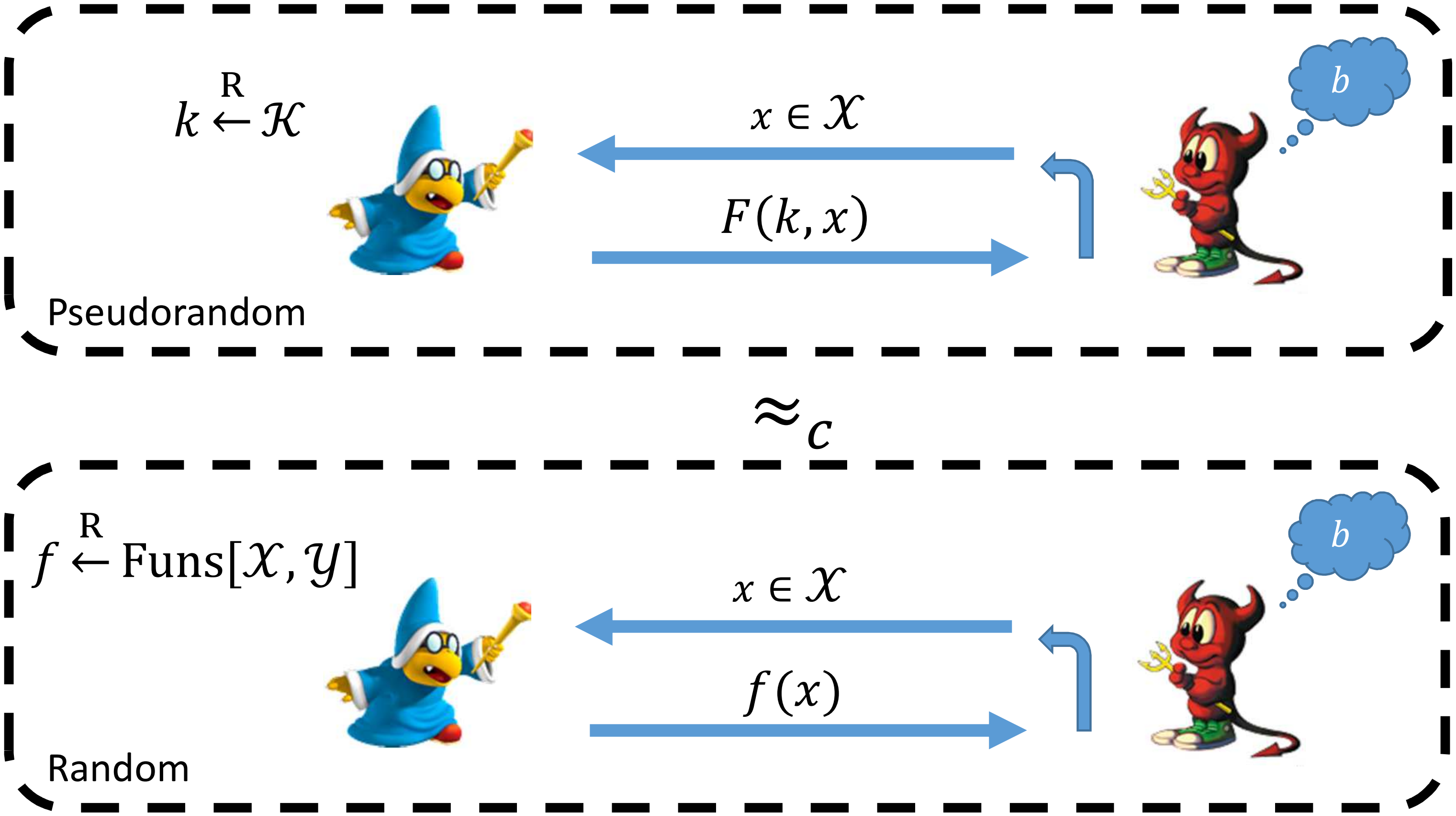
Constraining Pseudorandom Functions Privately

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

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

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

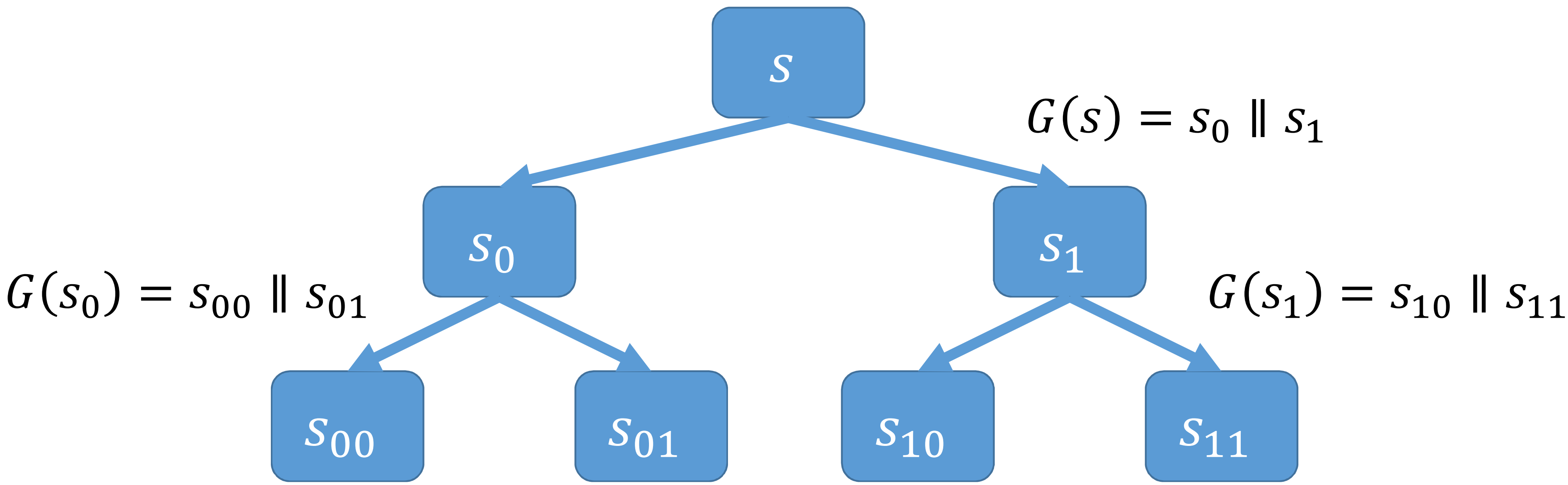
Constrained PRFs [BW13, BGI13, KPTZ13]

Many applications:

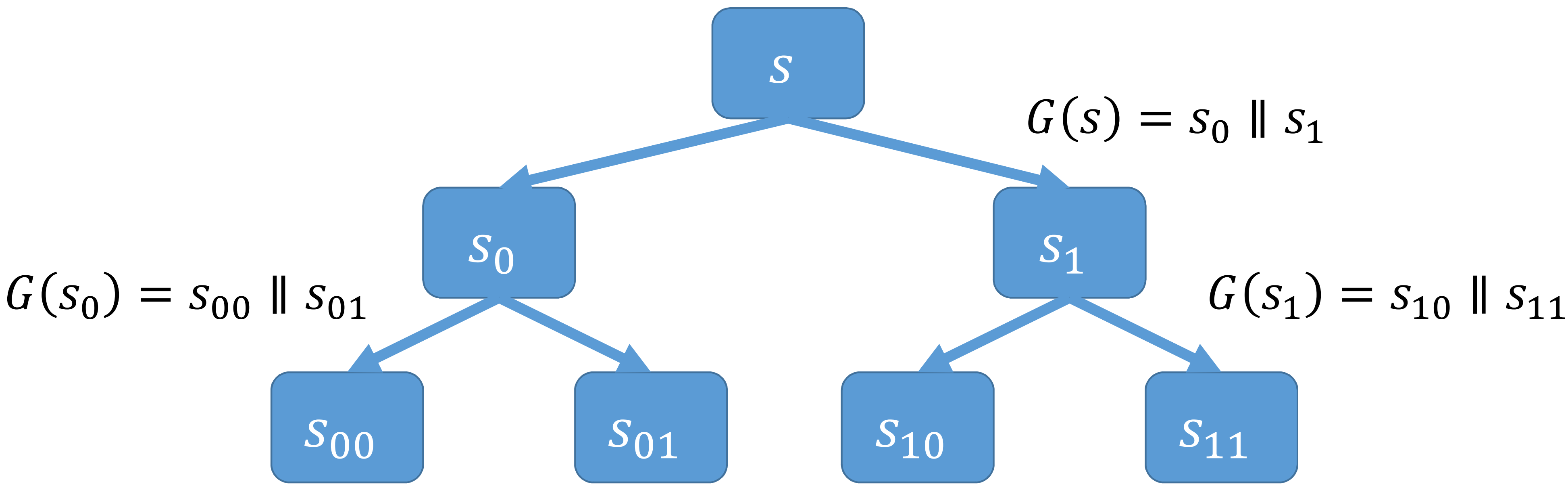
- Identity-Based Key Exchange, Optimal Broadcast Encryption [BW13]
- Punctured Programming Paradigm [SW14]
- Multiparty Key Exchange, Traitor Tracing [BZ14]

Puncturable PRFs from GGM

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

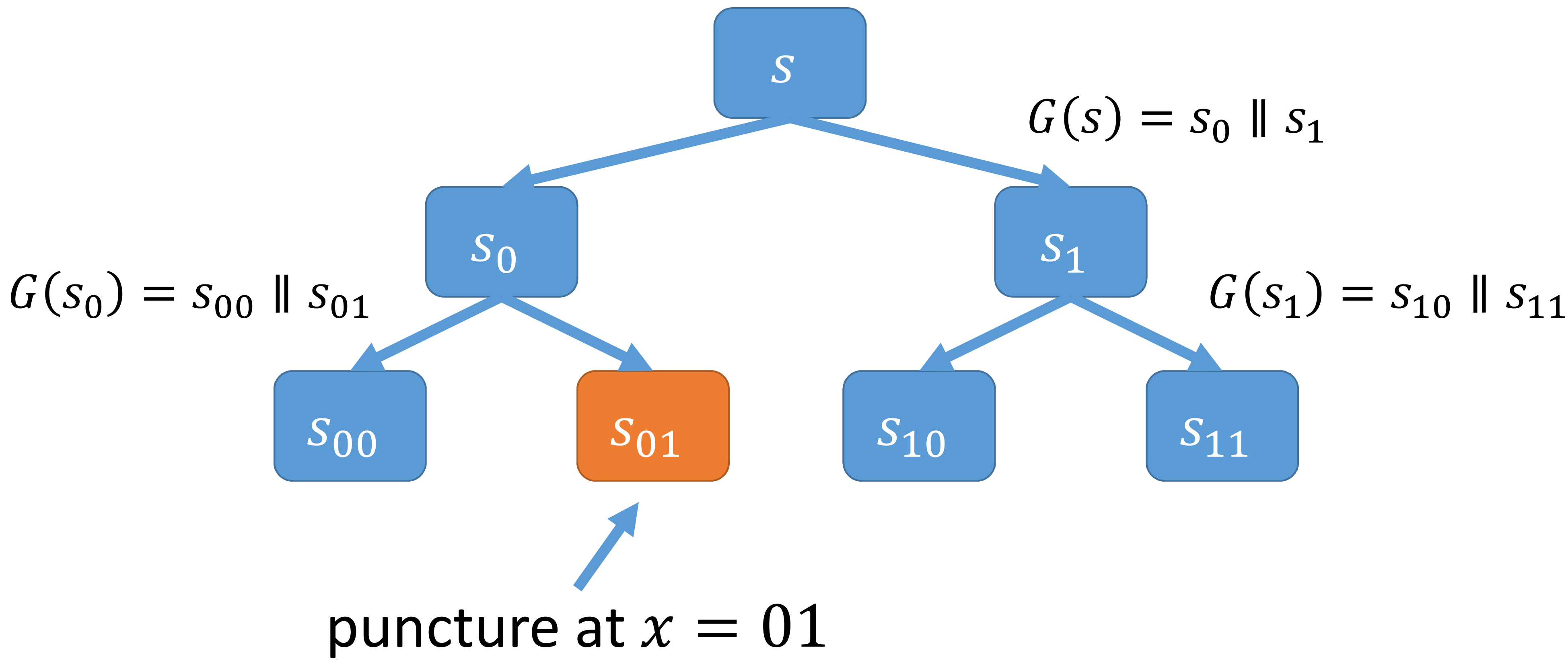


Puncturable PRFs from GGM

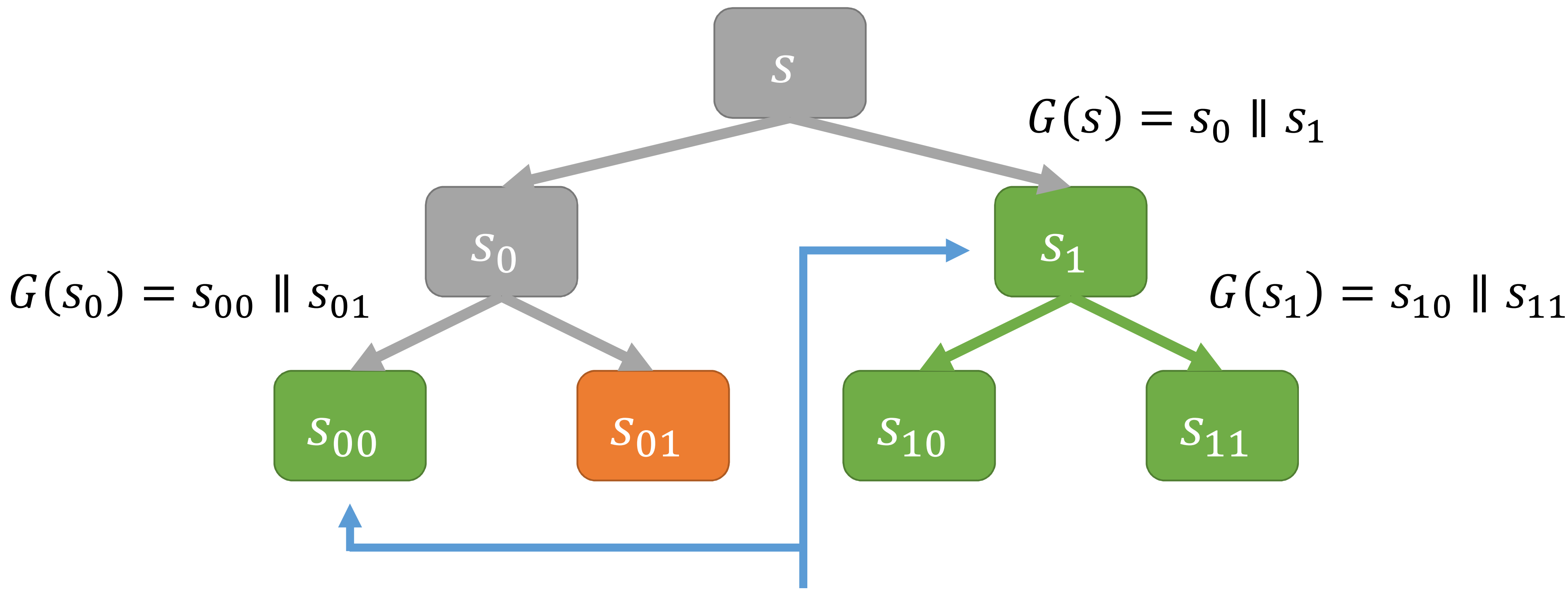


given root key s, can evaluate PRF everywhere

Puncturable PRFs from GGM

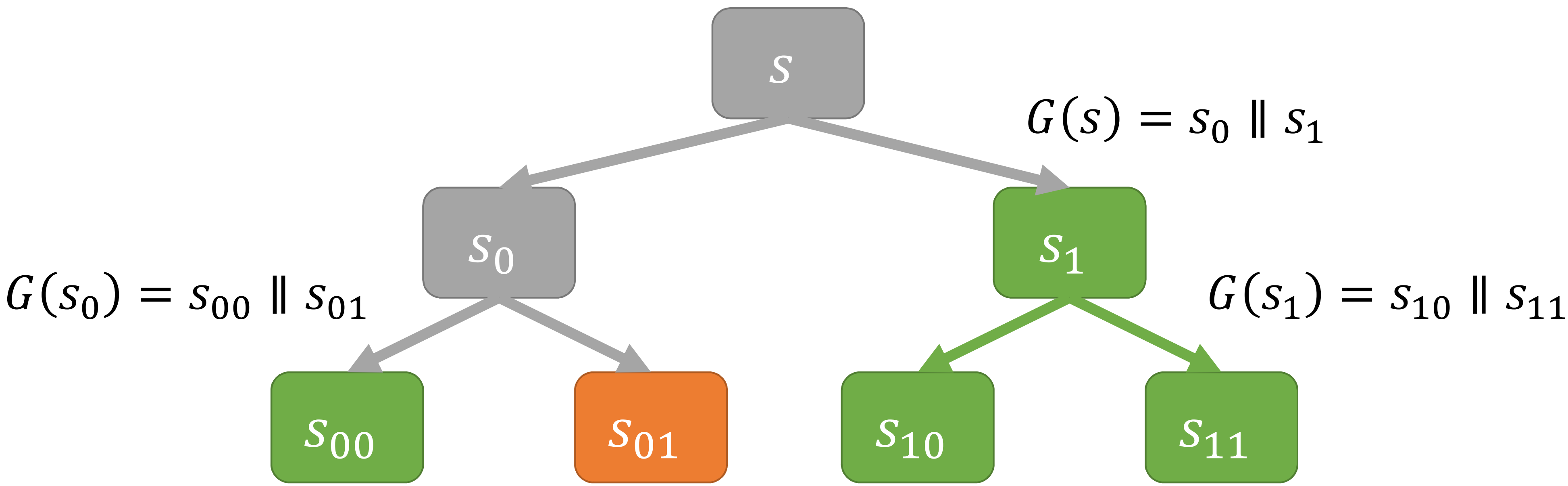


Puncturable PRFs from GGM



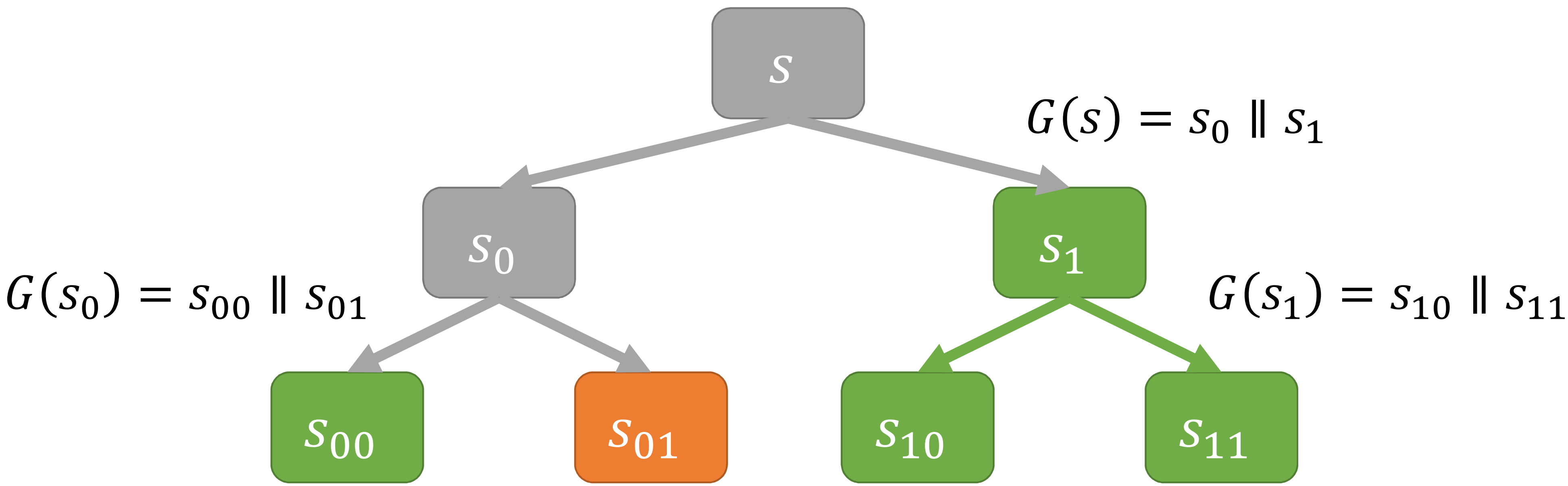
these two values suffice to evaluate at all other points

Puncturable PRFs from GGM



in general, punctured key consists of n nodes if domain of PRF is $\{0,1\}^n$

Puncturable PRFs from GGM

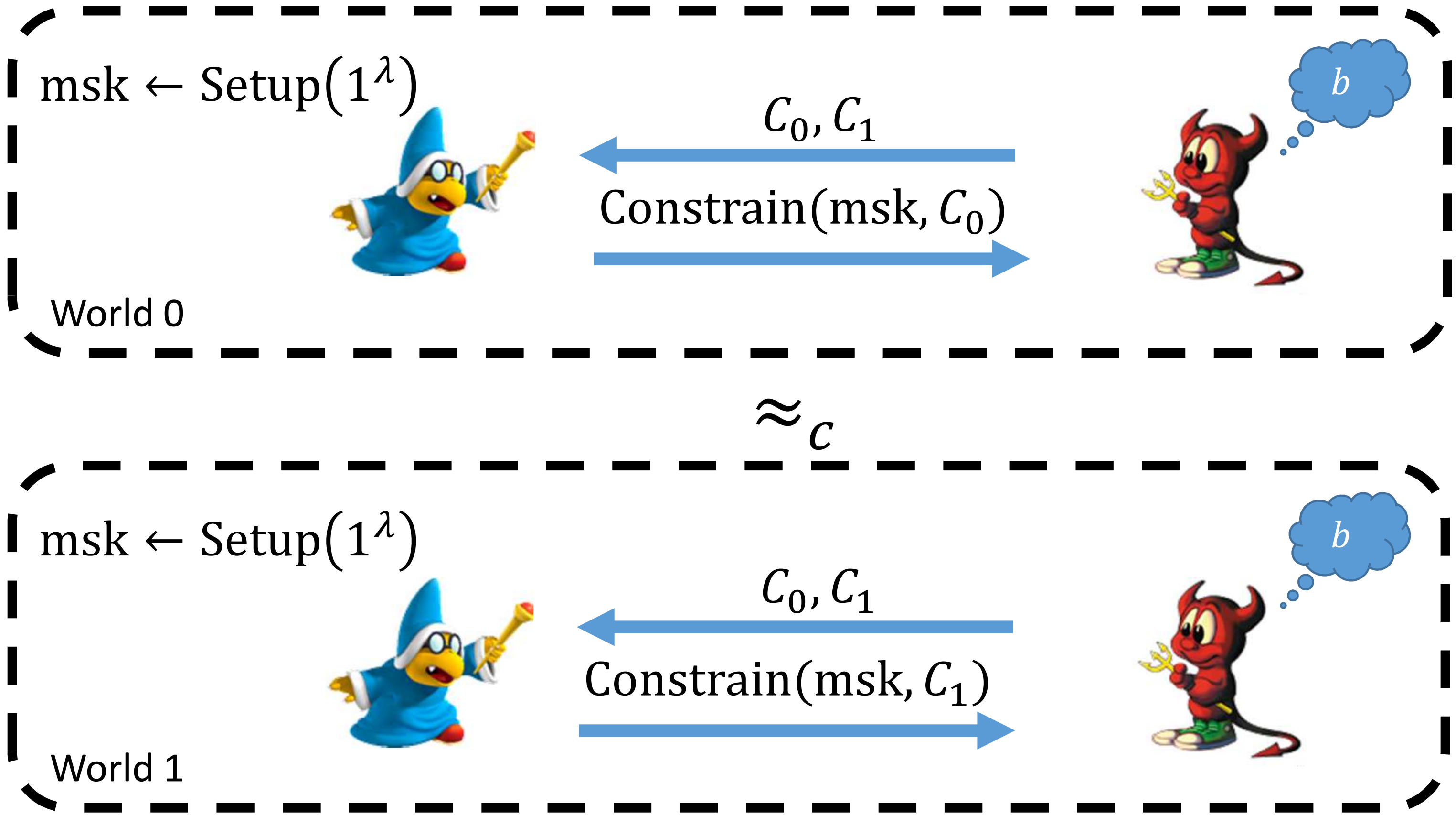


given s_1 and s_{00}, easy to tell that 01 is the punctured point

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

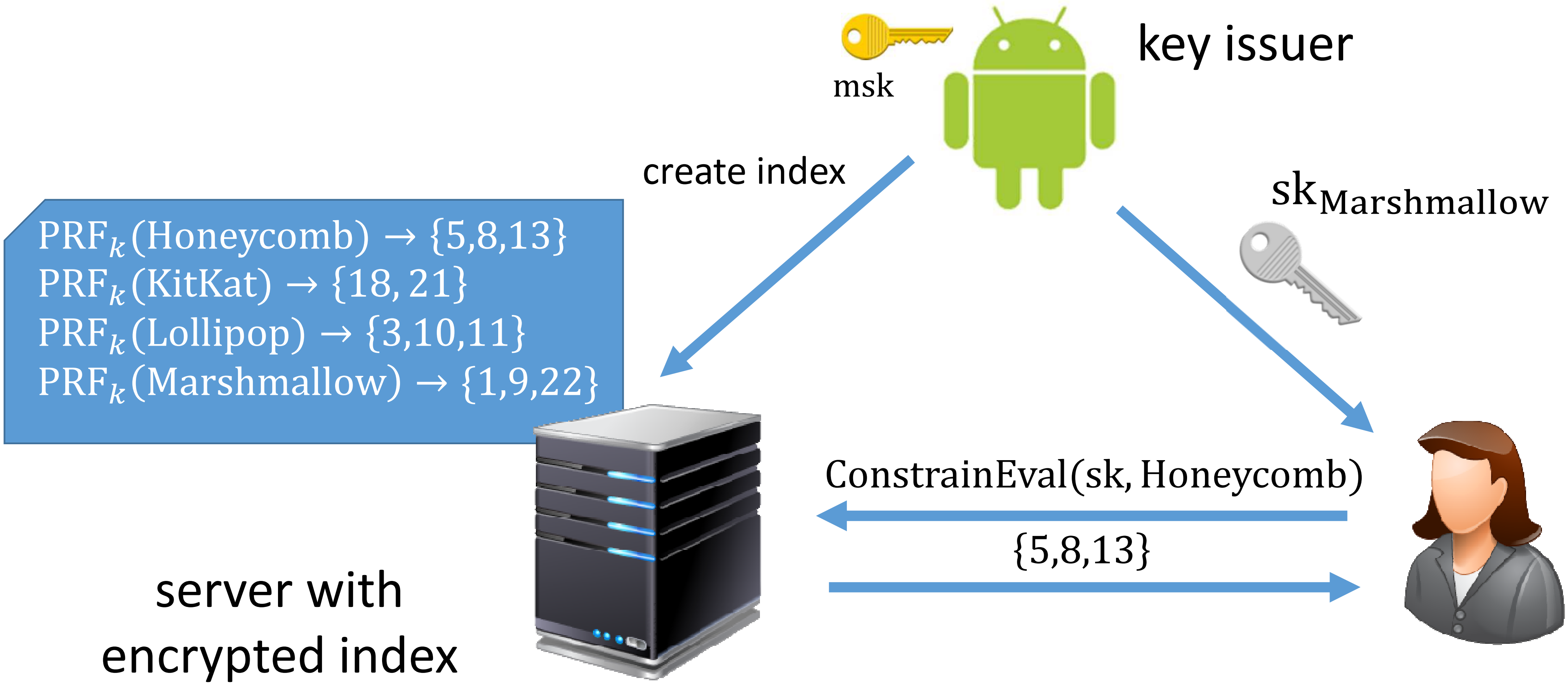
- **Correctness:** constrained evaluation at $x \neq z$ yields $F(k, x)$
- **Security:** $F(k, z)$ is indistinguishable from random
- **Privacy:** constrained key hides z

Implications of Privacy

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

- **<u>Security</u>**: Independent of $\text{Eval}(\text{msk}, z)$
- **<u>Privacy</u>**: Unguessable by the adversary

Using Privacy: Restricted Keyword Search



Using Privacy: Restricted Keyword Search

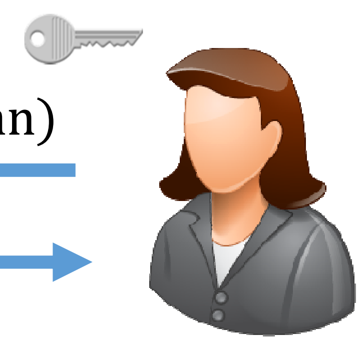
search for non-existent keyword

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

server with encrypted index

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

No results



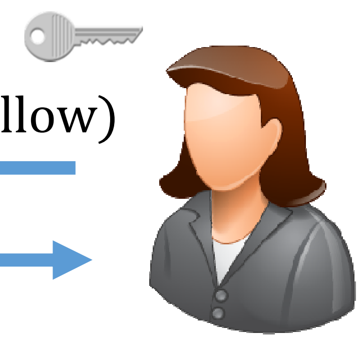
Using Privacy: Restricted Keyword Search

search for "restricted" keyword

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

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

No results



server with encrypted index

Using Privacy: Restricted Keyword Search

- **<u>Security</u>**: $\text{ConstrainEval}(\text{sk}, \text{Marshmallow}) \neq \text{Eval}(\text{msk}, \text{Marshmallow})$
- **<u>Privacy</u>**: 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

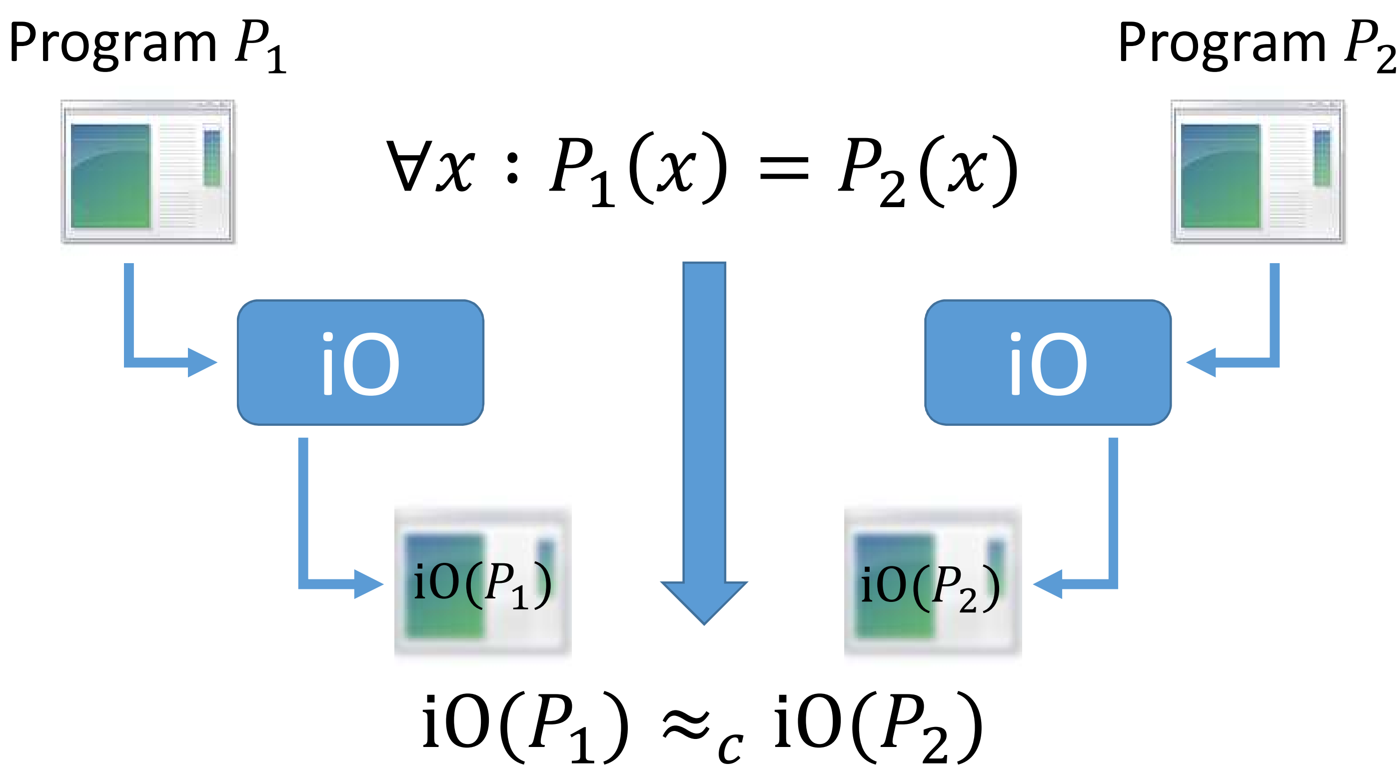
This talk

- From <u>concrete</u> assumptions on multilinear maps:
  - Private puncturable PRFs from subgroup hiding assumptions
  - Private bit-fixing PRF from multilinear Diffie-Hellman assumption

See paper

Constructing Private Constrained PRFs

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



Indistinguishability Obfuscation (iO)

- First introduced by Barak et al. [BGI^{+}01]
- First construction from multilinear maps [GGH^{+}13]
  - Subsequent constructions from multilinear maps [BR13, BGK^{+}14, AGIS14, Zim14, AB15, …]
  - Constructions also from (compact) functional encryption [AJ15, AJS15]

Indistinguishability Obfuscation (iO)

Many applications – "crypto complete"

- Functional encryption [GGH^{+}13]
- Deniable encryption [SW13]
- Witness encryption [GGSW13]
- Private broadcast encryption [BZ14]
- Traitor tracing [BZ14]
- Multiparty key exchange [BZ14]
- Multiparty computation [GGHR14]
- and more…

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

$P_z(x)$:

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

random value (hard coded)

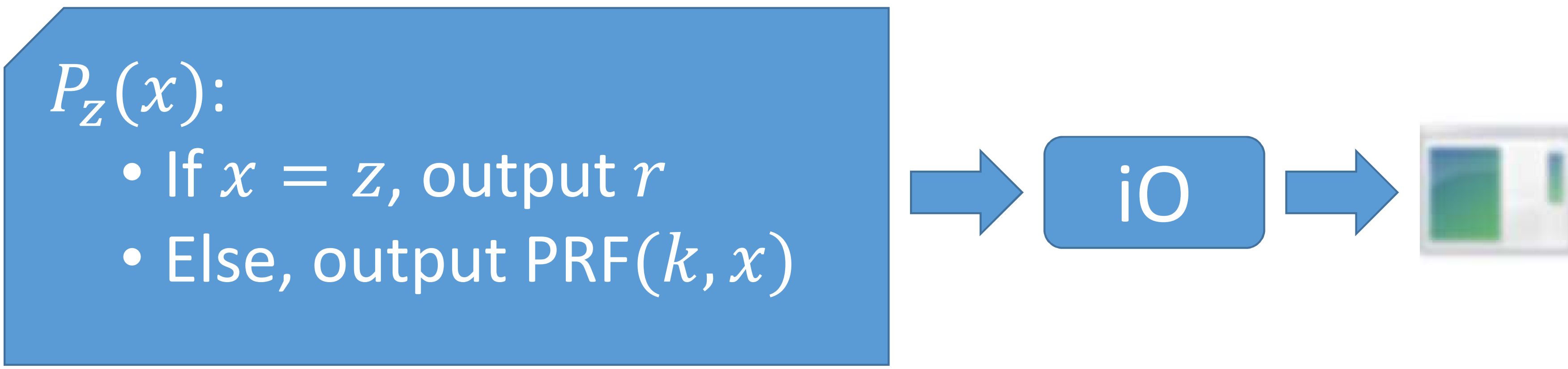
real value of the PRF

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 $x \in \mathcal{X}$: $\text{PRF}(k, x)$

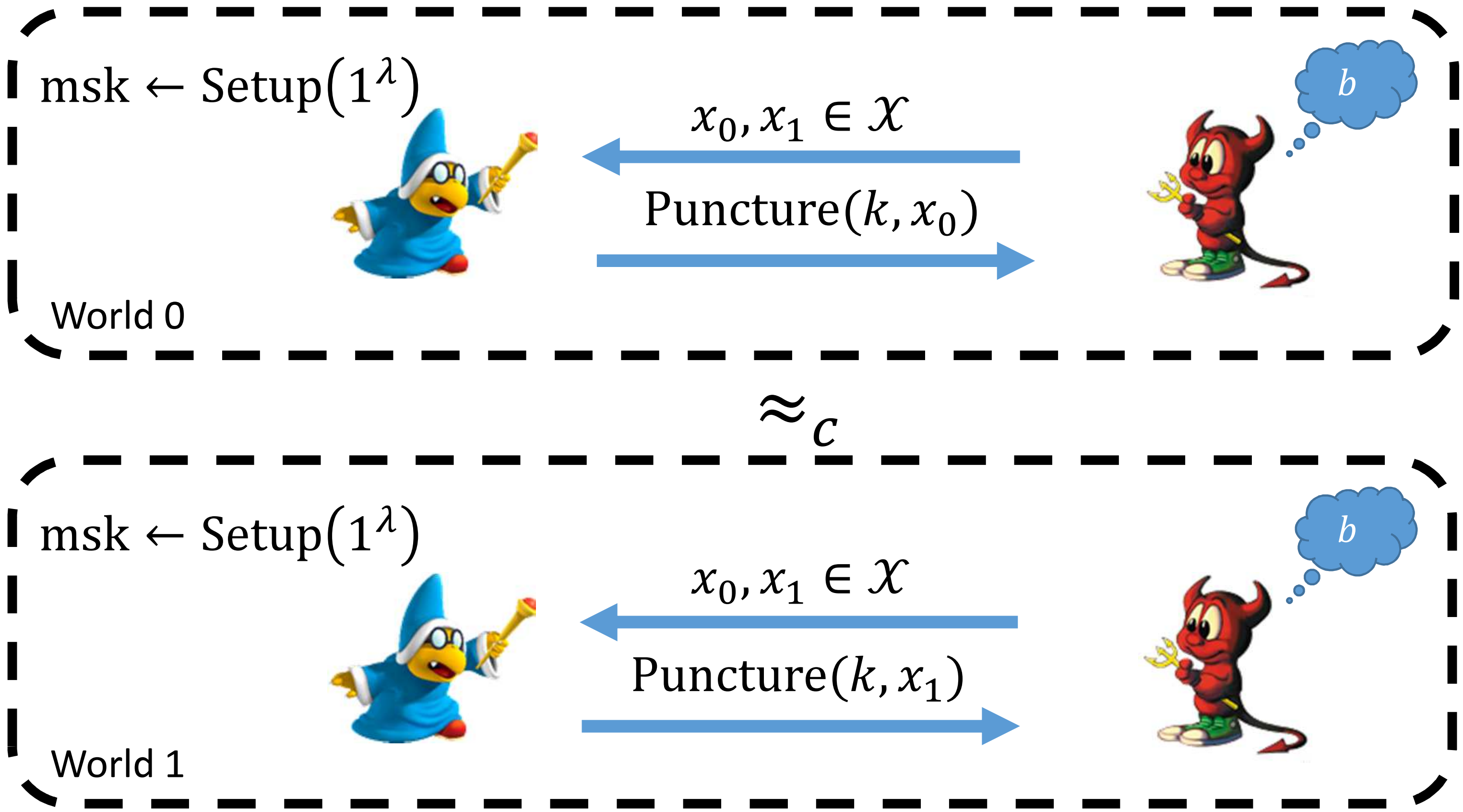


Punctured key for a point z is an obfuscated program

Constrained evaluation corresponds to evaluating obfuscated program

Private Puncturing from iO: Privacy

Recall privacy notion:



Private Puncturing from iO: Privacy

iO ($P_{x_0}(x)$:
- If $x = x_0$, output r
- Else, output $\text{PRF}(k, x)$

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

)

By correctness of puncturing, P_{x_0} and P'_{x_0} compute identical functions

Private Puncturing from iO: Privacy

iO ($P_{x_0}(x)$:
- If $x = x_0$, output r
- Else, output $\mathrm{PRF}(k, x)$)

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

Hybrid 0: Real game

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

Private Puncturing from iO: Privacy

Invoke puncturing security

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

Hybrid 1

$\approx_c$

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

Hybrid 2

Private Puncturing from iO: Privacy

Invoke iO security

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

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 $\text{PRF}(k, x)$

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

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

Security proof similar to that for private puncturable PRF

Requires exponential number of hybrids (one for each input), so require sub-exponential hardness for iO and one-way functions

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 can be constructed from OWFs
  - Can we construct private puncturable PRFs from OWFs?
  - 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?

Thanks!