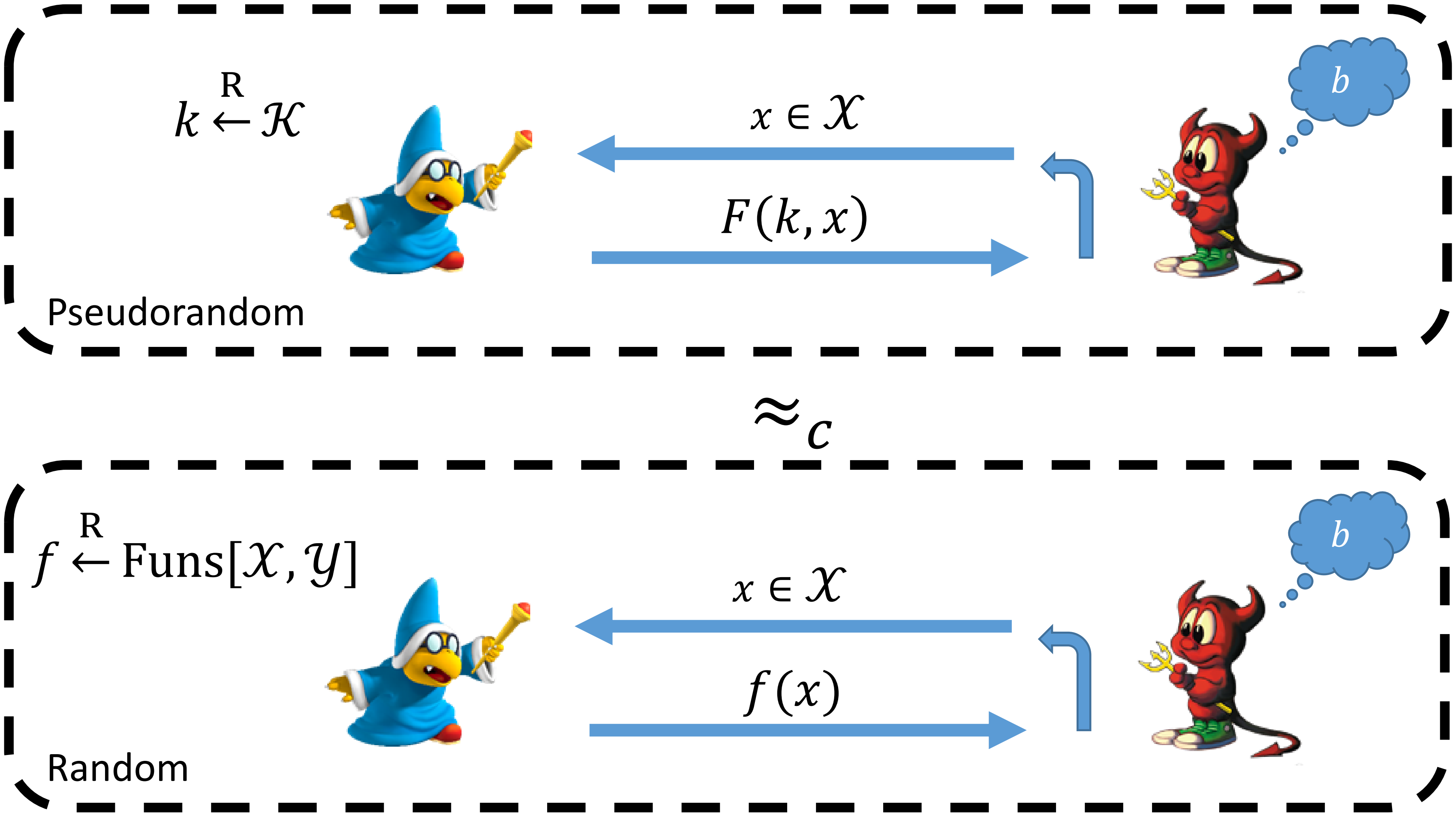
Part I: Constraining PRFs Privately

David Wu
Stanford University

Joint work with Dan Boneh and Kevin Lewi

Pseudorandom Functions (PRFs) [GGM84]



$F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

Constrained PRFs [BW13, BGI13, KPTZ13]

Constrained PRF: PRF with additional “constrain” functionality

PRF key

constrained key

$$F: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$$

can be used to evaluate at all points $x \in \mathcal{X}$ where $C(x) = 1$

Constrained PRFs [BW13, BGI13, KPTZ13]

Correctness: constrained evaluation at $x \in \mathcal{X}$ where $C(x) = 1$ yields PRF value at x

Security: PRF value at points $x \in \mathcal{X}$ where $C(x) = 0$ are indistinguishable from random

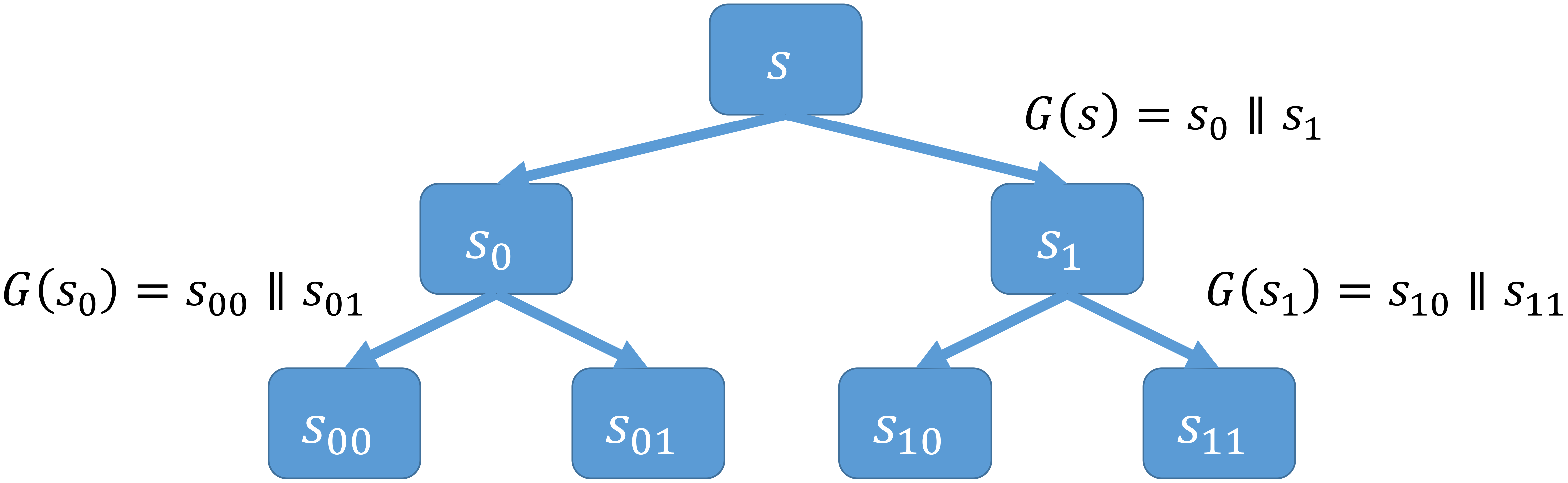
Constrained PRFs [BW13, BGI13, KPTZ13]

Many applications:

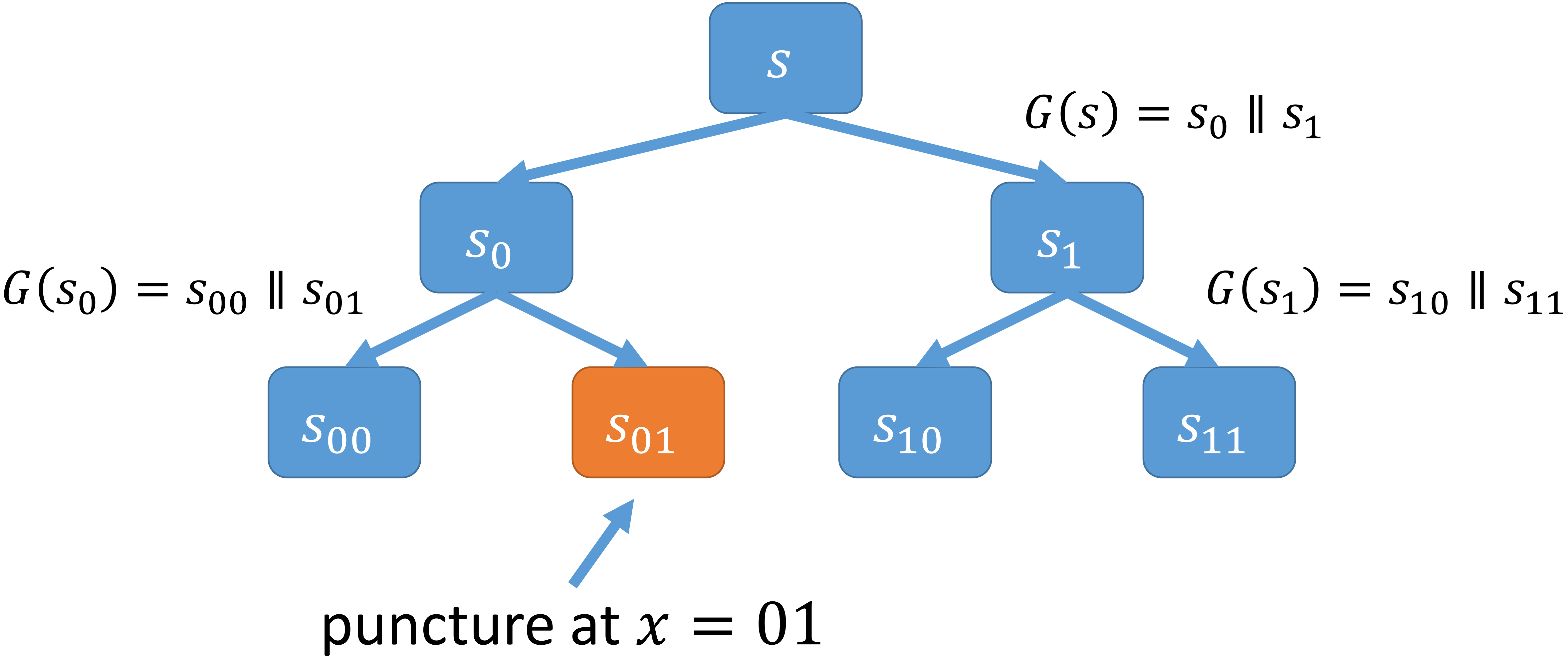
- Identity-Based Key Exchange, Optimal Broadcast Encryption [BW13]
- Punctured Programming Paradigm [SW14]
- Multiparty Key Exchange, Traitor Tracing [BZ14]

Puncturable PRFs from GGM

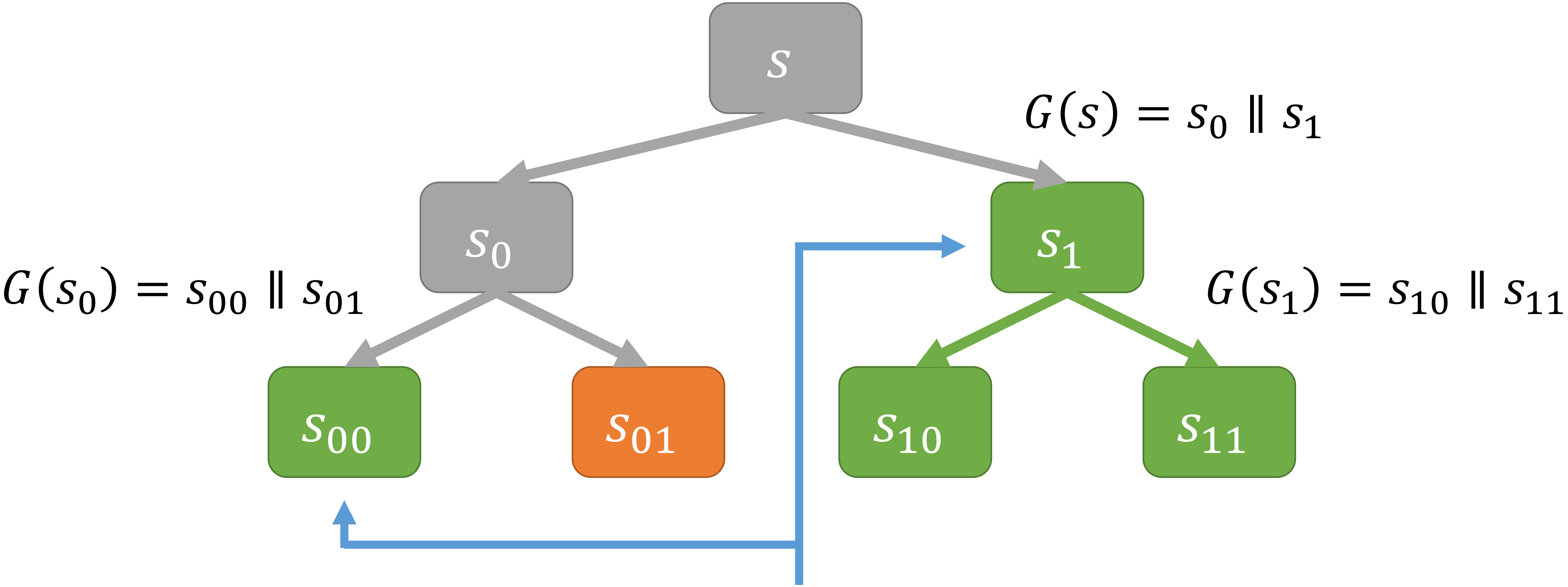
- Puncturable PRF: constrained keys allow evaluation at *all* but a single point
- Easily constructed from GGM:



Puncturable PRFs from GGM

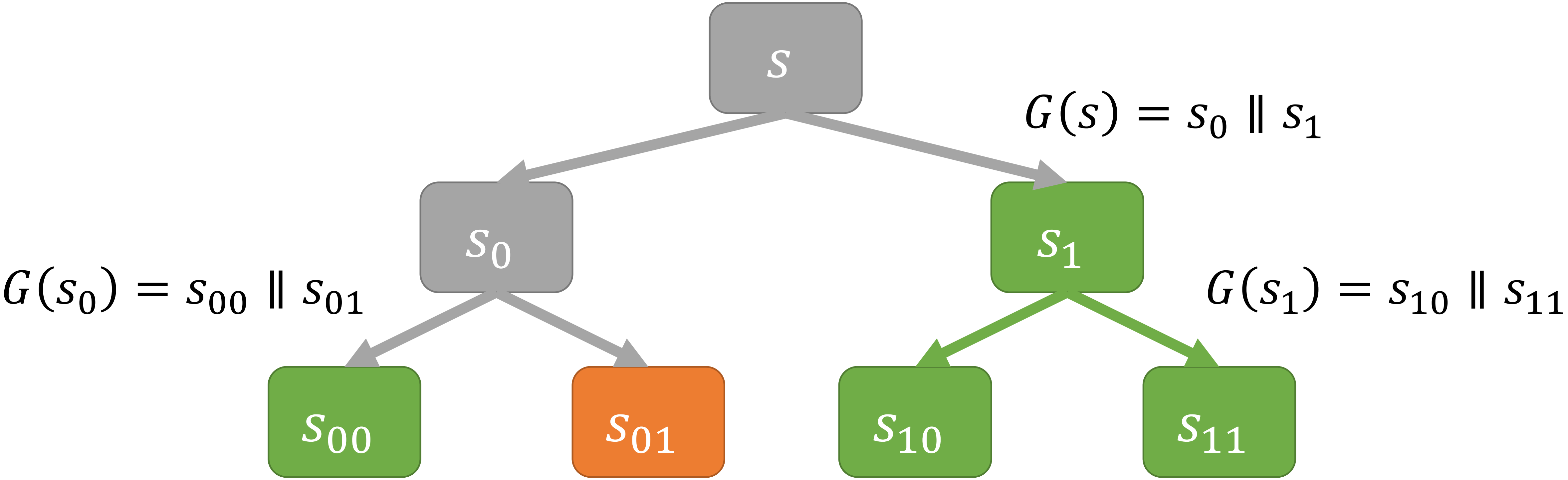


Puncturable PRFs from GGM



these two values suffice to evaluate at all other points

Puncturable PRFs from GGM

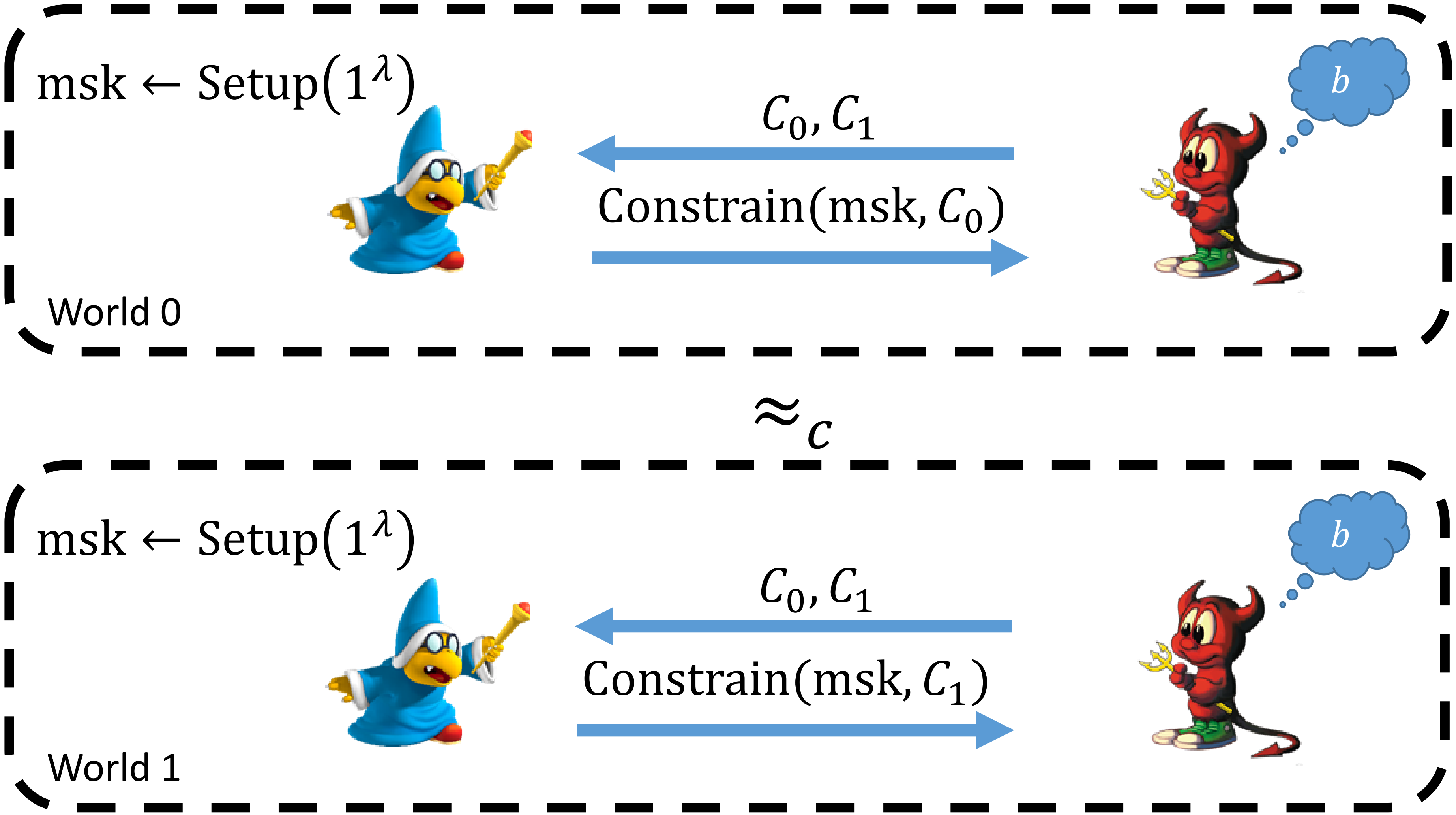


given s_1 and s_{00}, easy to tell that 01 is the punctured point

Constraining PRFs Privately

Can we build a constrained PRF where the constrained key for a circuit C hides C?

Constraining PRFs Privately



Single-key privacy

Definitions generalize to multi-key privacy. See paper for details.

Private Puncturing

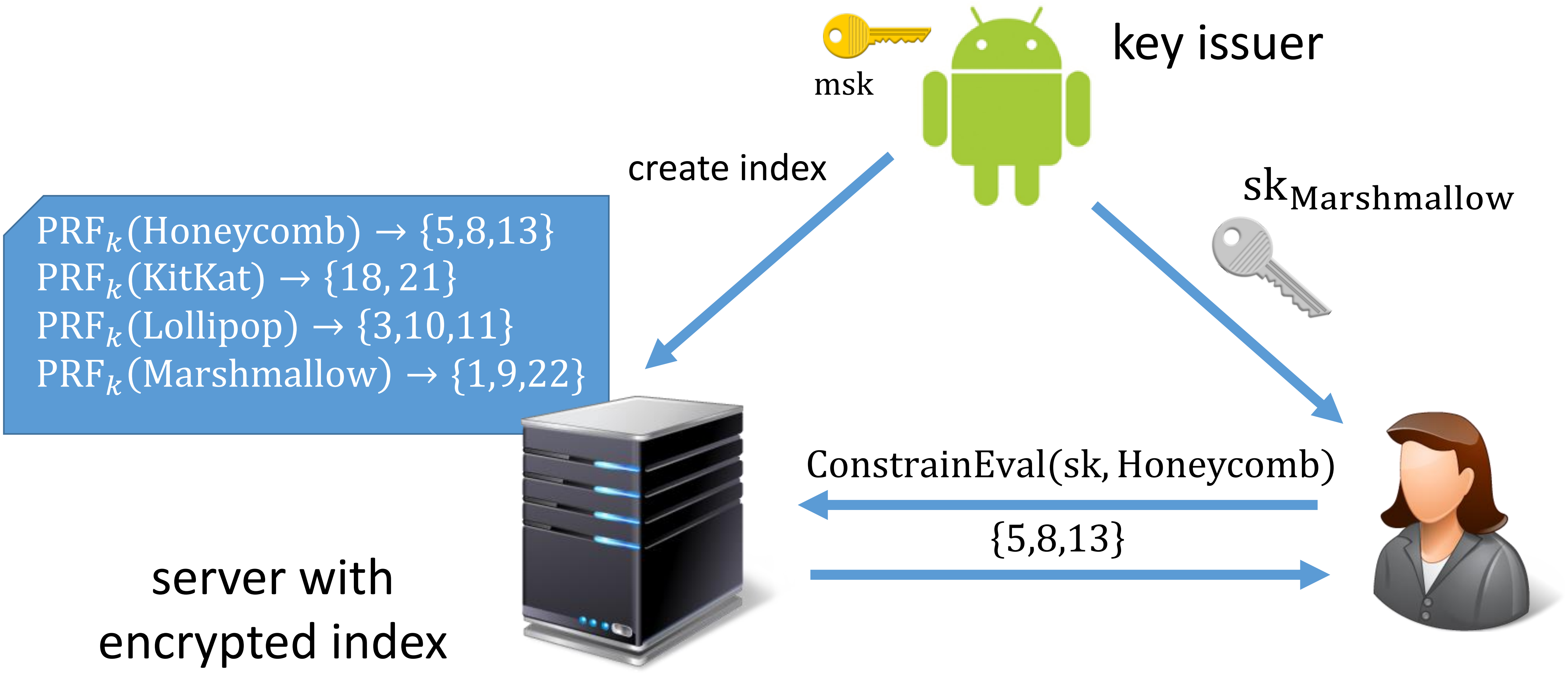
- <u>**Correctness:**</u> constrained evaluation at $x \neq z$ yields $F(k, x)$
- <u>**Security:**</u> $F(k, z)$ is indistinguishable from random
- <u>**Privacy:**</u> constrained key hides z

Implications of Privacy

Consider value of $\mathsf{ConstrainEval}(\mathsf{sk}_z, z)$:

- **Security**: Independent of $\mathsf{Eval}(\mathsf{msk}, z)$
- **Privacy**: Unguessable by the adversary

Using Privacy: Restricted Keyword Search



Using Privacy: Restricted Keyword Search

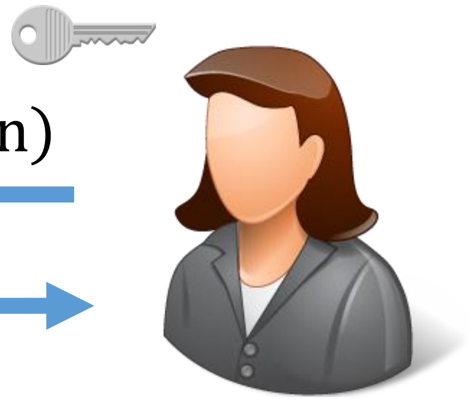
search for non-existent keyword

$\mathrm{PRF}_k(\mathrm{Honeycomb}) \to \{5,8,13\}$
$\mathrm{PRF}_k(\mathrm{KitKat}) \to \{18, 21\}$
$\mathrm{PRF}_k(\mathrm{Lollipop}) \to \{3,10,11\}$
$\mathrm{PRF}_k(\mathrm{Marshmallow}) \to \{1,9,22\}$

server with encrypted index

$\mathrm{ConstrainEval}(\mathrm{sk}, \mathrm{Jelly\ Bean})$

No results



Using Privacy: Restricted Keyword Search

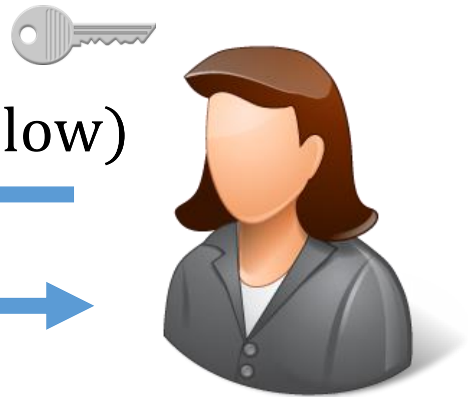
search for “restricted” keyword

$\mathrm{PRF}_k(\mathrm{Honeycomb}) \rightarrow \{5,8,13\}$
$\mathrm{PRF}_k(\mathrm{KitKat}) \rightarrow \{18, 21\}$
$\mathrm{PRF}_k(\mathrm{Lollipop}) \rightarrow \{3,10,11\}$
$\mathrm{PRF}_k(\mathrm{Marshmallow}) \rightarrow \{1,9,22\}$

server with encrypted index

$\mathrm{ConstrainEval}(\mathrm{sk}, \mathrm{Marshmallow})$

No results



Using Privacy: Restricted Keyword Search

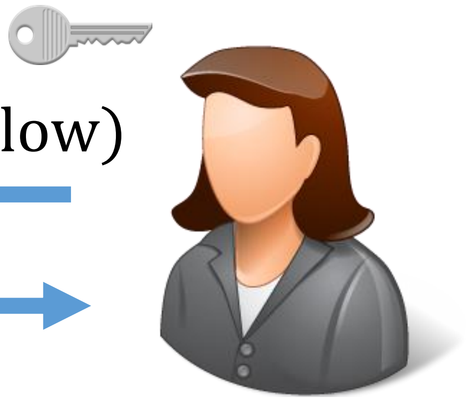
- **Security**: $\text{ConstrainEval}(\text{sk}, \text{Marshmallow}) \neq \text{Eval}(\text{msk}, \text{Marshmallow})$
- **Privacy**: Does not learn that no results were returned because no matches for keyword or if the keyword was restricted

$\text{PRF}_k(\text{Honeycomb}) \rightarrow \{5,8,13\}$
$\text{PRF}_k(\text{KitKat}) \rightarrow \{18, 21\}$
$\text{PRF}_k(\text{Lollipop}) \rightarrow \{3,10,11\}$
$\text{PRF}_k(\text{Marshmallow}) \rightarrow \{1,9,22\}$

server with encrypted index

$\text{ConstrainEval}(\text{sk}, \text{Marshmallow})$

No results



The Many Applications of Privacy

- Private constrained MACs
  - Parties can only sign messages satisfying certain policy (e.g., enforce a spending limit), but policies are hidden
- Symmetric Deniable Encryption [CDNO97]
  - Two parties can communicate using a symmetric encryption scheme
  - If an adversary has intercepted a sequence of messages and coerces one of the parties to produce a decryption key for the messages, they can produce a "fake" key that decrypts all but a subset of the messages
- Constructing a family of watermarkable PRFs
  - Can be used to embed a secret message within a PRF that is "unremovable" – useful for authentication [CHNVW15]

See paper for details!

Summary of our Constructions

- From indistinguishability obfuscation (iO):
  - Private puncturable PRFs from iO + one-way functions
  - Private circuit constrained PRFs from sub-exponentially hard iO + one-way functions
- From concrete assumptions on multilinear maps:
  - Private puncturable PRFs from subgroup hiding assumptions

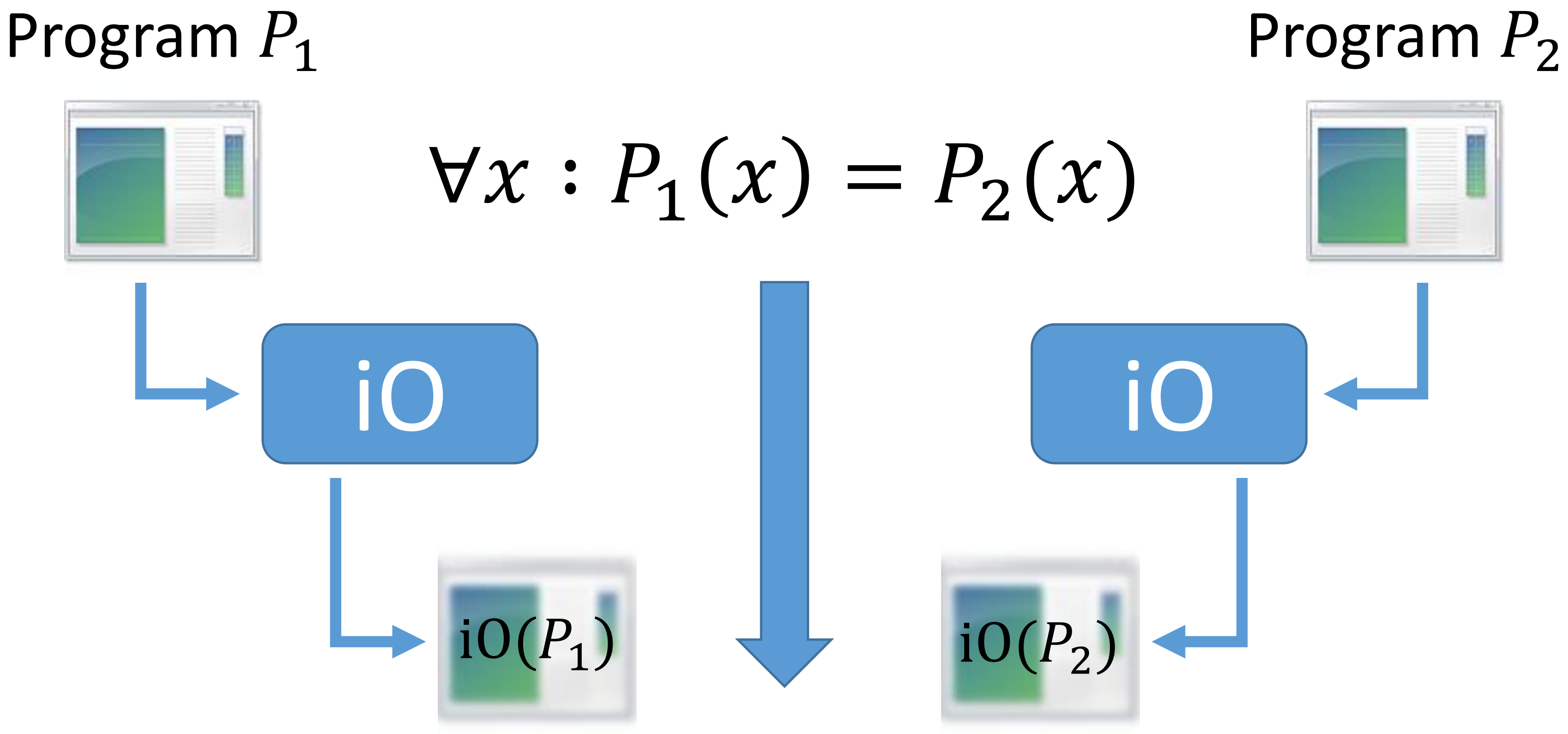
This talk

  - Private bit-fixing PRF from multilinear Diffie-Hellman assumption

See paper

Constructing Private Constrained PRFs

Tool: indistinguishability obfuscation [BGI+01, GGH+13]



$$\mathrm{iO}(P_1) \approx_c \mathrm{iO}(P_2)$$

Private Puncturing from iO

- Starting point: puncturable PRFs (e.g. GGM)

- Need a way to hide the point that is punctured
  - Intuition: obfuscate the puncturable PRF

- Question: what value to output at the punctured point?

Private Puncturing from iO

Use iO to hide the punctured point and output uniformly random value at punctured point

$P_z(x)$:

random value (hard coded)

- If $x = z$, output $\boxed{r}$
- Else, output $\boxed{\mathrm{PRF}(k, x)}$

real value of the PRF

Program for punctured PRF (punctured at z)

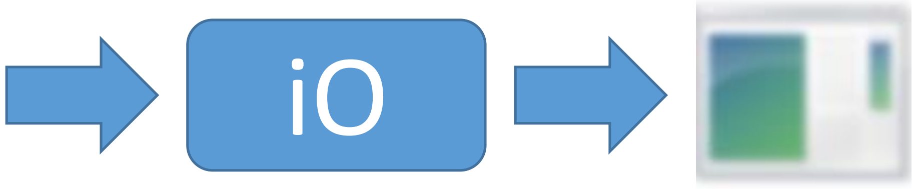
Private Puncturing from iO

Suppose PRF is puncturable (e.g., GGM)

- Master secret key: PRF key k
- PRF output at $x \in \mathcal{X}$: $\mathrm{PRF}(k, x)$

$P_z(x)$:

- If $x = z$, output r
- Else, output $\mathrm{PRF}(k, x)$

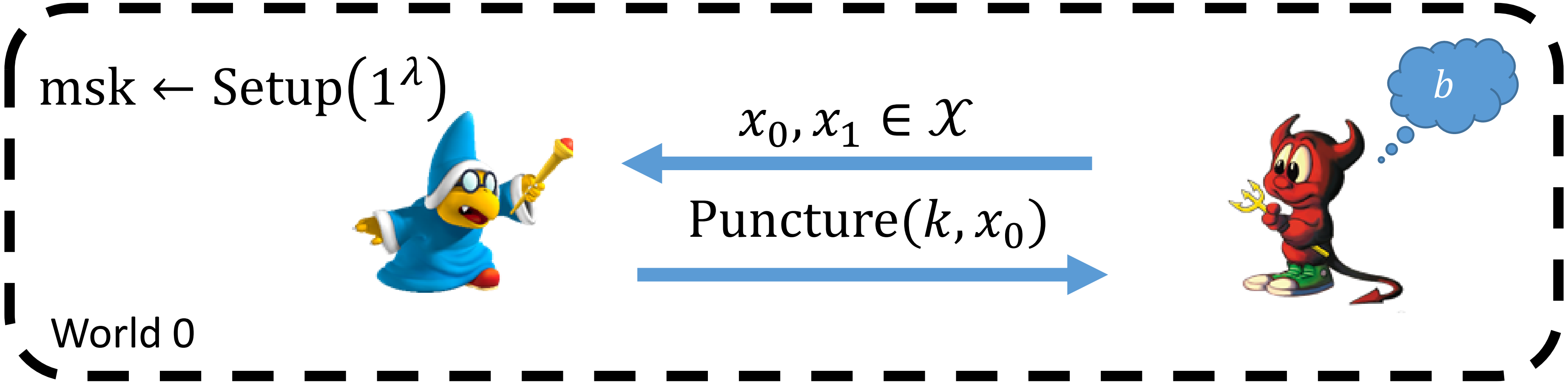


Punctured key for a point z is an obfuscated program

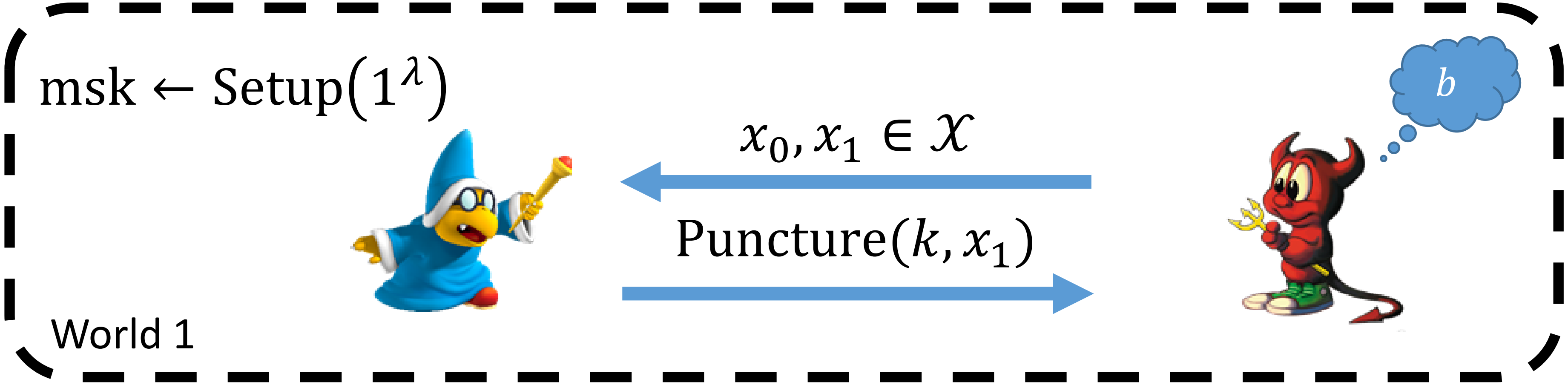
Constrained evaluation corresponds to evaluating obfuscated program

Private Puncturing from iO: Privacy

Recall privacy notion:



$$\approx_c$$



Private Puncturing from iO: Privacy

Proof is simple exercise in punctured programming

$P_z(x)$:

- If $x = z$, output $\boxed{r}$
- Else, output $\boxed{\mathrm{PRF}(k, x)}$

random value (hard coded)

real value of the PRF

Program for punctured PRF (punctured at z)

Private Puncturing from iO: Privacy

k_{x_0}: k punctured at x_0

iO (
$P_{x_0}(x)$:
- If $x = x_0$, output r
- Else, output PRF(k, x)
) $\approx_c$ iO (
$P'_{x_0}(x)$:
- If $x = x_0$, output r
- Else, output PRF(k_{x_0}, x)
)

Hybrid 0: Real game

Hybrid 1: Challenger responds to puncture query with iO of this program

Private Puncturing from iO: Privacy

Invoke puncturing security

$$\text{iO}\left(\begin{array}{l} P'_{x_0}(x): \\ \bullet\ \text{If } x = x_0\text{, output } r \\ \bullet\ \text{Else, output PRF}(k_{x_0}, x) \end{array}\right) \approx_c \text{iO}\left(\begin{array}{l} P''_{x_0}(x): \\ \bullet\ \text{If } x = x_0\text{, output PRF}(k, x_0) \\ \bullet\ \text{Else, output PRF}(k_{x_0}, x) \end{array}\right)$$

Hybrid 1

Hybrid 2

Private Puncturing from iO: Privacy

Invoke iO security

iO $\left(\begin{array}{l} P''_{x_0}(x): \\ \bullet \text{ If } x = x_0\text{, output PRF}(k, x_0) \\ \bullet \text{ Else, output PRF}(k_{x_0}, x) \end{array} \right) \approx_c$ iO $\left(\begin{array}{l} P'''_{x_0}(x): \\ \bullet \text{ Output PRF}(k, x) \end{array} \right)$

Hybrid 2 Hybrid 3

The program in Hybrid 3 is independent of x_0. Similar argument holds starting from $P_{x_1}(x)$.

Private Puncturing from iO: Summary

Use iO to hide the punctured point and output uniformly random value at punctured point

$P_z(x)$:

- If $x = z$, output r
- Else, output $\mathrm{PRF}(k, x)$

Private Circuit Constrained PRF from iO

Construction generalizes to circuit constraints, except random values now derived from another PRF

$P_C(x)$:

- If $C(x) = 0$, output $\boxed{\mathrm{PRF}(k', x)}$
- If $C(x) = 1$, output $\boxed{\mathrm{PRF}(k, x)}$

k' is independently sampled PRF key

"real" PRF value

Private Circuit Constrained PRF from iO

$P_C(x)$:

- If $C(x) = 0$, output PRF(k', x)
- If $C(x) = 1$, output PRF(k, x)

Recall intuitive requirements for private constrained PRF:

- **Security**: Values at constrained points independent of actual PRF value at those points
- **Privacy**: Values at constrained points are unguessable by the adversary

Private Circuit Constrained PRF from iO

$P_C(x)$:

- If $C(x) = 0$, output PRF(k', x)
- If $C(x) = 1$, output PRF(k, x)

Security proof similar to that for private puncturable PRF

Number of hybrids equal to number of points that differ across the two circuits, so sub-exponential hardness needed in general

Private Puncturing from Multilinear Maps

- Composite-order (ideal) multilinear maps* [BS04]
  - Fix composite modulus $N = pq$
  - Base group $\mathbb{G}_1$ and target group $\mathbb{G}_n$ (of order N) with canonical generators g_1 and g_n, respectively
  - Multilinear map operation:

$$e\left(g_1^{\alpha_1}, g_1^{\alpha_2}, \dots, g_1^{\alpha_n}\right) = g_n^{\alpha_1 \alpha_2 \cdots \alpha_n}$$

*For simplicity, we describe our construction using ideal multilinear maps. It is straightforward to translate our construction to use composite-order graded multilinear encodings [CLT13]

Private Puncturing from Multilinear Maps

- Composite-order (ideal) multilinear maps [BS04]
  - Let $\mathbb{G}_{1,p}$ be subgroup of order p of $\mathbb{G}_1$
  - <u>Subgroup decision assumption</u> [BGN05]: hard to distinguish random elements of the full group $\mathbb{G}_1$ from random elements of the subgroup $\mathbb{G}_{1,p}$

Private Puncturing from Multilinear Maps

Starting point: multilinear analog of Naor-Reingold [NR97, BW13]

master secret key:

$g_1^{\alpha_{1,0}}$	$g_1^{\alpha_{2,0}}$	$\cdots$	$g_1^{\alpha_{n,0}}$
$g_1^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$\cdots$	$g_1^{\alpha_{n,1}}$

collection of $2n$ <u>random</u> group elements from $\mathbb{G}_1$

Private Puncturing from Multilinear Maps

PRF evaluation via multilinear map

$g_1^{\alpha_{1,0}}$	$g_1^{\alpha_{2,0}}$	$g_1^{\alpha_{3,0}}$	$g_1^{\alpha_{4,0}}$	$g_1^{\alpha_{5,0}}$
$g_1^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$g_1^{\alpha_{3,1}}$	$g_1^{\alpha_{4,1}}$	$g_1^{\alpha_{5,1}}$

Private Puncturing from Multilinear Maps

PRF evaluation via multilinear map

$g_1^{\alpha_{1,0}}$	$g_1^{\alpha_{2,0}}$	$g_1^{\alpha_{3,0}}$	$g_1^{\alpha_{4,0}}$	$g_1^{\alpha_{5,0}}$
$g_1^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$g_1^{\alpha_{3,1}}$	$g_1^{\alpha_{4,1}}$	$g_1^{\alpha_{5,1}}$

$$F_k(01101) = e\left(g_1^{\alpha_{1,0}}, g_1^{\alpha_{2,1}}, g_1^{\alpha_{3,1}}, g_1^{\alpha_{4,0}}, g_1^{\alpha_{5,1}}\right)$$

Private Puncturing from Multilinear Maps

Puncture PRF by exploiting orthogonality

master secret key:

$g_{1,p}^{\alpha_{1,0}}$	$g_{1,p}^{\alpha_{2,0}}$	$g_{1,p}^{\alpha_{3,0}}$	$g_{1,p}^{\alpha_{4,0}}$	$g_{1,p}^{\alpha_{5,0}}$
$g_{1,p}^{\alpha_{1,1}}$	$g_{1,p}^{\alpha_{2,1}}$	$g_{1,p}^{\alpha_{3,1}}$	$g_{1,p}^{\alpha_{4,1}}$	$g_{1,p}^{\alpha_{5,1}}$

all elements in subgroup

puncture at 01101:

$g_1^{\alpha_{1,0}}$	$g_{1,p}^{\alpha_{2,0}}$	$g_{1,p}^{\alpha_{3,0}}$	$g_1^{\alpha_{4,0}}$	$g_{1,p}^{\alpha_{5,0}}$
$g_{1,p}^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$g_1^{\alpha_{3,1}}$	$g_{1,p}^{\alpha_{4,1}}$	$g_1^{\alpha_{5,1}}$

punctured components in full group

Private Puncturing from Multilinear Maps

Correctness

puncture at $x^* = 01101$:

$g_1^{\alpha_{1,0}}$	$g_{1,p}^{\alpha_{2,0}}$	$g_{1,p}^{\alpha_{3,0}}$	$g_1^{\alpha_{4,0}}$	$g_{1,p}^{\alpha_{5,0}}$
$g_{1,p}^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$g_1^{\alpha_{3,1}}$	$g_{1,p}^{\alpha_{4,1}}$	$g_1^{\alpha_{5,1}}$

Correctness by multilinearity (and CRT):

$$e\left(g_1^{\beta_1}, \dots, g_1^{\beta_n}\right) = e\left(g_{1,p}, \dots, g_{1,p}\right)^{\beta_1 \cdots \beta_n \pmod{p}} e\left(g_{1,q}, \dots, g_{1,q}\right)^{\beta_1 \cdots \beta_n \pmod{q}}$$

For all $x \neq x^*$, there is some i where $x_i \neq x_i^*$ so $\beta_{i,x_i^*} = 0 \pmod{q}$ where $\left(g^{\beta_{i,0}}, g^{\beta_{i,1}}\right)$ is the i^{th} component of the secret key

Private Puncturing from Multilinear Maps

Privacy

puncture at $x^* = 01101$:

$g_1^{\alpha_{1,0}}$	$g_{1,p}^{\alpha_{2,0}}$	$g_{1,p}^{\alpha_{3,0}}$	$g_1^{\alpha_{4,0}}$	$g_{1,p}^{\alpha_{5,0}}$
$g_{1,p}^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$g_1^{\alpha_{3,1}}$	$g_{1,p}^{\alpha_{4,1}}$	$g_1^{\alpha_{5,1}}$

Follows directly by subgroup decision: elements of $\mathbb{G}_1$ look indistinguishable from elements of $\mathbb{G}_{1,p}$

Private Puncturing from Multilinear Maps

Puncturing Security

puncture at $x^* = 01101$:

$g_1^{\alpha_{1,0}}$	$g_{1,p}^{\alpha_{2,0}}$	$g_{1,p}^{\alpha_{3,0}}$	$g_1^{\alpha_{4,0}}$	$g_{1,p}^{\alpha_{5,0}}$
$g_{1,p}^{\alpha_{1,1}}$	$g_1^{\alpha_{2,1}}$	$g_1^{\alpha_{3,1}}$	$g_{1,p}^{\alpha_{4,1}}$	$g_1^{\alpha_{5,1}}$

Follows from a multilinear Diffie-Hellman subgroup decision assumption on composite-order multilinear maps

See paper for details!

Private Puncturing and Distributed Point Functions [GI14]

$$\begin{tikzcd}[column sep=small] & \mathrm{Gen}(1^\lambda, x) \arrow[dl] \arrow[dr] & \\ k_0 & & k_1 \end{tikzcd}$$

Privacy: k_0 and k_1 individually must hide x

Correctness: k_0 and k_1 implement a point function

$$\mathrm{Eval}(k_0, x') \oplus \mathrm{Eval}(k_1, x') = \begin{cases} 1, & x' = x \\ 0, & x' \neq x \end{cases}$$

Private Puncturing and Distributed Point Functions [GI14]

A private puncturable PRF can be used to build a distributed point function (DPF):

$$\mathrm{Gen}(1^\lambda, x)$$

$$k_0 \leftarrow \mathrm{Setup}(1^\lambda) \qquad\qquad k_1 \leftarrow \mathrm{Puncture}(k_0, x)$$

Correctness: $\mathrm{Eval}(k_0,\cdot)$ and $\mathrm{ConstrainEval}(k_1,\cdot)$ agree everywhere except x

Privacy: k_0 is independent of x and k_1 hides x

Private Puncturing and Distributed Point Functions [GI14]

However, distributed point functions do not give a private puncturable PRF

Key difference:

- In a DPF, the point x is known at <u>setup</u> time: both k_0 and k_1 are generated together
- In a private puncturable PRF, the point x is known <u>after</u> the master parameters (the key k_0) are generated

Open question: Can constructions of DPFs be adapted to obtain a private puncturable PRF?

Conclusions

- New notion of <u>private</u> constrained PRFs
- Simple definitions, but require powerful tools to construct: iO / multilinear maps
- Private constrained PRFs immediately provide natural solutions to many problems

Open Questions

- Puncturable PRFs (and DPFs) can be constructed from OWFs
  - Can we construct private puncturable PRFs from OWFs?
  - Does private puncturing necessitate strong assumptions like multilinear maps?
  - Can we construct private circuit-constrained PRFs without requiring sub-exponentially hard iO?
- Most of our candidate applications just require private puncturable PRFs
  - New applications for more expressive families of constraints?

Part II: Practical Order-Revealing Encryption with Limited Leakage

Joint work with Nathan Chenette, Kevin Lewi, and Stephen A. Weis

Order-Revealing Encryption [BLRSZZ15]

sk

$$\begin{aligned} \mathrm{ct}_1 &= \mathrm{Enc}(\mathrm{sk}, 123) \\ \mathrm{ct}_2 &= \mathrm{Enc}(\mathrm{sk}, 512) \\ \mathrm{ct}_3 &= \mathrm{Enc}(\mathrm{sk}, 273) \end{aligned}$$

Client

Server

secret-key encryption scheme

Order-Revealing Encryption [BLRSZZ15]

$$\begin{aligned} ct_1 &= \mathrm{Enc}(sk, 123) \\ ct_2 &= \mathrm{Enc}(sk, 512) \\ ct_3 &= \mathrm{Enc}(sk, 273) \end{aligned}$$

Which is greater: the value encrypted by ct_1 or the value encrypted by ct_2?

Server

Application: range queries / binary search on encrypted data

Order-Revealing Encryption [BLRSZZ15]

given any two ciphertexts

$\mathrm{ct}_1 = \mathrm{Enc}(\mathrm{sk}, x)$

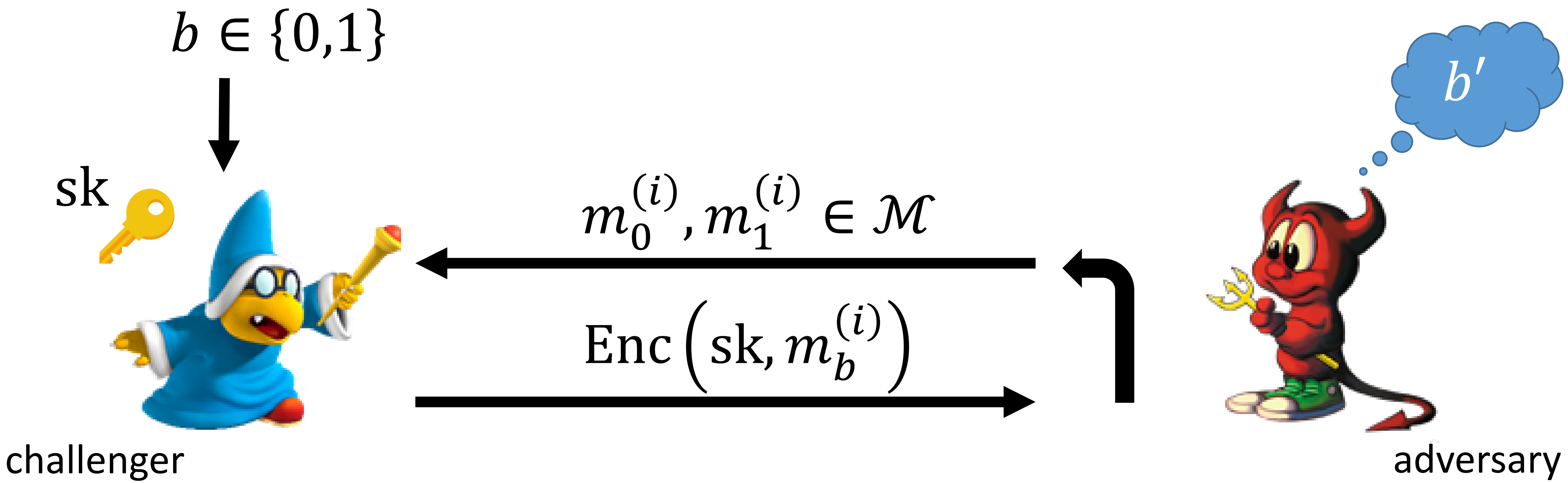
$\mathrm{ct}_2 = \mathrm{Enc}(\mathrm{sk}, y)$

there is a <u>publically</u> evaluatable function that evaluates the comparison function

$x > y$

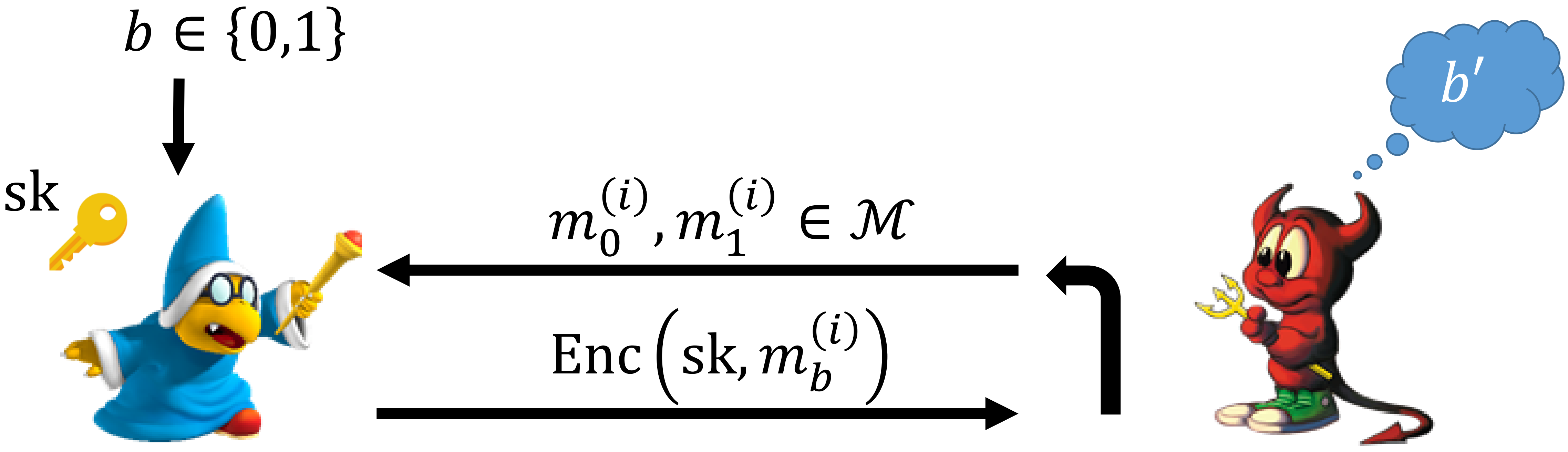
Defining Security

Starting point: semantic security (IND-CPA) [GM84]



semantic security: adversary cannot guess b (except with probability negligibly close to 1/2)

Best-Possible Security [BCLO09]



must impose restriction on messages: otherwise trivial to break semantic security using comparison operator

Best-Possible Security [BCLO09]

$b \in \{0,1\}$

sk

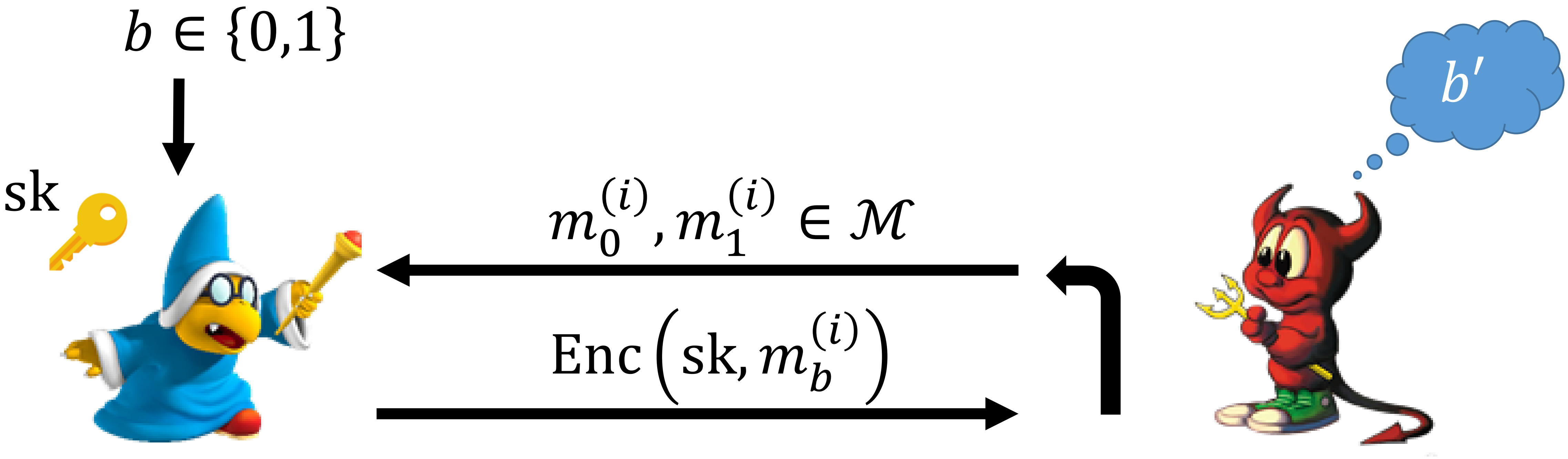
$m_0^{(i)}, m_1^{(i)} \in \mathcal{M}$

$\mathrm{Enc}\left(\mathrm{sk}, m_b^{(i)}\right)$

b'

$$\forall i,j: m_0^{(i)} < m_0^{(j)} \Leftrightarrow m_1^{(i)} < m_1^{(j)}$$

Best-Possible Security [BCLO09]



order of "left" set of messages same as order of "right" set of messages

Existing Approaches

General-Purpose Multi-Input Functional Encryption [GGGJKLSSZ14, BV15, AJ15]

- Powerful cryptographic primitive that fully subsumes ORE
- Achieves best-possible security
- Impractical (requires obfuscating a PRF)

Existing Approaches

Multilinear-map-based Solution [BLRSZZ15]

- Much more efficient than general purpose indistinguishability obfuscation
- Achieves best-possible security
- Security of multilinear maps not well-understood
- Still quite inefficient (e.g., ciphertexts on the order of GB)

Existing Approaches

Order-preserving encryption (OPE) [BCLO09, BCO11]:

- Comparison operation is <u>direct</u> comparison of ciphertexts:

$$x > y \Leftrightarrow \mathrm{Enc}(\mathrm{sk}, x) > \mathrm{Enc}(\mathrm{sk}, y)$$

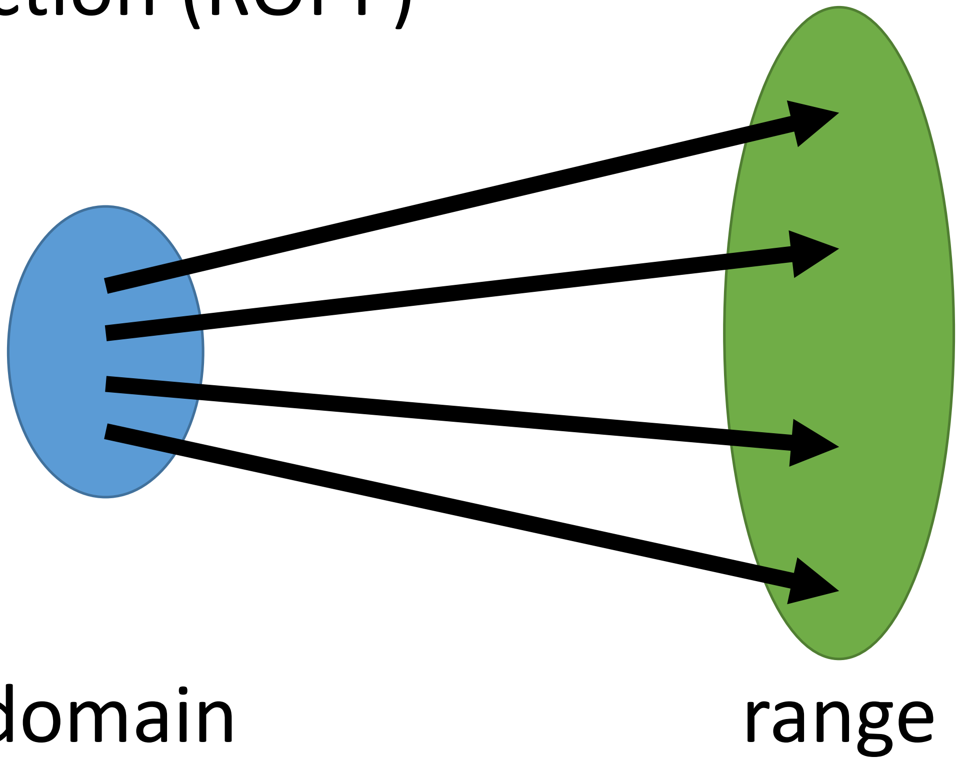
- Lower bound: no OPE scheme can satisfy "best-possible" security unless the size of the ciphertext space is <u>exponential</u> in the size of the plaintext space

Existing Approaches

Order-preserving encryption (OPE) [BCLO09, BCO11]:

- No "best-possible" security, so instead, compare with random order-preserving function (ROPF)

encryption function implements a random order-preserving function



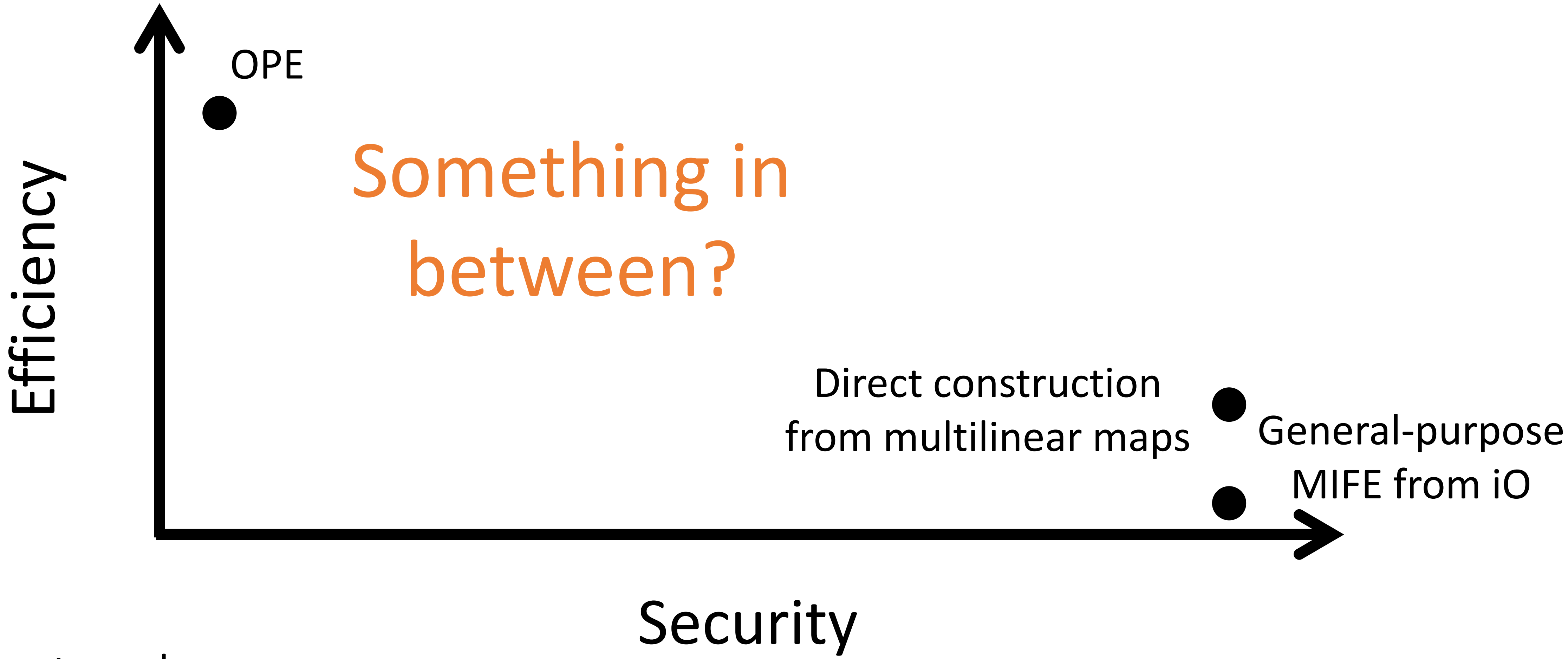
Existing Approaches

Properties of a random order-preserving function [BCO'11]:

- Each ciphertext roughly leaks half of the most significant bits
- Each pair of ciphertexts roughly leaks half of the most significant bits of their difference

No semantic security for
even a single message!

Existing Approaches



Not drawn to scale

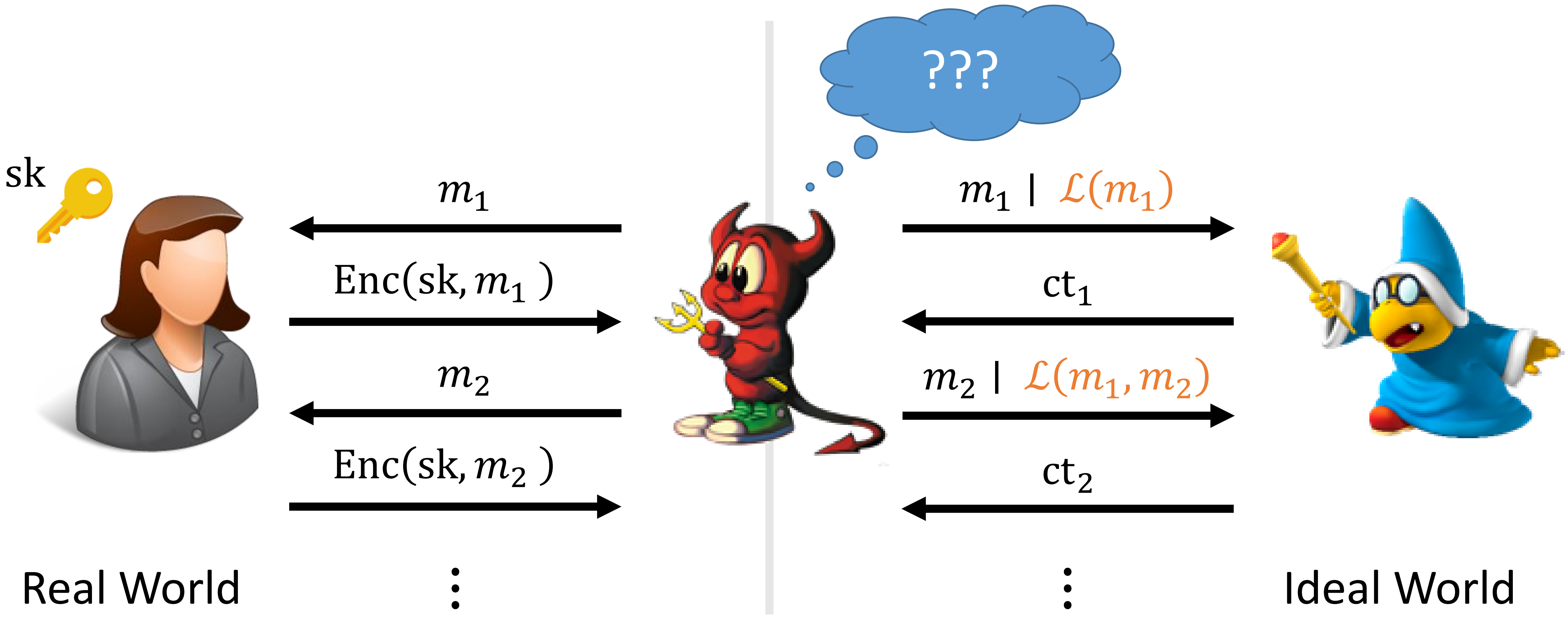
A New Security Notion

Two existing security notions:

- IND-OCPA: strong security, but hard to achieve efficiently
- ROPF-CCA: efficiently constructible, but lots of leakage, and difficult to precisely quantify the leakage

A New Security Notion: SIM-ORE

Idea: augment "best-possible" security with a leakage function $\mathcal{L}$



A New Security Notion: SIM-ORE

Similar to SSE definitions [CM05, CGKO06]

Leakage functions specifies exactly what is leaked

"Best-possible" simulation security:

$$\mathcal{L}(m_1, \ldots, m_q) = \{\, \mathbf{1}\{m_i < m_j\} \mid 1 \leq i < j \leq q \,\}$$

A New Security Notion: SIM-ORE

"Best-possible" simulation security:

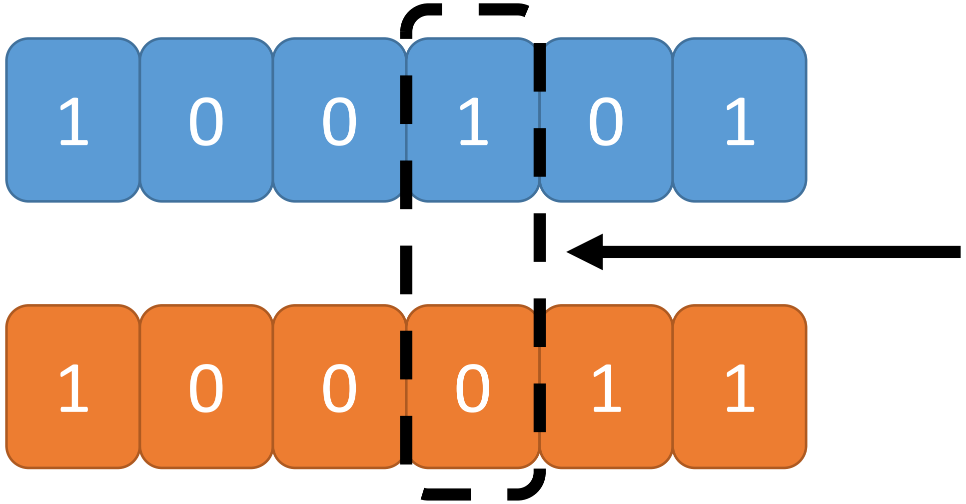
$$\mathcal{L}(m_1, \ldots, m_q) = \{\, 1\{m_i < m_j\} \mid 1 \leq i < j \leq q \,\}$$

Anything that can be computed given the ciphertexts can be computed given the ordering on the messages

Our Construction

Leak a little more than just the ordering:

$$\mathcal{L}(m_1, \dots, m_q) = \left\{ \left(1\{m_i < m_j\}, \mathrm{ind}_{\mathrm{diff}}(m_i, m_j) \right) \,\middle|\, 1 \le i < j \le q \right\}$$



$\mathrm{ind}_{\mathrm{diff}}(m_1, m_2)$: index of first bit that differs

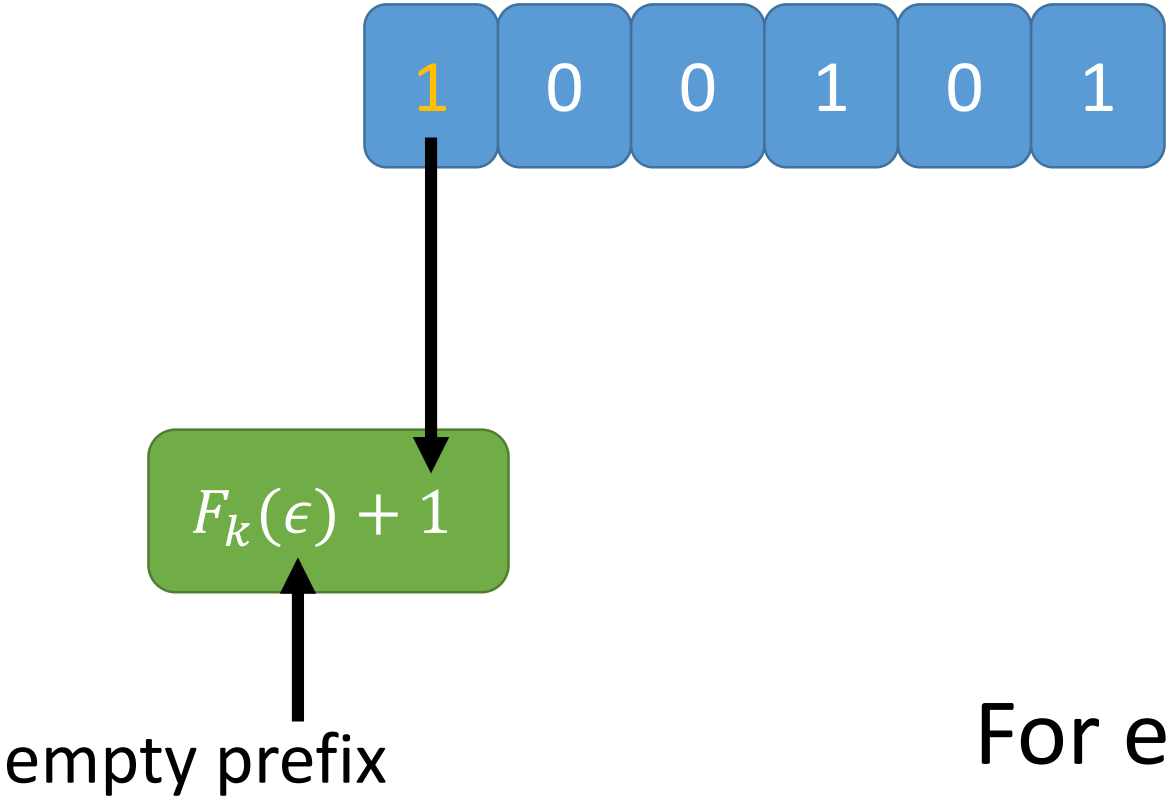
Our Construction

1	0	0	1	0	1

$F\colon \mathcal{K} \times \{0,1\}^* \to \mathbb{Z}_3$

For each index i, apply a PRF to the first $i-1$ bits, then add $b_i \pmod{n}$

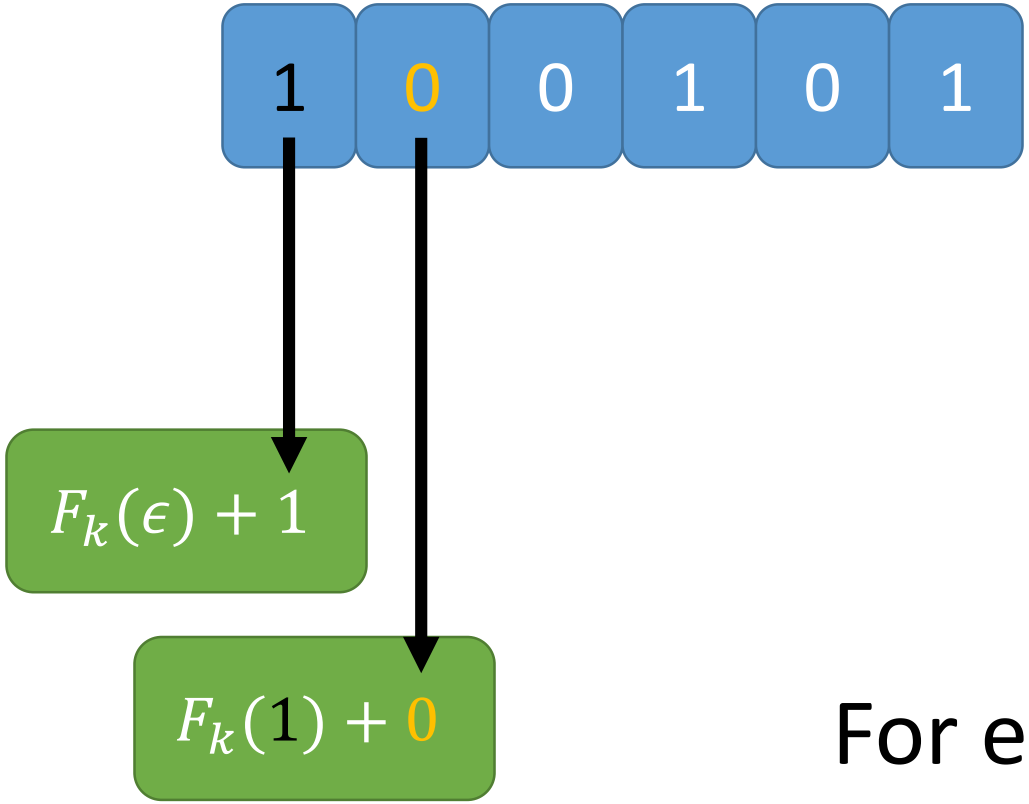
Our Construction



For each index i, apply a PRF to the first $i - 1$ bits, then add $b_i \pmod{n}$

$F: \mathcal{K} \times \{0,1\}^* \rightarrow \mathbb{Z}_3$

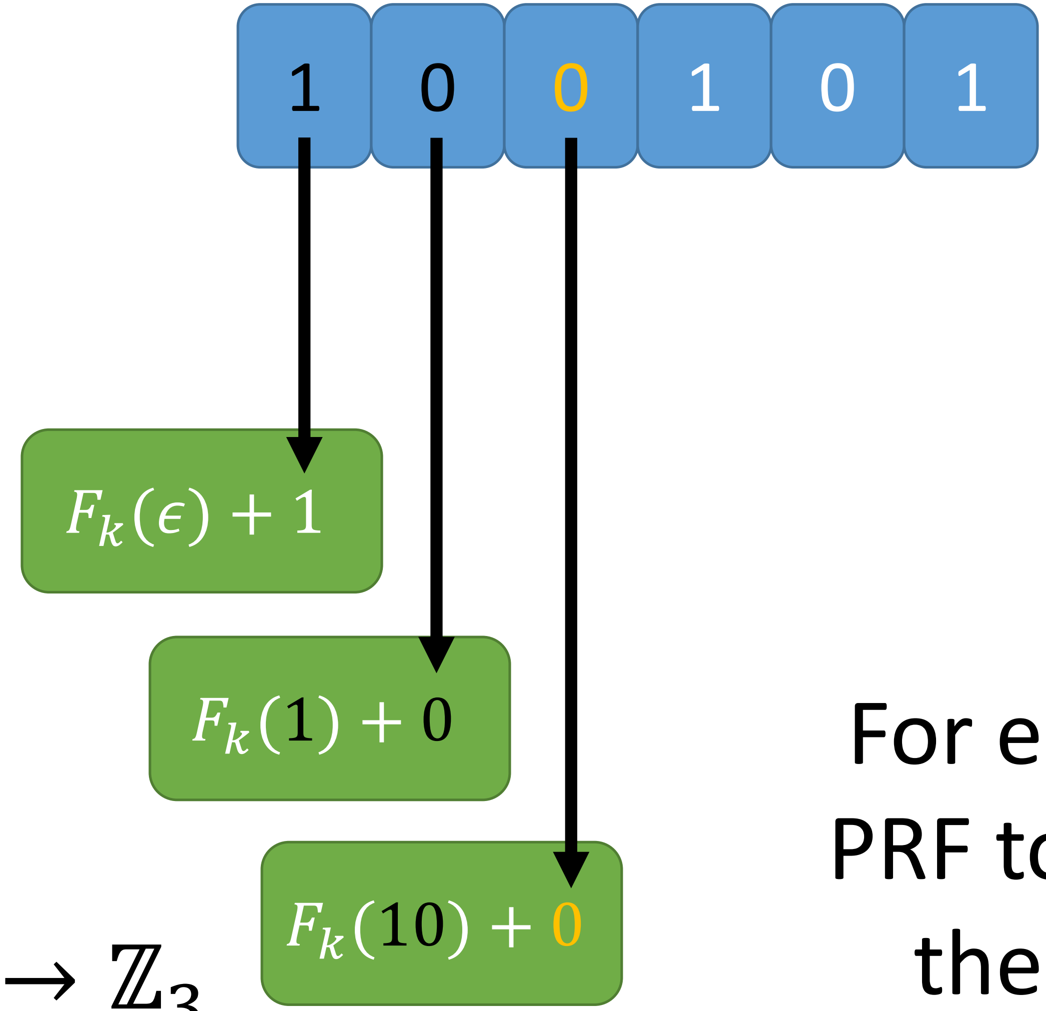
Our Construction



For each index i, apply a PRF to the first $i - 1$ bits, then add $b_i \pmod n$

$F\colon \mathcal{K} \times \{0,1\}^* \to \mathbb{Z}_3$

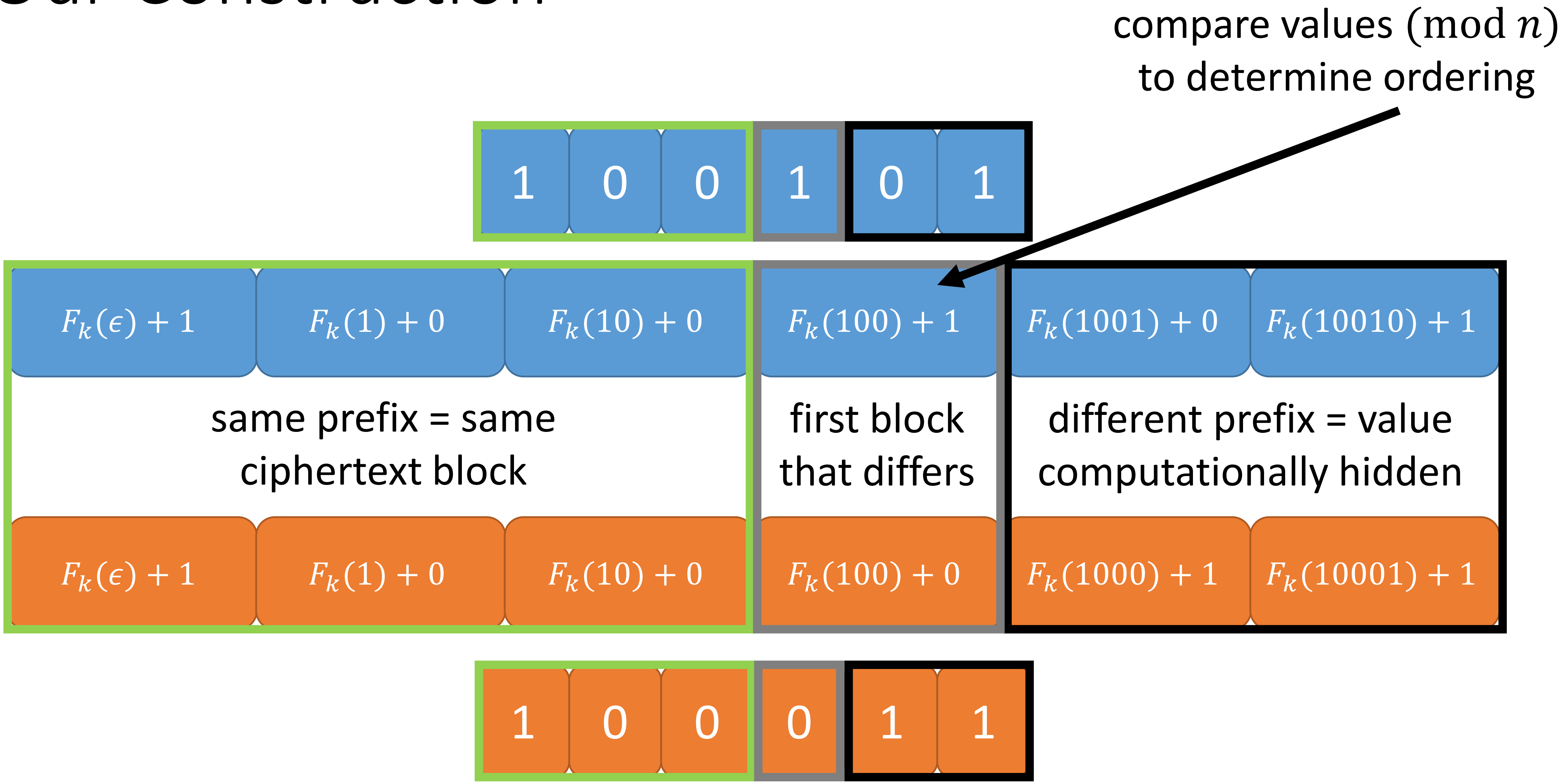
Our Construction



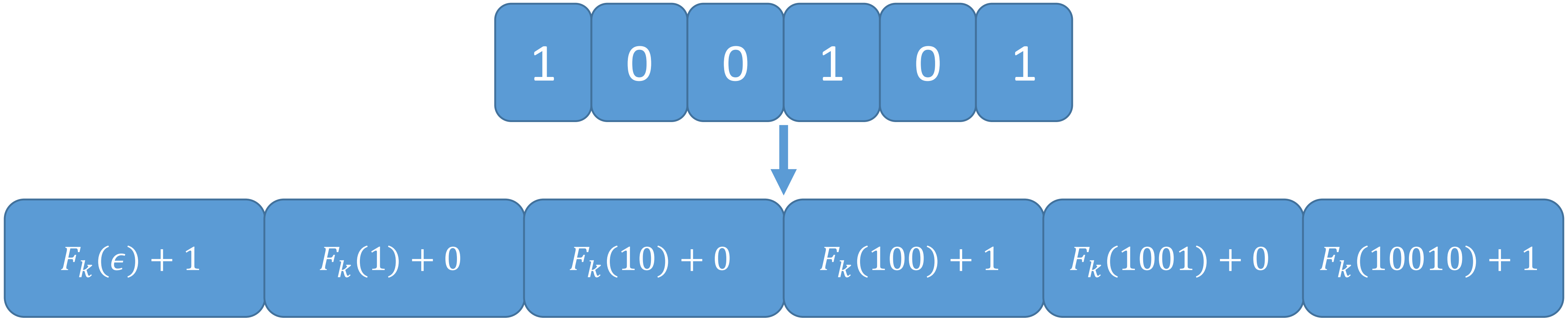
$F\colon \mathcal{K} \times \{0,1\}^* \to \mathbb{Z}_3$

For each index i, apply a PRF to the first $i - 1$ bits, then add $b_i \pmod{n}$

Our Construction



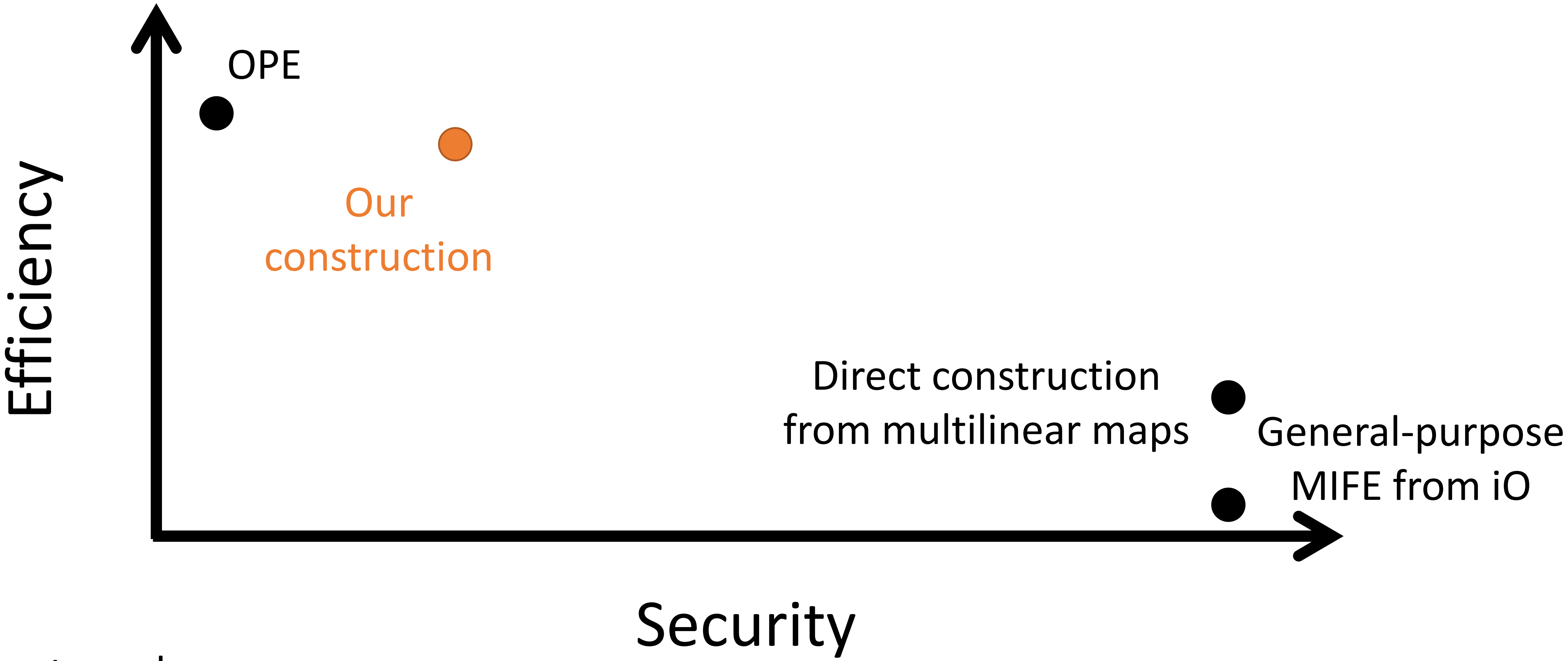
Our Construction: Security



Security follows directly from security of the PRF

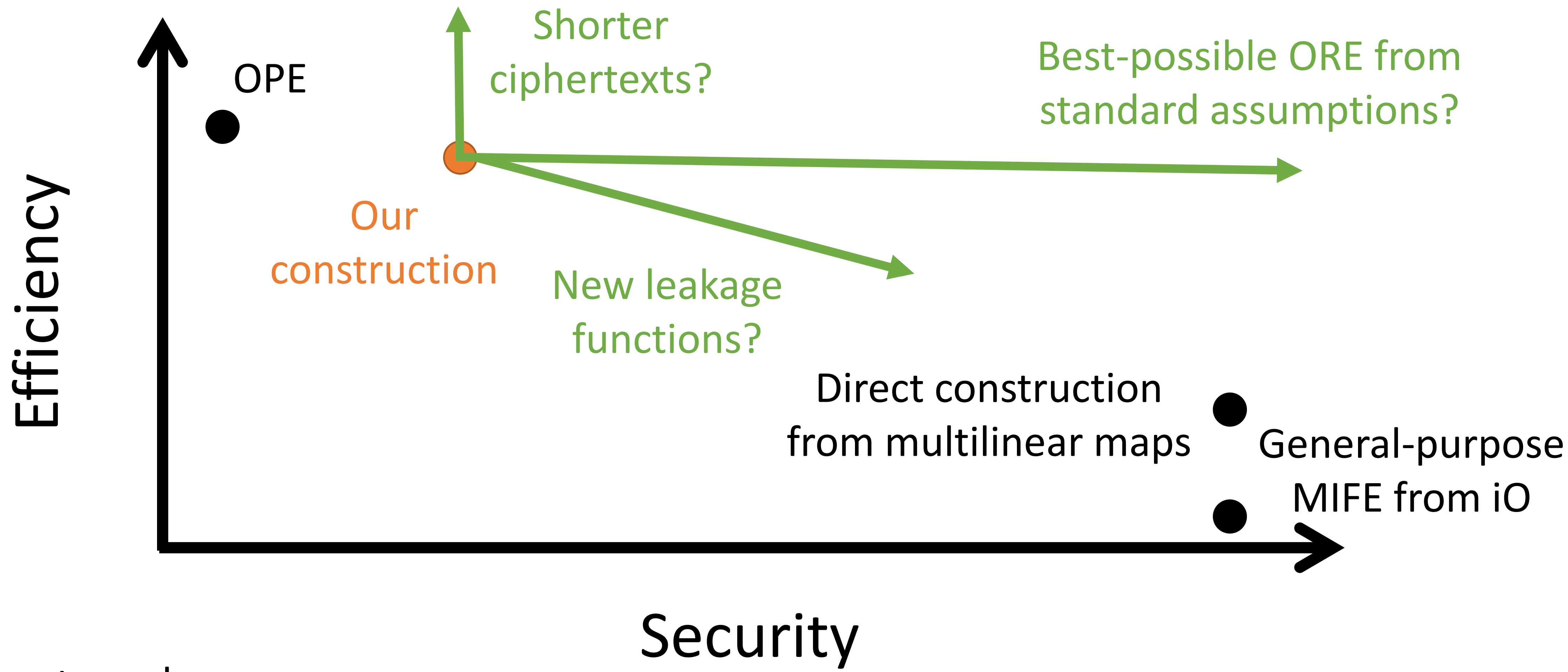
<u>Proof sketch</u>. Simulator responds to encryption queries using random strings. Maintains consistency using leakage information (first bit that differs).

The Landscape of OPE/ORE



Not drawn to scale

Directions for Future Research



Not drawn to scale

Questions?