}$. We say the robust DPRFs Π for an mNP access structure $\mathbb{A}$ is selectively pseudorandom if there exists a negligible function $negl(\lambda)$ such that for all non-uniform PPT adversaries $\mathcal{A}$, it holds that $\mathsf{Adv}^{\mathsf{pseudo}}_{\Pi,\mathcal{A}} \leq negl(\lambda)$.

4 General Construction of Robust Distributed PRFs for mNP

In this section, we describe our construction of robust distributed PRFs in details. We present a general construction of robust DPRFs from perfectly binding commitments, NIWIs, puncturable PRFs and indistinguishability obfuscation

$i\mathcal{O}$. We also prove that it satisfies the verifiability, robustness, correctness and pseudorandomness properties (as described in Section 3).

Let $(\mathcal{P}, \mathcal{V})$ be a NIWI proof system for a language $\mathcal{L}$ (which will be defined later), $(\mathsf{CS.Commit}, \mathsf{CS.Verify})$ be a perfectly binding commitment scheme with $\{\mathcal{M}_\lambda\}_\lambda$, $\{\mathcal{R}_\lambda\}_\lambda$ and $\{\mathcal{C}_\lambda\}_\lambda$ as the message, randomness and commitment space, and $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Puncture}, \mathsf{PRF.Eval})$ be a puncturable PRF with $\{\mathcal{X}_\lambda\}_\lambda$, $\{\mathcal{Y}_\lambda\}_\lambda$, $\{\mathcal{K}_\lambda\}_\lambda$ and $\{\mathcal{K}^p_\lambda\}_\lambda$, as its domain, range, key and punctured key spaces. Here we assume that $\mathcal{K}_\lambda \cup \mathcal{K}^p_\lambda \subseteq \mathcal{M}_\lambda$, which means that all the PRF master keys and punctured keys lie in the message space of the commitment scheme CS.

Define the language $\mathcal{L}$, which contains instances of the form $(c_1, c_2, c_3, x, y) \in \mathcal{C}^3_\lambda \times \mathcal{X}_\lambda \times \mathcal{Y}_\lambda$ with the following witness relation:

$$\begin{aligned} &\exists \imath, \jmath \in \{1,2,3\}, K, K' \in \mathcal{K}_\lambda \cup \mathcal{K}^p_\lambda, r, r' \in \mathcal{R}_\lambda \text{ such that} \\ &\imath \neq \jmath \wedge \mathsf{CS.Verify}(K, c_\imath, r) = 1 \wedge \mathsf{CS.Verify}(K', c_\jmath, r') = 1 \\ &\qquad \wedge \mathsf{PRF}(K, x) = \mathsf{PRF}(K', x) = y. \end{aligned}$$

It is obvious that the above language is in NP as it can be verified in polynomial time.

Next we describe our construction for selectively-secure robust DPRFs with message space $\{\mathcal{X}_\lambda\}_\lambda$ and range space $\{\mathcal{Y}_\lambda\}_\lambda$.

Setup$(1^\lambda, N, V_\mathbb{A})$**:** On input 1^λ, the number N and the verification procedure $V_\mathbb{A}$ for an mNP access structure $\mathbb{A}$, it does as following:

- Sample $N+1$ PRF keys for punctured PRF as $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [N]$, and for each $i \in [N]$ generate three independent commitments to the key K_i as $c_{i1} \leftarrow \mathsf{CS.Commit}(1^\lambda, K_i, r_{i1})$, $c_{i2} \leftarrow \mathsf{CS.Commit}(1^\lambda, K_i, r_{i2})$ and $c_{i3} \leftarrow \mathsf{CS.Commit}(1^\lambda, K_i, r_{i3})$ where r_{i1}, r_{i2}, r_{i3} are sampled randomly in $\mathcal{R}_\lambda$.
- Create an obfuscation for the program Recon defined below as described in Figure 3.
- Set $\mathsf{PP} = (\{c_{i1}, c_{i2}, c_{i3}\}_{i\in[N]}, i\mathcal{O}(\mathsf{Recon}))$ and $\mathsf{msk} = (K, \{(K_i, r_{i1}, r_{i2}, r_{i3})\}_{i\in[N]})$.

GEval(msk, x)**:** It parses $\mathsf{msk} = (K, \{(K_i, r_{i1}, r_{i2}, r_{i3})\}_{i\in[N]})$. On input $x \in \mathcal{X}$, it computes $y = \mathsf{PRF.Eval}(K, x)$.

Share(msk)**:** It parses $\mathsf{msk} = (K, \{(K_i, r_{i1}, r_{i2}, r_{i3})\}_{i\in[N]})$, and then sets $\mathsf{sk}_i = (K_i, r_{i1}, r_{i2}, r_{i3})$ for all $i \in [N]$.

PEval(i, sk_i, x)**:** It parses $\mathsf{sk}_i = ((K_i, r_{i1}, r_{i2}, r_{i3})$ and runs the PRF evaluation algorithm on x as $y_i = \mathsf{PRF.Eval}(K_i, x)$. It also computes a NIWI proof π_i for the statement $(c_{i1}, c_{i2}, c_{i3}, x, y_i) \in \mathcal{L}$ using the NIWI prover algorithm $\mathcal{P}$ with $(\imath = i1, \jmath = i2, K_i, K_i, r_{i1}, r_{i2})$ as witness, and outputs (y_i, π_i) as the partial evaluation and corresponding proof for the server S_i on input x.

PVerify$(\mathsf{PP}, i, x, y_i, \pi_i)$**:** It runs the NIWI verifier to check the proof π_i as $\mathcal{V}\big((c_{i1}, c_{i2}, c_{i3}, x, y_i), \pi_i\big)$ and accepts the proof (outputs 1) iff $\mathcal{V}$ outputs 1.

Comb$(\mathsf{PP}, V_{\mathbb{A}}, w, \{(x, y_i, \pi_i)\}_{i\in\Gamma})$**:** The client is expected to derive the function value from a set of qualified servers' partial evaluations and proofs. It executes the obfuscated program $i\mathcal{O}(\mathsf{Recon})$ on inputs w, PP and $\{(x, y_i, \pi_i)\}_{i\in\Gamma}$ to obtain y.

Recon

Hardwired into the circuit: $V_{\mathbb{A}}$, K, $\{c_{i1}, c_{i2}, c_{i3}\}_{i\in[N]}$.

Input to the circuit: x, w, $\{(y_i, \pi_i)\}_{i\in\Gamma}$.

Algorithm:

1. Execute the following tests:
   (a) For all $i \in \Gamma$ verify that $\mathcal{V}(c_{i1}, c_{i2}, c_{i3}, x, y_i, \pi_i) = 1$;
   (b) Verify that $V_{\mathbb{A}}(\Gamma, w) = 1$;
2. If any of the above tests fails, output $\perp$;
   otherwise, compute $y = \mathsf{PRF}(K, x)$ and output y.

Fig. 3: The description of the program Recon.

Theorem 1. *If $i\mathcal{O}$ is a secure indistinguishability obfuscator, $(\mathcal{P}, \mathcal{V})$ is a secure NIWI proof system for a language $\mathcal{L}$, $(\mathsf{CS.Commit}, \mathsf{CS.Verify})$ is a secure perfectly binding commitment scheme, and PRF is a secure puncturable PRF, then the above construction is a selectively-secure robust distributed PRF satisfying verifiability, robustness, correctness and selective pseudorandomness properties as described in Section 3.*

Verifiability. For the honestly generated keys $(\mathsf{PP}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, N, V_{\mathbb{A}})$ and $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathsf{msk})$ where $\mathsf{sk}_i = (K_i, r_{i1}, r_{i2}, r_{i3})$ and $\mathsf{PP} = (\{c_{i1}, c_{i2}, c_{i3}\}_{i\in[N]}, i\mathcal{O}(\mathsf{Recon}))$, we know that for every $i \in [N]$ both c_{i1} and c_{i2} are commitments to the PRF key K with r_{i1} and r_{i2} as the corresponding openings. Therefore, by the perfect correctness of the puncturable PRF and the NIWI proof system, we can conclude that $\mathsf{PVerify}(\mathsf{PP}, x, y_i, \pi_i) = 1$ for all $i \in [N]$.

Correctness. The *correctness* property can be verified in a straightforward manner from the proof of the verifiability property. Since on the one hand, we have shown that for every honestly generated keys $\mathsf{PP}, \mathsf{sk}_i$, by the perfect correctness of the puncturable PRF and NIWI proof system, for any set of qualified servers Γ with valid witness w (i.e., $V_{\mathbb{A}}(\Gamma, w) = 1$), we can conclude that $\mathsf{PartVerify}(\mathsf{pk}_i, x, (y_i, \sigma_i)) = 1$ for all $i \in \Gamma$. On the other hand, from the correctness property of $i\mathcal{O}$ we have $\mathsf{Comb}\big(\mathsf{PP}, V_{\mathbb{A}}, w, x, \{(y_i, \pi_i)\}_{i\in\Gamma}\big) = \mathsf{PRF}(K, x) = y$.

Robustness. We prove this by contradiction. Assume that for the corrupted server S_{j^*} holding the secret share K_{j^*} there exists (x', y''_j, π''_j) such that $y''_j \neq y'_j \ \wedge \ \mathsf{PVerify}(\mathsf{PP}, j, x', y''_j, \pi''_j) = 1$ where $(y'_j, \pi'_j) \leftarrow \mathsf{PEval}(j, K_j, x')$. From the

construction, it is implied that there exists $(x', (y''_j, \pi''_j), (y'_j, \pi'_j))$ such that $y''_j \neq y'_j$ and $\mathcal{V}(c_{ji}, c_{j2}, c_{j3}, x', y'_j, \pi'_j) = \mathcal{V}(c_{j1}, c_{j2}, c_{j3}, x', y''_j, \pi''_j) = 1$. However, this is not possible.

- Since the commitment scheme is perfectly binding, then for each $t \in \{1, 2, 3\}$ there exists at most one key K_j^t such that there exists an r_j^t, which is a valid opening for c_{jt}, *i.e.,* $\mathsf{CS.Verify}(K_j^t, c_{jt}, r_j^t) = 1$.
- Suppose c_{jt} is a commitment to the key K_j^t for $t \in \{1, 2, 3\}$, and $\mathsf{PRF}(K_j^1, x') = \mathsf{PRF}(K_j^2, x') = y'_j$. Now since $y''_j \neq y'_j$, thus even when $\mathsf{PRF}(K_j^3, x') = y''_j$ holds, we know that $(c_{j1}, c_{j2}, c_{j3}, x', y''_j) \notin \mathcal{L}$ because there are no two keys out of K_j^1, K_j^2, K_j^3 such that their function values equal to y''_j on input x'. Therefore, π''_j is a proof for an incorrect statement. Thus, it contradicts the statistical soundness of the NIWI proof system.

Pseudorandomness. To show that the *pseudorandomness* property holds, we prove it through a sequence of hybrid games. Let $\mathcal{A}$ be a PPT adversary that wins in the pseudorandomness game (described in Section 3) for the above construction of robust DPRFs with non-negligible probability. Let $Q = Q(\lambda)$ be a polynomial upper bound on the number of queries made by $\mathcal{A}$ to the partial evaluation oracles w.r.t. the uncorrupted servers. We argue that such an adversary must break the security of at least one of the underlying primitives.

To formally prove our theorem, we describe the following sequence of games, where the first game models the real pseudorandomness security game. Let $\mathsf{Adv}_{\mathcal{A},i}$ denote the advantage of an adversary $\mathcal{A}$ in the Game_i of guessing the bit b. We then show via a sequence of lemmas that if $\mathcal{A}$'s advantage is non-negligible in Game_i, then it has non-negligible advantage in Game_{i+1} as well. Note that in each successive hybrid, we only provide the steps that differ.

Game_0 this game is defined as the original selective pseudorandomness security game described in Section 3, instantiated by our construction of robust DPRFs.

1. $\mathcal{A}$ first chooses a challenge input x^* and an access structure $\mathbb{A} \in \mathsf{mNP}$, and outputs an unqualified set $T \subseteq [N]$ (that is, $T \notin \mathbb{A}$).
2. The challenger samples PRF keys $K, K_1, \ldots, K_N$, and for each $i \in [N]$, $t \in \{1, 2, 3\}$ generates three independent commitments to the key K_i as $c_{it} \leftarrow \mathsf{CS.Commit}(1^\lambda, K_i, r_{it})$, where r_{it} is sampled randomly in $\mathcal{R}_\lambda$. It creates an obfuscation for the program Recon defined in Figure 3. The challenger sets $\mathsf{PP} = (\{c_{i1}, c_{i2}, c_{i3}\}_{i \in [N]}, i\mathcal{O}(\mathsf{Recon}))$, and sends PP to the adversary $\mathcal{A}$.
3. Queries to the oracle $\mathsf{O_{PEval}}$: $\mathcal{A}$ sends the query (i, x) to $\mathsf{O_{PEval}}$ where $i \in [N] \backslash T$ and $x \neq x^*$, the challenger computes $y_i = \mathsf{PRF}(K_i, x)$ and a NIWI proof π_i for the statement $(c_{i1}, c_{i2}, c_{i3}, x, y_i) \in \mathcal{L}$ using the NIWI prover algorithm $\mathcal{P}$ with $(i1, i2, K_i, K_i, r_{i1}, r_{i2})$ as the witness.
4. The challenger chooses a random bit $b \leftarrow \{0, 1\}$. If $b = 1$, the challenger computes $y^* = \mathsf{PRF}(K, x^*)$, else it samples $y^* \in \mathcal{Y}$, and sends y^* to $\mathcal{A}$.
5. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b' = b$.

For each $i \in [N] \backslash T$, we define the following intermediate hybrid experiments, which are the hybrid games that are performed iteratively.

Game$_{1,i,1}$ the challenger uses $(i2, i3, K_i, K_i, r_{i2}, r_{i3})$ as the witness instead of $(i1, i2, K_i, K_i, r_{i1}, r_{i2})$ for generating the NIWI proof for the server S_i.

Game$_{1,i,2}$ the challenger generates c_{i1} as $c_{i1} \leftarrow \mathsf{Com}(K_i^*; r_{i1})$ instead of using a commitment to $c_{i1} \leftarrow \mathsf{Com}(K_i; r_{i1})$.

Game$_{1,i,3}$ the challenger uses $(i1, i3, K_i^*, K_i, r_{i1}, r_{i3})$ as the witness instead of $(i2, i3, K_i, K_i, r_{i2}, r_{i3})$ for generating the NIWI proof for server S_i.

Game$_{1,i,4}$ the challenger generates c_{i2} as $c_{i2} \leftarrow \mathsf{Com}(K_i^*; r_{i2})$ instead of using a commitment to $c_{i2} \leftarrow \mathsf{Com}(K_i; r_{i2})$.

Game$_{1,i,5}$ the challenger uses $(i1, i2, K_i^*, K_i^*, r_{i1}, r_{i2})$ as the witness instead of $(i1, i3, K_i^*, K_i, r_{i1}, r_{i3})$ for generating the NIWI proof for the server S_i.

Game$_2$ after the previous polynomial intermediate hybrid experiments, for each $i \in [N] \backslash T$ the commitments c_{i1}, c_{i2}, c_{i3} are generated as the commitments to the same punctured key K_i^* with different randomness. In this game, we replace the obfuscation of the program Recon which is defined in Figure 3 with the obfuscation of the program Recon^1 defined in Figure 4. Namely, the challenger computes $\hat{y} = \mathsf{PRF}(K, x^*)$ and creates an obfuscation for the program Recon^1 defined in Figure 4, which is hardwired with $K^*, x^*, \hat{y}, \{c_{i1}, c_{i2}, c_{i3}\}_{i \in [N]}$, instead of being hardwired only with K. Namely,

1. $\mathcal{A}$ first chooses a challenge input x^* and an access structure $\mathbb{A} \in \mathsf{mNP}$, and outputs an unqualified set $T \subseteq [N]$ (that is, $T \notin \mathbb{A}$).
2. The challenger generates the PRF keys $K, K_1, \ldots, K_N$ and computes the punctured key $K^* \leftarrow \mathsf{PRF.Punctured}(K, x^*)$.
3. For each $i \in T$, the challenger generates independent commitments to the key K_i as $c_{it} \leftarrow \mathsf{CS.Commit}(1^\lambda, K_i, r_{it})$ for each $t \in \{1, 2, 3\}$, where $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and $r_{it} \leftarrow \mathcal{R}_\lambda$. For each $i \in [N] \backslash T$, the challenger generates independent commitments to the key K_i^* as $c_{it} \leftarrow \mathsf{CS.Commit}(1^\lambda, K_i^*, r_{it})$ for each $t \in \{1, 2, 3\}$, where $K_i^* \leftarrow \mathsf{PRF.Punctured}(K_i, x^*)$ and $r_{it} \leftarrow \mathcal{R}_\lambda$. The challenger computes $\hat{y} = \mathsf{PRF}(K, x^*)$ and creates an obfuscation for the program Recon^1 defined in Figure 4. The challenger sets $\mathsf{PP} = (\{c_{i1}, c_{i2}, c_{i3}\}_{i \in [N]}, i\mathcal{O}(\mathsf{Recon}^1))$, and sends PP to the adversary $\mathcal{A}$.
4. Queries to the oracle $\mathsf{O}_{\mathsf{PEval}}$: $\mathcal{A}$ sends the query (i, x) to $\mathsf{O}_{\mathsf{PEval}}$ where $i \in [N] \backslash T$ and $x \neq x^*$, the challenger computes $y_i = \mathsf{PRF}(K_i^*, x)$ and a NIWI proof π_i for the statement $(c_{i1}, c_{i2}, c_{i3}, x, y_i) \in \mathcal{L}$ using the NIWI prover algorithm $\mathcal{P}$ with $(i1, i2, K_i^*, K_i^*, r_{i1}, r_{i2})$ as the witness.
5. The challenger chooses a random bit $b \leftarrow \{0, 1\}$. If $b = 1$, the challenger computes $y^* = [\mathsf{PRF}(K, x^*)]_1$, else it samples $y^* \in \{0, 1\}$, and sends y^* to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b' = b$.

The security of $i\mathcal{O}$ implies that this game is indistinguishable from Game$_{1,i,5}$.

Game$_3$ the challenger randomly chooses $\hat{y} \in \mathcal{Y}$ instead of computing from $\hat{y} = \mathsf{PRF}(K, x^*)$.

Game$_4$ the challenger replaces the obfuscation of the program Recon^1 with the obfuscation of the program Recon.

Recon^1

Hardwired into the circuit: $V_{\mathbb{A}}$, $\{c_{i1}, c_{i2}, c_{i3}\}_{i\in T}$, $\{c_{i1}, c_{i2}, c_{i3}\}_{i\in[N]\setminus T}$, Punctured key K^*, x^*, $\hat{y}$,.

Input to the circuit: x, w, $\{(y_i, \pi_i)\}_{i\in\Gamma}$.

Algorithm:

1. If $x \neq x^*$, execute the following tests:
   (a) For all $i \in \Gamma$ verify that $\mathcal{V}\big(c_{i1}, c_{i2}, c_{i3}, x, y_i, \pi_i\big) = 1$;
   (b) Verify that $V_{\mathbb{A}}(\Gamma, w) = 1$;
   If any of the above tests fails, output $\perp$;
   otherwise, compute $y = \mathsf{PRF}(K^*, x)$ and output y.
2. Else if $x = x^*$, execute (a) and (b);
   If both of the tests verify, output $\hat{y}$. Otherwise, output $\perp$;

Fig. 4: The description of the program Recon^1.

Game$_5$ the challenger replaces back the key to K_i from K_i^* committed in the (c_{i1}, c_{i2}, c_{i3}) for each $i \in [N]\setminus T$.

Due to space constraints, we will not present the proofs of the indistinguishability between each adjacent game in details. The complete proof is provided in the full version of this article.

5 Conclusions

Distributed PRFs is a very useful cryptographic primitive that can be employed in multiple settings in order to avoid single point of failures (*e.g.,* distributed key distribution centres, byzantine agreement protocols, consensus protocols). However, in many cases when a PRF value is computed in a distributed way, guarantees are needed for the correctness of the distributed partial evaluations of the PRF value. Robust distributed PRFs (DPRFs) can provide a proof of correctness π_i that a partial evaluation y_i is computed correctly. Existing robust DPRFs are suitable only for threshold access structures or monotone span programs, restricting thus, the application scenarios of robust DPRFs. In this paper, we investigated whether it is possible to construct robust distributed PRFs for a very general class of structures (mNP) rendering them appropriate for any access predicate (*i.e.,* a predicate satisfied by the distributed parties *e.g.,* all parties within a predefined distance) and thus, allowing a wider spectrum of applications. More precisely, we investigate whether it is possible to transform a PRF into a *robust distributed PRF* for monotone functions in NP, also known as mNP [20]. The answer is affirmative and we show that robust DPRFs for an mNP access structure can be constructed generally from the ingredients of puncturable PRFs [6,19,9], a non-interactive witness indistinguishable proof (NIWI) and indistinguishable obfuscation [11]. We believe that our robust DPRFs can have important impact in a broad range of application scenarios that require the distributed computation of a pseudorandom value.

6 Acknowledgements

This work was partially supported by the Swedish Research Council (Vetenskapsrådet) through the grant PRECIS (621-2014-4845).

References

1. B. Barak, S. J. Ong, and S. Vadhan. Derandomization in cryptography. *SIAM Journal on Computing*, 37(2):380–400, 2007.
2. N. Bitansky and O. Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 401–427. Springer, 2015.
3. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Y. G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 31–46. Springer Berlin Heidelberg, 2002.
4. D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In H. Shacham and A. Boldyreva, editors, *Proceedings of CRYPTO 2018*. Springer, 2018.
5. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key homomorphic prfs and their applications. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*. Springer Berlin Heidelberg, 2013.
6. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *Proceeding of ASIACRYPT 2013*, pages 280–300. Springer.
7. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In E. Oswald and M. Fischlin, editors, *Proceedings of EUROCRYPT 2015*. Springer, 2015.
8. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1292–1303. ACM, 2016.
9. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *Public-Key Cryptography – PKC 2014*. Springer Berlin Heidelberg, 2014.
10. A. De Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge with preprocessing. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*. Springer New York, 1990.
11. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of FOCS '13*, pages 40–49, Washington, DC, USA, 2013. IEEE Computer Society.
12. R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *Applied Cryptography and Network Security*. Springer, 2016.
13. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of rsa functions. In N. Koblitz, editor, *Advances in Cryptology — CRYPTO '96*. Springer Berlin Heidelberg, 1996.
14. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of STOC'15*, pages 469–477, New York, NY, USA, 2015. ACM.

15. R. Goyal, S. Hohenberger, V. Koppula, and B. Waters. A generic approach to constructing and proving verifiable random functions. Technical report, IACR Cryptology ePrint Archive, 2017.
16. M. Grigni and M. Sipser. Monotone complexity, 1990.
17. J. Groth, R. Ostrovsky, and A. Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)*, 59(3):11, 2012.
18. A. Jain, P. M. Rasmussen, and A. Sahai. Threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017:257, 2017. https://eprint.iacr.org/2017/257.
19. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, pages 669–684, New York, NY, USA, 2013. ACM.
20. I. Komargodski, M. Naor, and E. Yogev. Secret-sharing for np. *Journal of Cryptology*, 30(2):444–469, 2017.
21. I. Komargodski and M. Zhandry. Cutting-edge cryptography through the lens of secret sharing. In *Theory of Cryptography Conference*, pages 449–479. Springer, 2016.
22. B. Liang and A. Mitrokotsa. Distributed pseudorandom functions for general access structures in np. In *International Conference on Information and Communications Security*, pages 81–87. Springer, 2017.
23. B. Libert, D. Stehlé, and R. Titiu. Adaptively secure distributed prfs from lwe. *IACR Cryptology ePrint Archive*, 2018:927, 2018. https://eprint.iacr.org/2018/927.
24. R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Commun. ACM*, 24(9):583–584, Sept. 1981.
25. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *Proceedings of FOCS '99*, pages 120–130, 1999.
26. M. Naor, B. Pinkas, and O. Reingold. Distributed Pseudo-random Functions and KDCs. In *Advances in Cryptology–EUROCRYPT'99*, pages 327–346. Springer-Verlag, Berlin, Heidelberg, 1999.
27. J. B. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology – CRYPTO 2002*, pages 401–416. 2002.
28. N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.
29. A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of STOC '14*, pages 475–484. ACM, 2014.
30. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*. Springer New York, 1990.
31. V. Shoup. Practical threshold signatures. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*. Springer Berlin Heidelberg, 2000.
32. D. R. Stinson and R. Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In V. Varadharajan and Y. Mu, editors, *Information Security and Privacy*. Springer Berlin Heidelberg, 2001.

A Appendix

A.1 Preliminaries

Definition 5 (Indistinguishability obfuscation [11]). *A probabilistic polynomial time (PPT) algorithm $i\mathcal{O}$ is said to be an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$, if the following conditions are satisfied:*

- *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in C_\lambda$, for all inputs x, we have that*
$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1.$$
- *For any (not necessarily uniform) PPT adversaries (Samp, D), there exists a negligible function $negl(\cdot)$ such that the following holds: if $\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] > 1 - negl(\lambda)$, then we have:*
$$\Big|\Pr[D(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] - \Pr[D(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)]\Big| \leq negl(\lambda).$$

Definition 6 (Puncturable PRFs [29]). *A puncturable family of PRFs F mapping is given by a triple of Turing Machines $(\mathsf{Setup}_F, \mathsf{Puncture}_F, \text{and } \mathsf{Eval}_F)$, and a pair of computable functions $\tau_1(\cdot)$ and $\tau_2(\cdot)$, satisfying the following conditions:*

- ***(Functionality preserved under puncturing)*** *For every PPT adversary $\mathcal{A}$ such that $\mathcal{A}(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{\tau_1(\lambda)}$, then for all $x \in \{0,1\}^{\tau_1(\lambda)}$ where $x \notin S$, we have that:*
$$\Pr[\mathsf{Eval}_F(K, x) = \mathsf{Eval}_F(K_S, x) : K \leftarrow \mathsf{Setup}_F(1^\lambda), K_S = \mathsf{Puncture}_F(K, S)] = 1.$$
- ***(Pseudorandom at punctured points)*** *For every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{\tau_1(\lambda)}$ and state σ, consider an experiment where $K \leftarrow \mathsf{Setup}_F(1^\lambda)$ and $K_S = \mathsf{Puncture}_F(K, S)$. Then, we have:*
$$\Big|\Pr[\mathcal{A}_2(\sigma, K_S, S, \mathsf{Eval}_F(K, S)) = 1] - \Pr[\mathcal{A}_2(\sigma, K_S, S, U_{\tau_2(\lambda)\cdot|S|}) = 1]\Big| = negl(\lambda),$$

where $\mathsf{Eval}_F(K, S)$ denotes the concatenation of $\mathsf{Eval}_F(K, x_1), \ldots, \mathsf{Eval}_F(K, x_k)$ where $S = \{x_1, \ldots, x_k\}$ is the enumeration of the elements of S in lexicographic order, $negl(\cdot)$ is a negligible function, and $U_{\tau_2(\lambda)\cdot|S|}$ denotes the uniform distribution over $\tau_2(\lambda) \cdot |S|$ bits.

Theorem 2. *[29] If one-way functions exist, then for all efficiently computable functions $\tau_1(\lambda)$ and $\tau_2(\lambda)$, there exists a family of puncturable PRFs that maps $\tau_1(\lambda)$ bits to $\tau_2(\lambda)$ bits.*