# Distributed Pseudorandom Functions for General Access Structures in NP

Bei Liang[1] and Aikaterini Mitrokotsa[1]

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

**Abstract.** Distributed pseudorandom functions (DPRFs) originally introduced by Naor, Pinkas and Reingold (EUROCRYPT '99) are pseudorandom functions (PRFs), whose computation is distributed to multiple servers. Although by distributing the function computation, we avoid single points of failures, this distribution usually implies the need for multiple interactions with the parties (servers) involved in the computation of the function. In this paper, we take distributed pseudorandom functions (DPRFs) even further, by pursuing a very natural direction. We ask if it is possible to construct *distributed PRFs* for a general class of access mechanism going beyond the threshold access structure and the access structure that can be described by a polynomial-size monotone span programs. More precisely, our contributions are two-fold and can be summarised as follows: *(i)* we introduce the notion of single round distributed PRFs for a general class of access structure (monotone functions in $\mathsf{NP}$), *(ii)* we provide a provably secure general construction of distributed PRFs for every $\mathsf{mNP}$ access structure from puncturable PRFs based on indistinguishable obfuscation.

**Keywords**: Distributed pseudorandom functions, Puncturable PRFs, Function secret sharing.

## 1 Introduction

Distributing the computation of a function is a rather important method employed in order to avoid performance bottlenecks as well as single point of failures due to security compromises or even increased demand (*i.e.,* overloaded servers). Investigating the distribution of trapdoor functions for public key cryptography has already received a lot of attention [2, 3]. However, the computation of distributed functions that are useful in secret key cryptography *e.g.,* pseudorandom functions (PRFs) has received limited attention [7, 9, 8].

As a motivating example for the use of distributed PRFs, let us consider the scenario of a one-time password system (*e.g.,* RSA SecurID). Users obtain one-time passwords from this system by sending inputs. Each password should be random and independent from the other passwords, and asking the evaluating system on the same input twice should yield the same (random) output. In this system, one assumes the existence of an authentication server, who has a secret

key $K$ and responds with PRF outputs $\mathsf{PRF}_K(x)$ that are used as the users' one-time passwords. Since the server knows the secret PRF key, this authentication server is a prime target for attacks. The natural solution to this problem is to distribute the role of the authentication server among many servers. This leads to the notion of *distributed PRFs (DPRFs)*.

In this paper, we investigate whether it is possible to construct distributed PRFs for a general class of access mechanism, going beyond the existing threshold access structure (*i.e.,* at least $t$-out-of-$N$ servers are required to evaluate the PRF) and the access structure that can be described by a polynomial-size monotone span programs (*e.g.,* undirected connectivity in a graph).

More precisely our contributions are two-fold: *(i)* we introduce the notion of single round distributed PRFs for a general class of access structures (monotone functions in $\mathsf{NP}$), *(ii)* we provide a provably secure general construction of distributed PRFs for every $\mathsf{mNP}$ access structure from puncturable PRFs based on indistinguishable obfuscation.

**Distributed PRFs** *Distributed pseudorandom functions* (DPRFs), originally introduced by Naor *et al.* [7], 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.* [7] considered the setting where the PRF secret key is split among $N$ key servers and at least $t$ servers are needed to evaluate the PRF. The distributed PRF in this setting is known as *distributed PRF* for threshold access structure. Very importantly, evaluating the PRF is done without reconstructing the key at a single location. Naor *et al.* [7] also presented constructions of DPRFs based on general monotone access structures, such as monotone symmetric branching programs (contact schemes), and monotone span programs.

Although some distributed PRFs (DPRFs) schemes have been proposed, all previous constructions have some limitations. Naor *et al.* [7] gave several efficient constructions of certain weak variants of DPRFs. One of their DPRF constructions requires the use of random oracles. To eliminate the use of random oracles, Nielsen [9] provided the first regular DPRF by distributing a slightly modified variant of the Naor-Reingold PRF [8]. Unfortunately, the resulting DPRF is highly interactive among the servers and requires a lot of rounds.

Boneh *et al.* [1] gave an efficient construction of DPRF for $t$-out-of-$N$ threshold access structure from LWE using a key homomorphic PRF. Boneh *et al.* apply Shamir's $t$-out-of-$N$ threshold secret sharing scheme [2] on top of their LWE-based key homomorphic PRF scheme, which results in an one-round DPRF with no interaction among the key servers. However, the question of constructing single round, non-interactive distributed PRFs that support more general access structures such as monotone functions in $\mathsf{NP}$ remained open prior to this work.

**Our contributions** In this work, we consider *single round distributed PRFs* for a more general class of access structures than the existing monotone access structures: monotone functions in $\mathsf{NP}$, also known as $\mathsf{mNP}$ (firstly considered

by Komargodski *et al.* [6]). An access structure that is defined by a function in $\mathsf{mNP}$ is called an $\mathsf{mNP}$ access structure. We also give a generic construction of distributed PRFs for every $\mathsf{mNP}$ access structure from puncturable PRFs based on indistinguishable obfuscation [4].

Intuitively, a single round distributed PRF for an $\mathsf{mNP}$ access structure is defined as follows: given an access structure, the setup algorithm outputs a public parameter $\mathsf{PP}$ and a secret key $\alpha$ which defines a PRF. On input the secret key $\alpha$, there is an algorithm that allows to "split" $\alpha$ into many "shares" $\alpha_i$ and then to distribute the shares to a collection of servers. Using its own key, $\alpha_i$, each server can compute a partial evaluation on input $x$. For the "qualified" subsets, there is a witness attesting to this fact and given the witness and the partial values of these "qualified" servers it should be possible to reconstruct the evaluation of PRF on input $x$. On the other hand, for the "unqualified" subsets there is no witness, and so it should not be possible to reconstruct the PRF on input $x$. For example, consider the Hamiltonian access structure. In this access structure, the parties correspond to edges of a complete undirected graph, and a set of parties $X$ is said to be "qualified", if and only if the corresponding set of edges contains a Hamiltonian cycle and the set of parties knows a witness attesting to this fact.

Our central challenge is to reconstruct a function value on some input from a set of partial evaluations of the "qualified" servers. Prior solutions are based on specific PRFs with particular algebraic structures, in combination with Shamir's secret sharing scheme. These solutions employ the homomorphic property of the PRF to distribute the function value into different parts, from which in turn the PRF value can be reconstructed. Here, we explore a solution based on general PRFs with no algebraic structure. Our approach achieves these goals by employing program obfuscation. Both our general constructions of *distributed PRFs* are based on indistinguishability obfuscation [4] and we prove formally their security in the full version of this paper.

**Overview of Our Techniques** We now give a high level overview of our technical approach. A formal treatment is given in the main body of the paper.

We propose a general method that makes any puncturable PRF to be a distributed PRF for any $\mathsf{mNP}$ access structure based on indistinguishability obfuscation [4]. 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 $\mathsf{mNP}$ access structure $\mathbb{A}$, a trusted third party samples the PRF key $K$ as well as $N$ PRF keys $K_1, \ldots, K_N$, and creates an obfuscated program $i\mathcal{O}(\mathsf{Prog})$ which with keys $K$, $K_1, \ldots, K_N$ and $V_{\mathbb{A}}$ hardwired, takes as input the valid witness $w$ of the set of qualified servers $\Gamma \subseteq S_1, \ldots, S_N$, $\{\sigma_i\}_{i \in \Gamma}$ and $x$ and checks if the condition $V_{\mathbb{A}}(\Gamma, w) = 1$ and $\sigma_i = \mathsf{PRF}(K_i, x)$ holds for every $i \in \Gamma$. If the condition holds, the program $\mathsf{Prog}$ outputs $\mathsf{PRF}(K, x)$; otherwise it outputs $\perp$. Each server's key is given as $K_i$. For input $x$, each server computes $\sigma_i = \mathsf{PRF}(K_i, x)$ and outputs $\sigma_i$ as the partial share of the function value $\mathsf{PRF}(K, x)$. In order to reconstruct the function value on input $x$ from a set of shares of qualified servers $\Gamma$, with witness $w$ the client runs the public obfuscated program $i\mathcal{O}(\mathsf{Prog})$. We also show that the resulting distributed PRF 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.

## 2 Preliminaries

### 2.1 Monotone-NP and Access Structures

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 $\Gamma \subseteq \Gamma'$ 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 ([5]).** *We say that a function $L$ is in $\mathsf{mNP}$ if there exists a uniform family of polynomial-size monotone non-deterministic circuit that computes $L$.*

**Lemma 1 ([5]).** $\mathsf{mNP} = \mathsf{NP} \bigcap \mathsf{mono}$*, where* $\mathsf{mono}$ *is the set of all monotone functions.*

**Definition 2 (Access structure [6]).** *An 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 $\Gamma'$ such that $\Gamma \subseteq \Gamma' \subseteq \mathcal{S}$ it holds that $\Gamma' \in \mathbb{A}$.*

We may think of $\mathbb{A}$ 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 $\Gamma$ is in the access structure, namely $\Gamma \in \mathbb{A}$. We view $\mathbb{A}$ either as a function or as a language. Throughout this paper, we deal with distributed PRFs for access structures over $N$ servers $\mathcal{S} = \mathcal{S}_N = \{S_1, \ldots, S_N\}$.

## 3 Distributed Pseudorandom Functions

We now formally define the syntax and security notion of distributed pseudorandom functions (DPRFs) for any $\mathsf{mNP}$ access structure $\mathbb{A}$ on $N$ servers. It is a natural generalization of the definition of DPRFs given in [7] (which was proposed for threshold access structures).

Consider a PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm: on input $(K, x) \in \mathcal{K} \times \mathcal{X}$ the algorithm outputs $F(K, x) \in \mathcal{Y}$. To define distributed PRFs, we follow the partial exposition of Naor *et al.* [7]. The model comprises of $N$ servers $S_1, \ldots, S_N$ and for each of the servers $S_i$ the share space is $\mathcal{Z}_i$. Note we identify a server $S_i$ with its index $i$.

**Definition 3 (Distributed PRFs).** *For $N \in \mathbb{N}$, let $\mathbb{A}$ be an $\mathsf{mNP}$ access structure on $N$ servers $S_1, \ldots, S_N$. A distributed PRF for $\mathbb{A}$ is a tuple of polynomial time algorithms $\Pi =$ (Setup, Func, Gen, Eval, Comb) with the following syntax:*

- $\mathsf{Setup}(1^\lambda, N, V_{\mathbb{A}})$ : *On input the security parameter* $\lambda$*, 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* $\alpha$*.*
- $\mathsf{Func}(\alpha, x)$ : *On input the master secret key* $\alpha$ *and an input string* $x \in \mathcal{X}$*, the function evaluation algorithm outputs a function value* $y \in \mathcal{Y}$*.*
- $\mathsf{Gen}(\alpha)$ : *On input the master secret key* $\alpha$*, the key generation algorithm outputs* $N$ *keys,* $(\alpha_1, \ldots, \alpha_N)$*.*
- $\mathsf{Eval}(i, \alpha_i, x)$ : *On input a server index* $i$*, key* $\alpha_i$ *and input string* $x \in \mathcal{X}$*, the partial evaluation algorithm outputs a pair* $(x, y_i)$ *where* $y_i \in \mathcal{Z}_i$ *is the server's* $S_i$ *share of function value* $\mathsf{Func}(\alpha, x)$*.*
- $\mathsf{Comb}\big(\mathsf{PP}, V_{\mathbb{A}}, w, \{\mathsf{Eval}(i, \alpha_i, x)\}_{i \in \Gamma}\big)$ : *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* $\{\mathsf{Eval}(i, \alpha_i, x)\}_{i \in \Gamma}$ *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* correctness *and* pseudorandomness *requirements:*

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

$$\mathbf{Pr}[(\mathsf{PP}, \alpha) \leftarrow \mathsf{Setup}(1^\lambda, N, V_{\mathbb{A}}), (\alpha_1, \ldots, \alpha_N) \leftarrow \mathsf{Gen}(\alpha) : \mathsf{Comb}\big(\mathsf{PP}, V_{\mathbb{A}}, w, \{\mathsf{Eval}(i, \alpha_i, x)\}_{i \in \Gamma}\big) = \mathsf{Func}(\alpha, x)] = 1.$$

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

1. *On input the security parameter* $1^\lambda$ *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* $(\alpha_1, \ldots, \alpha_N) \leftarrow \mathsf{Gen}(\alpha)$*, 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, and for each query* $x_j$ *the challenger responds with* $\mathsf{Eval}(i, \alpha_i, x_j)$ *for all* $i \in [N] \backslash T$*.*
5. *The adversary submits a challenge query* $x^* \notin \{x_1, \ldots, x_Q\}$ *to the challenger. 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* $y^* = \mathsf{Func}(\alpha, x^*)$*.*
6. *The adversary continues to issue polynomially more queries of the form* $x_j \neq x^*$*, to which the challenger responds with* $\{\mathsf{Eval}(i, \alpha_i, x_j)\}_{i \in [N] \backslash T}$*.*
7. *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 distributed PRFs* $\Pi$ *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 Distributed PRFs

In this section, we describe our construction of DPRFs in details. The construction is parameterized over a security parameter $\lambda$ and a number $N$, has input space $\mathcal{X} = \{0,1\}^{\mathsf{inp}(\lambda)}$ and range space $\mathcal{Y} = \{0,1\}^{\mathsf{out}(\lambda)}$ for some polynomial functions $\mathsf{inp}(\cdot)$ and $\mathsf{out}(\cdot)$. It relies on the following primitives:

- A puncturable PRF $(\mathsf{Setup}_{\mathsf{PRF}}, \mathsf{Puncture}_{\mathsf{PRF}}, \mathsf{Eval}_{\mathsf{PRF}})$, that accepts inputs of length $\mathsf{inp}(\lambda)$ and outputs strings of length $\mathsf{out}(\lambda)$.
- $N$ puncturable PRFs $(\mathsf{Setup}_{\mathsf{PRF}_i}, \mathsf{Puncture}_{\mathsf{PRF}_i}, \mathsf{Eval}_{\mathsf{PRF}_i})$, for each $i \in [N]$, that accepts inputs of length $\mathsf{inp}(\lambda)$ and outputs strings of length $\ell(\lambda)$.

**Prog**

**Hardwired into the circuit:** $V_{\mathbb{A}}$, $K$ and $\{K_i\}_{i\in[N]}$.

**Input to the circuit:** $\{\sigma_i\}_{i\in\Gamma}$, $x$, $w$.

**Algorithm:**

1. Check $V_{\mathbb{A}}(\Gamma, w) = 1$. If it is true, go to the next step; else output $\perp$.
2. For all $i \in \Gamma$ check $\sigma_i = \mathsf{PRF}(K_i, x)$.
   If it is true, output $y = \mathsf{PRF}(K, x)$; else output $\perp$.

Fig. 1: The description of the programs Prog.

Our construction of a DPRF is composed of the following algorithms:

**Setup**$(1^\lambda, N, V_{\mathbb{A}})$**:** On input $1^\lambda$, $N$ and $V_{\mathbb{A}}$, it does as follows:
- sample PRF key $K \leftarrow \mathsf{Setup}_{\mathsf{PRF}}(1^\lambda)$ and $N$ keys $K_i \leftarrow \mathsf{Setup}_{\mathsf{PRF}_i}(1^\lambda)$.
- create an obfuscated program $i\mathcal{O}(\mathsf{Prog})$, where the program Prog is defined in Figure 1.
- set the secret key $\alpha = (K, \{K_i\}_{i\in[N]})$ and the public parameters $\mathsf{PP} = i\mathcal{O}(\mathsf{Prog})$.

**Func**$(\alpha, x)$**:** It parses $\alpha = (K, \{K_i\}_{i\in[N]})$. The function value for input $x \in \{0,1\}^{\mathsf{inp}}$ is defined as: $\mathsf{Func}(\alpha, x) = \mathsf{PRF}(K, x) = y$.

**Gen**$(\alpha)$**:** It parses $\alpha = (K, \{K_i\}_{i\in[N]})$ and sets $\alpha_i = K_i$ for every $i \in [N]$, where $\alpha_i$ is the key used to compute the corresponding server's share of the function value.

**Eval**$(i, \alpha_i, x)$**:** It parses $\alpha_i = K_i$, computes $\sigma_i = \mathsf{PRF}(K_i, x)$ and outputs $(\sigma_i, x)$, where $\sigma_i$ is the server's $S_i$ share of the function value $\mathsf{Func}(\alpha, x)$.

$\mathsf{Comb}\big(\mathsf{PP}, V_{\mathbb{A}}, w, \{\mathsf{Eval}(i, \alpha_i, x)\}_{i\in\Gamma}\big)$**:** It parses $\mathsf{Eval}(i, \alpha_i, x) = (\sigma_i, x)$ for every $i \in \Gamma$ and runs $y \leftarrow i\mathcal{O}(\mathsf{Prog})(\{\sigma_i\}_{i\in\Gamma}, x, w)$ to obtain value $y$. Output $y$.

**Theorem 1.** *If $i\mathcal{O}$ is a secure indistinguishability obfuscator, $\mathsf{PRF}(K, \cdot)$ is a secure puncturable PRF, and for each $i \in [N]$ $\mathsf{PRF}(K_i, \cdot)$ is a secure puncturable PRF, then our distributed PRF given above is a selectively pseudorandom distributed PRF for the $\mathsf{mNP}$ access structure $\mathbb{A}$, as defined in Definition 3.*

The complete proof is provided in the full version of this article.

## 5 Conclusion

In this paper, we consider single round distributed PRFs for a more general class of access structures: monotone functions in NP. We also give a generic construction of distributed PRFs for every mNP access structure from puncturable PRFs based on indistinguishable obfuscation.

## 6 Acknowledgements

This work was partially supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement n 608743 and the STINT grant IB 2015-6001.

## References

1. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key homomorphic PRFs and their applications. In *Advances in Cryptology–CRYPTO 2013*, pages 410–428. Springer, 2013.
2. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proceedings of STOC '94*, pages 522–533, New York, NY, USA, 1994. ACM.
3. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology–CRYPTO '89*, pages 307–315. Springer-Verlag, London, UK, UK, 1990.
4. 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, DC, SA, 2013. IEEE Computer Society.
5. M. Grigni and M. Sipser. Monotone complexity, 1990.
6. I. Komargodski, M. Naor, and E. Yogev. Secret-sharing for np. *Journal of Cryptology*, 30(2):444–469, 2017.
7. 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.
8. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.
9. J. B. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology – CRYPTO 2002*, pages 401–416. 2002.