Adaptively Secure Distributed PRFs from LWE

Benoît Libert[1,2] **Damien Stehlé**[2] **Radu Ţiţiu**[2,3]

[1]CNRS (France)
[2]ENS de Lyon (France)
[3]Bitdefender (Romania)

June 13, 2019

Nanyang Technological University

Outline

1. Distributed PRFs
2. Known Constructions from Key-homomorphic PRFs
3. Achieving Adaptive Security
4. Robustness against Malicious Adversaries

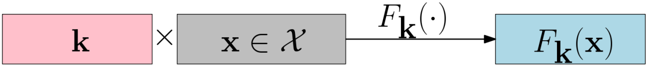
Pseudorandom functions (PRFs) (Goldreich-Goldwasser-Micali; FOCS'84)

Is an efficiently computable function, $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

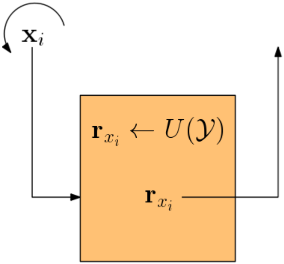
$$\boxed{\mathbf{k}} \times \boxed{\mathbf{x} \in \mathcal{X}} \xrightarrow{F_{\mathbf{k}}(\cdot)} \boxed{F_{\mathbf{k}}(\mathbf{x})}$$

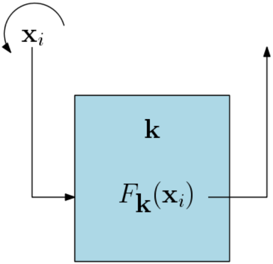
Pseudorandom functions (PRFs) (Goldreich-Goldwasser-Micali; FOCS'84)

Is an efficiently computable function, $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$



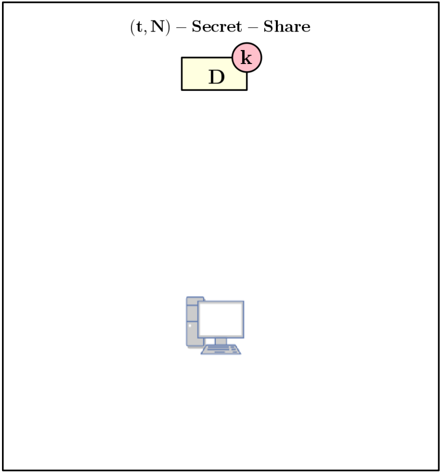
s.t. for $k \leftarrow U(\mathcal{K})$, the following boxes are indistinguishable:





Distributed PRFs (Micali-Sidney; Crypto'95)

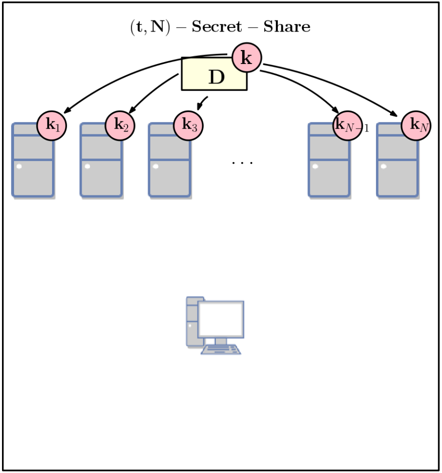
Threshold **sharing** of the **key** among N servers



- Dealer runs $(pp, k) \leftarrow \mathsf{Setup}(\lambda)$

Distributed PRFs (Micali-Sidney; Crypto'95)

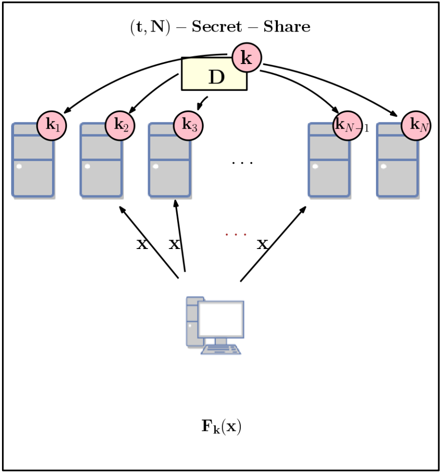
Threshold **sharing** of the **key** among N servers



- Dealer runs $(pp, k) \leftarrow \mathsf{Setup}(\lambda)$
- Dealer runs $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$ and gives k_i to server i

Distributed PRFs (Micali-Sidney; Crypto'95)

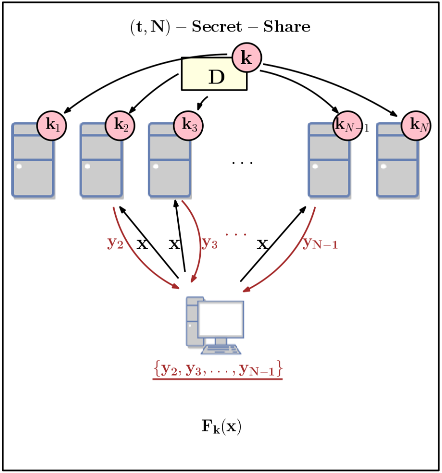
Threshold **sharing** of the **key** among N servers



- Dealer runs $(pp, k) \leftarrow \mathsf{Setup}(\lambda)$
- Dealer runs $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$ and gives k_i to server i
- On input x, each server computes $y_i = \mathsf{PEval}(k_i, x)$

Distributed PRFs (Micali-Sidney; Crypto'95)

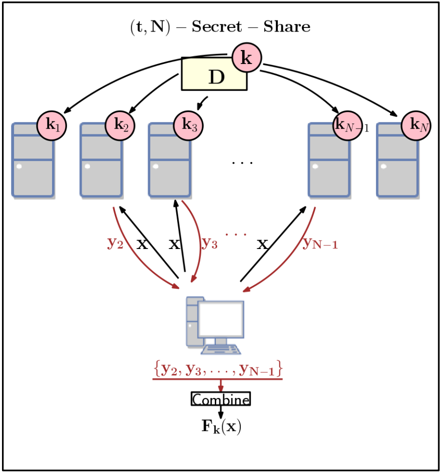
Threshold **sharing** of the **key** among N servers



- Dealer runs $(pp, k) \leftarrow \mathsf{Setup}(\lambda)$
- Dealer runs $(k_1, \dots, k_N) \leftarrow \mathsf{Share}(k)$ and gives k_i to server i
- On input x, each server computes $y_i = \mathsf{PEval}(k_i, x)$

Distributed PRFs (Micali-Sidney; Crypto'95)

Threshold **sharing** of the **key** among N servers



- Dealer runs $(pp, k) \leftarrow \mathsf{Setup}(\lambda)$
- Dealer runs $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$ and gives k_i to server i
- On input x, each server computes $y_i = \mathsf{PEval}(k_i, x)$
- Using any t partial evaluations, the user computes

$$F_k(x) \leftarrow \mathsf{Combine}(y_{i_1}, y_{i_2}, \ldots, y_{i_t})$$

Motivations

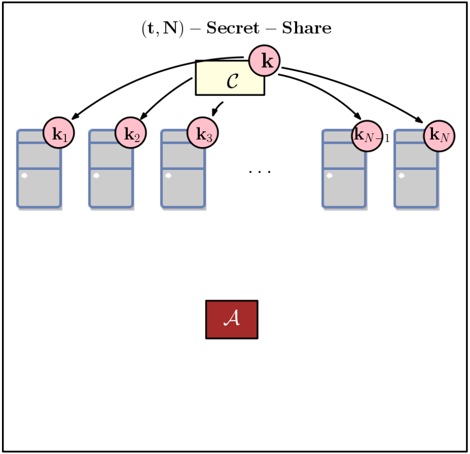
- Distributed symmetric encryption
`(Agrawal-Mohassel-Mukherjee-Rindal; CCS 2018)`

- Distributed key distribution centers
(`Naor-Pinkas-Reingold; Eurocrypt'99`)

- Distributed coin tossing: e.g., allows threshold Cramer-Shoup
`(Canetti-Goldwasser; Eurocrypt'99)` without pre-shared randomness

Statically secure DPRFs

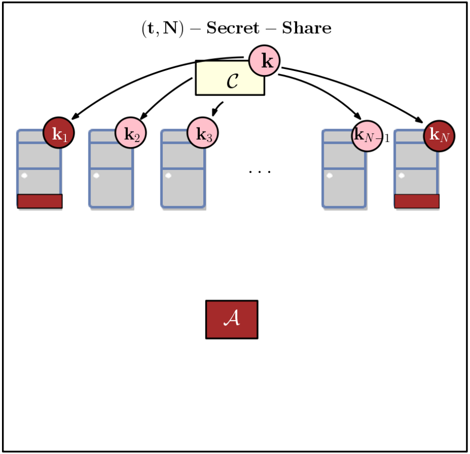
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$

Statically secure DPRFs

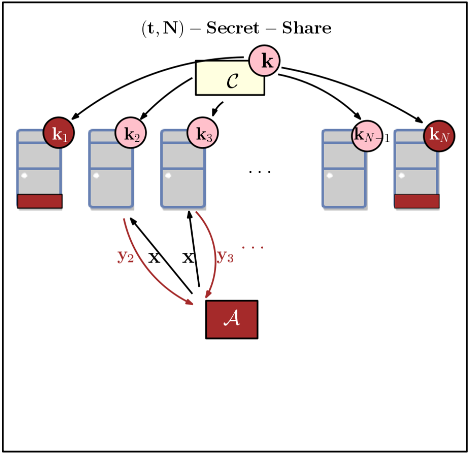
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \dots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ chooses $\{i_1, i_2, \dots, i_{t-1}\} \subset [N]$ and gets $\{k_{i_1}, k_{i_2}, \dots, k_{i_{t-1}}\} \subset [N]$

Statically secure DPRFs

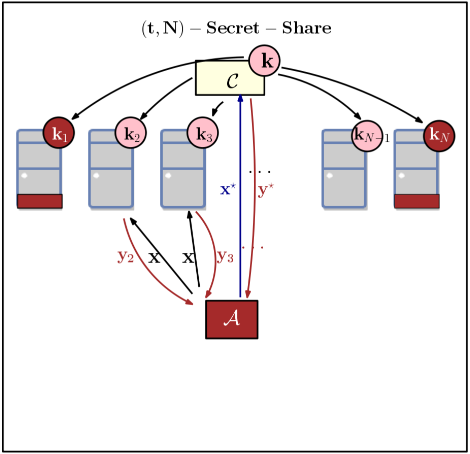
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ chooses $\{i_1, i_2, \ldots, i_{t-1}\} \subset [N]$ and gets $\{k_{i_1}, k_{i_2}, \ldots, k_{i_{t-1}}\} \subset [N]$
3. $\mathcal{A}$ adaptively learns partial evaluations $y_i = \mathsf{PEval}(k_i, x)$

Statically secure DPRFs

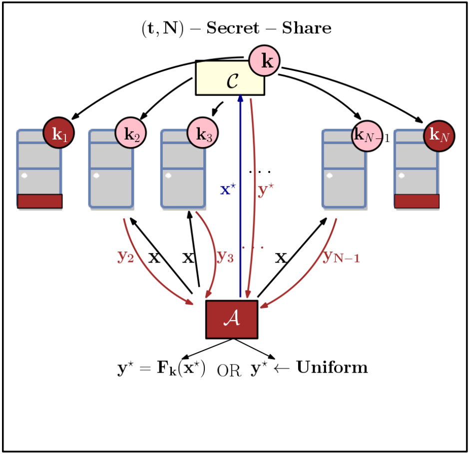
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ chooses $\{i_1, i_2, \ldots, i_{t-1}\} \subset [N]$ and gets $\{k_{i_1}, k_{i_2}, \ldots, k_{i_{t-1}}\} \subset [N]$
3. $\mathcal{A}$ adaptively learns partial evaluations $y_i = \mathsf{PEval}(k_i, x)$
4. $\mathcal{A}$ chooses $x^\star$; $\mathcal{C}$ flips $b \leftarrow \{0, 1\}$ $\mathcal{C}$ returns $F_k(x^\star)$ or *random*

Statically secure DPRFs

Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$

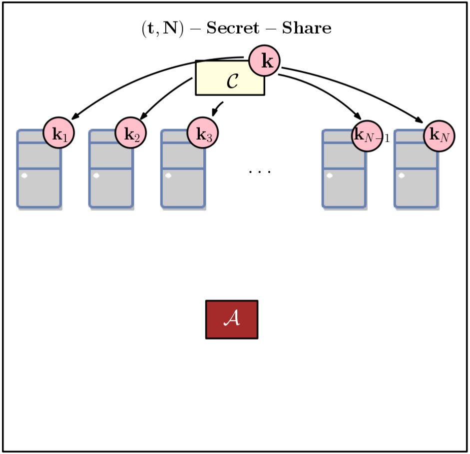


1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ chooses $\{i_1, i_2, \ldots, i_{t-1}\} \subset [N]$ and gets $\{k_{i_1}, k_{i_2}, \ldots, k_{i_{t-1}}\} \subset [N]$
3. $\mathcal{A}$ adaptively learns partial evaluations $y_i = \mathsf{PEval}(k_i, x)$
4. $\mathcal{A}$ chooses $x^\star$; $\mathcal{C}$ flips $b \leftarrow \{0,1\}$ $\mathcal{C}$ returns $F_k(x^\star)$ or *random*
5. $\mathcal{A}$ makes more queries and outputs $b' \in \{0,1\}$

We want: $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$

Adaptively secure DPRFs

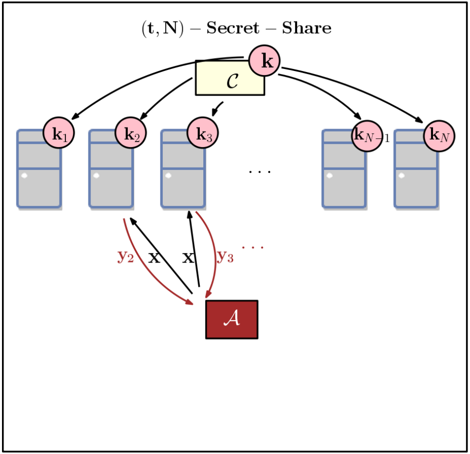
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$

Adaptively secure DPRFs

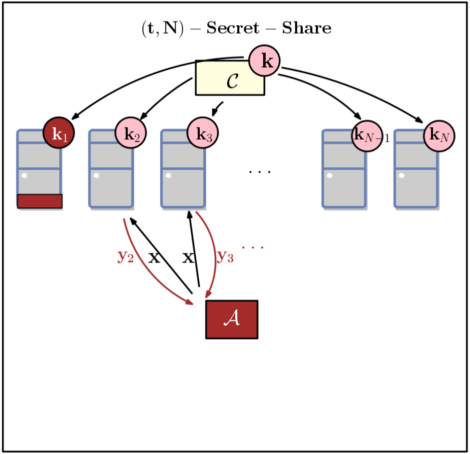
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ interleaves corruption queries and partial evaluation queries

Adaptively secure DPRFs

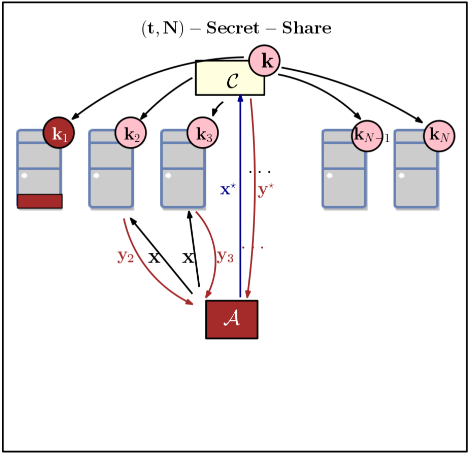
Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ interleaves corruption queries and partial evaluation queries

Adaptively secure DPRFs

Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



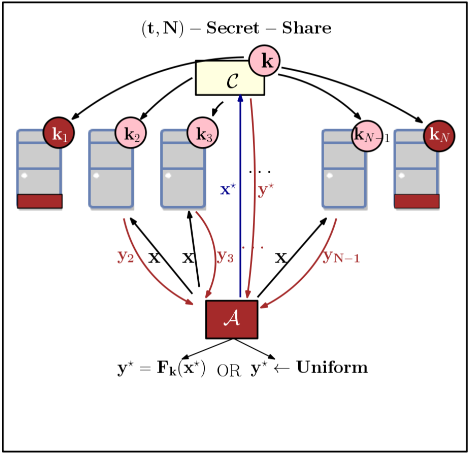
1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$

2. $\mathcal{A}$ interleaves corruption queries and partial evaluation queries

3. $\mathcal{A}$ chooses $x^\star$; $\mathcal{C}$ flips $b \leftarrow \{0, 1\}$
   $\mathcal{C}$ replies with $F_k(x^\star)$ or *random*

Adaptively secure DPRFs

Security modeled as an interactive game between $\mathcal{A}$ and $\mathcal{C}$



1. $\mathcal{C}$ runs $(pp, k) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(k)$
2. $\mathcal{A}$ interleaves corruption queries and partial evaluation queries
3. $\mathcal{A}$ chooses $x^\star$; $\mathcal{C}$ flips $b \leftarrow \{0, 1\}$ $\mathcal{C}$ replies with $F_k(x^\star)$ or *random*
4. $\mathcal{A}$ makes more queries and outputs $b' \in \{0, 1\}$

We want: $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$

Related work on DPRFs

Multiple interaction rounds:

- Nielsen (Crypto'02), Dodis (PKC'03), Dodis-Yampolskiy-Yung (TCC'06), ...

Non-interactive:

- Micali and Sidney (Crypto'95): very small of very large t w.r.t N
- Naor, Pinkas and Reingold (Eurocrypt'99): uses random oracles
- Boneh *et al.* (Crypto'13): generic construction from key-homomorphic PRFs
- Boneh *et al.* (Crypto'18): generic construction from threshold FHE

All secure in the **static corruption** setting

This work

Our results: We build the first DPRF which is:

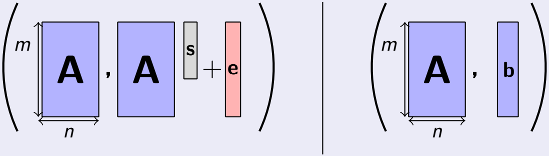
- secure under **adaptive corruptions**
- secure under the **standard LWE assumption**
- **non-interactive**
- **without random oracles**

Hardness assumptions

The **Learning-With-Errors** (LWE) problem `(Regev, STOC'05)`

Parameters: dimension n, number of samples $m \geq n$, modulus q.

For $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e}$ a small error $\approx \alpha q$, distinguish



for uniform $\mathbf{b} \xleftarrow{R} \mathbb{Z}_q^m$

- As hard as standard *worst-case* lattice problems when $\alpha q > \sqrt{n}$
- Conjectured quantum-resistant
- Enables powerful functionalities (e.g., FHE, ABE for circuits)

Key-homomorphic PRFs $\Rightarrow$ static DPRFs

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x)$$

Key-homomorphic PRFs $\Rightarrow$ static DPRFs

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x)$$

To obtain a **DPRF**, use Shamir's secret sharing:

- $k \leftarrow \mathcal{K}, \quad (k_1, k_2, \ldots, k_N) \leftarrow \mathsf{ShamirSS}(k, t, N)$

Key-homomorphic PRFs $\Rightarrow$ static DPRFs

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x)$$

To obtain a **DPRF**, use Shamir's secret sharing:

- $k \leftarrow \mathcal{K}, \quad (k_1, k_2, \ldots, k_N) \leftarrow \mathsf{ShamirSS}(k, t, N)$
- $\mathsf{PEval}(k_i, x) := F(k_i, x)$

Key-homomorphic PRFs $\Rightarrow$ static DPRFs

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x)$$

To obtain a **DPRF**, use Shamir's secret sharing:

- $k \leftarrow \mathcal{K}, \quad (k_1, k_2, \ldots, k_N) \leftarrow \mathsf{ShamirSS}(k, t, N)$
- $\mathsf{PEval}(k_i, x) := F(k_i, x)$
- $\mathit{Combine}(y_{i_1}, \ldots, y_{i_t})$: compute $\lambda_{W,i}$ such that:

$$k = \sum_{i \in W} \lambda_{W,i} \cdot k_i$$

  recover the evaluation as:

$$\sum_{i \in W} \lambda_{W,i} \cdot F(k_i, x) = F(k, x)$$

Almost key-homomorphic PRFs $\Rightarrow$ static DPRFs

Almost key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) + \mathsf{e}$$

for all $k_1, k_2 \in \mathcal{K}, x \in \mathcal{X}$ and some small error e

Almost key-homomorphic PRFs ⇒ static DPRFs

Almost key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) + \boxed{\mathsf{e}}$$

for all $k_1, k_2 \in \mathcal{K}, x \in \mathcal{X}$ and some small error $\boxed{\mathsf{e}}$

Theorem (Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Secure almost key-homomorphic PRF ⇒ **statically** *secure DPRF*

Almost key-homomorphic PRFs $\Rightarrow$ static DPRFs

Almost key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) + \boxed{\mathsf{e}}$$

for all $k_1, k_2 \in \mathcal{K}, x \in \mathcal{X}$ and some small error $\boxed{\mathsf{e}}$

Theorem (Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Secure almost key-homomorphic PRF $\Rightarrow$ **statically** *secure DPRF*

Almost key-homomorphic PRFs from LWE

Use $F_{\mathbf{s}} : \{0,1\}^L \to \mathbb{Z}_p^m$, with $F_{\mathbf{s}}(x) = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor_p = \left\lfloor \frac{p}{q} \cdot \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor$

Almost key-homomorphic PRFs $\Rightarrow$ static DPRFs

Almost key-homomorphic PRFs are secure PRFs that also satisfy:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) + \boxed{e}$$

for all $k_1, k_2 \in \mathcal{K}, x \in \mathcal{X}$ and some small error $\boxed{e}$

Theorem (Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

Secure almost key-homomorphic PRF $\Rightarrow$ **statically** *secure DPRF*

Almost key-homomorphic PRFs from LWE

Use $F_{\mathbf{s}} : \{0,1\}^L \to \mathbb{Z}_p^m$, with $F_{\mathbf{s}}(x) = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor_p = \left\lfloor \frac{p}{q} \cdot \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor$

$$\left\lfloor \boxed{\mathbf{A}(x)} \cdot \left(\boxed{\mathbf{s_1}} + \boxed{\mathbf{s_2}} \right) \right\rfloor_p = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s_1}} \right\rfloor_p + \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s_2}} \right\rfloor_p + \boxed{\mathbf{e}}$$

$$F_{\mathbf{s_1}+\mathbf{s_2}}(x) = F_{\mathbf{s_1}}(x) + F_{\mathbf{s_2}}(x) + \mathbf{e}, \text{ with } \mathbf{e} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{m}}$$

Towards adaptive security

Starting point: Modification of (Boneh *et al.*; Crypto'13)

- Let $\boxed{\mathbf{A}_0}, \boxed{\mathbf{A}_{i,b}} \leftarrow U(\mathbb{Z}_q^{m \times n})$ for each $i \in [L]$, $b \in \{0,1\}$:

$$F_{\mathbf{s}} : \{0,1\}^L \to \mathbb{Z}_p^m, \quad \text{given by} \quad F_{\mathbf{s}}(x) = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor_p = \left\lfloor \frac{p}{q} \cdot \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor$$

where $\boxed{\mathbf{s}} \leftarrow U(\mathbb{Z}_q^n)$ is the secret key

Towards adaptive security

Starting point: Modification of `(Boneh` *et al.*; `Crypto'13)`

- Let $\boxed{\mathbf{A}_0}, \boxed{\mathbf{A}_{i,b}} \leftarrow U(\mathbb{Z}_q^{m \times n})$ for each $i \in [L]$, $b \in \{0,1\}$:

$$F_{\mathbf{s}} : \{0,1\}^L \to \mathbb{Z}_p^m, \quad \text{given by} \quad F_{\mathbf{s}}(x) = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor_p = \left\lfloor \frac{p}{q} \cdot \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor$$

where $\boxed{\mathbf{s}} \leftarrow U(\mathbb{Z}_q^n)$ is the secret key

- Here,

$$\boxed{\mathbf{A}(x)} := \prod_{i=1}^{L} \mathbf{G}^{-1}\left(\boxed{\mathbf{A}_{i,x[i]}} \right) \cdot \boxed{\mathbf{A}_0}$$

- $\mathbf{G} = \mathbf{I_n} \otimes \left[\mathbf{1}, \mathbf{2}, \ldots, \mathbf{2}^{\lceil \log \mathbf{q} \rceil}\right]^\top$ the gadget matrix; $\mathbf{G}^{-1}$ binary decomposition

- Our proof idea: exploit the connection with fully homomorphic encryption `(Gentry-Sahai-Waters; Crypto'13)`

Modified Almost key-homomorphic PRF

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

We change it slightly:

Modified Almost key-homomorphic PRF

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

We change it slightly:

1. **Admissible hash function** (Boneh-Boyen; Crypto'04)

$$\mathsf{AHF} : \{0,1\}^{\ell} \to \{0,1\}^{L}$$

   encodes the input as $x := \mathsf{AHF}(X)$

Modified Almost key-homomorphic PRF

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

We change it slightly:

1. **Admissible hash function** (Boneh-Boyen; Crypto'04)

$$\mathsf{AHF} : \{0,1\}^{\ell} \to \{0,1\}^{L}$$

   encodes the input as $x := \mathsf{AHF}(X)$

2. Use a **deterministic randomness extractor** π (Trevisan-Vadhan, FOCS'00; Dodis, PhD thesis, 2000): output is $\pi(\boxed{\mathbf{z}})$, where

$$\boxed{\mathbf{z}} = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor_p$$

Modified Almost key-homomorphic PRF

(Boneh-Lewi-Montgomery-Raghunathan; Crypto'13)

We change it slightly:

1. **Admissible hash function** (Boneh-Boyen; Crypto'04)

   $$\mathsf{AHF} : \{0,1\}^{\ell} \to \{0,1\}^{L}$$

   encodes the input as $x := \mathsf{AHF}(X)$

2. Use a **deterministic randomness extractor** π (Trevisan-Vadhan, FOCS'00; Dodis, PhD thesis, 2000): output is $\pi(\boxed{\mathbf{z}})$, where

   $$\boxed{\mathbf{z}} = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}} \right\rfloor_p$$

3. DPRF obtained by sharing $\boxed{\mathbf{s}}$ using **Linear Integer Secret Sharing** (LISS) (Damgård-Thorbek; PKC'06): allows "small" shares

Building block: Linear integer secret sharing

LISS (Damgård-Thorbek; PKC'06)

To share an integer $s \in [-2^l, 2^l]$ among parties $[N]$, use a matrix $\mathbf{M} \in \mathbb{Z}^{d \times e}$,

- Choose random $\rho_2, \ldots \rho_e$ and define $\vec{\rho} = (s, \rho_2, \ldots, \rho_e)^\top$
- Compute $\vec{s} = (s_1, \ldots, s_d)^\top = \mathbf{M} \cdot \vec{\rho}$
- Give s_i to party $\psi(i)$ for some function $\psi : [d] \to [N]$

Integer Span Programs (Cramer-Fehr; Crypto'02)

Let $\mathbf{M} \in \mathbb{Z}^{d \times e}$ and a surjective function $\psi : [d] \to [N]$. Then, $(\mathbf{M}, \psi)$ is an ISP for a monotone access structure Γ iff

- $\forall A \in \Gamma, \;\; \exists \vec{\lambda}_A \in \mathbb{Z}^{d_A}$ s.t. $\vec{\lambda}_A^\top \cdot \mathbf{M}_A = (1, 0, \ldots, 0)^\top$
- $\forall A \notin \Gamma, \;\; \exists \vec{\kappa}_A \in \mathbb{Z}^{e}$ s.t. $\mathbf{M}_A \cdot \vec{\kappa}_A = 0$ and $\vec{\kappa}_A^\top \cdot (1, 0, \ldots, 0)^\top = 1$

LISS/ISP: Required properties

Need an ISP $(\mathbf{M} \in \mathbb{Z}^{d \times e},\ \psi : [d] \to [N])$ for threshold functions

- For correctness: $\forall A \in \Gamma,\ \exists$ small $\vec{\lambda}_A \in \mathbb{Z}^{d_A}$ s.t. $\vec{\lambda}_A^\top \cdot \mathbf{M}_A = \mathbf{1}^\top$
- Security proof requires: $\forall A \notin \Gamma,\ \exists$ small $\vec{\kappa}_A \in \mathbb{Z}^e$ s.t. $\mathbf{M}_A \cdot \vec{\kappa}_A = 0$
- ...and small-magnitude shares $s_i = \mathbf{M}_i \cdot \vec{\rho}$

Useful facts:

- Damgård and Thorbek build a LISS/ISP: construction based on Benaloh-Leichter (Crypto'88) yields $\|\vec{\lambda}_A\|_\infty = 1$ and $\|\vec{\kappa}_A\|_\infty = 1$

LISS/ISP: Required properties

Need an ISP $(\mathbf{M} \in \mathbb{Z}^{d \times e},\ \psi : [d] \to [N])$ for threshold functions

- For correctness: $\forall A \in \Gamma,\ \exists$ small $\vec{\lambda}_A \in \mathbb{Z}^{d_A}$ s.t. $\vec{\lambda}_A^\top \cdot \mathbf{M}_A = \mathbf{1}^\top$
- Security proof requires: $\forall A \notin \Gamma,\ \exists$ small $\vec{\kappa}_A \in \mathbb{Z}^e$ s.t. $\mathbf{M}_A \cdot \vec{\kappa}_A = 0$
- ...and small-magnitude shares $s_i = \mathbf{M}_i \cdot \vec{\rho}$

Useful facts:

- Damgård and Thorbek build a LISS/ISP: construction based on Benaloh-Leichter (Crypto'88) yields $\|\vec{\lambda}_A\|_\infty = 1$ and $\|\vec{\kappa}_A\|_\infty = 1$
- Any threshold function $T_{t,N}$ is computable by a Boolean formula of depth $O(\log N)$ and size $d = O(N^{5.3})$ (Valiant; J. of Algorithms, 1984)

LISS/ISP: Required properties

Need an ISP ($\mathbf{M} \in \mathbb{Z}^{d \times e}$, $\psi : [d] \rightarrow [N]$) for threshold functions

- For correctness: $\forall A \in \Gamma$, $\exists$ small $\vec{\lambda}_A \in \mathbb{Z}^{d_A}$ s.t. $\vec{\lambda}_A^\top \cdot \mathbf{M}_A = \mathbf{1}^\top$
- Security proof requires: $\forall A \notin \Gamma$, $\exists$ small $\vec{\kappa}_A \in \mathbb{Z}^e$ s.t. $\mathbf{M}_A \cdot \vec{\kappa}_A = 0$
- ...and small-magnitude shares $s_i = \mathbf{M}_i \cdot \vec{\rho}$

Useful facts:

- Damgård and Thorbek build a LISS/ISP: construction based on Benaloh-Leichter `(Crypto'88)` yields $\|\vec{\lambda}_A\|_\infty = 1$ and $\|\vec{\kappa}_A\|_\infty = 1$
- Any threshold function $T_{t,N}$ is computable by a Boolean formula of depth $O(\log N)$ and size $d = O(N^{5.3})$ `(Valiant; J. of Algorithms, 1984)`
- For a Boolean formula f, [DT06,BL88] provides $\mathbf{M} \in \{0,1\}^{d \times e}$ with $d, e = O(\mathrm{size}(f))$ and $\mathrm{depth}(f)$ non-zero entries

LISS: In our setting

Our construction:

- Dealer shares Gaussian vectors $\boxed{\mathbf{s}} \in \mathbb{Z}^n$ with Gaussian randomness $\{\vec{\rho}_i\}_{i=1}^n$, where $n = \mathrm{poly}(\lambda)$, and standard deviation $\sigma = O(n^{1/2} \cdot N^{1.7})$
- Each share $\boxed{\mathbf{s}_i} \in \mathbb{Z}^n$ has entries smaller than $O(n \cdot N^{1.7} \cdot \log N)$
- Using moduli q and p with $q/p > 2^{\mathrm{poly}(\lambda)}$, security proof exploits that

$$\left\lfloor \left(\boxed{\mathbf{B}(x)} + \text{noise} \right) \cdot \boxed{\mathbf{s}_i} \right\rfloor_p = \left\lfloor \boxed{\mathbf{B}(x)} \cdot \boxed{\mathbf{s}_i} \right\rfloor_p$$

w.h.p. for each share $\boxed{\mathbf{s}_i} \in \mathbb{Z}^n$ of the secret key $\boxed{\mathbf{s}} \in \mathbb{Z}^n$

Adaptively secure DPRF construction

- Setup : pick $\boxed{\mathbf{A}_0} \in \mathbb{Z}_q^{m\times n}$, $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m\times n} \right\}_{i=1}^{L}$, a k-wise independent function π

  Secret key: $\boxed{\mathbf{s}} \leftarrow D_{\mathbb{Z}^n,\sigma}$

Adaptively secure DPRF construction

- Setup : pick $\boxed{\mathbf{A}_0} \in \mathbb{Z}_q^{m\times n}$, $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m\times n} \right\}_{i=1}^{L}$, a k-wise independent function π

  Secret key: $\boxed{\mathbf{s}} \leftarrow D_{\mathbb{Z}^n,\sigma}$

- Share$\left(\boxed{\mathbf{s}}, t, N\right)$: Compute

$$\left(\boxed{\mathbf{s}_1}, \boxed{\mathbf{s}_2}, \ldots, \boxed{\mathbf{s}_N}\right) \leftarrow \mathsf{LISS}\left(\boxed{\mathbf{s}}, t, N\right)$$

Adaptively secure DPRF construction

- Setup : pick $\boxed{\mathbf{A}_0} \in \mathbb{Z}_q^{m \times n}$, $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m \times n} \right\}_{i=1}^{L}$, a k-wise independent function π

  Secret key: $\boxed{\mathbf{s}} \leftarrow D_{\mathbb{Z}^n,\sigma}$

- Share$\left(\boxed{\mathbf{s}}, t, N\right)$: Compute

$$\left(\boxed{\mathbf{s}_1}, \boxed{\mathbf{s}_2}, \ldots, \boxed{\mathbf{s}_N}\right) \leftarrow \mathsf{LISS}\left(\boxed{\mathbf{s}}, t, N\right)$$

- PEval$\left(\boxed{\mathbf{s}_i}, X\right)$: Let $x = \mathsf{AHF}(X) \in \{0,1\}^L$, output

$$\boxed{\mathbf{y}_i} = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}_i} \right\rfloor_p$$

Adaptively secure DPRF construction

- Setup : pick $\boxed{\mathbf{A}_0} \in \mathbb{Z}_q^{m \times n}$, $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m \times n} \right\}_{i=1}^{L}$, a k-wise independent function π

  Secret key: $\boxed{\mathbf{s}} \leftarrow D_{\mathbb{Z}^n,\sigma}$

- Share$\left(\boxed{\mathbf{s}}, t, N\right)$: Compute

$$\left(\boxed{\mathbf{s}_1}, \boxed{\mathbf{s}_2}, \ldots, \boxed{\mathbf{s}_N}\right) \leftarrow \mathsf{LISS}\left(\boxed{\mathbf{s}}, t, N\right)$$

- PEval$\left(\boxed{\mathbf{s}_i}, X\right)$: Let $x = \mathsf{AHF}(X) \in \{0,1\}^L$, output

$$\boxed{\mathbf{y}_i} = \left\lfloor \boxed{\mathbf{A}(x)} \cdot \boxed{\mathbf{s}_i} \right\rfloor_p$$

- Combine$(\{y_i\}_{i \in W})$: Compute $\lambda_{W,i}$ s.t. $\boxed{\mathbf{s}} = \sum_{i \in W} \lambda_{W,i} \cdot \boxed{\mathbf{s}_i}$ and output

$$\pi\left(\left\lfloor \sum_{i \in W} \lambda_{W,i} \cdot \boxed{\mathbf{y}_i} \mod p \right\rfloor_u \right)$$

Proof component 1: Lossy mode of LWE

(Goldwasser et al., ICS 2010)

- $f_{\mathsf{LWE}} : [-\sigma_x, \sigma_x]^m \times [-\sigma_e, \sigma_e]^m \to \mathbb{Z}_q^m$

$$f_{\mathsf{LWE}}(\mathbf{x}, \mathbf{e}) := \mathbf{A} \cdot \mathbf{x} + \mathbf{e}$$

is **injective** w.h.p. for a random $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$

Proof component 1: Lossy mode of LWE

(Goldwasser et al., ICS 2010)

- $f_{\mathrm{LWE}} : [-\sigma_x, \sigma_x]^m \times [-\sigma_e, \sigma_e]^m \to \mathbb{Z}_q^m$

$$f_{\mathrm{LWE}}(\mathbf{x}, \mathbf{e}) \; := \; \mathbf{A} \cdot \mathbf{x} + \mathbf{e}$$

is **injective** w.h.p. for a random $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$

- $f_{\mathrm{LWE}}(\mathbf{x}, \mathbf{e})$ is a **lossy** function if $\mathbf{A}$ is of the form

$$\underbrace{\mathbf{A}}_{m \times n} = \underbrace{\mathbf{B}}_{m \times n'} \cdot \underbrace{\mathbf{C}}_{n' \times n} + \mathbf{F}, \qquad n' < n$$

for uniform $\mathbf{B} \xleftarrow{R} \mathbb{Z}_q^{m \times n'}$, $\mathbf{C} \xleftarrow{R} \mathbb{Z}_q^{n' \times n}$ and small $\mathbf{F} \leftarrow \mathbb{Z}^{m \times n}$

Proof component 2: The GSW encodings

(Gentry-Sahai-Waters, Crypto'13)

- Public key is $PK := \boxed{\mathbf{A}} \in \mathbb{Z}_q^{m \times n}$
- $\mu \in \{0,1\}$ is encoded by picking $\boxed{\mathbf{R}} \xleftarrow{R} \{0,1\}^{m \times m}$ and computing

$$\underbrace{\mathbf{C}}_{m \times n} = \mathbf{R}\,\mathbf{A} + \mu\,\mathbf{G}$$

Proof component 2: The GSW encodings

`(Gentry-Sahai-Waters, Crypto'13)`

- Public key is $PK := \mathbf{A} \in \mathbb{Z}_q^{m \times n}$
- $\mu \in \{0,1\}$ is encoded by picking $\mathbf{R} \stackrel{R}{\leftarrow} \{0,1\}^{m \times m}$ and computing

$$\underbrace{\mathbf{C}}_{m \times n} = \mathbf{R} \, \mathbf{A} + \mu \, \mathbf{G}$$

- Given encodings $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{m \times n}$ of $\mu_1, \mu_2 \in \{0,1\}$,

$$\mathbf{G}^{-1}(\mathbf{C}_1) \cdot \mathbf{C}_2 \text{ encodes } \mu_1 \cdot \mu_2 \in \{0,1\}$$

Proof idea:

- Public parameters contain $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m \times n} \right\}_{i=1}^{L}$

Proof idea:

- Public parameters contain $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m \times n} \right\}_{i=1}^{L}$

- Each $\boxed{\mathbf{A}_{i,b}}$ is "programmed" as

$$\underbrace{\overbrace{\boxed{\mathbf{A}_{i,b}}}^{n}}_{m \times n} = \boxed{\mathbf{R}_{i,b}} \; \boxed{\mathbf{A}} + \mu_{i,b} \boxed{\mathbf{G}}$$

Proof idea:

- Public parameters contain $\left\{ \mathbf{A}_{i,b} \in \mathbb{Z}_q^{m \times n} \right\}_{i=1}^{L}$

- Each $\mathbf{A}_{i,b}$ is "programmed" as

$$\underbrace{\mathbf{A}_{i,b}}_{m \times n} = \mathbf{R}_{i,b}\, \mathbf{A} + \mu_{i,b}\, \mathbf{G}$$

- $\mathbf{A}(x)$ now equals

$$\mathbf{A}(x) = \mathbf{R}_x \cdot \mathbf{A} + \left(\prod_{i=1}^{L} \mu_{i,x[i]} \right) \cdot \mathbf{G}$$

Proof idea:

- Public parameters contain $\left\{ \boxed{\mathbf{A}_{i,b}} \in \mathbb{Z}_q^{m\times n} \right\}_{i=1}^{L}$

- Each $\boxed{\mathbf{A}_{i,b}}$ is "programmed" as

$$\underbrace{\boxed{\mathbf{A}_{i,b}}}_{m \times n} = \boxed{\mathbf{R}_{i,b}}\ \boxed{\mathbf{A}} + \mu_{i,b}\ \boxed{\mathbf{G}}$$

- $\boxed{\mathbf{A}(x)}$ now equals

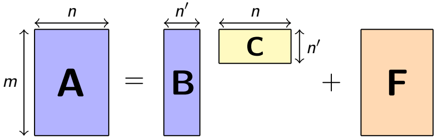
$$\boxed{\mathbf{A}(x)} = \boxed{\mathbf{R}_x} \cdot \boxed{\mathbf{A}} + \left(\prod_{i=1}^{L} \mu_{i,x[i]}\right) \cdot \boxed{\mathbf{G}}$$

- With noticeable probability, encoding $x := \mathsf{AHF}(X)$ allows:

$$\boxed{\mathbf{A}(x)} = \boxed{\mathbf{R}_x} \cdot \boxed{\mathbf{A}} \qquad\qquad \boxed{\mathbf{A}(x^\star)} = \boxed{\mathbf{R}_{x^\star}} \cdot \boxed{\mathbf{A}} + \boxed{\mathbf{G}}$$

Proof idea:

- Now, switch **A** to the lossy mode:



Proof idea:

- Now, switch $\mathbf{A}$ to the lossy mode:

$$\underbrace{\overset{n}{\mathbf{A}}}_{m} = \overset{n'}{\mathbf{B}} \cdot \underbrace{\overset{n}{\mathbf{C}}}_{n'} + \mathbf{F}$$

- Partial evaluations (LISS allows **small** shares $\mathbf{s}_i$)

$$\lfloor \mathbf{A}(x) \cdot \mathbf{s_i} \rfloor_p = \lfloor \mathbf{R}_x \cdot \left(\mathbf{B} \cdot \mathbf{C} + \mathbf{F} \right) \cdot \mathbf{s_i} \rfloor_p = \lfloor \mathbf{R}_x \cdot \mathbf{B} \cdot \left(\mathbf{C} \cdot \mathbf{s_i} \right) \rfloor_p$$

reveals no more than $\mathbf{C} \, \mathbf{s_i} \in \mathbb{Z}_q^{n'}$

Proof idea:

- Now, switch $\mathbf{A}$ to the lossy mode:

$$\underbrace{\mathbf{A}}_{m \times n} = \underbrace{\mathbf{B}}_{m \times n'} \cdot \underbrace{\mathbf{C}}_{n' \times n} + \mathbf{F}$$

- Partial evaluations (LISS allows **small** shares $\mathbf{s}_i$)

$$\left\lfloor \mathbf{A}(x) \cdot \mathbf{s_i} \right\rfloor_p = \left\lfloor \mathbf{R}_x \cdot \left(\mathbf{B} \cdot \mathbf{C} + \mathbf{F} \right) \cdot \mathbf{s_i} \right\rfloor_p = \left\lfloor \mathbf{R}_x \cdot \mathbf{B} \cdot \left(\mathbf{C} \cdot \mathbf{s_i} \right) \right\rfloor_p$$

reveals no more than $\mathbf{C} \cdot \mathbf{s_i} \in \mathbb{Z}_q^{n'}$

- All queries reveal the same information about the secret $\mathbf{s}_i$
- (Deterministic) randomness extraction argument applies:

$$\mathrm{H}_\infty(\mathbf{s} \mid \text{PEvals, Corrupted Keys}) \geq \text{sufficient entropy}$$

Proof idea: Summary

Why we can prove adaptive security:

- Non-modular use of key-homomorphic PRFs
- Reduction knows all secret key shares at any time

Proof idea:

- Switch matrix $\mathbf{A}$ to lossy mode
- Make sure that all evaluations reveal the same information about shares
- Lower bound on the entropy of the secret when information is leaked:

$$\mathrm{H}_\infty(\mathbf{s} \mid \text{PEvals, Corrupted Keys}) \geq \text{sufficient entropy}$$

- Extract randomness using a deterministic extractor π

Robustness against malicious adversaries

Robustness

Malicious servers cannot send wrong partial evaluations $y_i = F_{SK_i}(x)$ and prevent the reconstruction of $y = F_{SK}(x)$

- Need to prove that each partial evaluation $y_i = \mathsf{PEval}(SK_i, x)$ is correct
- ... possibly using some public commitment to SK_i
- **Possible using generic NIZK:**
  - NIZK with pre-processing: under the LWE assumption (Kim-Wu, Crypto'18)
  - ... or general NIZK from LWE (Peikert-Shiehian, Crypto'19)

Robustness from homomorphic signatures

Leveled homomorphic signatures (Gorbunov *et al.*; STOC'15)

Given (m, sig_m), anyone can compute $(C(m), sig_{C(m)} = \mathsf{SigEval}((m, sig), C))$

Direct construction: following Boneh *et al.* (Crypto'18)

- Dealer signs each secret key share SK_i using a homomorphic signature
- Given its signed share (SK_i, sig_i), each server i
  1. Computes $y_i = \mathsf{PEval}(SK_i, x)$
  2. Using the circuit $C_x(.)$ such that $C_x(SK_i) = F_{SK_i}(x)$, computes

$$sig_{y_i} \leftarrow \mathsf{SigEval}((SK_i, sig_i), C_x)$$

Robustness from homomorphic signatures

Leveled homomorphic signatures (Gorbunov *et al.*; STOC'15)

Given (m, sig_m), anyone can compute $(C(m), sig_{C(m)} = \mathsf{SigEval}((m, sig), C))$

Direct construction: following Boneh *et al.* (Crypto'18)

- Dealer signs each secret key share SK_i using a homomorphic signature
- Given its signed share (SK_i, sig_i), each server i
  1. Computes $y_i = \mathsf{PEval}(SK_i, x)$
  2. Using the circuit $C_x(.)$ such that $C_x(SK_i) = F_{SK_i}(x)$, computes

$$sig_{y_i} \leftarrow \mathsf{SigEval}((SK_i, sig_i), C_x)$$

- Combiner checks that (y_i, sig_{y_i}) is a valid pair w.r.t. $C_x(.)$
- (Simulation-based) **context-hiding** property of homomorphic signatures preserves the secrecy of SK_i

Summary

First DPRF construction in the standard model which is

- Non-interactive
- Secure under adaptive corruptions for $N \in \text{poly}(\lambda)$ servers

Requires large LWE parameters

- Exponentially large moduli p, q in λ (but not in N)
- Yamada's technique `(Crypto'17)` allows shorter public parameters ($O(\log^2 \lambda)$ matrices instead of $O(\lambda)$)

Open questions:

- More efficient parameters
- DPRFs from more standard LWE assumptions
- Other applications of LISS combined with discrete Gaussians?

Thanks!

Questions?