Constrained Pseudorandom Functions for Inner-Product Predicates from Weaker Assumptions

Sacha Servan-Schreiber

MIT CSAIL

This talk: New ways of building constrained PRFs

Overview

- Background on PRFs and constrained PRFs
- A secret sharing perspective on constrained PRFs
- Construction in the random oracle model
- Our framework and instantiations
- Implementation
- Open problems

Constrained PRFs

Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a PRF if:

Standard PRF security

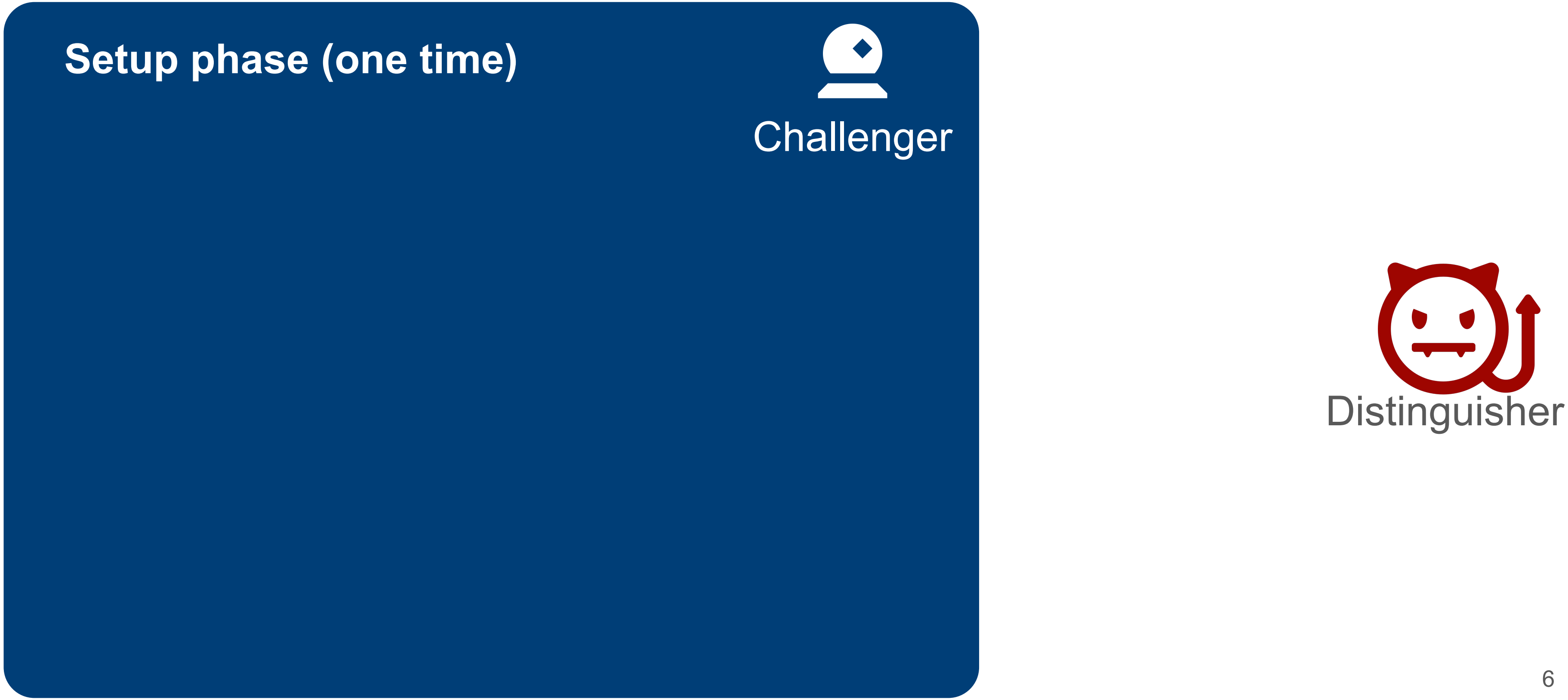
A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a PRF if:

Setup phase (one time)

Challenger

Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a PRF if:



Setup phase (one time)

Challenger

Distinguisher

Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a PRF if:

Setup phase (one time)

Challenger

1 $k \xleftarrow{R} \mathcal{K}$

Distinguisher

Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a PRF if:

Setup phase (one time)

Challenger

1. $k \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$

Distinguisher

Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a PRF if:

Setup phase (one time)

Challenger

① $k \xleftarrow{R} \mathcal{K}$

② $R \xleftarrow{R} \mathcal{F}uns\,(\mathcal{X}, \mathcal{Y})$

③ $b \xleftarrow{R} \{0, 1\}$

Distinguisher

Standard PRF security

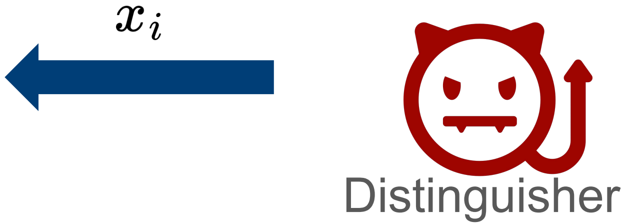
A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a PRF if:

Setup phase (one time)

Challenger

1. $k \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\,(\mathcal{X}, \mathcal{Y})$
3. $b \overset{R}{\leftarrow} \{0, 1\}$

Query phase (repeatable)



Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a PRF if:

Setup phase (one time)

Challenger

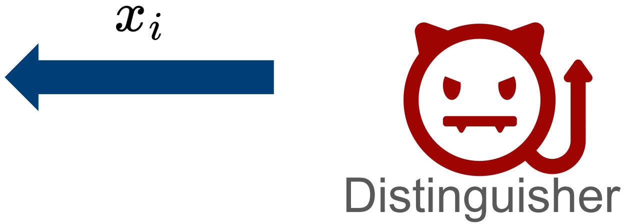
① $k \overset{R}{\leftarrow} \mathcal{K}$

② $R \overset{R}{\leftarrow} \mathcal{F}uns\,(\mathcal{X}, \mathcal{Y})$

③ $b \overset{R}{\leftarrow} \{0, 1\}$

Query phase (repeatable)

④ $$y_i := \begin{cases} F\,(k,\ x_i) & \text{if } b = 0 \\ R\,(x_i) & \text{if } b = 1 \end{cases}$$



Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a PRF if:

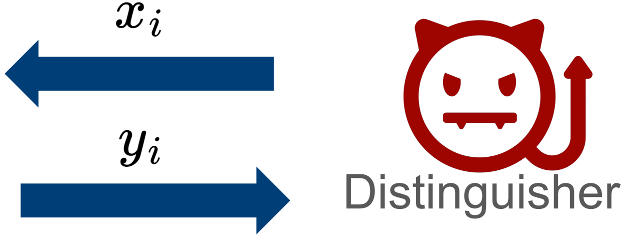
Setup phase (one time)

Challenger

1. $k \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $b \overset{R}{\leftarrow} \{0,1\}$

Query phase (repeatable)

4. $y_i := \begin{cases} F\left(k,\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$



x_i

y_i

Distinguisher

Standard PRF security

A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a PRF if:

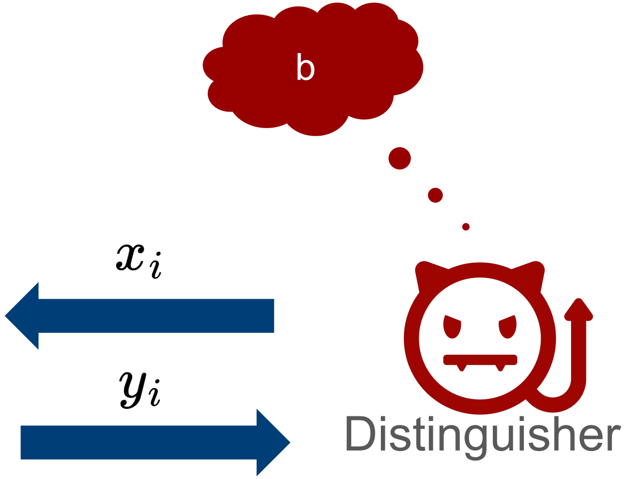
Setup phase (one time)

Challenger

1. $k \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $b \overset{R}{\leftarrow} \{0, 1\}$

Query phase (repeatable)

4. $$y_i := \begin{cases} F\left(k,\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$$



Constrained Pseudorandom Function (**C**PRF)

Constrained Pseudorandom Function (**C**PRF)

CPRFs have an additional **constrain** functionality:

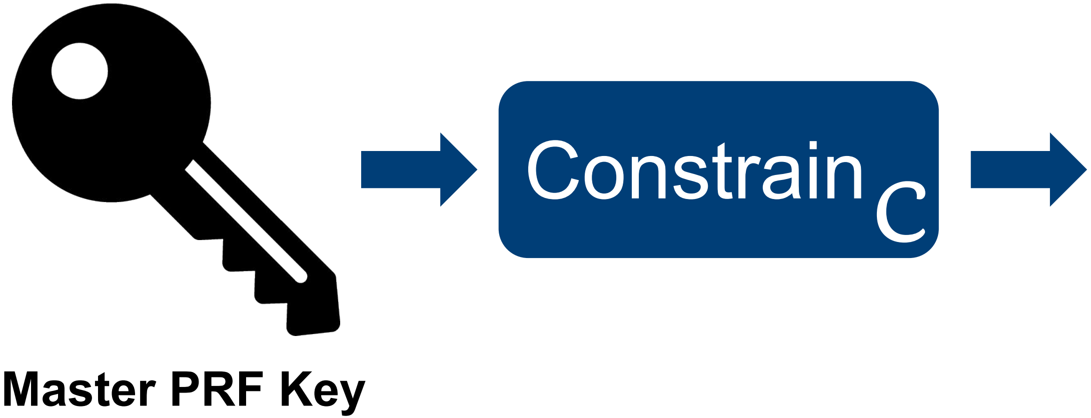
Constrained Pseudorandom Function (**C**PRF)

CPRFs have an additional **constrain** functionality:

Master PRF Key

Constrained Pseudorandom Function (**C**PRF)

CPRFs have an additional **constrain** functionality:

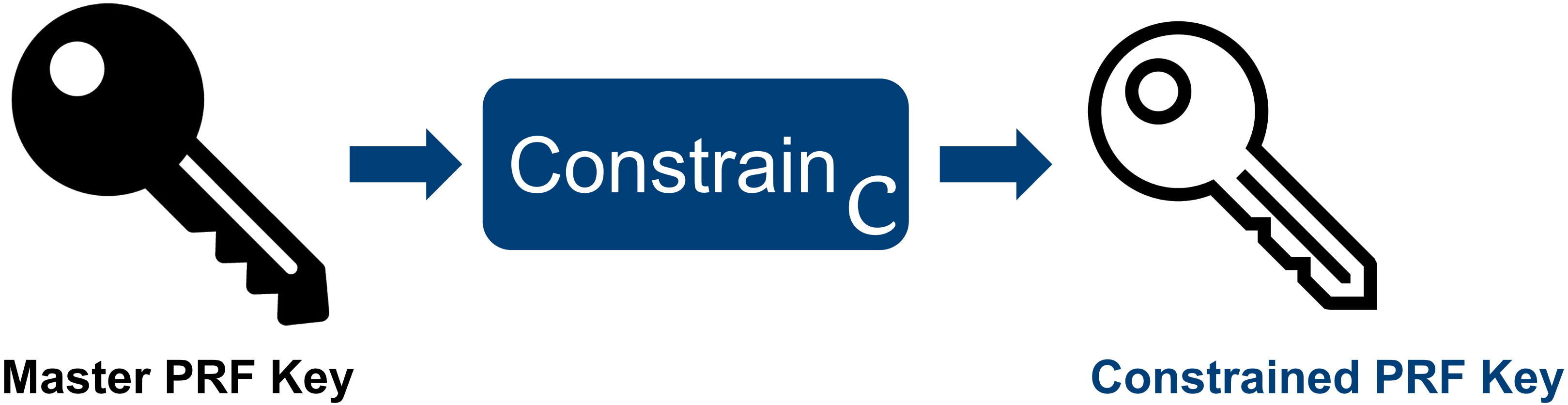


Master PRF Key

Constrain$_C$

Constrained Pseudorandom Function (**C**PRF)

CPRFs have an additional **constrain** functionality:



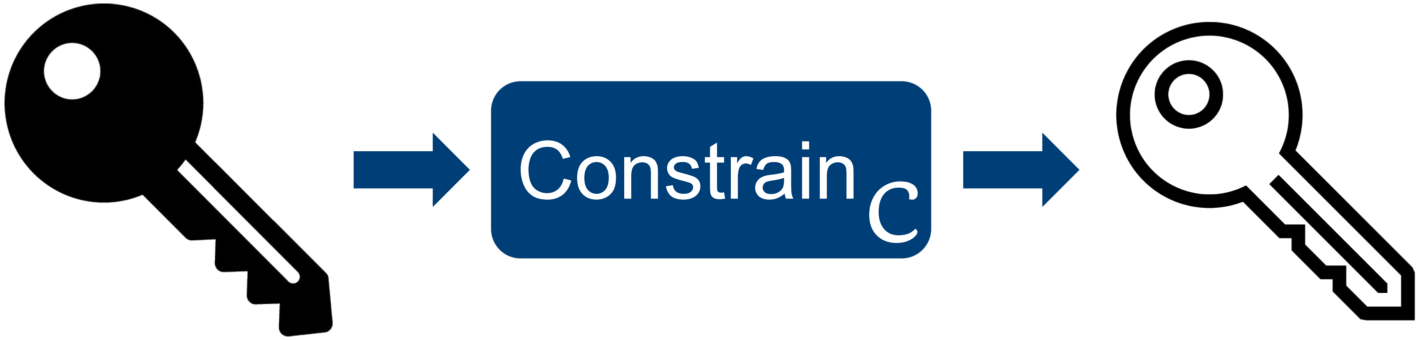
Master PRF Key

Constrain$_C$

Constrained PRF Key

Constrained Pseudorandom Function (**C**PRF)

CPRFs have an additional **constrain** functionality:



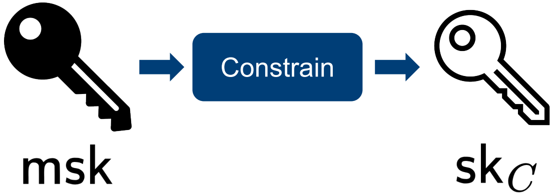
Constrain$_C$

Master PRF Key

Constrained PRF Key

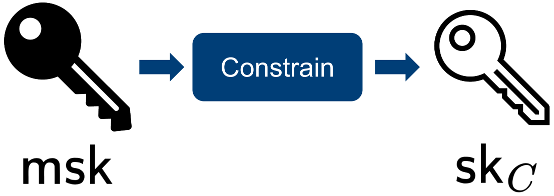
Constrained key 🔑 can be used to evaluate $F(\text{🔑}, x)$ for all $x \in \mathcal{X}$ where $C(x) = 0$

Constrained Pseudorandom Function (**C**PRF)



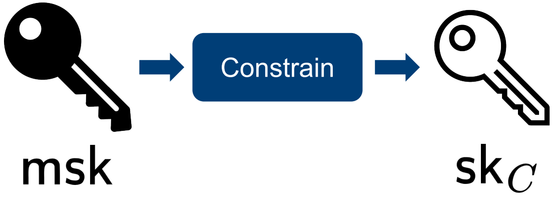
msk → Constrain → sk_C

Constrained Pseudorandom Function (CPRF)



$$C\left(x\right) = \begin{cases} 0 & \text{authorized} \\ 1 & \text{unauthorized} \end{cases}$$

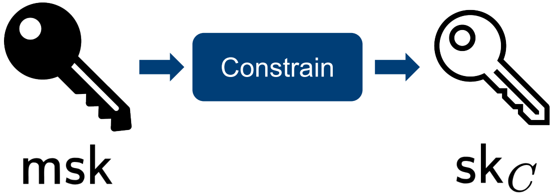
Constrained Pseudorandom Function (CPRF)



$$C(x) = \begin{cases} 0 & \text{authorized} \\ 1 & \text{unauthorized} \end{cases}$$

Correctness: If $C(x) = 0$ then $F(\mathsf{msk}, x) = F(\mathsf{sk}_C, x)$

Constrained Pseudorandom Function (**C**PRF)

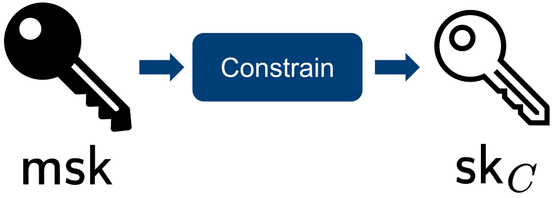


$$C(x) = \begin{cases} 0 & \text{authorized} \\ 1 & \text{unauthorized} \end{cases}$$

Correctness: If $C(x) = 0$ then $F(\mathsf{msk}, x) = F(\mathsf{sk}_C, x)$

Pseudorandomness: If $C(x) \neq 0$ then $F(\mathsf{msk}, x)$ is pseudorandom given sk_C

Constrained Pseudorandom Function (CPRF)



$$C(x) = \begin{cases} 0 & \text{authorized} \\ 1 & \text{unauthorized} \end{cases}$$

Correctness: If $C(x) = 0$ then $F(\mathsf{msk}, x) = F(\mathsf{sk}_C, x)$

Pseudorandomness: If $C(x) \neq 0$ then $F(\mathsf{msk}, x)$ is pseudorandom given sk_C

Hiding (optional): C is hidden given sk_C

Constrained Pseudorandom Function (**C**PRF)

Our focus: Inner-product predicates

Constrained Pseudorandom Function (**C**PRF)

Our focus: Inner-product predicates

$$C(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle \in \mathbb{F} \ \text{ where } \mathbf{z}, \mathbf{x} \in \mathbb{F}^{\ell}$$

Constrained Pseudorandom Function (CPRF)

Our focus: Inn

Predicate satisfied if and only if the inner product is zero

$$C(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle \in \mathbb{F} \text{ where } \mathbf{z}, \mathbf{x} \in \mathbb{F}^{\ell}$$

Constrained Pseudorandom Function (CPRF)

Our focus: Inn

Predicate satisfied if and only if the inner product is zero

$$C(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle \in \mathbb{F} \text{ where } \mathbf{z}, \mathbf{x} \in \mathbb{F}^{\ell}$$

Can be used to build other predicates, generically:

Constrained Pseudorandom Function (CPRF)

Our focus: Inn

Predicate satisfied if and only if the inner product is zero

$$C(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle \in \mathbb{F} \text{ where } \mathbf{z}, \mathbf{x} \in \mathbb{F}^{\ell}$$

Can be used to build other predicates, generically:

- **t-CNF** predicates (for constant t) [DKN+20]

Constrained Pseudorandom Function (**C**PRF)

Our focus: Inr

Predicate satisfied if and only if the inner product is zero

$$C(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle \in \mathbb{F} \;\; \text{where}\, \mathbf{z}, \mathbf{x} \in \mathbb{F}^{\ell}$$

Can be used to build other predicates, generically:

- **t-CNF** predicates (for constant t) [DKN+20]
- **Bit-fixing** predicates (special case of t-CNF) [DKN+20]

Constrained Pseudorandom Function (CPRF)

Our focus: Inn

Predicate satisfied if and only if the inner product is zero

$$C(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle \in \mathbb{F} \ \text{ where } \mathbf{z}, \mathbf{x} \in \mathbb{F}^{\ell}$$

Can be used to build other predicates, generically:

- **t-CNF** predicates (for constant t) [DKN+20]
- **Bit-fixing** predicates (special case of t-CNF) [DKN+20]
- **Matrix-product** predicates (folklore & this work)

Security Definitions

(1-key, selective) CPRF security game

Setup phase (one time)

Challenger

Distinguisher

(1-key, selective) CPRF security game

Setup phase (one time)

① $\text{msk} \xleftarrow{R} \mathcal{K}$

Challenger

Distinguisher

(1-key, selective) CPRF security game

Setup phase (one time)

Challenger

① $\mathsf{msk} \xleftarrow{R} \mathcal{K}$

② $R \xleftarrow{R} \mathcal{F}uns\,(\mathcal{X}, \mathcal{Y})$

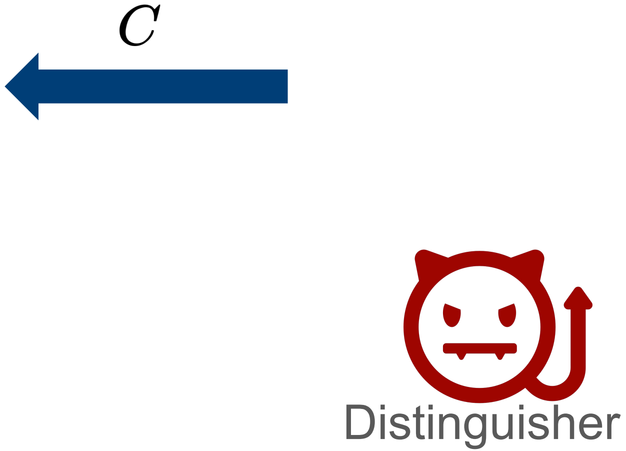
Distinguisher

(1-key, selective) CPRF security game

Setup phase (one time)

Challenger

1. $\text{msk} \xleftarrow{R} \mathcal{K}$
2. $R \xleftarrow{R} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$



(1-key, selective) CPRF security game

Setup phase (one time)

1. $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$

Challenger

C

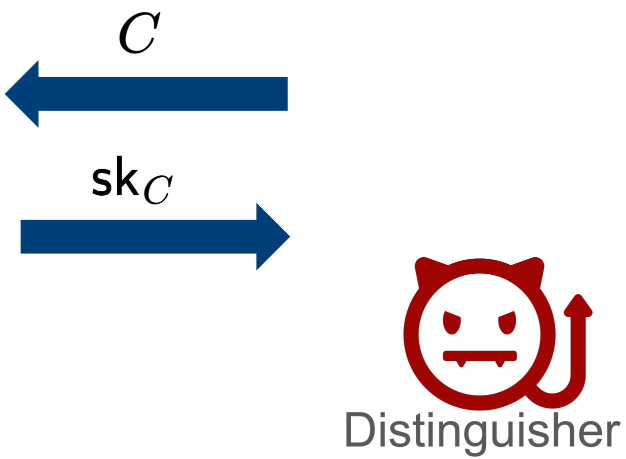
Distinguisher

(1-key, selective) CPRF security game

Setup phase (one time)

1. $\mathsf{msk} \xleftarrow{R} \mathcal{K}$
2. $R \xleftarrow{R} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$

Challenger

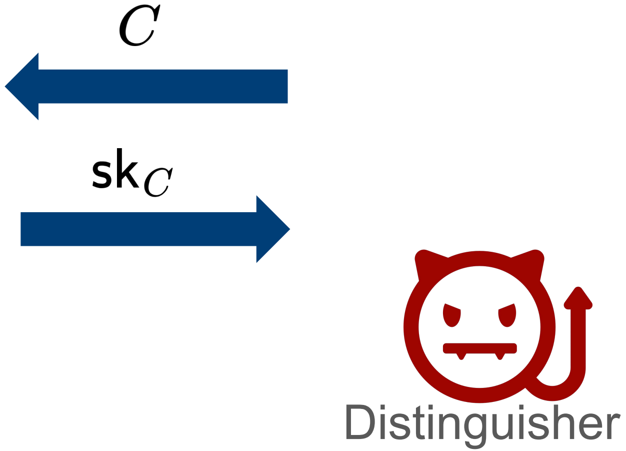


(1-key, selective) CPRF security game

Setup phase (one time)

Challenger

1. $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\,(\mathcal{X},\mathcal{Y})$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\,(\mathsf{msk}, C)$
4. $b \overset{R}{\leftarrow} \{0,1\}$



(1-key, selective) CPRF security game

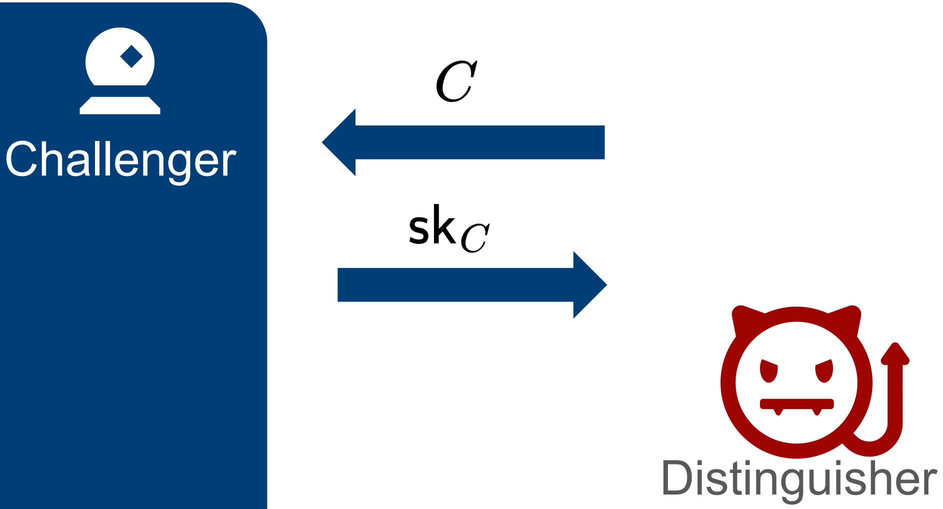
Setup phase (one time)

① $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$

② $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$

③ $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$

④ $b \overset{R}{\leftarrow} \{0,1\}$



Query phase (repeatable)

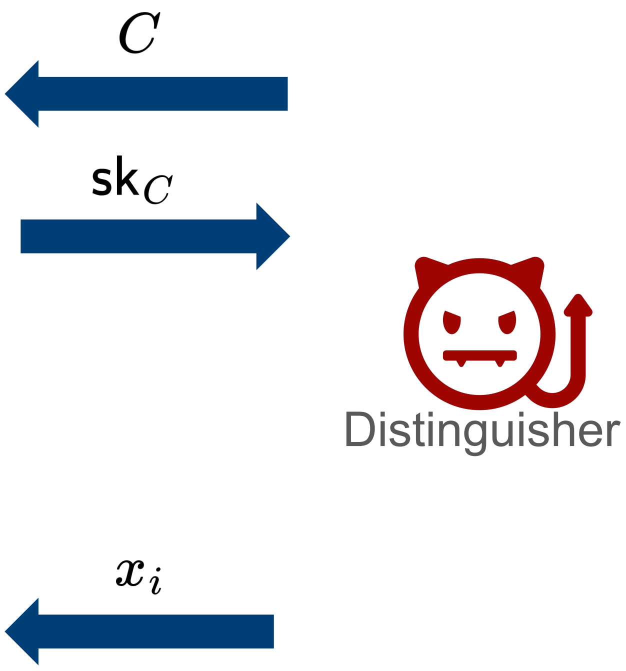
(1-key, selective) CPRF security game

Challenger

Setup phase (one time)

1. $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$
4. $b \overset{R}{\leftarrow} \{0,1\}$

Query phase (repeatable)



(1-key, selective) CPRF security game

Setup phase (one time)

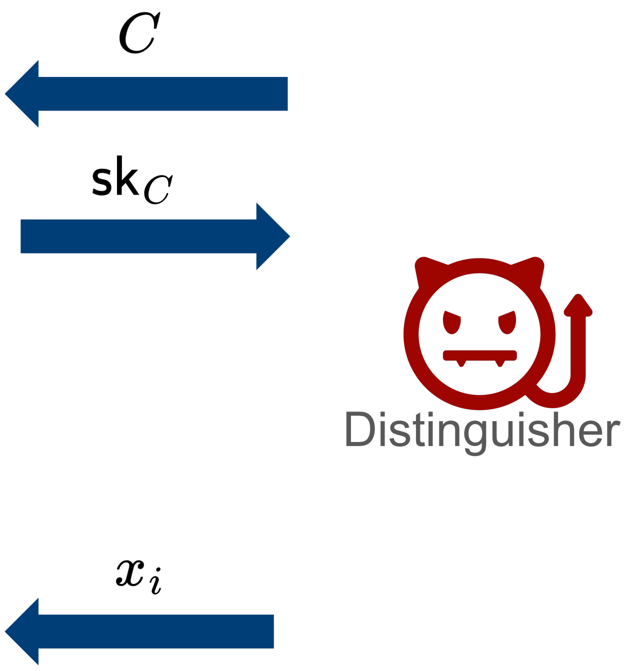
Challenger

1. $\mathsf{msk} \xleftarrow{R} \mathcal{K}$
2. $R \xleftarrow{R} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$
4. $b \xleftarrow{R} \{0,1\}$

Query phase (repeatable)

5. $y_i := \begin{cases} F\left(\mathsf{msk},\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$

Need: $C\left(x_i\right) \neq 0$ for all $x_i \in \mathcal{X}$.



(1-key, selective) CPRF security game

Setup phase (one time)

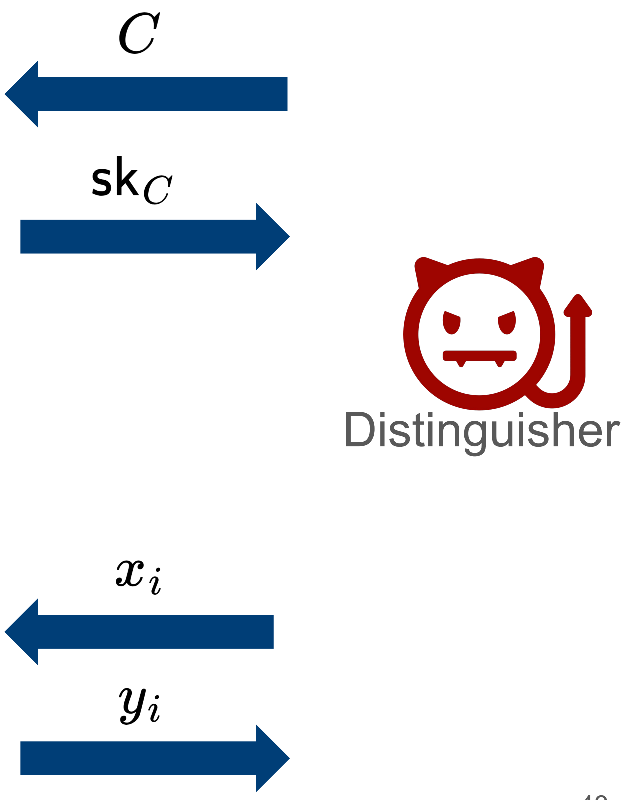
Challenger

1. $\mathsf{msk} \xleftarrow{R} \mathcal{K}$
2. $R \xleftarrow{R} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$
4. $b \xleftarrow{R} \{0, 1\}$

Query phase (repeatable)

5. $y_i := \begin{cases} F\left(\mathsf{msk},\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$

Need: $C\left(x_i\right) \neq 0$ for all $x_i \in \mathcal{X}$.



C

sk_C

Distinguisher

x_i

y_i

(1-key, selective) CPRF security game

Setup phase (one time)

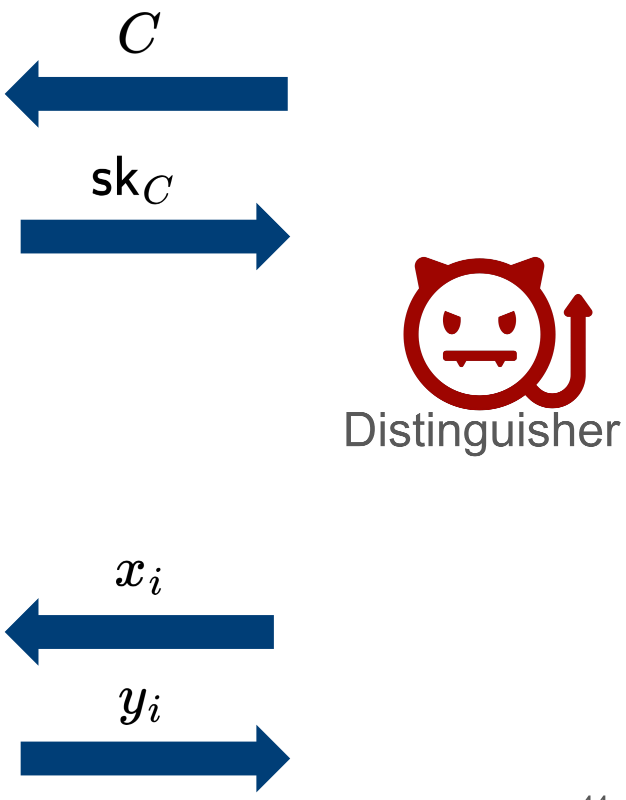
Challenger

1. $\mathsf{msk} \xleftarrow{R} \mathcal{K}$
2. $R \xleftarrow{R} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C\right)$
4. $b \xleftarrow{R} \{0, 1\}$

Query phase (repeatable)

5. $y_i := \begin{cases} F\left(\mathsf{msk},\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$

Need: $C\left(x_i\right) \neq 0$ for all $x_i \in \mathcal{X}$.



(1-key, adaptive) CPRF security game

Setup phase (one time)

Challenger

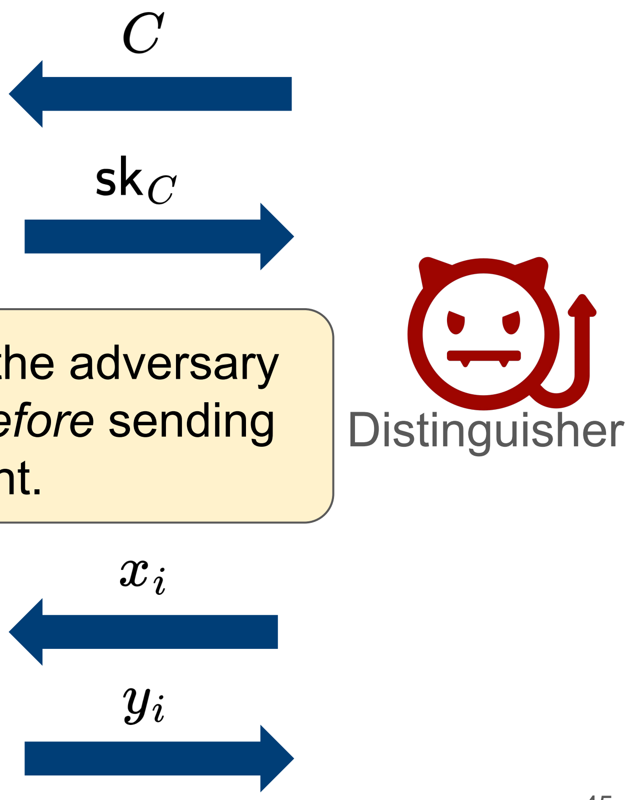
1. $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $\mathsf{sk}_C \leftarrow$ Constrai
4. $b \overset{R}{\leftarrow} \{0, 1\}$

Query phase (repeatable)

5. $$y_i := \begin{cases} F\left(\mathsf{msk},\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$$

Need: $C\left(x_i\right) \neq 0$ for all $x_i \in \mathcal{X}$.

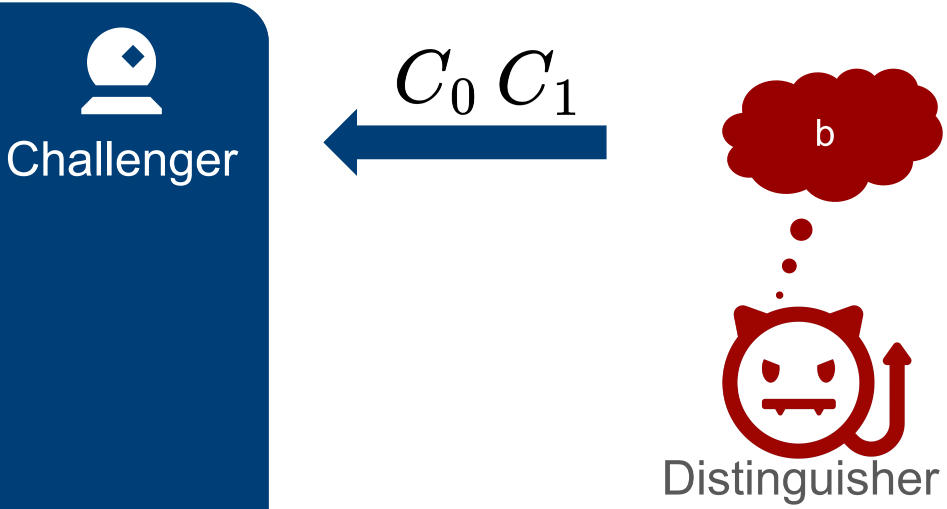
ⓘ **Adaptive** security lets the adversary query the challenger *before* sending the constraint.



Constraint-hiding security game

Setup phase (one time)

1. msk $\overset{R}{\leftarrow}\mathcal{K}$
2. $b \overset{R}{\leftarrow} \{0,1\}$

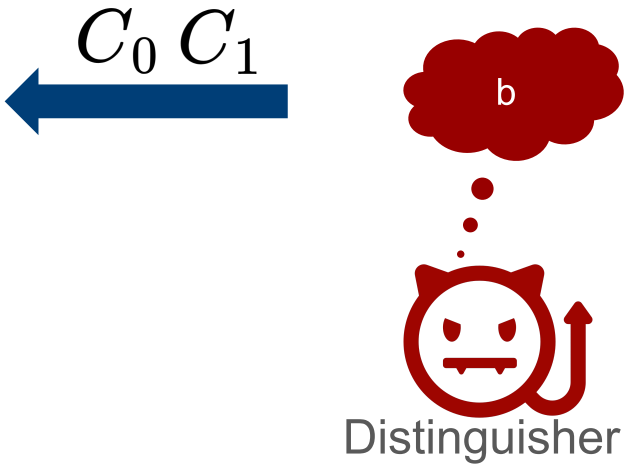


Constraint-hiding security game

Setup phase (one time)

Challenger

1. $\mathsf{msk} \xleftarrow{R} \mathcal{K}$
2. $b \xleftarrow{R} \{0,1\}$
3. $\mathsf{sk}_{C_b} \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C_b\right)$

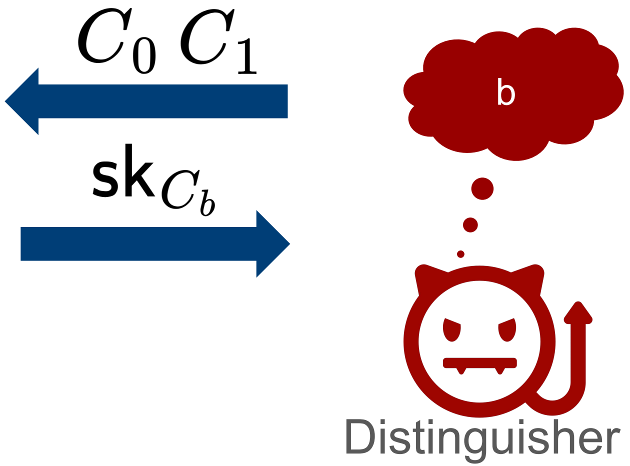


Constraint-hiding security game

Setup phase (one time)

Challenger

1. $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$
2. $b \overset{R}{\leftarrow} \{0,1\}$
3. $\mathsf{sk}_{C_b} \leftarrow \mathsf{Constrain}\left(\mathsf{msk}, C_b\right)$



$C_0 \; C_1$

sk_{C_b}

b

Distinguisher

Constraint-hiding security game

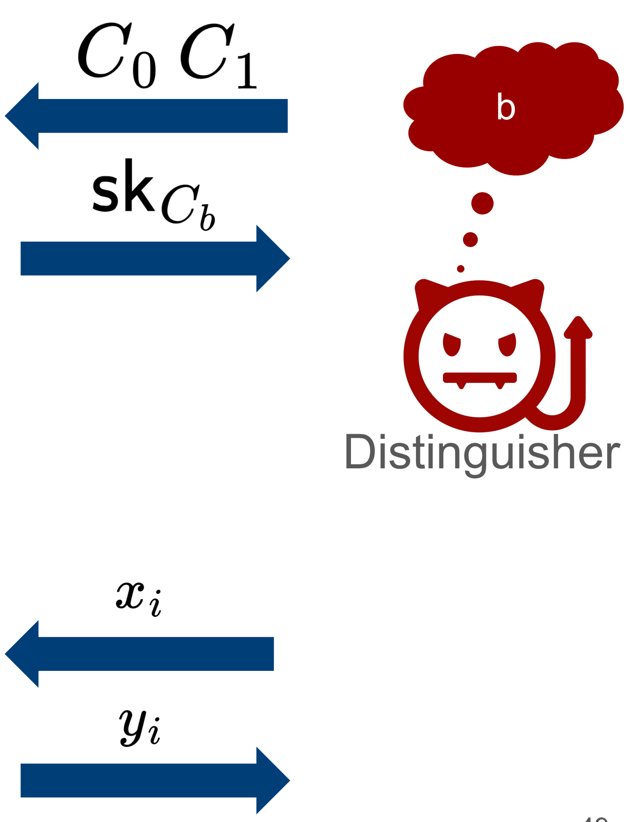
Setup phase (one time)

Challenger

1. $\mathsf{msk} \overset{R}{\leftarrow} \mathcal{K}$
2. $b \overset{R}{\leftarrow} \{0,1\}$
3. $\mathsf{sk}_{C_b} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C_b)$

Query phase (repeatable)

4. $y_i := F(\mathsf{msk}, x_i)$



Constraint-hiding security game

Setup phase (one time)

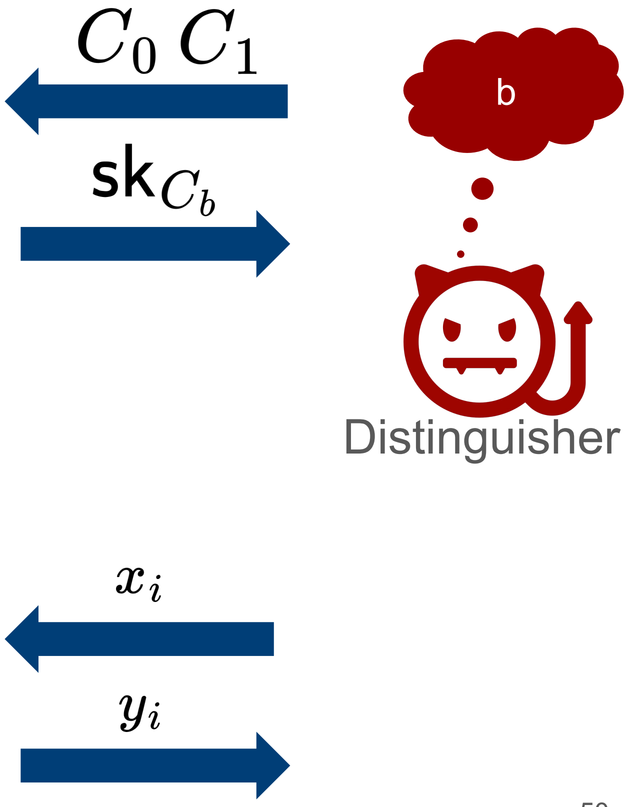
Challenger

1. $\text{msk} \xleftarrow{R} \mathcal{K}$
2. $b \xleftarrow{R} \{0, 1\}$
3. $\text{sk}_{C_b} \leftarrow \text{Constrain}\left(\text{msk}, C_b\right)$

Query phase (repeatable)

4. $y_i := F\left(\text{msk}, x_i\right)$

Must satisfy $C_0\left(x\right) = C_1\left(x\right)$
for all queries $x_i \in \mathcal{X}$



The current landscape

The current landscape

	Assumptions	Security	Hiding	Comments

The current landscape

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly

The current landscape

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1

The current landscape

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1

The current landscape

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	

Can we build CPRFs from weaker assumptions?

New results

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	

Can we build CPRFs for inner-product predicates using random oracles?

New results

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	

New results

	Assumptions	**Security**	**Hiding**	**Comments**
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC[1]
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC[1]
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	

Can we build CPRFs for inner-product predicates from DDH?

New results

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	
This work	**DDH**	**Selective**	✓	

New results

	Assumptions	Security	Hiding	Comments
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	
This work	**DDH**	**Selective**	✓	

Can we build CPRFs for inner-product predicates from LPN?

New results

	Assumptions	**Security**	**Hiding**	**Comments**
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	
This work	**DDH**	**Selective**	✓	
This work	**VDLPN**	**Selective**	✓	Weak CPRF (random inputs)

New results

	Assumptions	**Security**	**Hiding**	**Comments**
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	
This work	**DDH**	**Selective**	✓	
This work	**VDLPN**	**Selective**	✓	Weak CPRF (random inputs)

Can we build CPRFs for inner-product predicates from OWF?

New results

	Assumptions	**Security**	**Hiding**	**Comments**
Generic CPRFs	LWE or iO	Selective	✓	For NC and P/poly
[AMN+18]	L-DDHI + DDH	Selective	✗	For NC^1
[AMN+18]	L-DDHI + ROM	Adaptive	✗	For NC^1
[CMPR23]	DCR	Selective	✓	
This work	**ROM**	**Adaptive**	✓	
This work	**DDH**	**Selective**	✓	
This work	**VDLPN**	**Selective**	✓	Weak CPRF (random inputs)
This work	**OWF**	**Selective**	✓	Only for a polynomial domain

A secret sharing perspective on constrained PRFs

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

👩
Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

👨
Bob

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

👩 Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

👨 Bob

$$\mathsf{msk} = \mathbf{z}_0$$

$$\mathsf{sk}_{\mathbf{z}} = \mathbf{z}_1$$

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

👩 Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

👨 Bob

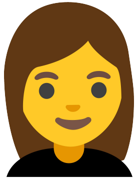
$$\mathsf{msk} = \mathbf{z}_0$$

One share (master share) can be sampled independently of the constraint vector **z**

$$\mathsf{sk}_{\mathbf{z}} = \mathbf{z}_1$$

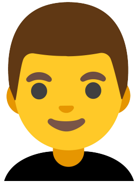
A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:



Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$



Bob

$$\mathsf{msk} = \mathbf{z}_0$$

For an input $\mathbf{x}$:

$$\mathsf{sk}_{\mathbf{z}} = \mathbf{z}_1$$

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

👩

Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

👨

Bob

$$\mathsf{msk} = \mathbf{z}_0$$

For an input $\mathbf{x}$:

$$\mathsf{sk}_{\mathbf{z}} = \mathbf{z}_1$$

$$k_A := \langle \mathbf{z}_0, \mathbf{x} \rangle$$

$$k_B := \langle \mathbf{z}_1, \mathbf{x} \rangle$$

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

👩 Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

👨 Bob

$$\mathsf{msk} = \mathbf{z}_0$$

$$k_A := \langle \mathbf{z}_0, \mathbf{x} \rangle$$

For an input $\mathbf{x}$:

$$k_A - k_B = \langle \mathbf{z}, \mathbf{x} \rangle$$

$$\mathsf{sk}_{\mathbf{z}} = \mathbf{z}_1$$

$$k_B := \langle \mathbf{z}_1, \mathbf{x} \rangle$$

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

ob

$$k_A = k_B \text{ when } \langle \mathbf{z}, \mathbf{x} \rangle = \mathbf{0}$$

$$\mathsf{msk} = \mathbf{z}_0$$

$$= \mathbf{z}_1$$

$$k_A := \langle \mathbf{z}_0, \mathbf{x} \rangle$$

$$k_A - k_B = \langle \mathbf{z}, \mathbf{x} \rangle$$

$$k_B := \langle \mathbf{z}_1, \mathbf{x} \rangle$$

A secret-sharing perspective

Idea: view msk and $\mathsf{sk}_{\mathbf{z}}$ as being secret shares of the constraint vector **z**:

Alice

$$\mathbf{z}_0 - \mathbf{z}_1 = \mathbf{z}$$

Bob

$$\mathsf{msk} = \mathbf{z}_0$$

$$k_A := \langle \mathbf{z}_0, \mathbf{x} \rangle$$

$$F\left(k_A, \mathbf{x}\right)$$

For an input $\mathbf{x}$:

$$k_A - k_B = \langle \mathbf{z}, \mathbf{x} \rangle$$

$$\mathsf{sk}_{\mathbf{z}} = \mathbf{z}_1$$

$$k_B := \langle \mathbf{z}_1, \mathbf{x} \rangle$$

$$F\left(k_B, \mathbf{x}\right)$$

Same PRF output

A first attempt at constructing a CPRF

$$\mathsf{msk} := \boldsymbol{z_0}$$

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

A first attempt at constructing a CPRF

$$\mathsf{msk} := \mathbf{z_0}$$

For a constraint vector $\mathbf{z}$:

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

Eval($\mathsf{msk}, \mathbf{x}$**):**

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

Eval($\mathsf{msk}, \mathbf{x}$ **):**

```
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
```

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

```
Eval($\mathsf{msk}, \mathbf{x}$):
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F\left(k, \mathbf{x}\right)$
```

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

Eval(msk **,** $\mathbf{x}$ **):**

1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F\left(k, \mathbf{x}\right)$

CEval($\mathsf{sk}_{\mathbf{z}}$ **,** $\mathbf{x}$ **):**

A first attempt at constructing a CPRF

$\mathsf{msk} := \mathbf{z_0}$

For a constraint vector $\mathbf{z}$:

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$

```
Eval($\mathsf{msk}$, $\mathbf{x}$):
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$
```

```
CEval($\mathsf{sk}_{\mathbf{z}}$, $\mathbf{x}$):
1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
```

A first attempt at constructing a CPRF

$$\mathsf{msk} := \mathbf{z_0}$$

```
Eval($\mathsf{msk}$, $\mathbf{x}$):
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$
```

For a constraint vector $\mathbf{z}$:

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

```
CEval($\mathsf{sk}_{\mathbf{z}}$, $\mathbf{x}$):
1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$
```

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

```
Eval(msk, x):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Is this correct?

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$\mathsf{msk} := \mathbf{z_0}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$

```
Eval(msk, x):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$\mathsf{msk} := \mathbf{z_0}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$

```
Eval(msk, x):
  1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
  2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
  1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
  2.  Output $F(k, \mathbf{x})$
```

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

```
Eval(msk, x):
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$
```

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

$$\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z}, \mathbf{x} \rangle + \langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$$

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

```
Eval(msk, $\mathbf{x}$):
  1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
  2. Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, $\mathbf{x}$):
  1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
  2. Output $F(k, \mathbf{x})$
```

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

Is this secure?

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

```
Eval(msk, x):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

Is this secure? No, because $\mathbf{z_0} = \mathbf{z_1} + \mathbf{z}$

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$\mathsf{msk} := \mathbf{z_0}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$

Eval(msk, $\mathbf{x}$):

1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$

CEval($\mathsf{sk}_{\mathbf{z}}$, $\mathbf{x}$):

1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

Is this secure? No, because $\mathbf{z_0} = \mathbf{z_1} + \mathbf{z}$

A first attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$\mathsf{msk} := \mathbf{z_0}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$

```
Eval(msk, x):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

Is this secure? No, because $\mathbf{z_0} = \mathbf{z_1} + \mathbf{z}$; **possible to recover the master key!**

A second attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$$

Eval($\mathsf{msk}, \mathbf{x}$**)**:

```
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$
```

CEval($\mathsf{sk}_{\mathbf{z}}, \mathbf{x}$**):**

```
1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$
```

A second attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \Delta\mathbf{z}$$

```
Eval(msk, x):
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$
```

A second attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\Delta \overset{R}{\leftarrow} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \Delta\mathbf{z}$$

Eval($\mathsf{msk}, \mathbf{x}$**):**

1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$

CEval($\mathsf{sk}_{\mathbf{z}}, \mathbf{x}$**):**

1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

A second attempt at constructing a CPRF

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1} = \mathbf{z_0} - \Delta \mathbf{z}$$

Eval($\mathsf{msk}, \mathbf{x}$ **):**

1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$

CEval($\mathsf{sk}_{\mathbf{z}}, \mathbf{x}$ **):**

1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$

Is this correct? Yes, because when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$.

$$\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \Delta\mathbf{z}, \mathbf{x} \rangle + \langle \mathbf{z_1}, \mathbf{x} \rangle = \Delta\langle \mathbf{z}, \mathbf{x} \rangle + \langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$$

Problem: keys are highly correlated

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta\mathbf{z} = \mathbf{z_1}$$

```
Eval(msk, $\mathbf{x}$):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, $\mathbf{x}$):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Construction using a random oracle

Problem: keys are highly correlated

$$\mathsf{msk} := \mathbf{z_0}$$

```
Eval($\mathsf{msk}$, $\mathbf{x}$):
1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2. Return $F(k, \mathbf{x})$
```

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta \mathbf{z} = \mathbf{z_1}$$

```
CEval($\mathsf{sk}_{\mathbf{z}}$, $\mathbf{x}$):
1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2. Output $F(k, \mathbf{x})$
```

Problem: keys are highly correlated

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta\mathbf{z} = \mathbf{z_1}$$

```
Eval(msk, x):
  1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
  2. Return $H(k, \mathbf{x})$
```

```
CEval(sk_z, x):
  1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
  2. Output $H(k, \mathbf{x})$
```

Okay if we replace the PRF with a RO

Let $H : \mathbb{F} \times \mathbb{F}^n \to \mathcal{Y}$ be a random oracle (RO).

Full construction using a random oracle

Let $H : \mathbb{F} \times \mathbb{F}^{\ell} \to \mathcal{Y}$ be a random oracle (RO).

```
KeyGen($1^{\lambda}$):
    1. $\mathbf{z_0} \overset{R}{\leftarrow} \mathbb{F}^{\ell}$
    2. Return msk $:= \mathbf{z_0}$
```

```
Constrain(msk, $\mathbf{z}$):
    1. $\Delta \overset{R}{\leftarrow} \mathbb{F}$
    2. $\mathbf{z_1} := \mathbf{z_0} - \Delta\mathbf{z}$
    3. Return sk$_{\mathbf{z}} := \mathbf{z_1}$
```

```
Eval(msk, $\mathbf{x}$):
    1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
    2. Return $H(k, \mathbf{x})$
```

```
CEval(sk$_{\mathbf{z}}$, $\mathbf{x}$):
    1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
    2. Output $H(k, \mathbf{x})$
```

⚠️ Simplified construction. See paper for full details.

Full construction using a random oracle

Proof of security

1. Think of $\mathbf{z_0}$ as $\mathbf{z_1} + \Delta\mathbf{z}$.

Full construction using a random oracle

Proof of security

1. Think of $\mathbf{z_0}$ as $\mathbf{z_1} + \Delta\mathbf{z}$.
2. For all $\mathbf{z}$ and $\mathbf{x}$ such that $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$, $\langle \mathbf{z_1}, \mathbf{x} \rangle$ is independent of $\langle \mathbf{z_0}, \mathbf{x} \rangle$

Full construction using a random oracle

Proof of security

1. Think of $\mathbf{z_0}$ as $\mathbf{z_1} + \Delta\mathbf{z}$.
2. For all $\mathbf{z}$ and $\mathbf{x}$ such that $\langle\mathbf{z},\mathbf{x}\rangle \neq 0$, $\langle\mathbf{z_1},\mathbf{x}\rangle$ is independent of $\langle\mathbf{z_0},\mathbf{x}\rangle$ because Δ is chosen uniformly and independently of $\mathbf{z_0}$ and we can write:

$$\langle\mathbf{z_1},\mathbf{x}\rangle = \langle\mathbf{z_0},\mathbf{x}\rangle + \Delta\langle\mathbf{z},\mathbf{x}\rangle.$$

Full construction using a random oracle

Proof of security

1. Think of $\mathbf{z_0}$ as $\mathbf{z_1} + \Delta\mathbf{z}$.
2. For all $\mathbf{z}$ and $\mathbf{x}$ such that $\langle\mathbf{z},\mathbf{x}\rangle \neq 0$, $\langle\mathbf{z_1},\mathbf{x}\rangle$ is independent of $\langle\mathbf{z_0},\mathbf{x}\rangle$ because Δ is chosen uniformly and independently of $\mathbf{z_0}$ and we can write:
$$\langle\mathbf{z_1},\mathbf{x}\rangle = \langle\mathbf{z_0},\mathbf{x}\rangle + \Delta\langle\mathbf{z},\mathbf{x}\rangle.$$
3. Therefore, **one constrained evaluation query** is equivalent to a evaluating the PRF using an independent key from the point of view of the adversary.

Full construction using a random oracle

Proof of security

1. Think of $\mathbf{z_0}$ as $\mathbf{z_1} + \Delta\mathbf{z}$.
2. For all $\mathbf{z}$ and $\mathbf{x}$ such that $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$, $\langle \mathbf{z_1}, \mathbf{x} \rangle$ is independent of $\langle \mathbf{z_0}, \mathbf{x} \rangle$ because Δ is chosen uniformly and independently of $\mathbf{z_0}$ and we can write:
$$\langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle + \Delta \langle \mathbf{z}, \mathbf{x} \rangle.$$
3. Therefore, **one constrained evaluation query** is equivalent to a evaluating the PRF using an independent key from the point of view of the adversary.
4. [AMN+18]: any CPRF that satisfies security with one constrained evaluation query can be made to provide adaptive security with a random oracle.

A general framework

Problem: keys are highly correlated

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta\mathbf{z} = \mathbf{z_1}$$

```
Eval(msk, x):
    1. $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
    2. Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
    1. $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
    2. Output $F(k, \mathbf{x})$
```

Problem: keys are highly correlated

$$\mathsf{msk} := \mathbf{z_0}$$

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta\mathbf{z} = \mathbf{z_1}$$

```
Eval(msk, x):
1.    $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.    Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1.    $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.    Output $F(k, \mathbf{x})$
```

Requires security against correlated keys

Problem: keys are highly correlated

For a constraint vector $\mathbf{z}$:

$$\Delta \xleftarrow{R} \mathbb{F}$$

$$\mathsf{msk} := \mathbf{z_0}$$

$$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta\mathbf{z} = \mathbf{z_1}$$

```
Eval(msk, x):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

```
CEval(sk_z, x):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Requires security against correlated keys

Let $F : \mathbb{F} \times \mathbb{F}^n \to \mathcal{Y}$ be a related-key attack (RKA) security.

Regular security for a PRF

A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a secure PRF if:

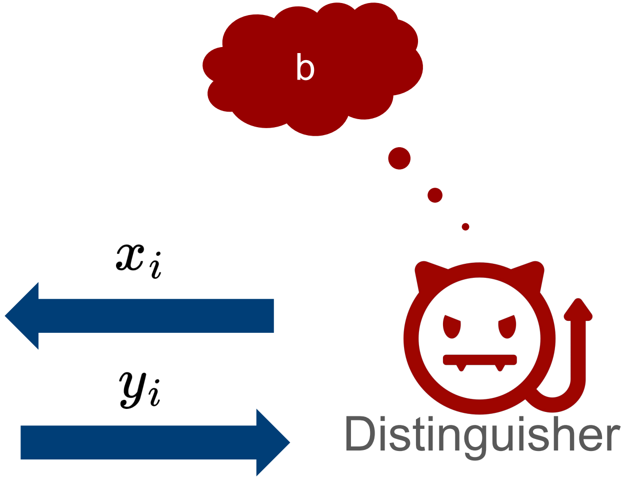
Setup phase (one time)

Challenger

1. $k \overset{R}{\leftarrow} \mathcal{K}$
2. $R \overset{R}{\leftarrow} \mathcal{F}uns\left(\mathcal{X}, \mathcal{Y}\right)$
3. $b \overset{R}{\leftarrow} \{0, 1\}$

Query phase (repeatable)

5. $$y_i := \begin{cases} F\left(k,\ x_i\right) & \text{if } b = 0 \\ R\left(x_i\right) & \text{if } b = 1 \end{cases}$$



Related Key Attack (RKA) security for a PRF

A function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is an **RKA-secure** PRF if:

Setup phase (one time)

Challenger

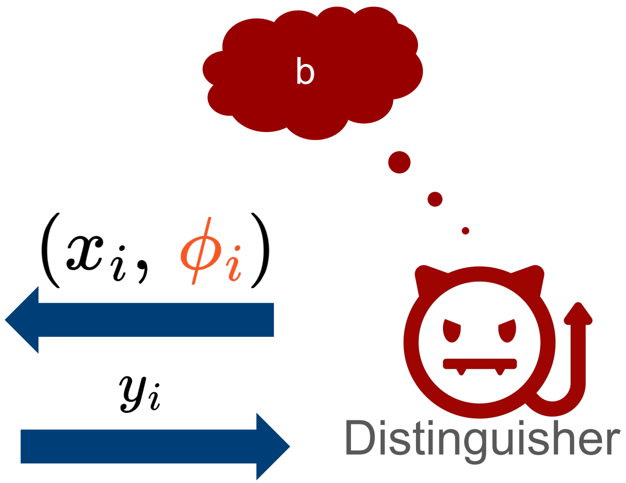
1 $k \xleftarrow{R} \mathcal{K}$

2 $R \xleftarrow{R} \mathcal{F}uns\left(\left(\mathcal{X}, \Phi\right), \mathcal{Y}\right)$

3 $b \xleftarrow{R} \{0, 1\}$

Query phase (repeatable)

5 $$y_i := \begin{cases} F\left(\phi_i\left(k\right),\ x_i\right) & \text{if } b = 0 \\ R\left(x_i, \phi_i\right) & \text{if } b = 1 \end{cases}$$



For a class of key derivation functions $\Phi : \mathcal{K} \rightarrow \mathcal{K}$

Solution: Use a PRF with related-key security

The inner product $\langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \boldsymbol{\Delta} \langle \mathbf{z}, \mathbf{x} \rangle$

Solution: Use a PRF with related-key security

The inner product $\langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \Delta \langle \mathbf{z}, \mathbf{x} \rangle$

is an ***affine*** function of Δ, determined by $\mathbf{x}$

Solution: Use a PRF with related-key security

The inner product $\langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \Delta \langle \mathbf{z}, \mathbf{x} \rangle$

is an ***affine*** function of Δ, determined by $\mathbf{x}$

msk $:= \mathbf{z_0}$

```
Eval(msk, x):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta \mathbf{z} = \mathbf{z_1}$

```
CEval(sk_z, x):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Solution: Use a PRF with related-key security

The inner product $\langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \Delta \langle \mathbf{z}, \mathbf{x} \rangle$

is an ***affine*** function of Δ, determined by $\mathbf{x}$

$\mathsf{msk} := \mathbf{z_0}$

```
Eval(msk, $\mathbf{x}$):
1.  $k := \langle \mathbf{z_0}, \mathbf{x} \rangle$
2.  Return $F(k, \mathbf{x})$
```

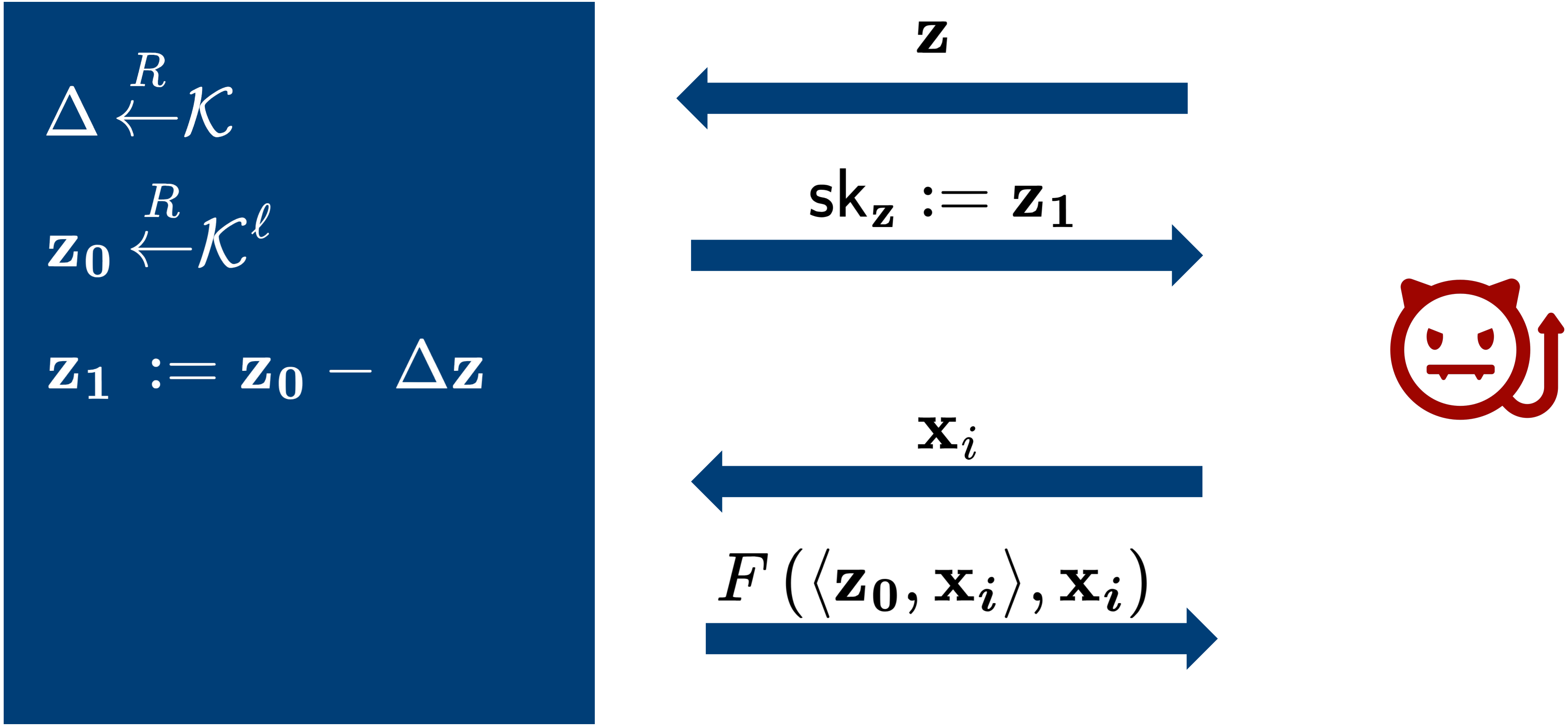
$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_0} - \Delta \mathbf{z} = \mathbf{z_1}$

```
CEval($\mathsf{sk}_{\mathbf{z}}$, $\mathbf{x}$):
1.  $k := \langle \mathbf{z_1}, \mathbf{x} \rangle$
2.  Output $F(k, \mathbf{x})$
```

Need F to be RKA-secure for affine functions

Reduction to RKA security

Step 1: The (1 key, selective) CPRF security game



$\Delta \xleftarrow{R} \mathcal{K}$

$\mathbf{z}_0 \xleftarrow{R} \mathcal{K}^\ell$

$\mathbf{z}_1 := \mathbf{z}_0 - \Delta\mathbf{z}$

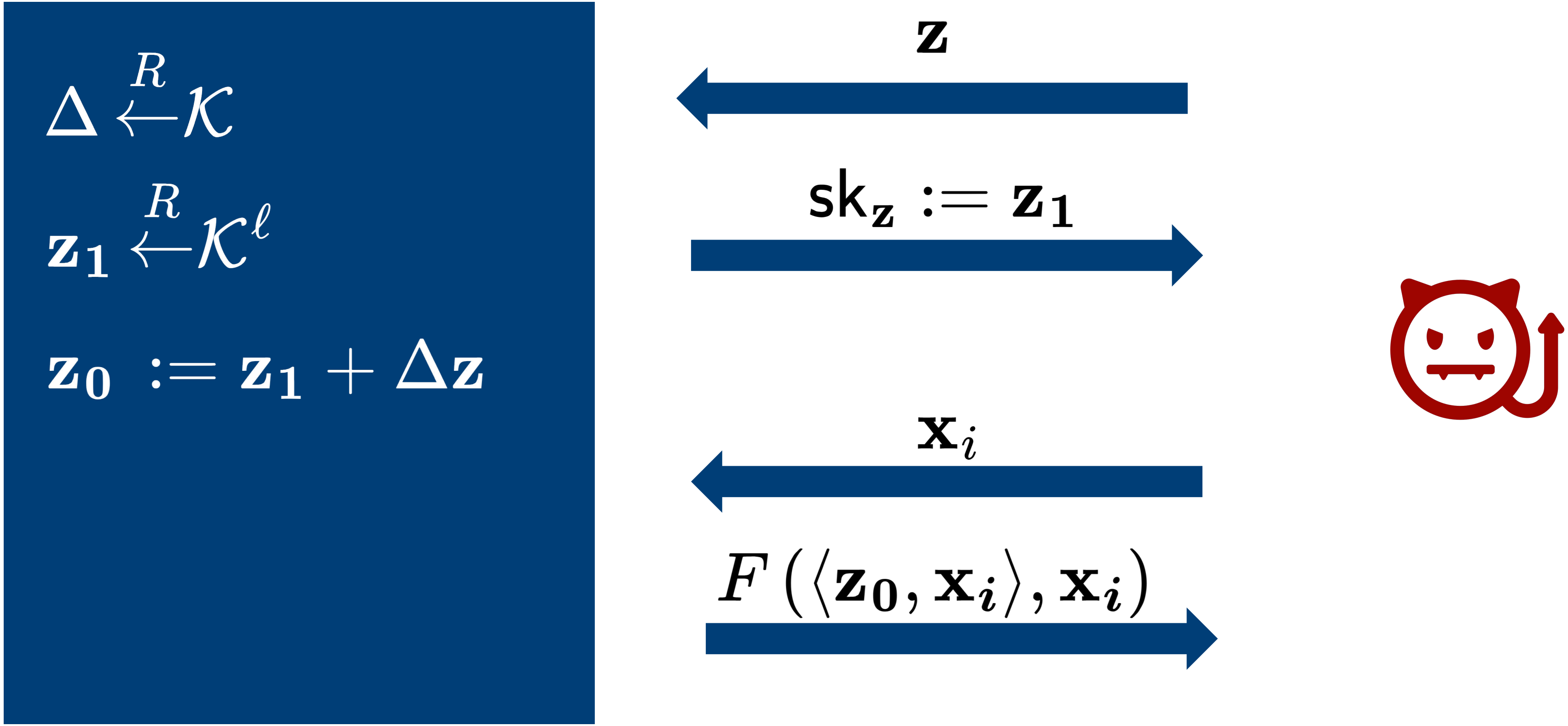
$\mathbf{z}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z}_1$

$\mathbf{x}_i$

$F\left(\langle \mathbf{z}_0, \mathbf{x}_i \rangle, \mathbf{x}_i\right)$

Step 2: Change definition of $\mathbf{z}_0$ to be in terms of $\mathbf{z}_1$



$$\Delta \xleftarrow{R} \mathcal{K}$$

$$\mathbf{z}_1 \xleftarrow{R} \mathcal{K}^{\ell}$$

$$\mathbf{z}_0 := \mathbf{z}_1 + \Delta \mathbf{z}$$

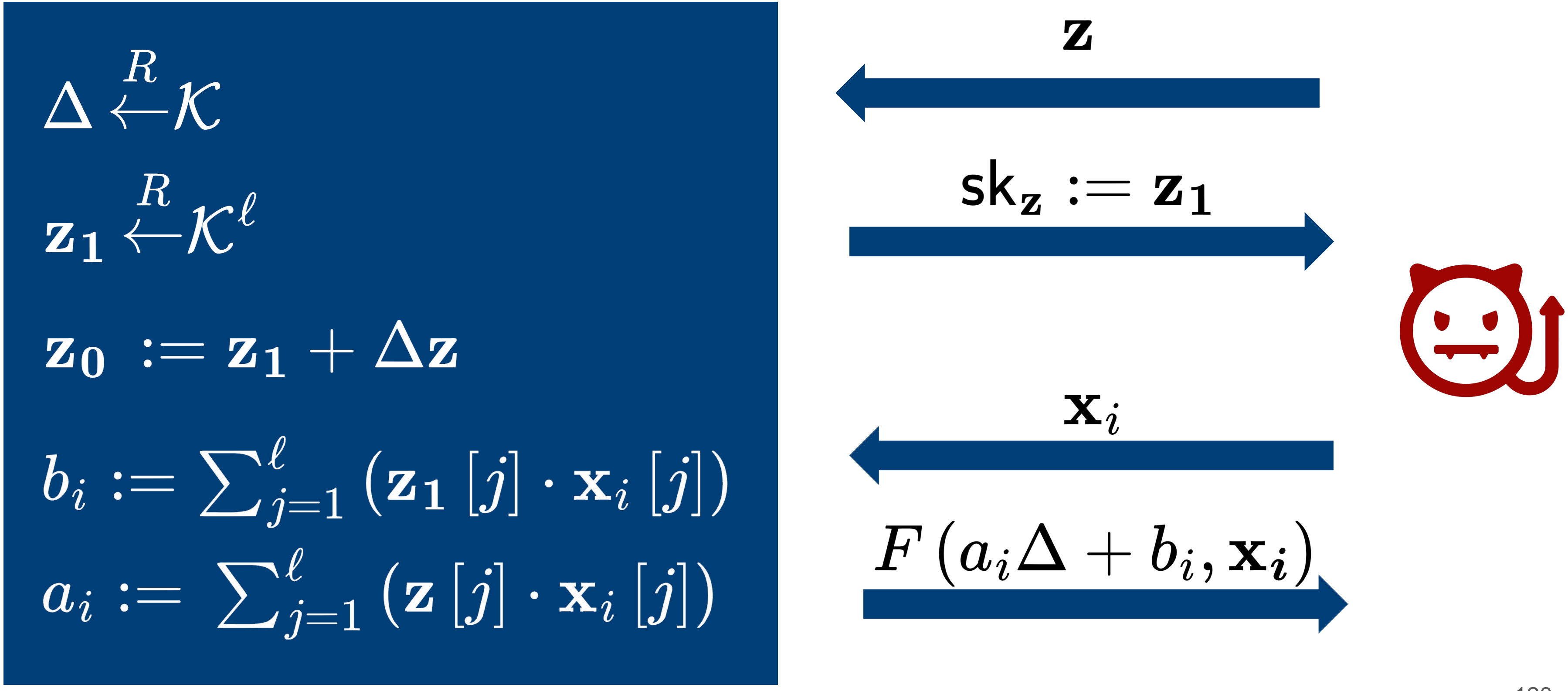
$\mathbf{z}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z}_1$

$\mathbf{x}_i$

$F\left(\langle \mathbf{z}_0, \mathbf{x}_i \rangle, \mathbf{x}_i\right)$

Step 3: Define the inner-product as an affine function



$$\Delta \overset{R}{\leftarrow} \mathcal{K}$$

$$\mathbf{z_1} \overset{R}{\leftarrow} \mathcal{K}^{\ell}$$

$$\mathbf{z_0} := \mathbf{z_1} + \Delta \mathbf{z}$$

$$b_i := \sum_{j=1}^{\ell} \left(\mathbf{z_1}[j] \cdot \mathbf{x}_i[j] \right)$$

$$a_i := \sum_{j=1}^{\ell} \left(\mathbf{z}[j] \cdot \mathbf{x}_i[j] \right)$$

$$\xleftarrow{\mathbf{z}}$$

$$\xrightarrow{\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1}}$$

$$\xleftarrow{\mathbf{x}_i}$$

$$\xrightarrow{F\left(a_i \Delta + b_i, \mathbf{x_i}\right)}$$

Step 3: Define the inner-product as an affine function

$$\Delta \overset{R}{\leftarrow} \mathcal{K}$$

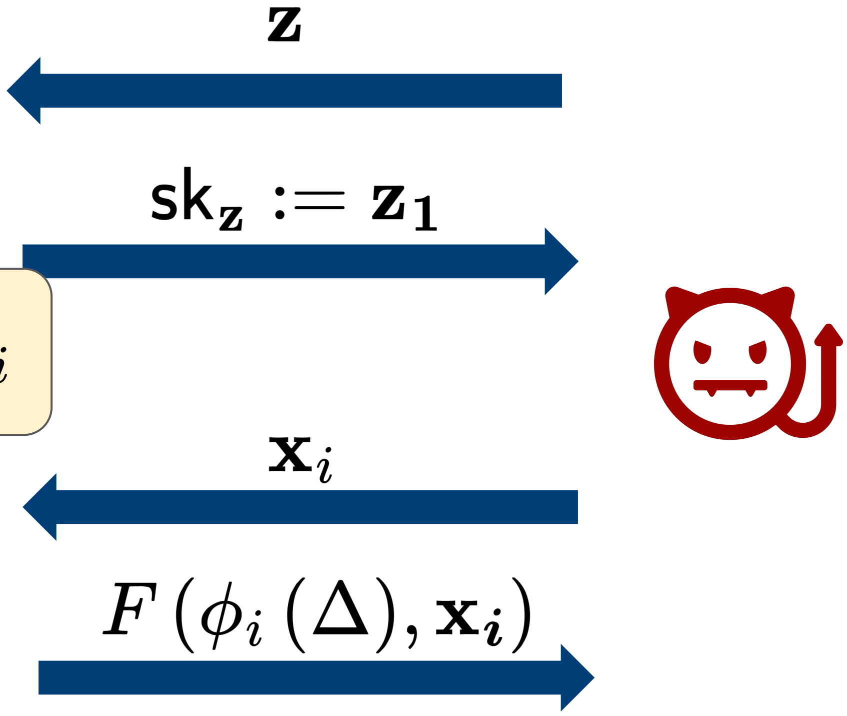
$$\mathbf{z_1} \overset{R}{\leftarrow} \mathcal{K}^{\ell}$$

$$\mathbf{z_0} := \mathbf{z}$$

$$\phi_i : \Delta \mapsto a_i \Delta + b_i$$

$$b_i := \sum_{j=1}^{\ell} \left(\mathbf{z_1}[j] \cdot \mathbf{x}_i[j]\right)$$

$$a_i := \sum_{j=1}^{\ell} \left(\mathbf{z}[j] \cdot \mathbf{x}_i[j]\right)$$



$\mathbf{z}$

$\mathsf{sk}_{\mathbf{z}} := \mathbf{z_1}$

$\mathbf{x}_i$

$F\left(\phi_i\left(\Delta\right), \mathbf{x_i}\right)$

Step 4: Reduce to RKA security

The key Δ is not sampled anymore…

$$\mathbf{z}_1 \xleftarrow{R} \mathcal{K}^\ell$$

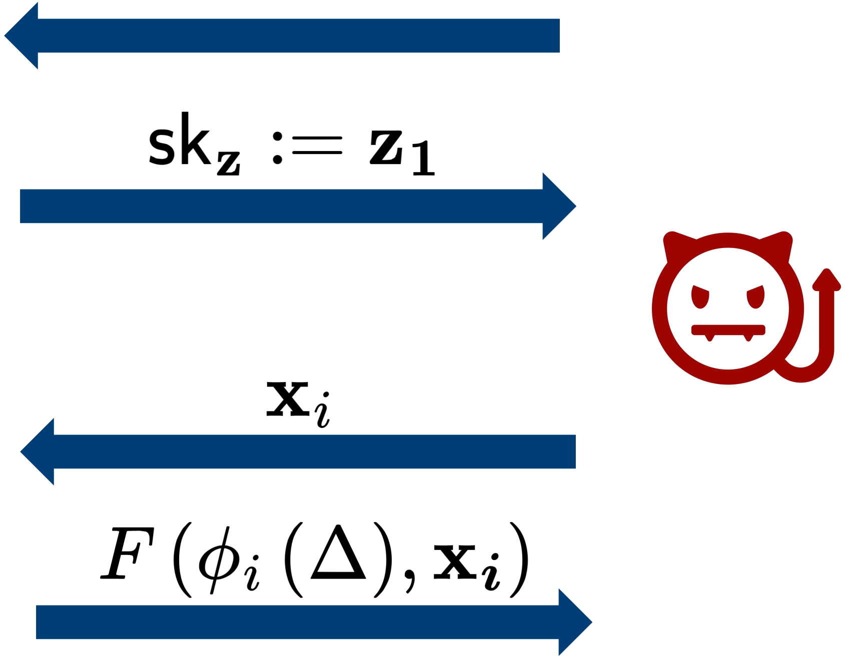
$$b_i := \sum_{j=1}^{\ell} \left(\mathbf{z}_1[j] \cdot \mathbf{x}_i[j]\right)$$

$$a_i := \sum_{j=1}^{\ell} \left(\mathbf{z}[j] \cdot \mathbf{x}_i[j]\right)$$

Query RKA PRF challenger on input:

$$\left(\phi_i := (a_i, b_i),\ \mathbf{x}_\mathrm{i}\right)$$

And get back: $F\left(\phi_i(\Delta), \mathbf{x}_i\right)$



$\mathsf{sk}_{\mathbf{z}} := \mathbf{z}_1$

$\mathbf{x}_i$

$F\left(\phi_i(\Delta), \mathbf{x}_i\right)$

Constructions from RKA-secure PRFs

Constructions of CPRFs from RKA-secure PRFs

Constructions of CPRFs from RKA-secure PRFs

From **DDH** via variant of the Naor-Reingold PRF [ABP+14]

Directly follows from the construction of [ABP+14] affine-function RKA security

Constructions of CPRFs from RKA-secure PRFs

From **DDH** via variant of the Naor-Reingold PRF [ABP+14]

Directly follows from the construction of [ABP+14] affine-function RKA security

From **Variable Density LPN** via [BCG+20]

Only works for random inputs since the VDLPN candidate is a weak PRF

Constructions of CPRFs from RKA-secure PRFs

From **DDH** via variant of the Naor-Reingold PRF [ABP+14]

Directly follows from the construction of [ABP+14] affine-function RKA security

From **Variable Density LPN** via [BCG+20]

Only works for random inputs since the VDLPN candidate is a weak PRF

From **OWF** via t-wise independent hashing [AW14]

Needs some additional technical work over the construction of [AW14]

OWF-based instantiation

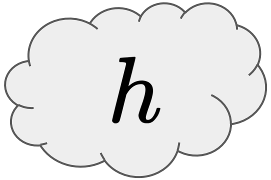
RKA-secure **bounded**-PRF construction of [AW14]

Problem 1: Only provides RKA security for additive key derivation functions.

Problem 2: Requires the adversary to use most $T = T(\lambda) \in \textbf{poly}(\lambda)$ unique RKA functions.

Fixed at setup time

OWF-based instantiation



“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

$$\{h\left(\phi_1\left(\Delta\right)\right), \ldots, h\left(\phi_T\left(\Delta\right)\right)\} \approx_s \{r_1, \ldots, r_T\}$$

Implied by a $\Omega\left(\lambda T^2\right)$-wise independent hash function [AW14]

OWF-based instantiation

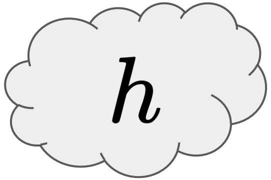
$$h$$

“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

$$\left\{h\left(\langle \mathbf{z}, \mathbf{x_1}\rangle \Delta + \langle \mathbf{z_0}, \mathbf{x_1}\rangle\right), \ldots, h\left(\langle \mathbf{z}, \mathbf{x_T}\rangle \Delta + \langle \mathbf{z_0}, \mathbf{x_T}\rangle\right)\right\} \approx_s \{r_1, \ldots, r_T\}$$

OWF-based instantiation

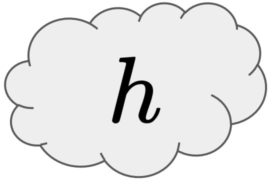


“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

$$\{h\left(a_1\Delta + b_1\right), \ldots, h\left(a_T\Delta + b_T\right)\}$$
$$\approx_s \{r_1, \ldots, r_T\}$$

OWF-based instantiation



“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

$$\{h\left(\phi_1\left(\Delta\right)\right), \ldots, h\left(\phi_T\left(\Delta\right)\right)\}$$
$$\approx_s \{r_1, \ldots, r_T\}$$

OWF-based instantiation

RKA-secure bounded PRF construction of [AW14]

Problem 1: Only provides RKA security for additive key derivation functions.

OWF-based instantiation

RKA-secure bounded PRF construction of [AW14]

Problem 1: Only provides RKA security for additive key derivation functions.

We upgrade the “T-good” hash function of [AW14] to the affine case

OWF-based instantiation

RKA-secure bounded PRF construction of [AW14]

Problem 1: Only provides RKA security for additive key derivation functions.

We upgrade the "T-good" hash function of [AW14] to the affine case

Problem 2: Requires the adversary to use most $T = T(\lambda) \in$ **poly** (λ) unique RKA functions.

OWF-based instantiation

RKA-secure bounded PRF construction of [AW14]

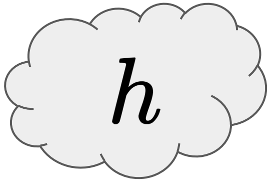
Problem 1: Only provides RKA security for additive key derivation functions.

We upgrade the "T-good" hash function of [AW14] to the affine case

Problem 2: Requires the adversary to use most $T = T(\lambda) \in \mathbf{poly}(\lambda)$ unique RKA functions.

We make the input domain polynomial in the security parameter

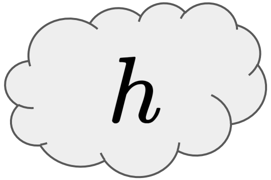
OWF-based instantiation



“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

OWF-based instantiation

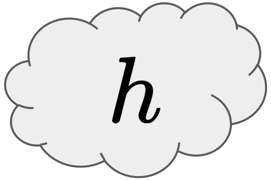


“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

Define the set $S = \left\{ \langle \mathbf{a}, \mathbf{x} \rangle \;\mid \mathbf{x} \in \{0, \ldots, B\}^{\ell} \right\}$

OWF-based instantiation



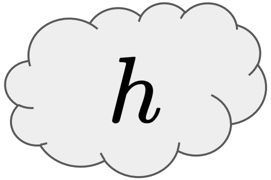
“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

Define the set $S = \left\{ \langle \mathbf{a}, \mathbf{x} \rangle \ \mid \mathbf{x} \in \{0, \ldots, B\}^{\ell} \right\}$

Note that: $|S| \leq B^{\ell}$ so we need to set parameters such that $B^{\ell} \leq T$

OWF-based instantiation



“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

Define the set $S = \left\{ \langle \mathbf{a}, \mathbf{x} \rangle \;\middle|\; \mathbf{x} \in \{0, \ldots, B\}^{\ell} \right\}$

Note that: $|S| \leq B^{\ell}$ so we need to set parameters such that $B^{\ell} \leq T$

$B = O(1) \wedge \ell = \ell(\lambda) \in \textsf{poly}(\lambda)$

OWF-based instantiation

$$h$$

“T-good” hash function:

“Acts like a random oracle for up to T unique inputs”

Define the set $S = \left\{ \langle \mathbf{a}, \mathbf{x} \rangle \;\middle|\; \mathbf{x} \in \{0, \ldots, B\}^{\ell} \right\}$

Note that: $|S| \leq B^{\ell}$ so we need to set parameters such that $B^{\ell} \leq T$

Polynomially-bounded input domain

Implementation and Evaluation

Artifact Badges: Available, Functional, and Reproduced.

https://github.com/sachaservan/cprf

Evaluation of the **random oracle** based CPRF

ℓ **(length of vector)**	**Evaluation time**
10	2 μs
50	10 μs
100	19 μs
500	98 μs
1000	200 μs

Implemented in Go (v1.20) without any significant optimizations

Bottleneck: inner-product computation in the finite field

Evaluation of the **DDH-based** CPRF

ℓ **(length of vector)**	**Evaluation time**
10	8 ms
50	11 ms
100	16 ms
500	46 ms
1000	85 ms

Implemented in Go (v1.20) without any significant optimizations

Bottleneck: exponentiations in the group

Open Questions

Concrete applications for CPRFs with inner product predicates?

Open Questions

Concrete applications for CPRFs with inner product predicates?

Extending constructions to NC^1 constraints?

Open Questions

Concrete applications for CPRFs with inner product predicates?

Extending constructions to NC^1 constraints?

Open Questions

Instantiating the framework under more assumptions?

Concrete applications for CPRFs with inner product predicates?

Extending constructions to NC^1 constraints?

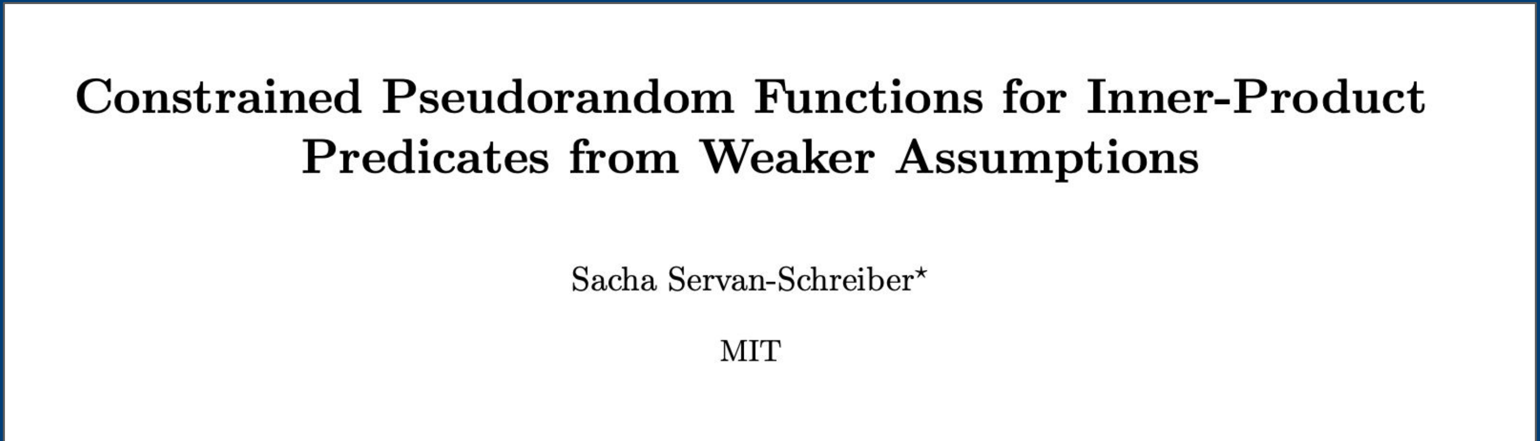
Open Questions

Instantiating the framework under more assumptions?

OWF construction with superpolynomial domain?

Thank you!

Email: 3s@mit.edu
ePrint: ia.cr/2024/058



Constrained Pseudorandom Functions for Inner-Product Predicates from Weaker Assumptions

Sacha Servan-Schreiber*

MIT

References

[ABP+14]: Abdalla, Michel, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. "Related-key security for pseudorandom functions beyond the linear barrier." *CRYPTO 2014.*

[AW14]: Applebaum, Benny, and Eyal Widder. "Related-key secure pseudorandom functions: The case of additive attacks." *ePrint Archive* (2014).

[AMN+18]: Attrapadung, Nuttapong, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. "Constrained PRFs for in traditional groups." CRYPTO 2018.

[BCG+20]: Boyle, Elette, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. "Correlated pseudorandom functions from variable-density LPN." FOCS 2020.

[CMPR23]: Couteau, Geoffroy, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. "Constrained Pseudorandom Functions from Homomorphic Secret Sharing." EUROCRYPT 2023.

[DKN+20]: Davidson, Alex, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. "Adaptively secure constrained pseudorandom functions in the standard model." CRYPTO 2020.