Course Summary

CS 598 DH

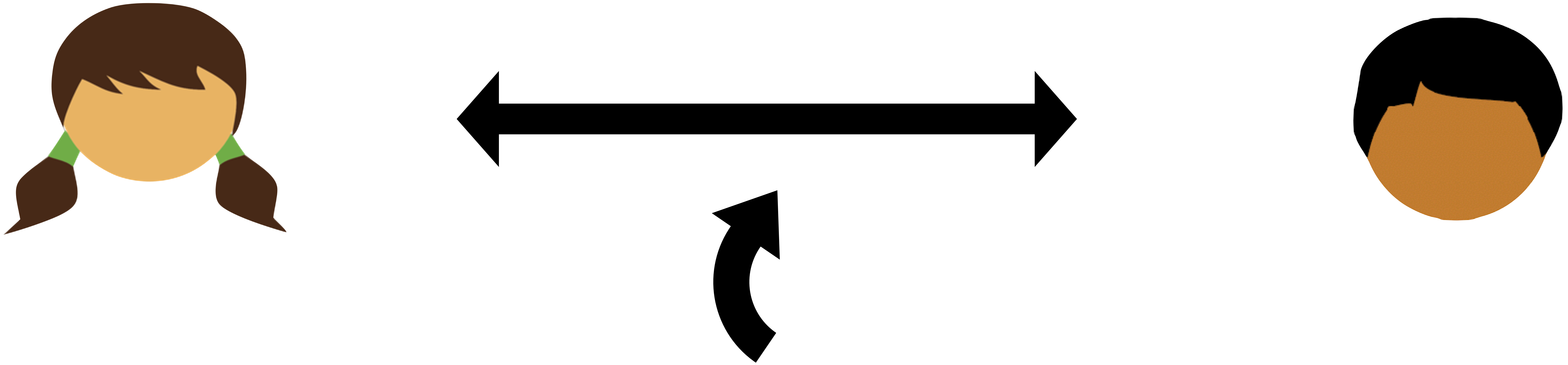
What is this course about?

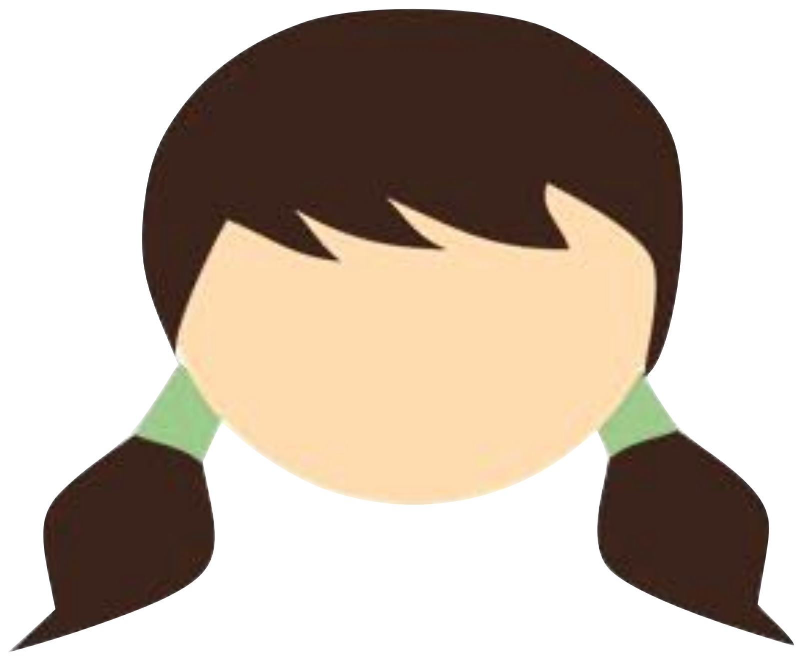
Use cryptography to run computer programs on “encrypted” data.

By doing so, we can solve problems while keeping the underlying data private.

What is this course *not* about?

classic cryptography setting





Privacy
Authenticity

Differentially Private Secure Multi-Party Computation for Federated Learning in Financial Applications

David Byrd
db@gatech.edu
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia

Antigoni Polychroniadou
antigoni.poly@jpmorgan.com
J.P. Morgan AI Research
New York, New York

ABSTRACT

Federated Learning enables a population of clients, working with a trusted server, to collaboratively learn a shared machine learning model while keeping each client's data within its own local systems. This reduces the risk of exposing sensitive data, but it is still possible to reverse engineer information about a client's private data set from communicated model parameters. Most federated learning systems therefore use differential privacy to introduce noise to the parameters. This adds uncertainty to any attempt to reveal private client data, but also reduces the accuracy of the shared model, limiting the useful scale of privacy-preserving noise. A system can further reduce the coordinating server's ability to recover private client information, without additional accuracy loss, by also including secure multiparty computation. An approach combining both techniques is especially relevant to financial firms as it allows new possibilities for collaborative learning without exposing sensitive client data. This could produce more accurate models for important tasks like optimal trade execution, credit origination, or fraud detection. The key contributions of this paper are: We present a privacy-preserving federated learning protocol to a non-specialist audience, demonstrate it using logistic regression on a real-world credit card fraud data set, and evaluate it using an open-source simulation platform which we have adapted for the development of federated learning systems.

KEYWORDS

federated learning, simulation, multiagent, finance, privacy

ACM Reference Format:
David Byrd and Antigoni Polychroniadou. 2020. Differentially Private Secure Multi-Party Computation for Federated Learning in Financial Applications. In *ACM International Conference on AI in Finance (ICAIF '20), October 15–16, 2020, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3383455.3422562

ICAIF '20, October 15–16, 2020, New York, NY, USA

ACM ISBN 978-1-4503-7584-9/20/10...$15.00
https://doi.org/10.1145/3383455.3422562

1 INTRODUCTION

Modern financial firms routinely need to conduct analysis of large data sets stored across multiple servers or devices. A typical response is to combine these data sets into a single central database, but this approach introduces a number of privacy challenges: The institution may not have appropriate authority or permission to transfer locally stored information, the owner of the data may not *want* it shared, and centralization of the data may worsen the potential consequences of a data breach.

For example, the mobile app ai.type collected personal data from its users' phones and uploaded this information to a central database. Security researchers gained access to the database and obtained the names, email addresses, passwords, and other sensitive information of 31 million users of the Android version of the app. Such incidents highlight the risks and challenges associated with centralized data solutions. [5]

In this section, we motivate our approach while providing an extensive non-technical overview of the underlying techniques.

1.1 Federated Learning

One approach to mitigate the mentioned privacy concerns is to analyze the multiple data sets separately and share only the resulting insights from each analysis. This approach is realized in a recently-introduced technique called federated analysis. [2] Federated *learning*, already adopted by large companies like Google, allows users to share insights (perhaps the parameters of a trained model) from the data on their laptops or mobile devices *without* ever sharing the data itself, typically as follows:

1. Users train a local model on their individual data.
2. Each user sends their model weights to a trusted server.
3. The server computes an average-weight shared model.
4. The shared model is returned to all of the users.
5. Users retrain a local model starting from the shared model.

For instance, email providers could use federated learning to reduce the amount of spam their customers receive. Instead of each provider using its own spam filter trained from its customers' reported spam email, the providers could combine their models to create a shared spam-detection mechanism, without sharing their individual customers' reported spam emails. For a survey of recent advances in federated learning, see Kairouz et al. [13]

It is still possible, however, for a malicious party to potentially compromise the privacy of the individual users by inferring details of a training data set from the trained model's weights or parameters [16, 19]. It is important to protect sensitive user information while still providing highly accurate inferences.

Secure Auctions

Privacy-preserving studies

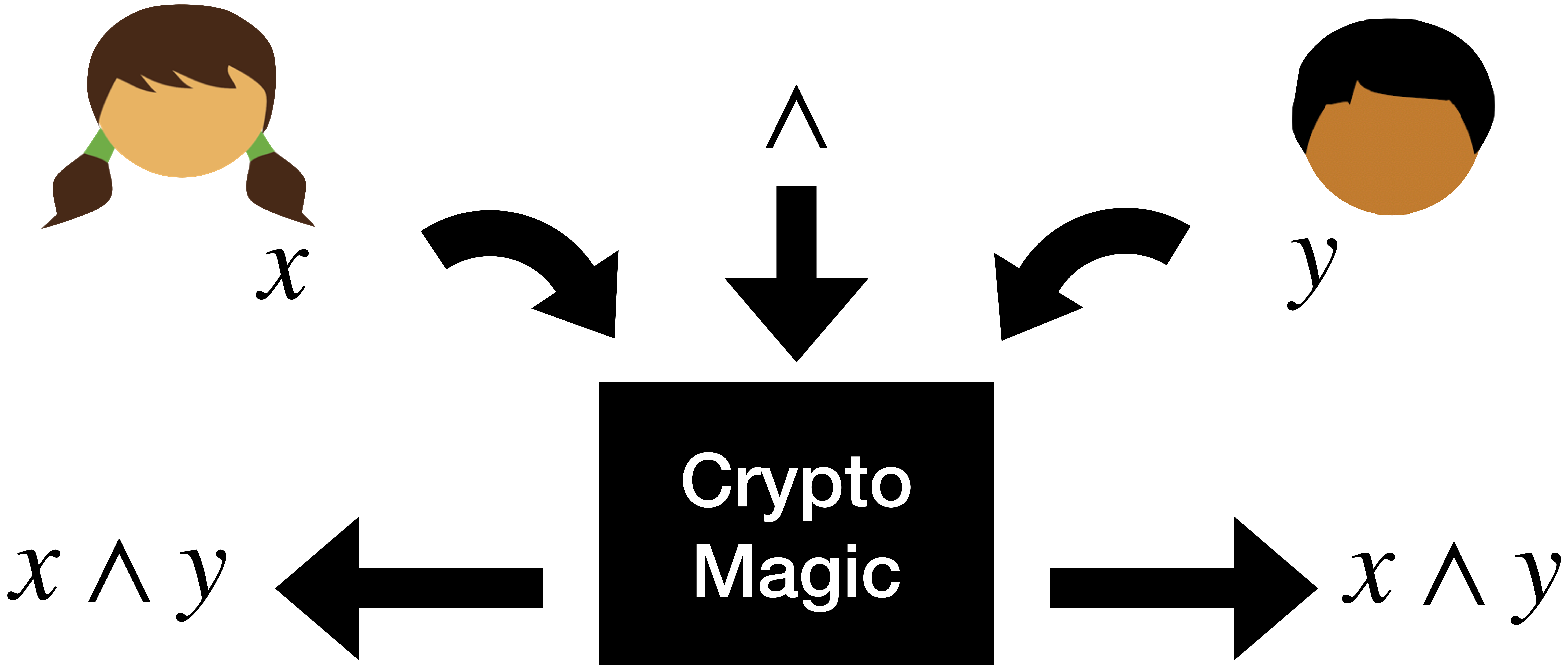
Privacy-preserving advertising

Privacy-preserving analytics

(Secure Machine Learning)

Financial Fraud Detection

…and much more

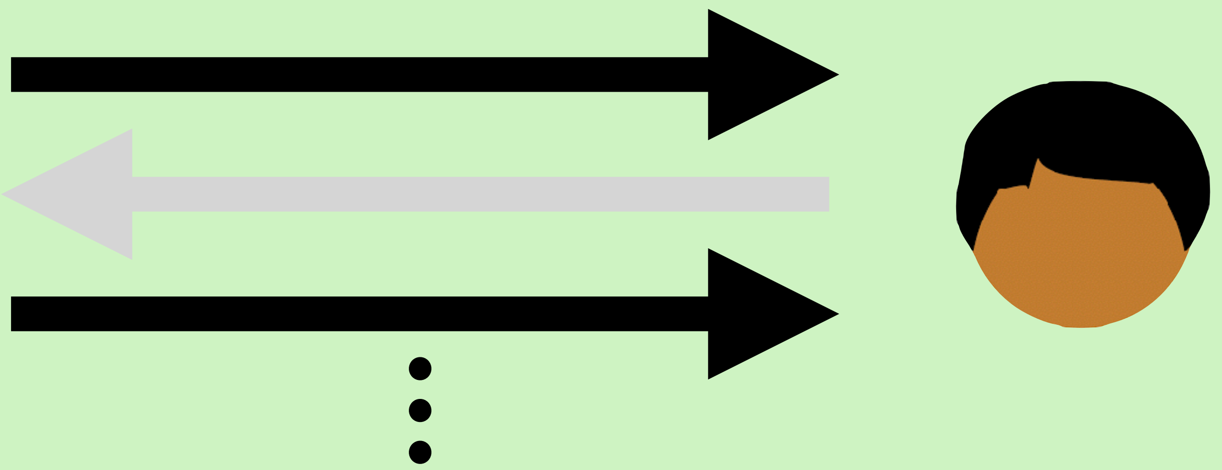


x
$\wedge$
y
Crypto
Magic
$x \wedge y$
$x \wedge y$

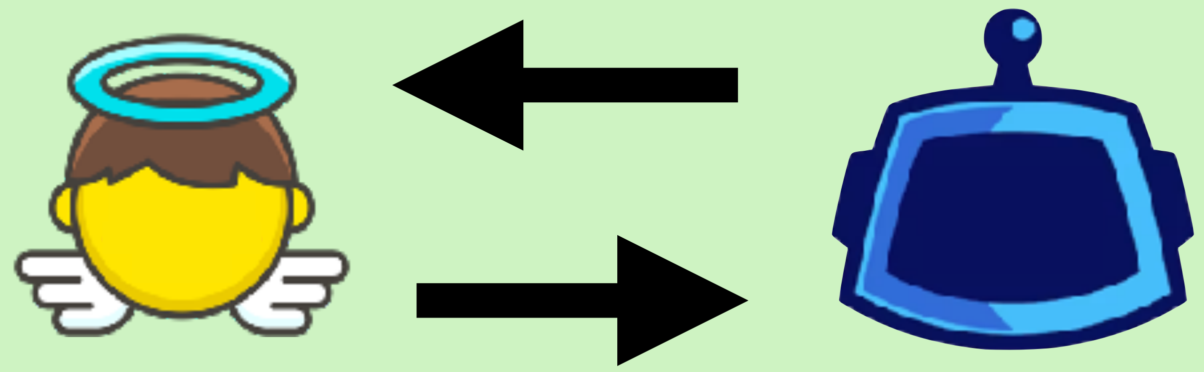
Simulator

Real

Ideal



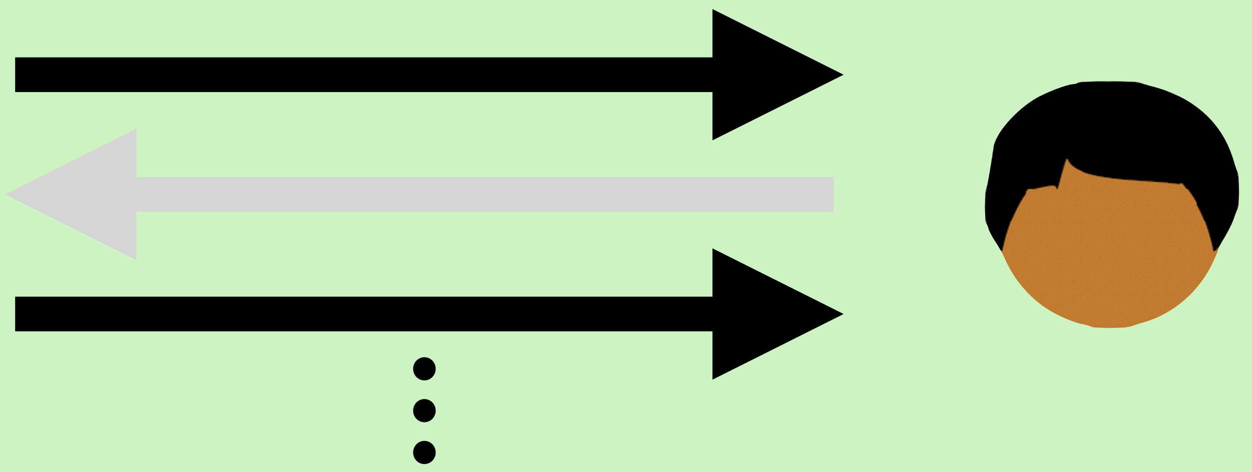
$\text{View}^{\Pi}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$



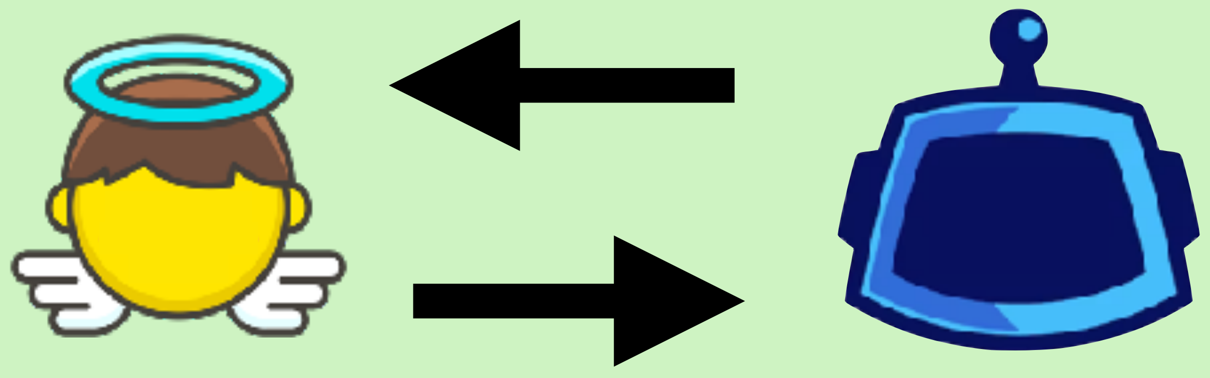
$\text{Output}^{\text{Sim}}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$

These should "look the same"

“No efficient algorithm can tell these two things apart”



$$\mathrm{View}^{\Pi}_{\mathrm{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$$



$$\mathrm{Output}^{\mathrm{Sim}}_{\mathrm{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$$

Three notions of “hard to tell apart”

$X \equiv Y$ Identically distributed

$X \approx Y$ Statistically close — As we increase a parameter, the distributions **quickly** become close together.

$X \overset{c}{=} Y$ Indistinguishable — As we increase a parameter, it **quickly** becomes difficult for programs to tell the distributions apart.

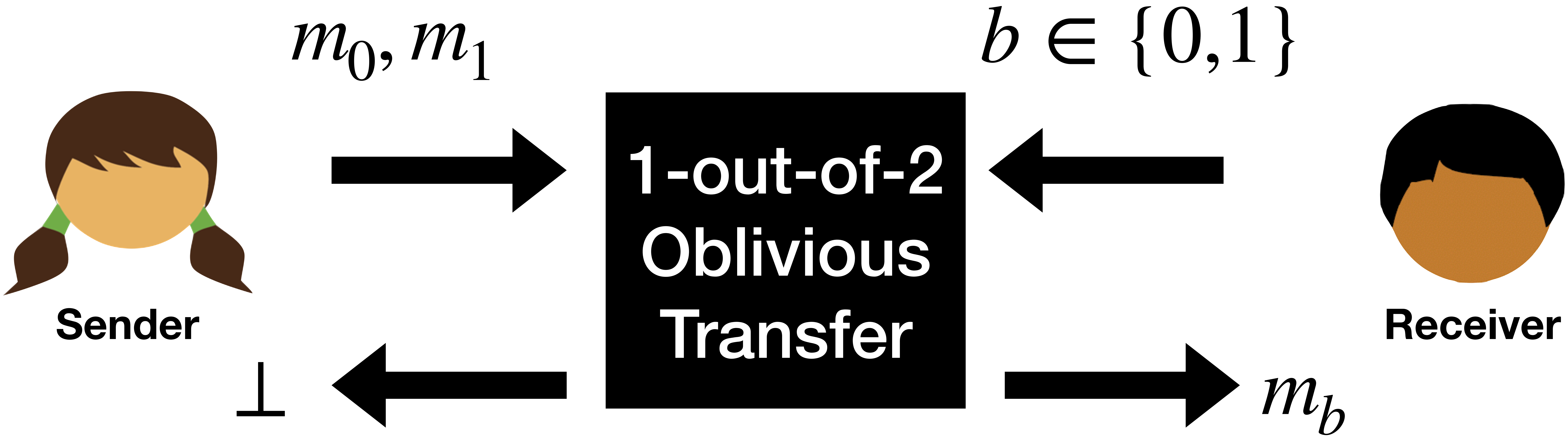
Two-Party Semi-Honest Security

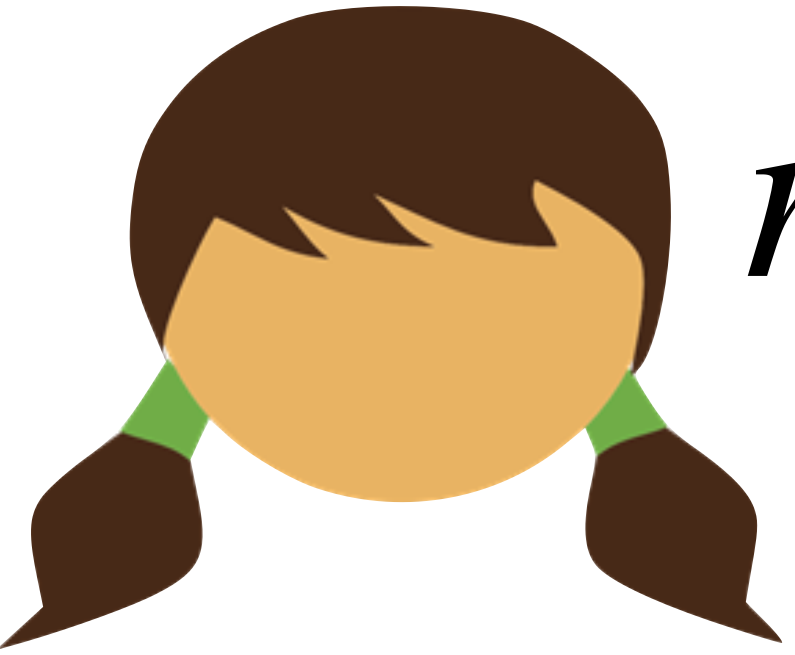
Let f be a functionality. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each party $i \in \{0,1\}$ there exists a polynomial time simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

$$\{\mathrm{View}_i^{\Pi}(x_0, x_1), \mathrm{Output}^{\Pi}(x_0, x_1)\}$$

$$\overset{c}{=}$$

$$\{\mathcal{S}_i(x_i, y_i), (y_0, y_1) \mid (y_0, y_1) \leftarrow f(x_0, x_1)\}$$


m_0, m_1
$b \in \{0,1\}$
1-out-of-2
Oblivious
Transfer
Sender
Receiver
$\perp$
m_b

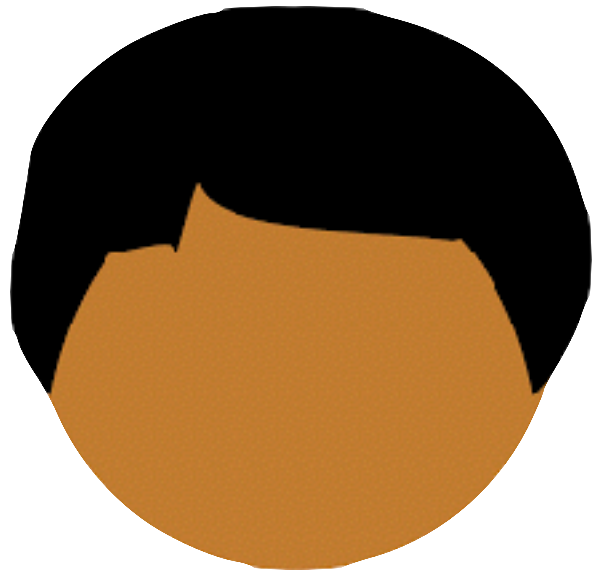


m_0, m_1

Sender

$r_0 \xleftarrow{\$} \mathbb{Z}_q$

$r_1 \xleftarrow{\$} \mathbb{Z}_q$



b

Receiver

$a \xleftarrow{\$} \mathbb{Z}_q$

$h_b \leftarrow g^a$

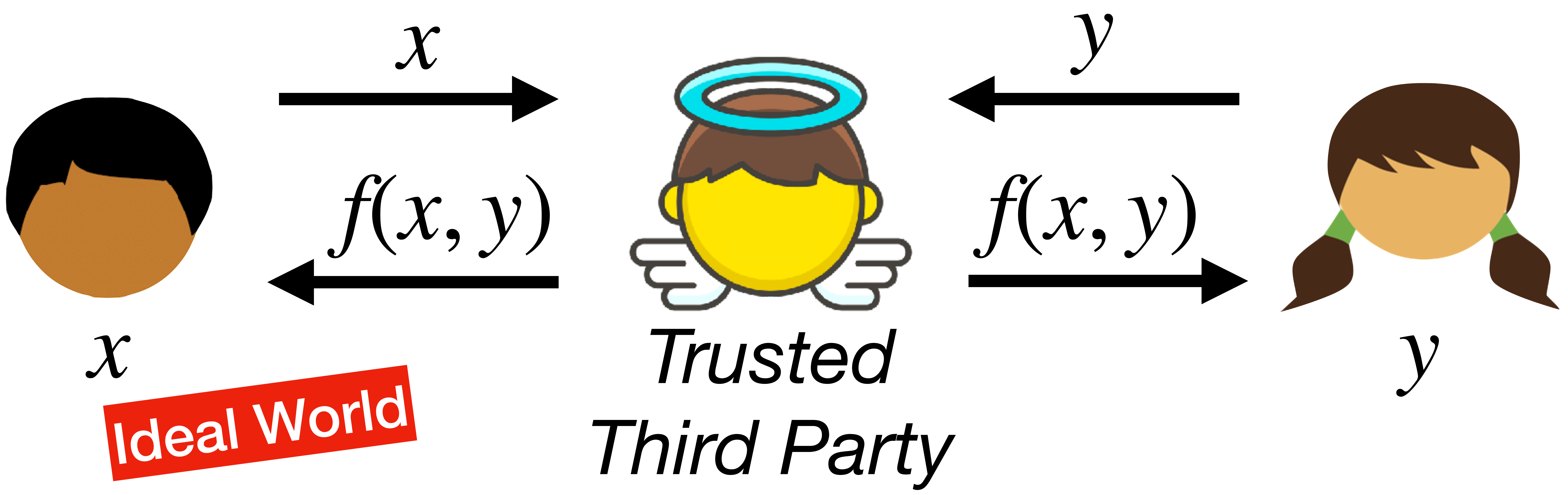
$h_{1-b} \xleftarrow{\$} G$

$$\xleftarrow{\quad h_0, h_1 \quad}$$

$$\xrightarrow{\quad g^{r_0} \quad g^{r_1} \quad h_0^{r_0} \cdot m_0 \quad h_1^{r_1} \cdot m_1 \quad}$$

$$\frac{h_b^{r_b} \cdot m_b}{\left(g^{r_b}\right)^a}$$

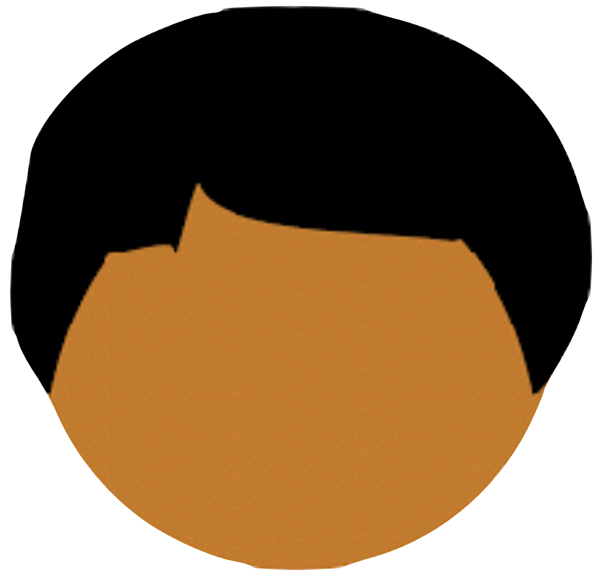
$$\frac{h_b^{r_b} \cdot m_b}{\left(g^{r_b}\right)^a} = \frac{(g^a)^{r_b} \cdot m_b}{\left(g^{r_b}\right)^a} = \frac{g^{a \cdot r_b} \cdot m_b}{g^{a \cdot r_b}} = m_b$$



GMW Protocol

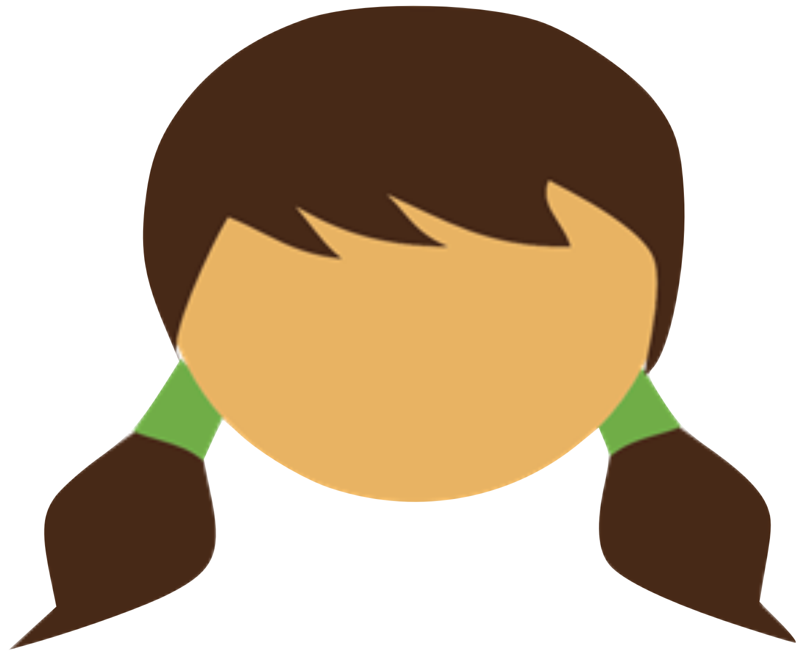
Hint: Lots of OT

Real World



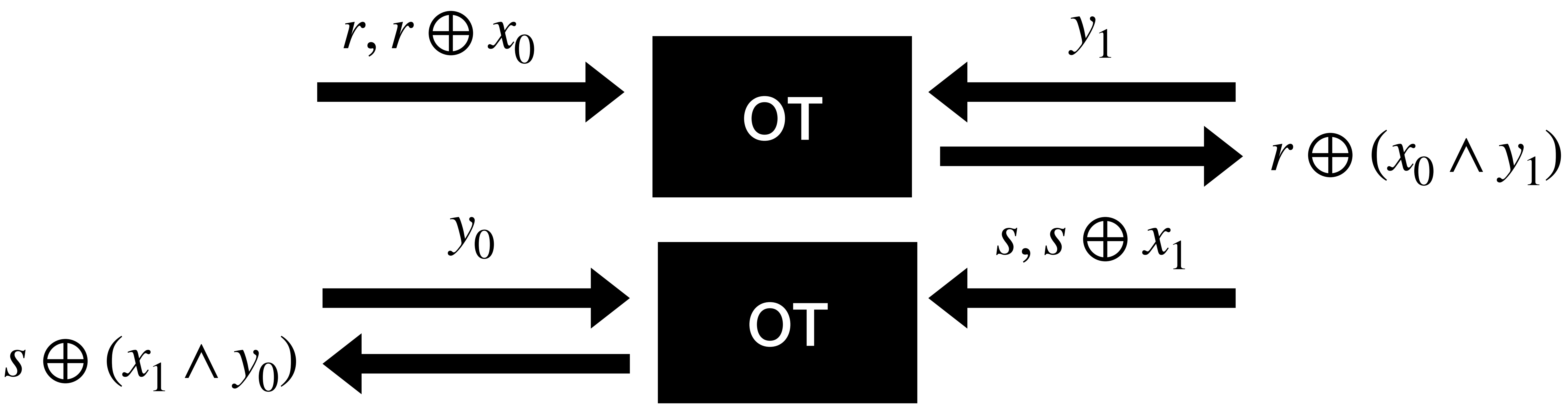
How do we AND two shares?

Goal: given gate input wires holding $[x]$, $[y]$, put $[x \wedge y]$ on the gate output



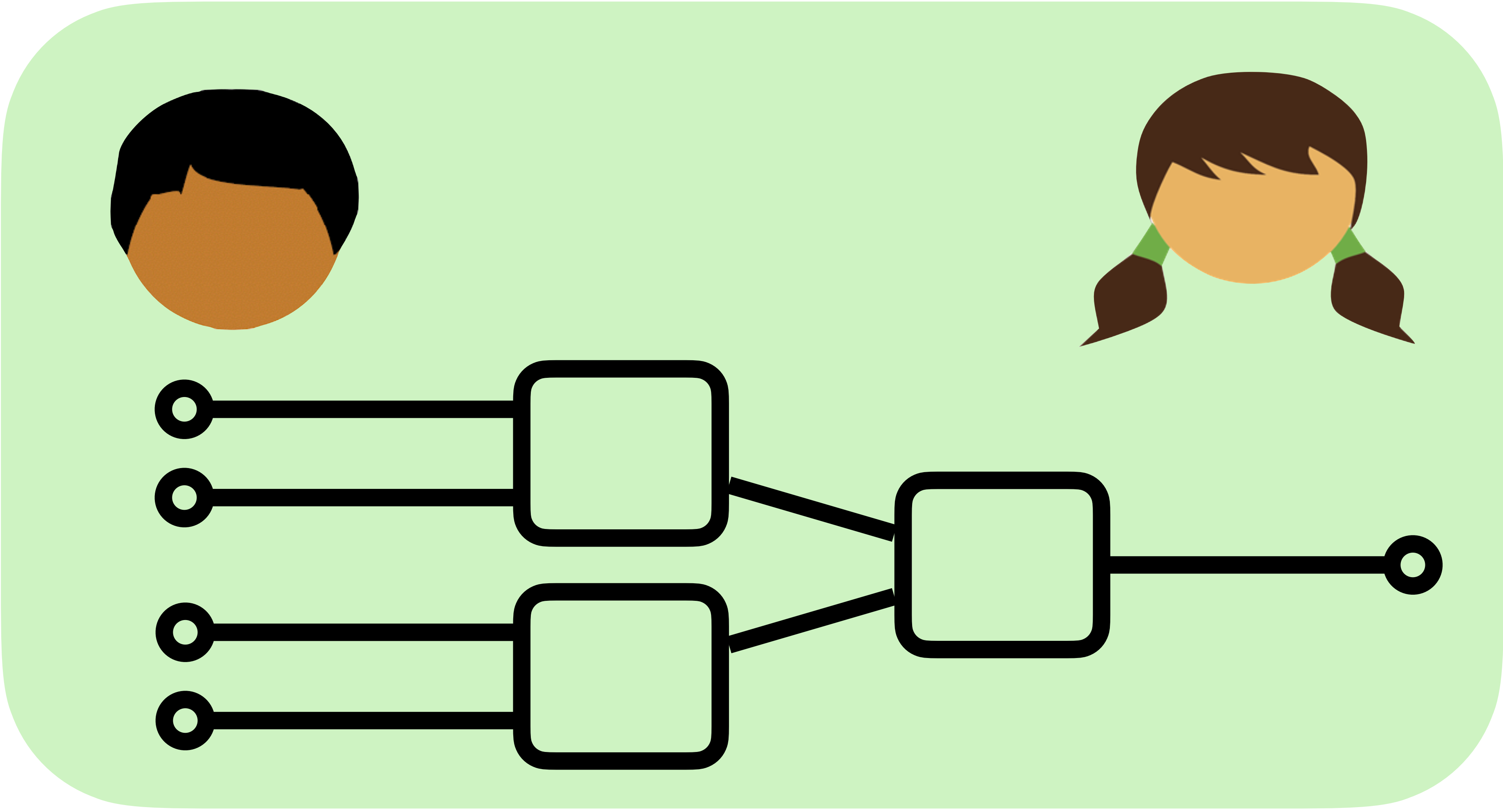
$r \xleftarrow{\$} \{0,1\}$

$s \xleftarrow{\$} \{0,1\}$



$$\langle r \oplus (s \oplus x_1 \wedge y_0) \oplus (x_0 \wedge y_0), s \oplus (r \oplus x_0 \wedge y_1) \oplus (x_1 \wedge y_1) \rangle$$

$$= [x \wedge y]$$



In GMW, Number of protocol rounds scales with multiplicative depth of C

Our protocol's efficiency is fundamentally bounded by the speed of light

Pseudorandom Function (PRF)

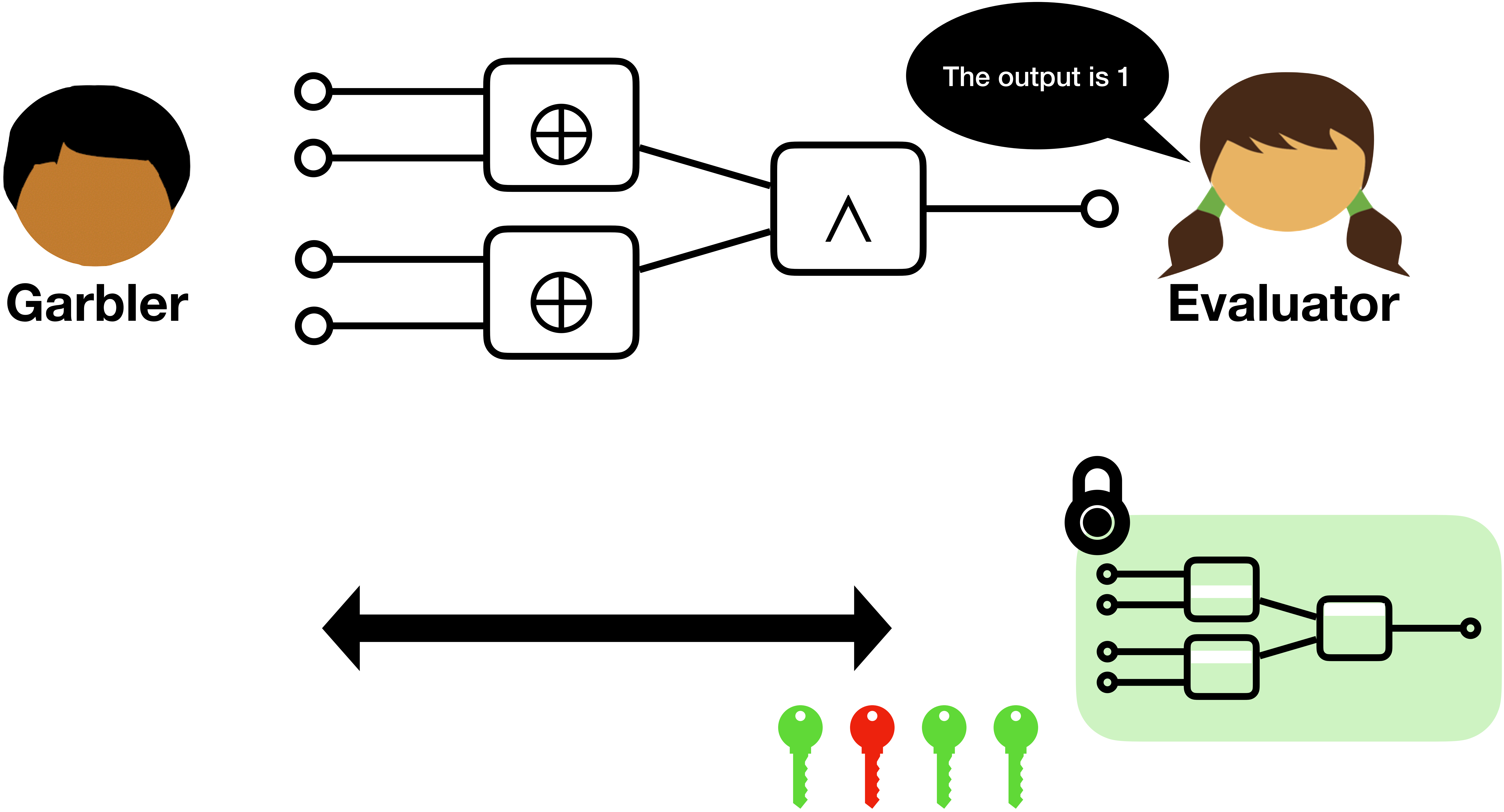
A function family F is considered pseudorandom *if the following indistinguishability holds*

```
Real:
    k <-$- {0,1}^λ

    lookup(m):
        return F(k, m)
```

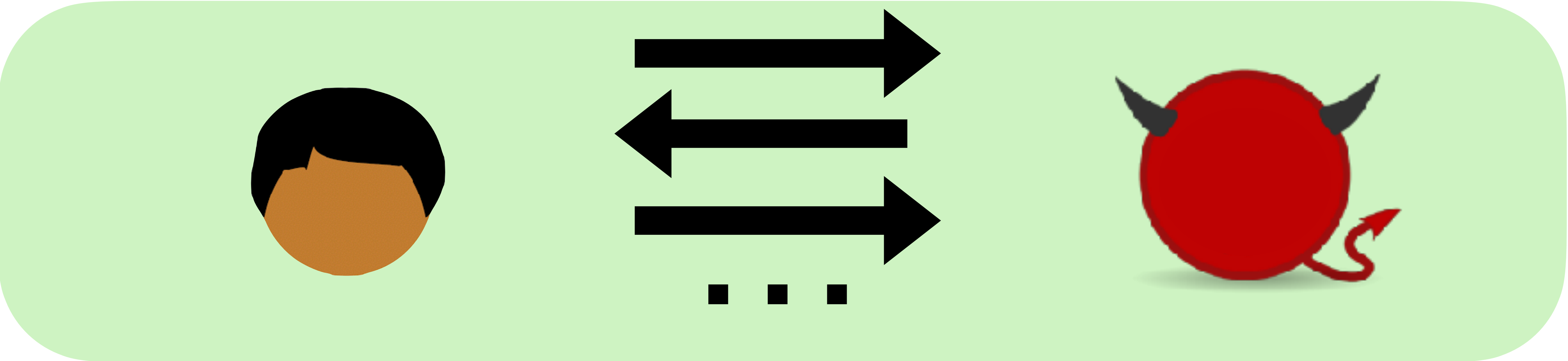
$$\overset{c}{=}$$

```
Ideal:
    T ← EmptyMap

    lookup(m):
        if m ∉ T:
            T[m] <-$- {0,1}^out
        return T[m]
```

"If you don't know the key, F looks random"


Garbler
The output is 1
Evaluator

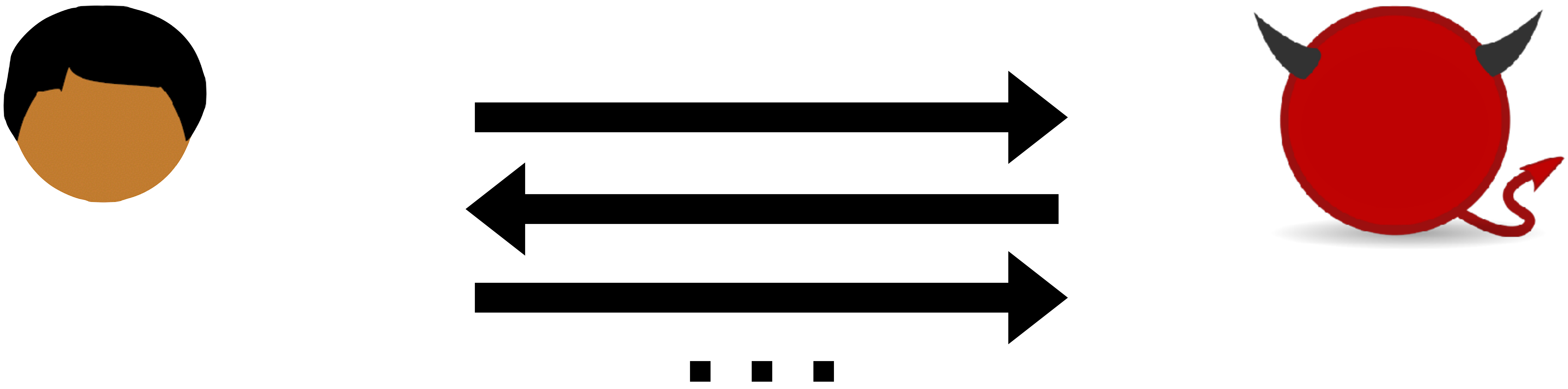
Malicious Security



*A protocol Π securely realizes a functionality f in the presence of a malicious adversary if for **every** real-world adversary $\mathscr{A}$ corrupting party i, **there exists** an ideal-world adversary $\mathscr{S}_i$ (a simulator) such that for all inputs x, y the following holds:*

$$\mathrm{Real}^{\Pi}_{\mathscr{A}}(x, y) \approx \mathrm{Ideal}^{f}_{\mathscr{S}_i}(x, y)$$

Ensemble of outputs of ***each*** party



What can go *in terms of outcomes*?

Cause honest party to output wrong answer ✓

Learn too much information about other party's input ✓

Prevent honest party from learning output ✗

Malicious security ideal-world execution

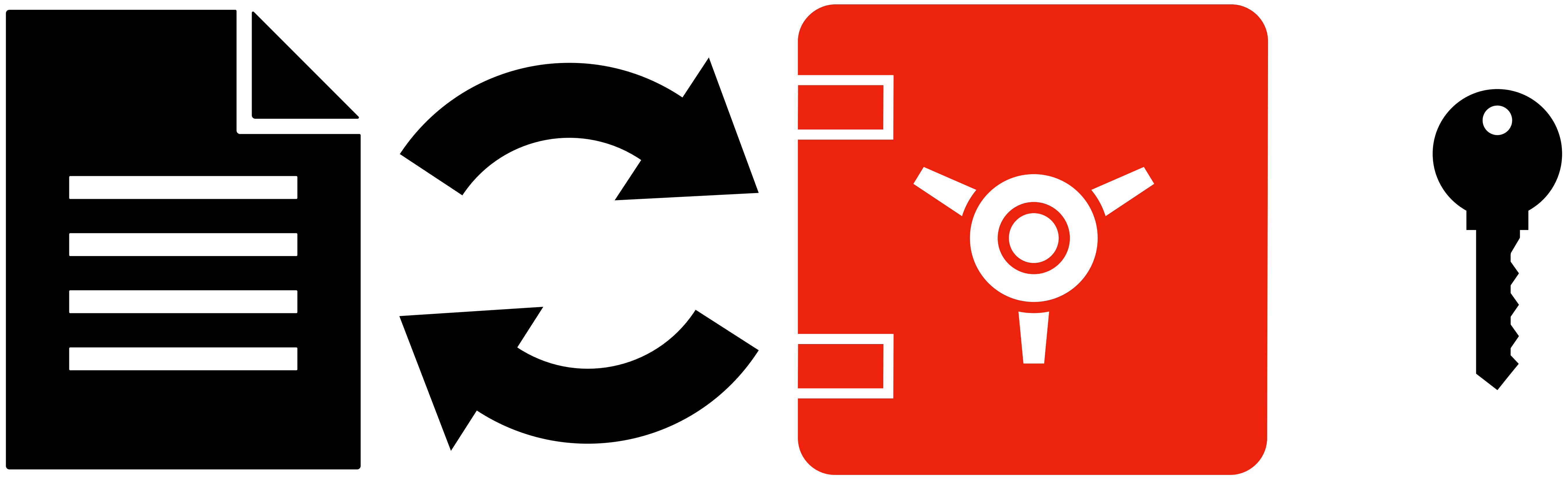
honest party outputs

$f(x, y')$

adversary outputs… ?

whatever it wants

Commitment Scheme



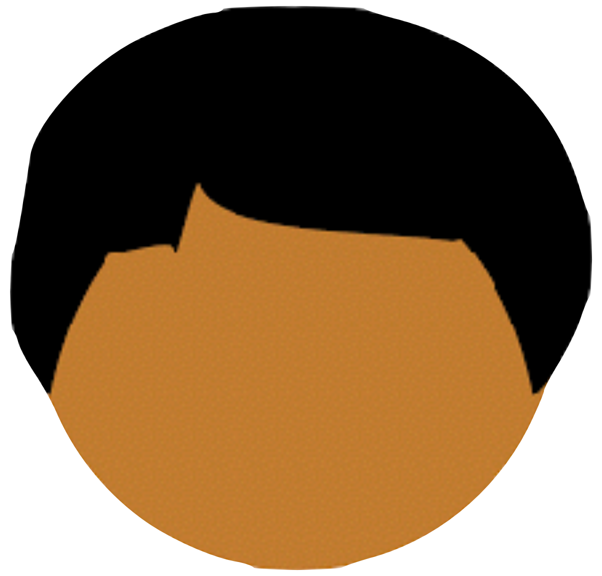
Hiding

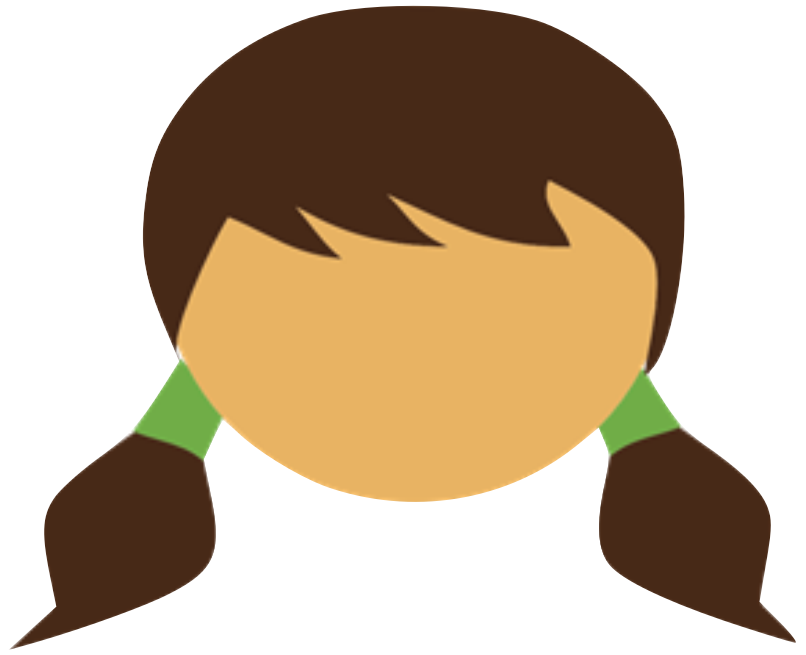
I am confident you cannot open the box without the key

You are confident I cannot tamper with the content of the box

Binding

$$f(\cdot) = \{\ r \mid r \xleftarrow{\$} \{0,1\}\ \}$$





$b_0 \xleftarrow{\$} \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^\lambda$

$b_1 \xleftarrow{\$} \{0,1\}$

$c = \mathrm{Com}(b_0; r) \longrightarrow$

$\longleftarrow b_1$

$b_0, r \longrightarrow$

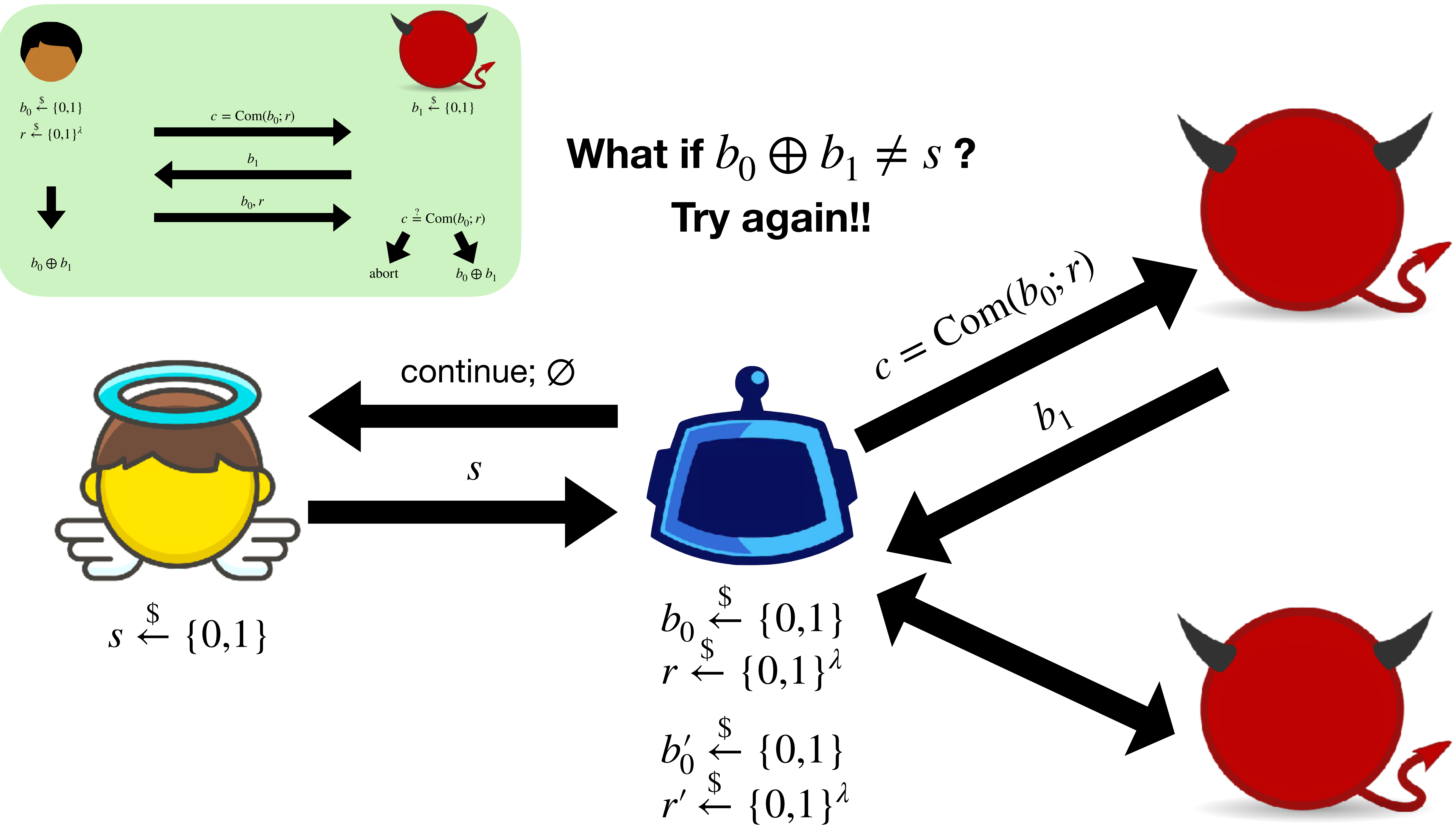
$c \stackrel{?}{=} \mathrm{Com}(b_0; r)$

abort

$b_0 \oplus b_1$

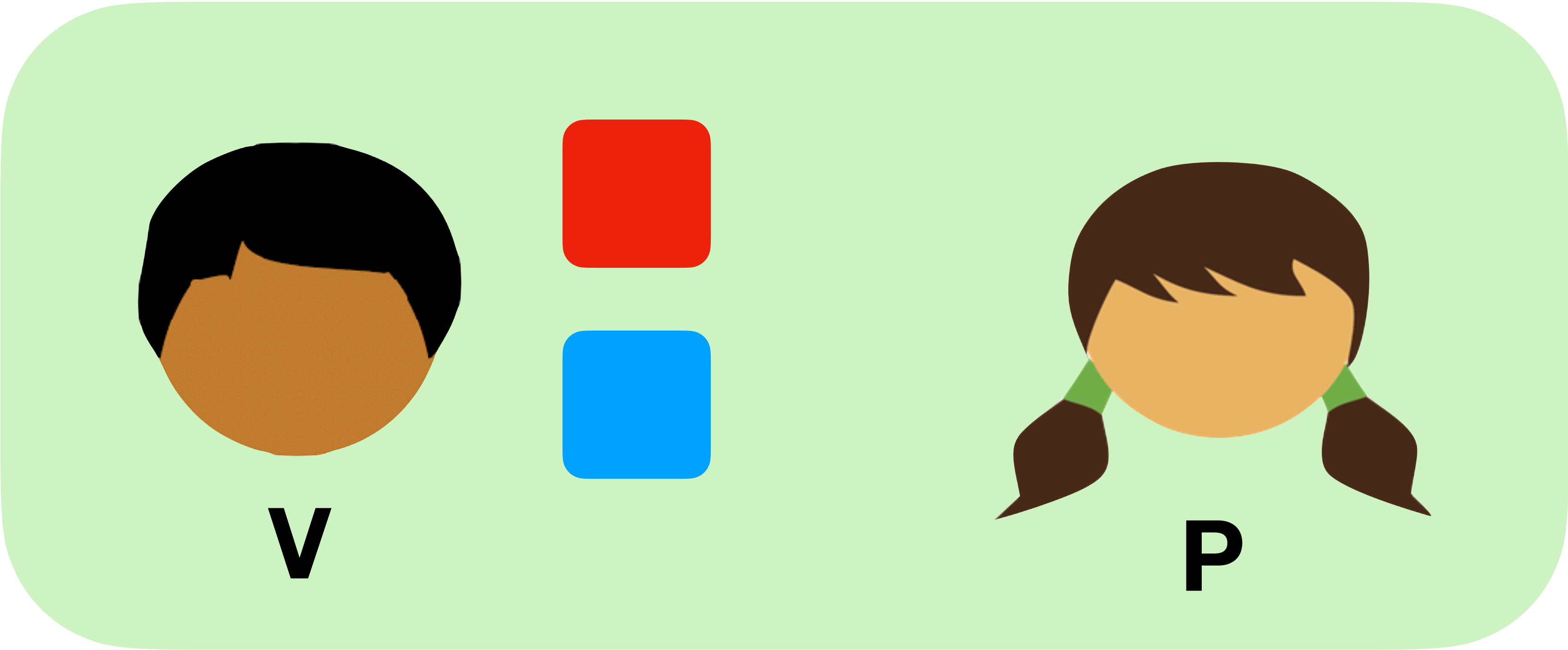
$b_0 \oplus b_1$

($b_1 = 0$ if Alice aborts)



$b_0 \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \{0,1\}^\lambda$
$b_1 \xleftarrow{\$} \{0,1\}$
$c = \mathrm{Com}(b_0; r)$
b_1
b_0, r
$c \stackrel{?}{=} \mathrm{Com}(b_0; r)$
$b_0 \oplus b_1$
abort
$b_0 \oplus b_1$
What if $b_0 \oplus b_1 \neq s$?
Try again!!
continue; $\varnothing$
s
$c = \mathrm{Com}(b_0; r)$
b_1
$s \xleftarrow{\$} \{0,1\}$
$b_0 \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \{0,1\}^\lambda$
$b'_0 \xleftarrow{\$} \{0,1\}$
$r' \xleftarrow{\$} \{0,1\}^\lambda$

What is a *zero-knowledge* proof?



Completeness: If $x \in \mathscr{L}$ and if P and V are honest, then V accepts the proof (except with negligible probability)

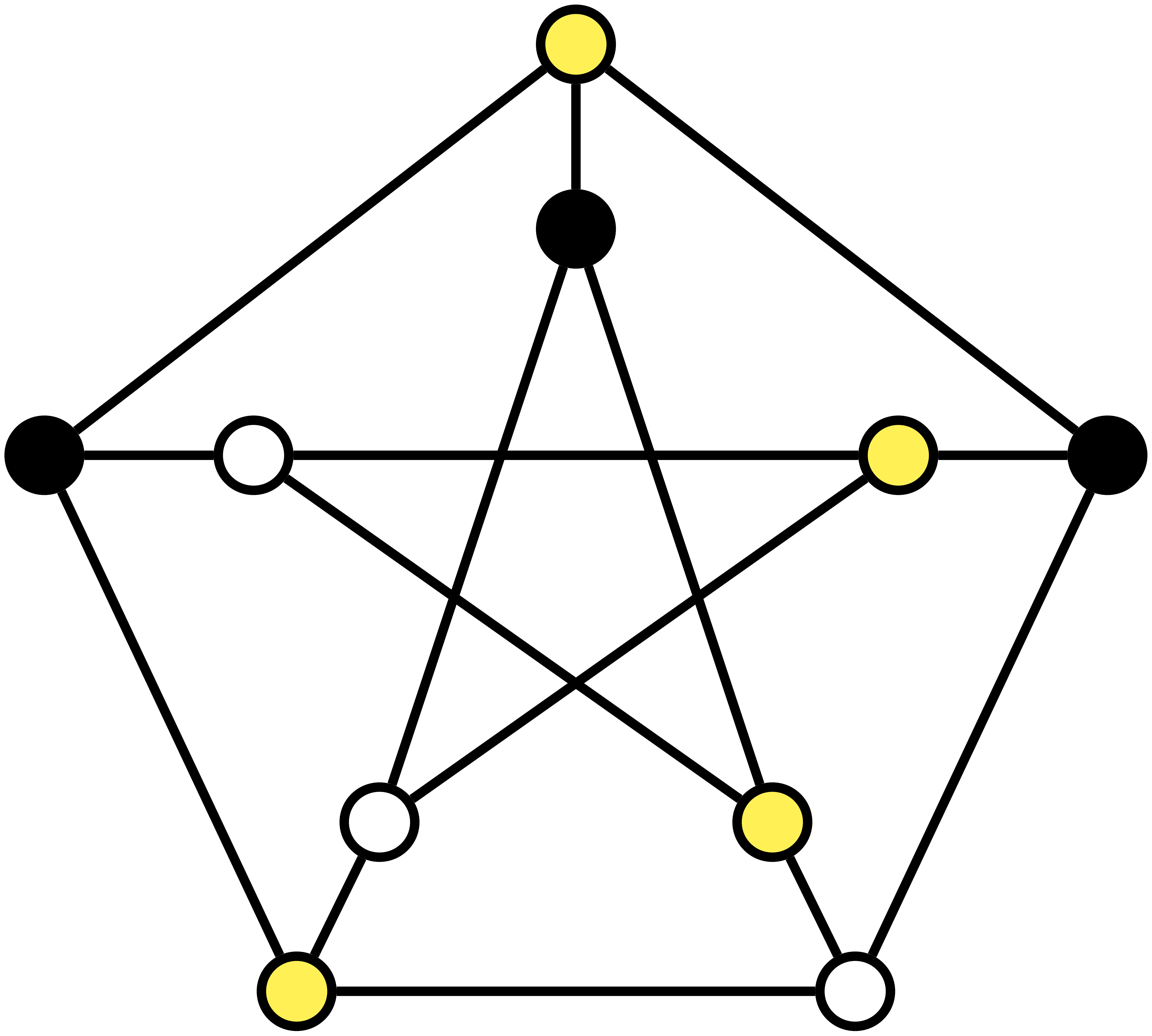
"P can prove true things"

Soundness: If $x \notin \mathscr{L}$, even malicious P cannot cause honest V to accept the proof

"P cannot prove false things"

Zero Knowledge: "V learns nothing except that the thing is true"

Graph 3-Coloring



ZK Proof system for 3-colorability

Statement: a graph
“this graph is 3-colorable”

Witness: a coloring

Basic cryptographic tool: Commitments

Setting

Semi-honest Security

GMW
Compiler

Malicious Security

Zero Knowledge

General-Purpose Tools

GMW Protocol

Multi-party

Multi-round

Garbled Circuit

Constant Round

Two Party

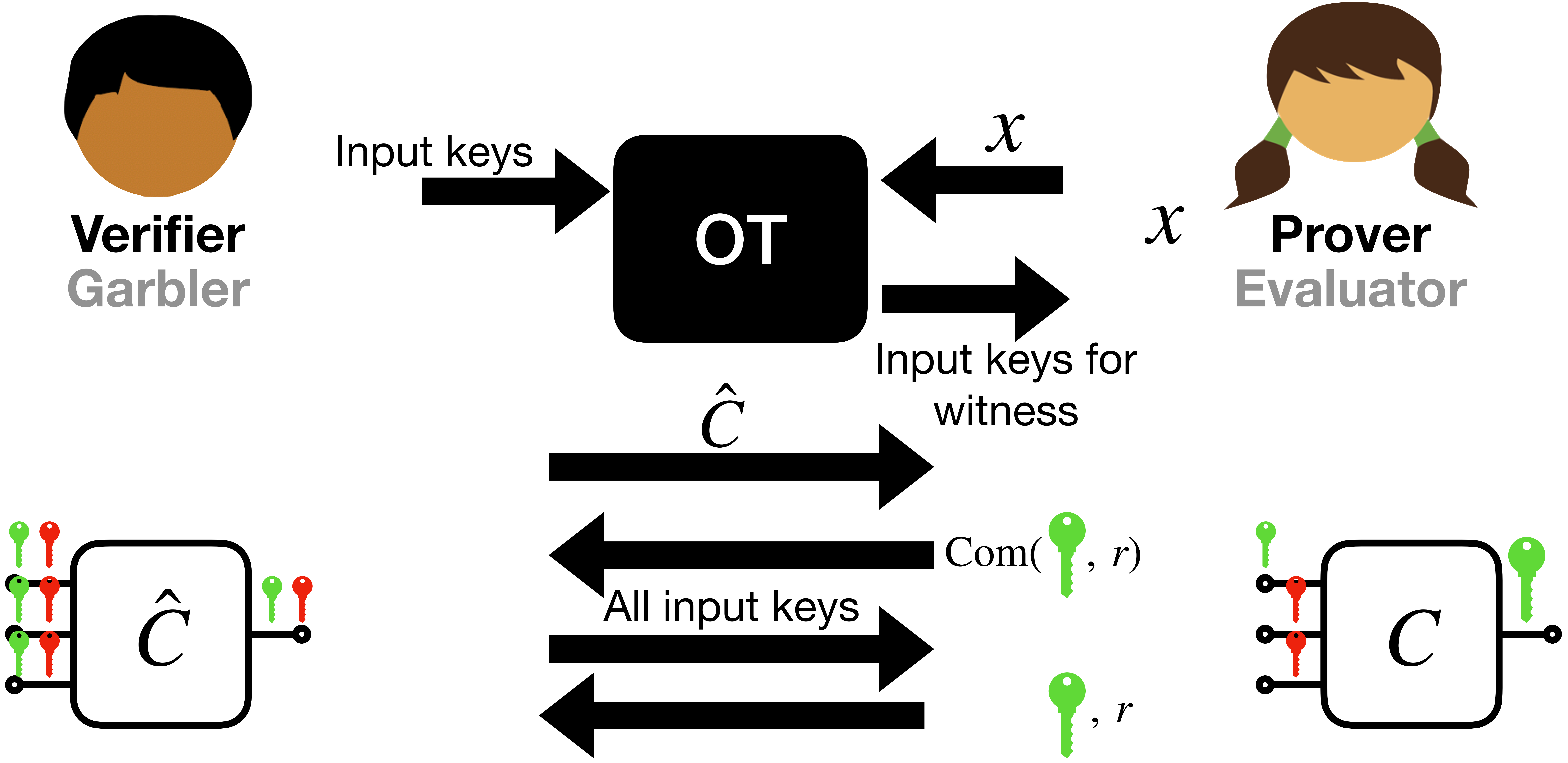
Primitives

Oblivious Transfer

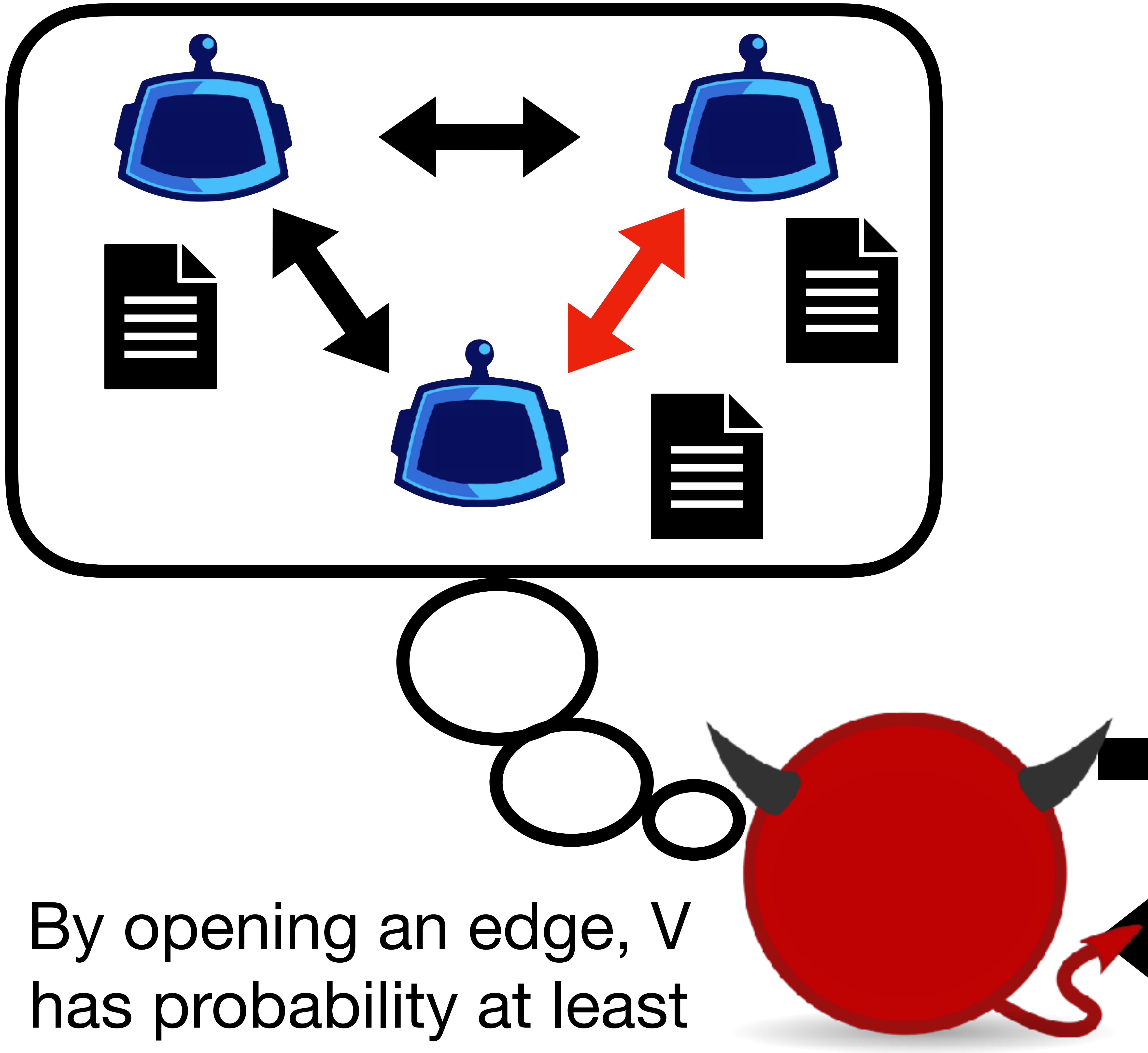
Pseudorandom functions/encryption

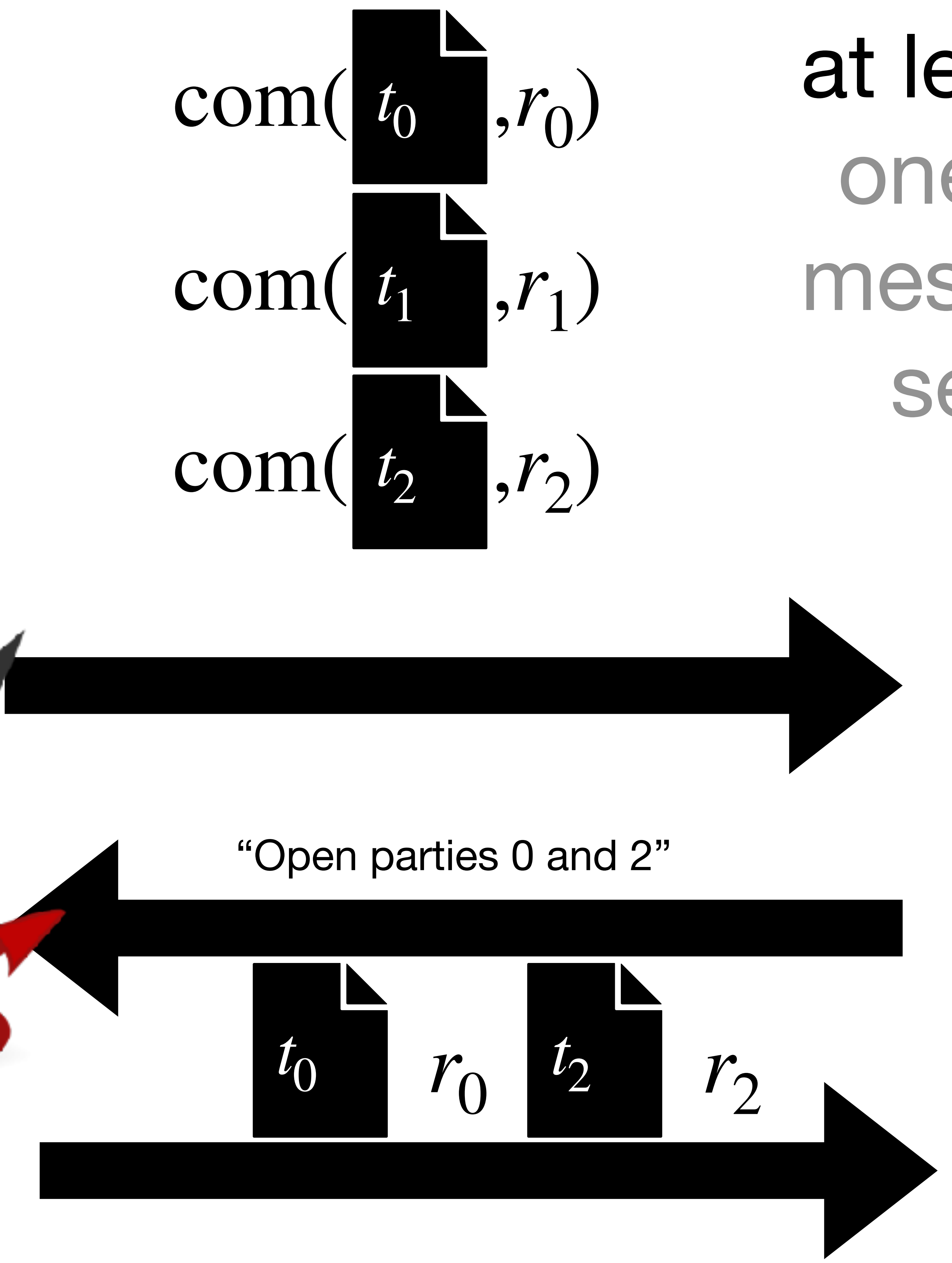
Commitments

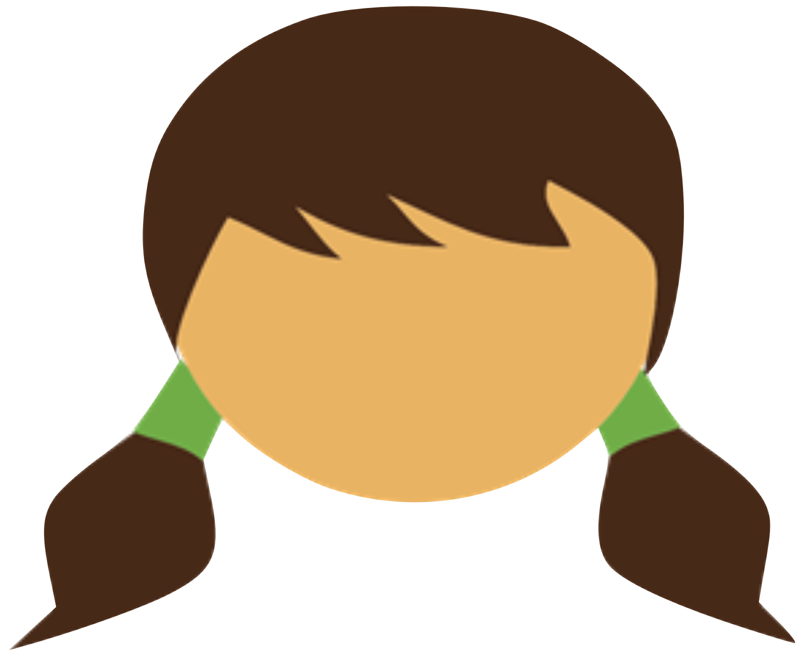
Zero Knowledge from Garbled Circuits



ZK from MPC in the Head







P

V

Soundness?

To cheat, P must corrupt at least one edge (i.e., one party receives a message that was not sent by the other)

By opening an edge, V has probability at least 1/3 to catch cheating P

Repeat to obtain desired soundness

How To Prove Yourself:
Practical Solutions to Identification
and Signature Problems

Amos Fiat and Adi Shamir
Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100, Israel

Abstract.

In this paper we describe simple identification and signature schemes which enable any user to prove his identity and the authenticity of his messages to any other user without shared or public keys. The schemes are provably secure against any known or chosen message attack if factoring is difficult, and typical implementations require only 1% to 4% of the number of modular multiplications required by the RSA scheme. Due to their simplicity, security and speed, these schemes are ideally suited for microprocessor-based devices such as smart cards, personal computers, and remote control systems.

1. Introduction

Creating unforgeable ID cards based on the emerging technology of smart cards is an important problem with numerous commercial and military applications. The problem becomes particularly challenging when the two parties (the prover A and the verifier B) are adversaries, and we want to make it impossible for B to misrepresent himself as A even after he witnesses and verifies arbitrarily many proofs of identity generated by A. Typical applications include passports (which are often inspected and photocopied by hostile governments), credit cards (whose numbers can be copied to blank cards or used over the phone), computer passwords (which are vulnerable to hackers and wire tappers) and military command and control systems (whose terminals may fall into enemy hands). We distinguish between three levels of protection:

1) Authentication schemes: A can prove to B that he is A, but someone else cannot prove to B that he is A.

2) Identification schemes: A can prove to B that he is A, but B cannot prove to someone else that he is A.

3) Signature schemes: A can prove to B that he is A, but B cannot prove even to himself that he is A.

Authentication schemes are useful only against external threats when A and B cooperate. The distinction between identification and signature schemes is subtle, and manifests itself mainly when the proof is interactive and the verifier later wants to prove its existence to a judge: In identification schemes B can create a credible transcript of an imaginary communication by carefully choosing both the questions and the answers in the dialog, while in signature schemes only real communication with A could generate a credible transcript. However, in many commercial and military applications the main problem is to detect forgeries in real time and to deny the service,

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 186-194, 1987.

Fiat Shamir Heuristic

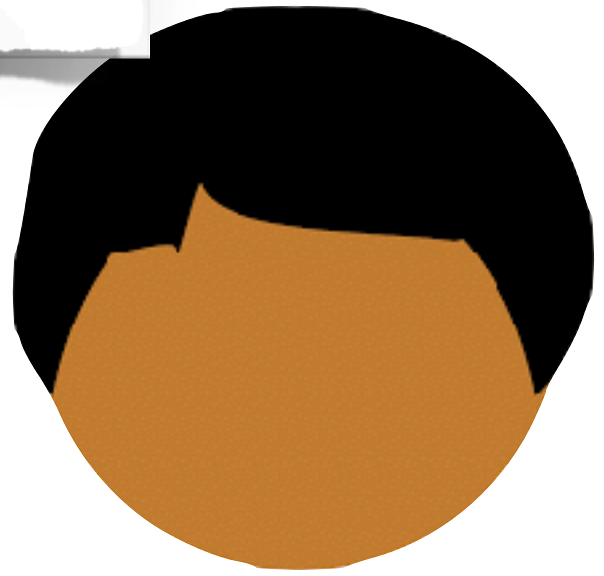
Public coin ZK can be made **non-interactive**

Simple idea: P can choose the challenge itself

Cryptographic hash function (e.g. SHA 256)
Formally, a random oracle

commitment

challenge = H(commitment)



P

response

Garbler

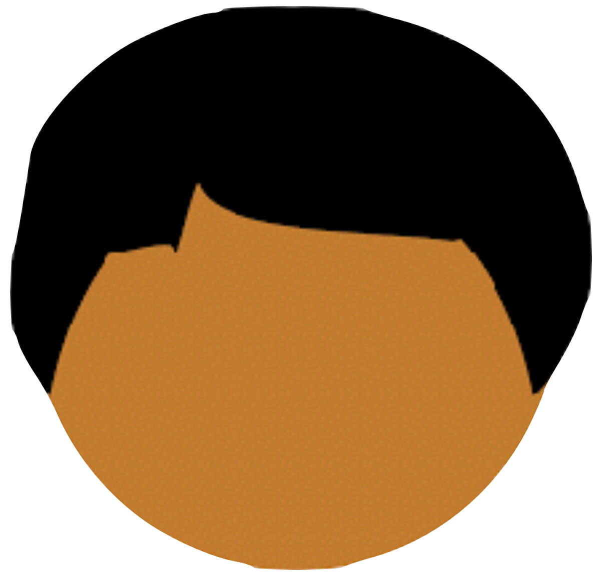
$$\mathrm{Enc}(K_a^0, \mathrm{Enc}(K_b^0, K_c^0))$$
$$\mathrm{Enc}(K_a^0, \mathrm{Enc}(K_b^1, K_c^0))$$
$$\mathrm{Enc}(K_a^1, \mathrm{Enc}(K_b^0, K_c^0))$$
$$\mathrm{Enc}(K_a^1, \mathrm{Enc}(K_b^1, K_c^1))$$

Why can't we simulate G?

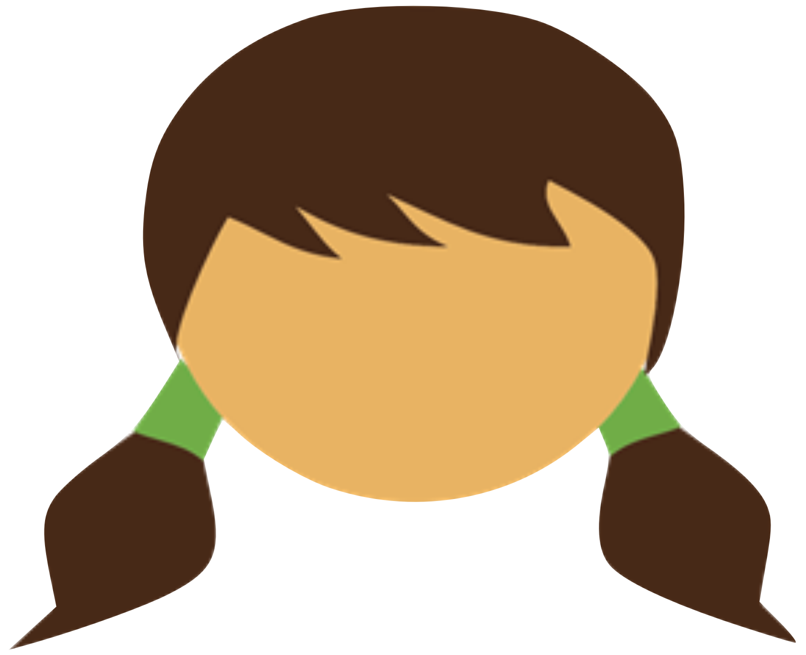
G can encrypt each gate *freely*

E has no way to tell if gate it correctly garbled

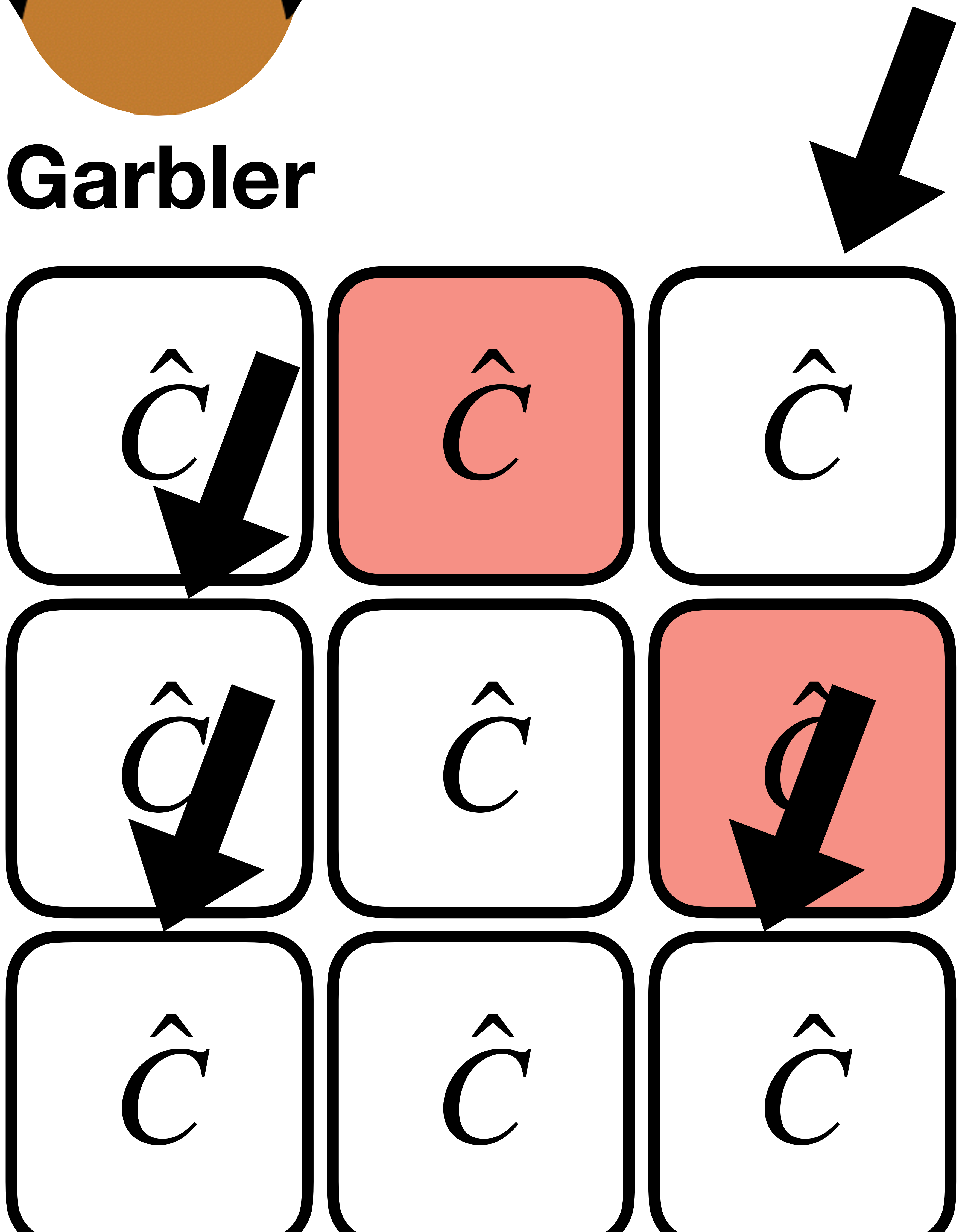
Cut and Choose



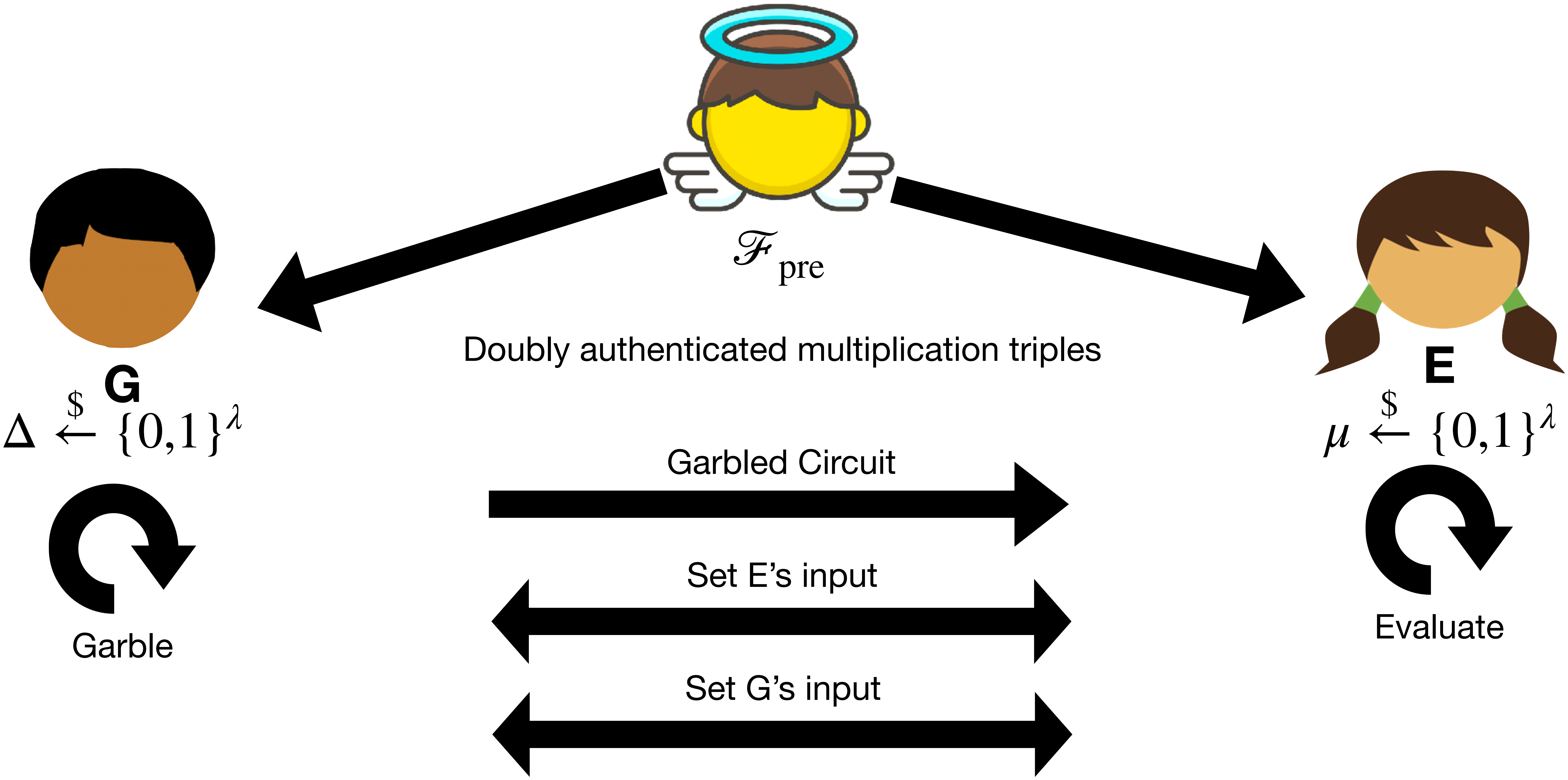
Garbler



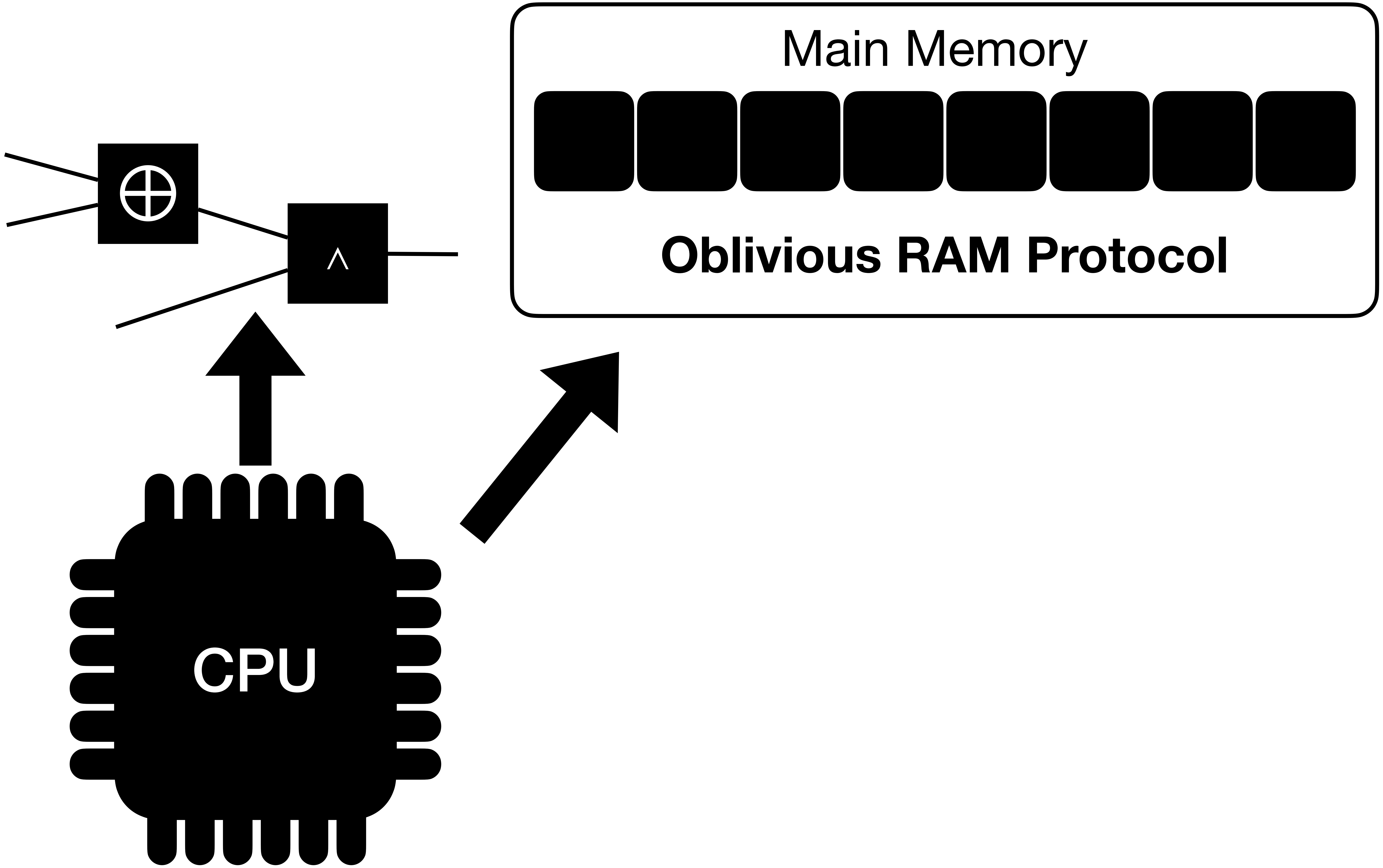
Evaluator



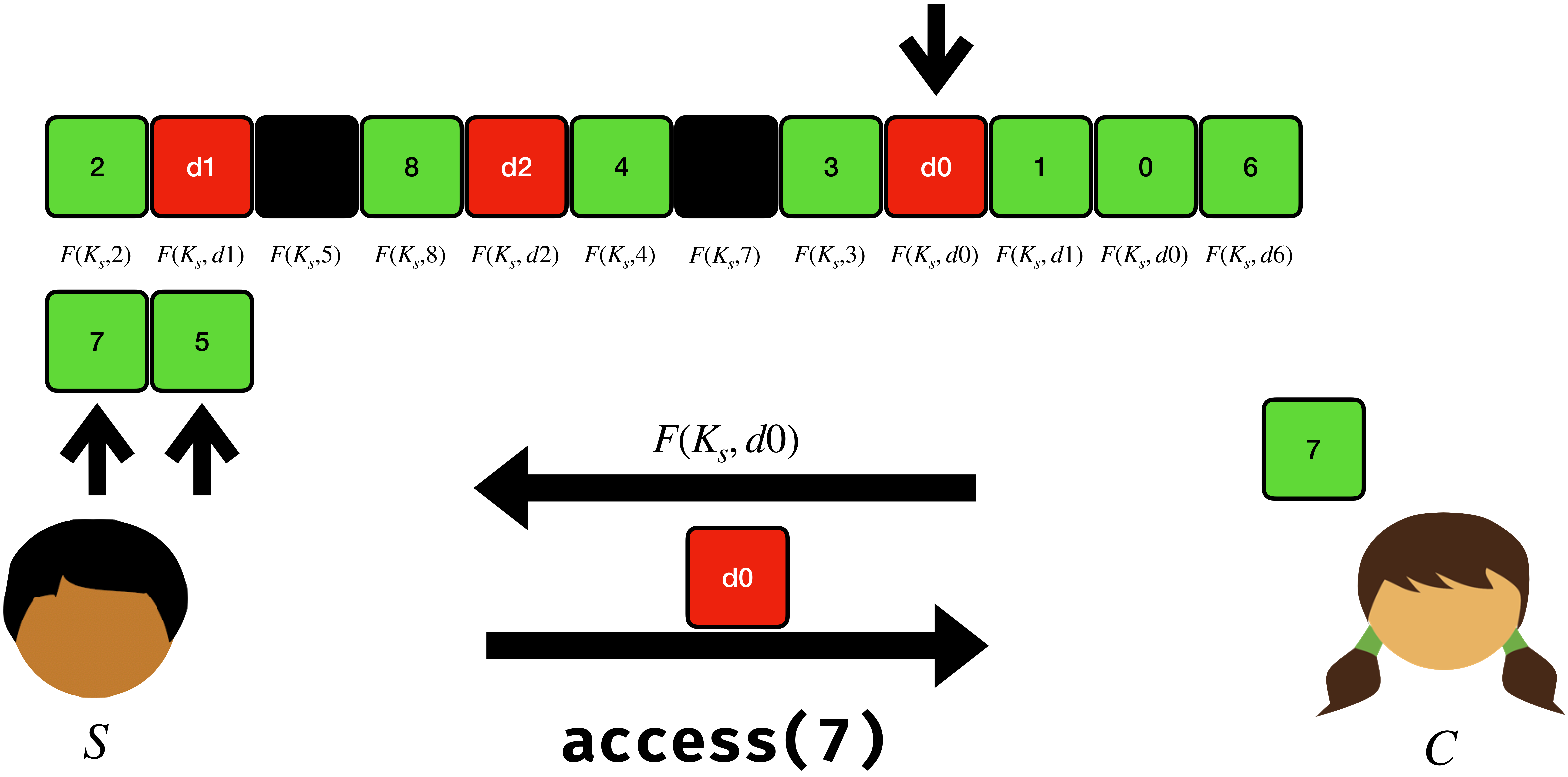
If all opened GC are well-formed, parties continue

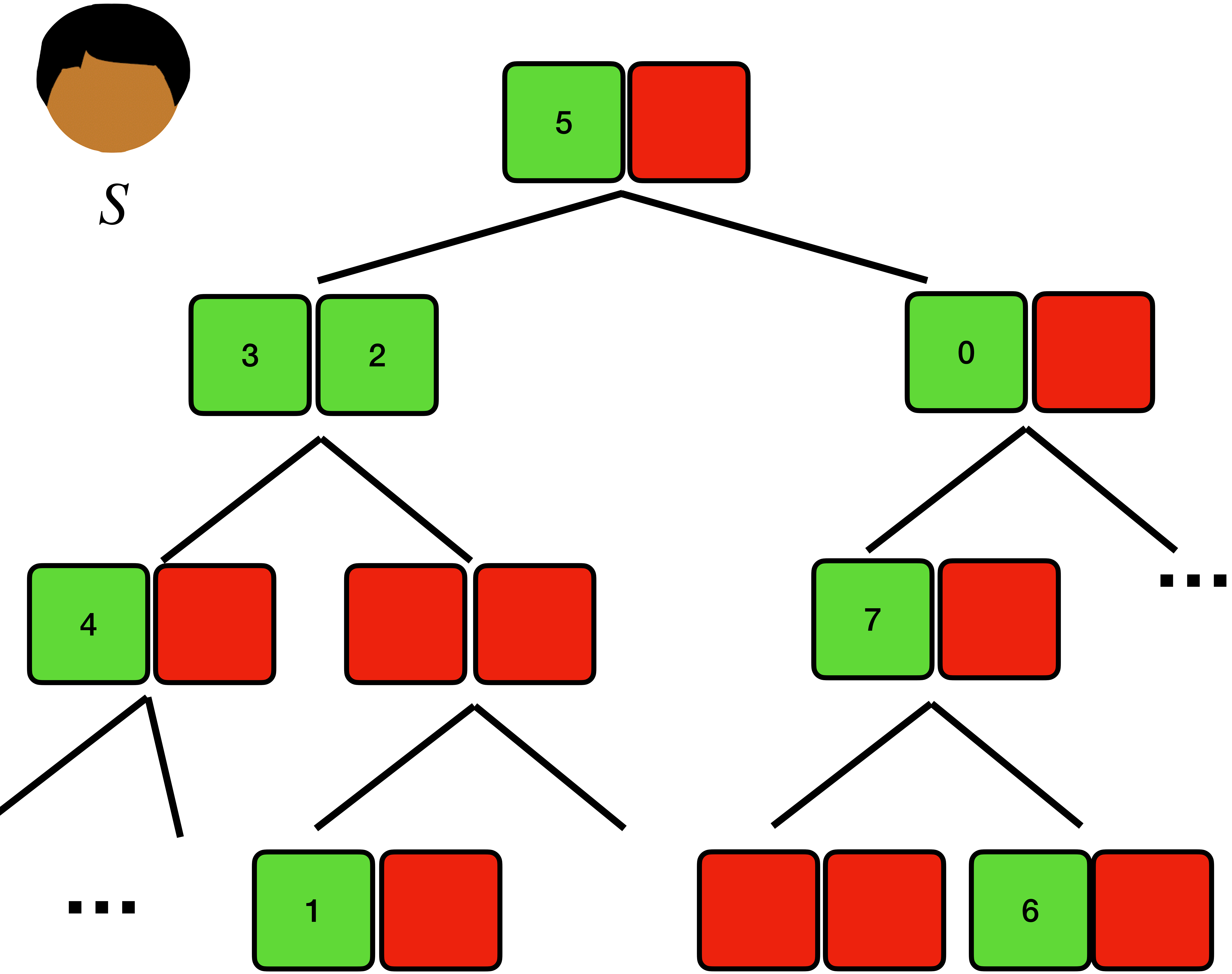


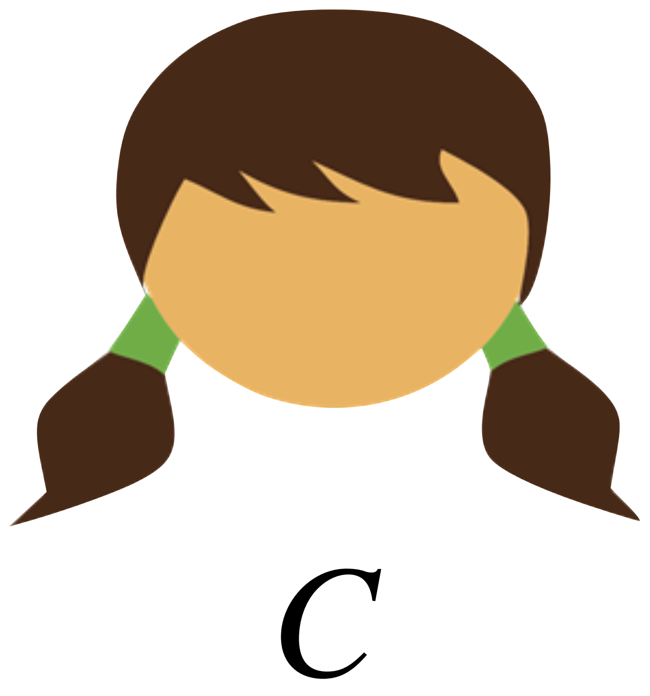
$\mathcal{F}_{\text{pre}}$
Doubly authenticated multiplication triples
G
$\Delta \xleftarrow{\$} \{0,1\}^\lambda$
Garble
E
$\mu \xleftarrow{\$} \{0,1\}^\lambda$
Evaluate
Garbled Circuit
Set E's input
Set G's input



random access machine


2
d1
8
d2
4
3
d0
1
0
6
$F(K_s,2)$
$F(K_s,d1)$
$F(K_s,5)$
$F(K_s,8)$
$F(K_s,d2)$
$F(K_s,4)$
$F(K_s,7)$
$F(K_s,3)$
$F(K_s,d0)$
$F(K_s,d1)$
$F(K_s,d0)$
$F(K_s,d6)$
7
5
$F(K_s,d0)$
d0
7
S
access(7)
C





Path Invariant: Each node is assigned a uniformly random leaf

Logical address	Leaf
0	10
1	5
2	7
...	...

Position Map

ORAM Lower Bound

Natural question: How low can we go in terms of overhead?

Yes, There is an Oblivious RAM Lower Bound!

Kasper Green Larsen* and Jesper Buus Nielsen**

[1] Computer Science. Aarhus University
[2] Computer Science & DIGIT, Aarhus University

Abstract. An Oblivious RAM (ORAM) introduced by Goldreich and Ostrovsky [JACM'96] is a (possibly randomized) RAM, for which the memory access pattern reveals no information about the operations performed. The main performance metric of an ORAM is the bandwidth overhead, i.e., the multiplicative factor extra memory blocks that must be accessed to hide the operation sequence. In their seminal paper introducing the ORAM, Goldreich and Ostrovsky proved an amortized $\Omega(\lg n)$ bandwidth overhead lower bound for ORAMs with memory size n. Their lower bound is very strong in the sense that it applies to the "offline" setting in which the ORAM knows the entire sequence of operations ahead of time.
However, as pointed out by Boyle and Naor [ITCS'16] in the paper "Is there an oblivious RAM lower bound?", there are two caveats with the lower bound of Goldreich and Ostrovsky: (1) it only applies to "balls in bins" algorithms, i.e., algorithms where the ORAM may only shuffle blocks around and not apply any sophisticated encoding of the data, and (2), it only applies to statistically secure constructions. Boyle and Naor showed that removing the "balls in bins" assumption would result in super linear lower bounds for sorting circuits, a long standing open problem in circuit complexity. As a way to circumventing this barrier, they also proposed a notion of an "online" ORAM, which is an ORAM that remains secure even if the operations arrive in an online manner. They argued that most known ORAM constructions work in the online setting as well.
Our contribution is an $\Omega(\lg n)$ lower bound on the bandwidth overhead of any online ORAM, even if we require only computational security and allow arbitrary representations of data, thus greatly strengthening the lower bound of Goldreich and Ostrovsky in the online setting. Our lower bound applies to ORAMs with memory size n and any word size $r \geq 1$. The bound therefore asymptotically matches the known upper bounds

Fact (informal): Any secure ORAM must incur overhead at least $\Omega(\log n)$

Combines two concepts:

- All access patterns should look the same to the server
- Certain access patterns will **force** the client to save its data on the server, then retrieve it later

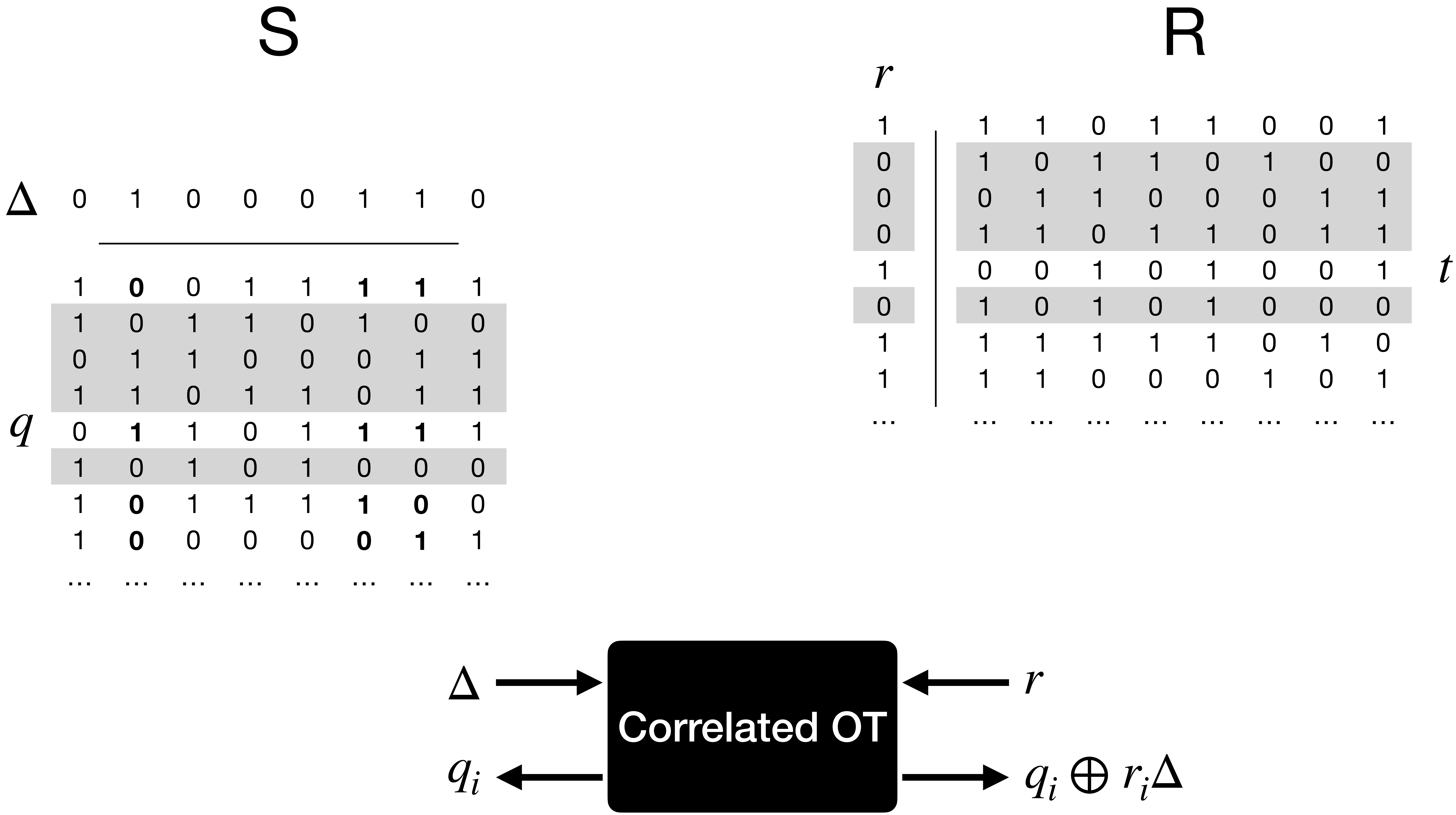
OT Extension

In MPC (e.g., GMW), we need lots of short OTs

Can we turn a few OTs into a lot of OTs?

λ *base* OTs → n *extended* OTs

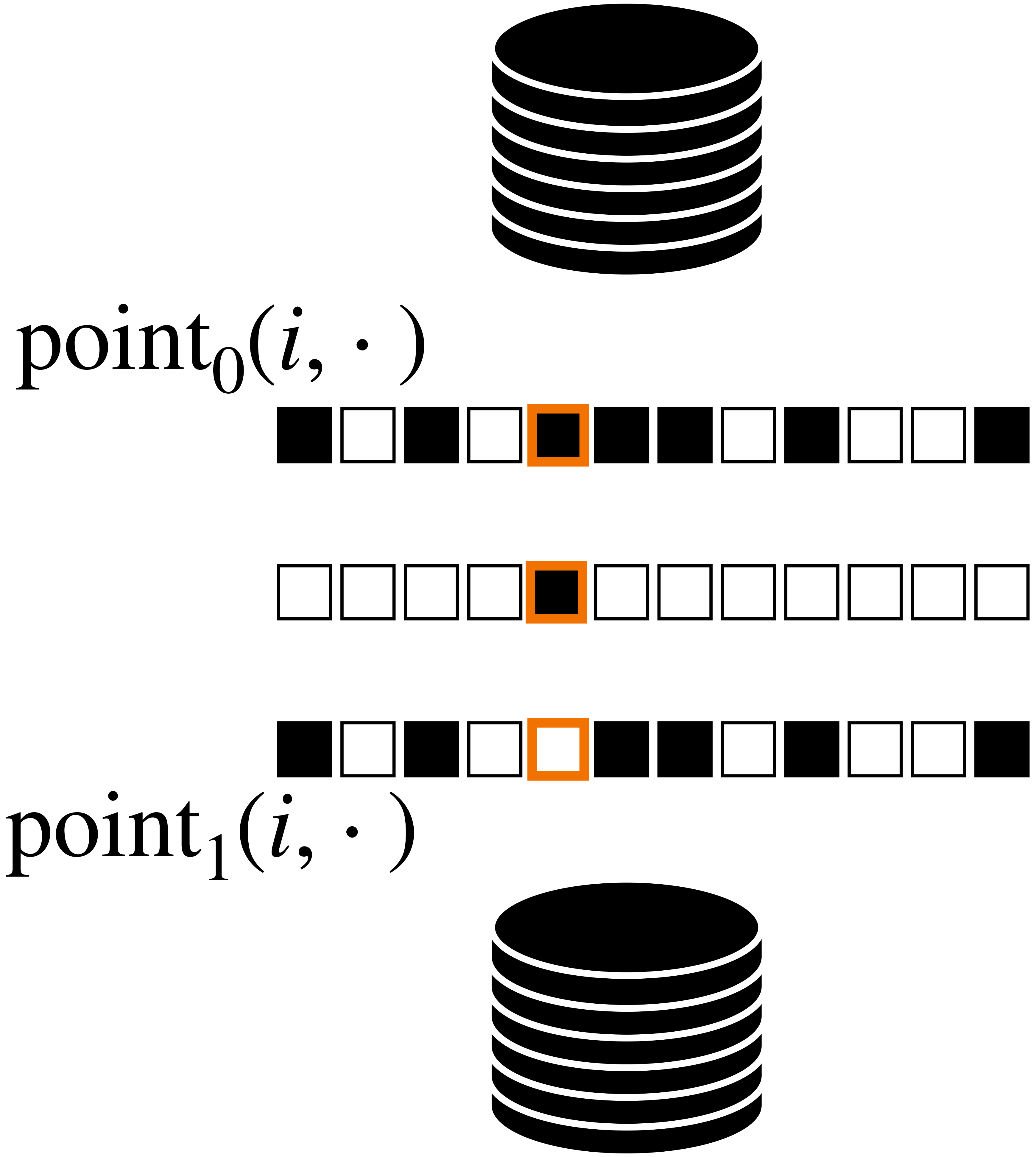
Public key   Symmetric key


S
R
r
Δ 0 1 0 0 0 1 1 0
1 0 0 1 1 1 1 1
1 0 1 1 0 1 0 0
0 1 1 0 0 0 1 1
1 1 0 1 1 0 1 1
q 0 1 1 0 1 1 1 1
1 0 1 0 1 0 0 0
1 0 1 1 1 1 0 0
1 0 0 0 0 0 1 1
1 | 1 1 0 1 1 0 0 1
0 | 1 0 1 1 0 1 0 0
0 | 0 1 1 0 0 0 1 1
0 | 1 1 0 1 1 0 1 1
1 | 0 0 1 0 1 0 0 1 t
0 | 1 0 1 0 1 0 0 0
1 | 1 1 1 1 1 0 1 0
1 | 1 1 0 0 0 1 0 1
Δ
r
Correlated OT
qi
qi ⊕ riΔ

Distributed Point Function

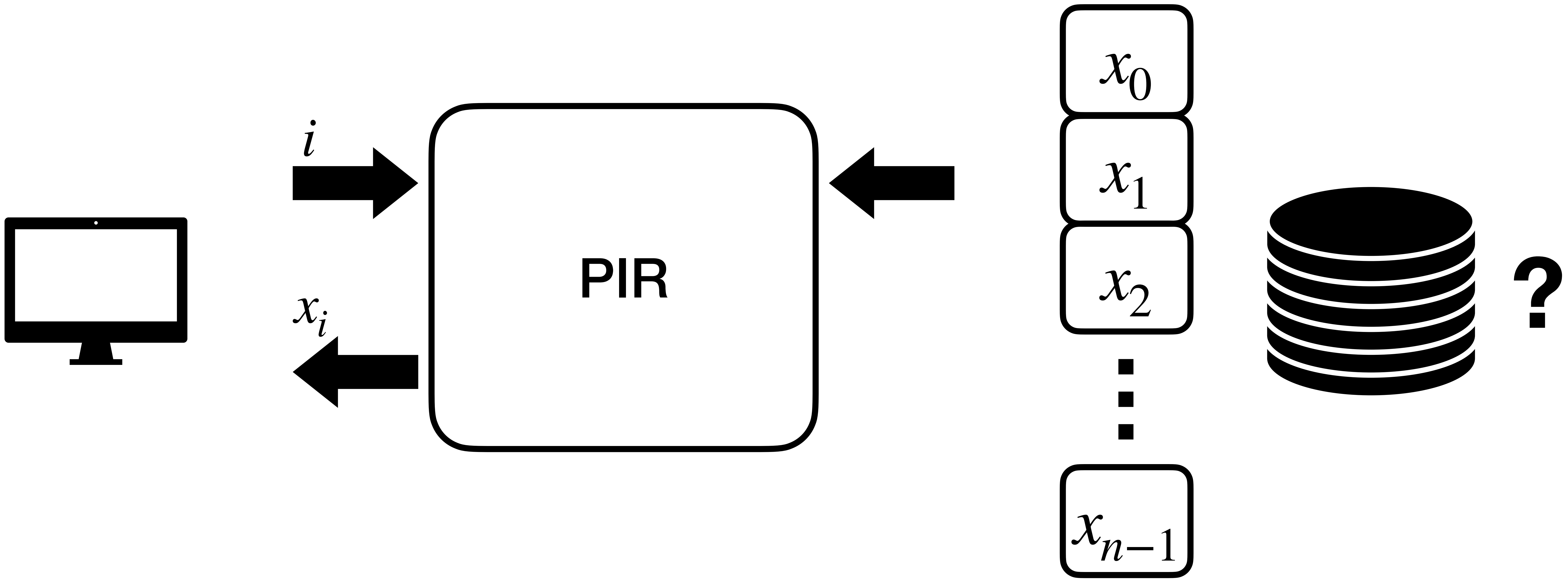
$f \in \mathscr{F}$

$$
\begin{tikzcd}
 & f_0 \arrow[rr, "\mathrm{Eval}(f_0, x)"] & & f_0(x) \arrow[dr] & \\
f \arrow[ur, "\mathrm{Gen}"'] \arrow[dr] & & & \oplus & f(x) \\
 & f_1 \arrow[rr, "\mathrm{Eval}(f_1, x)"] & & f_1(x) \arrow[ur] &
\end{tikzcd}
$$

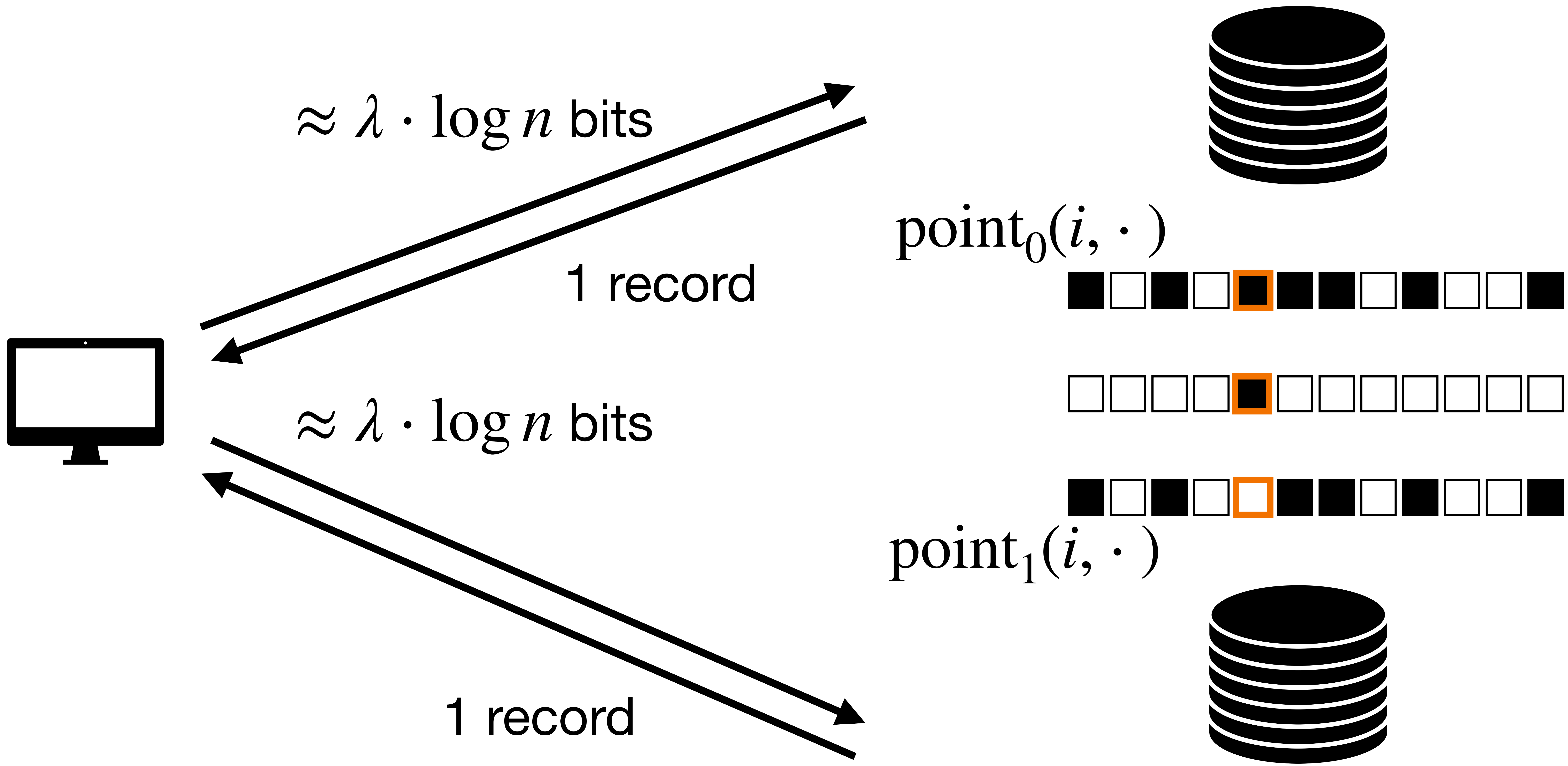


$$\mathrm{point}(i, x) = \begin{cases} 1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases}$$

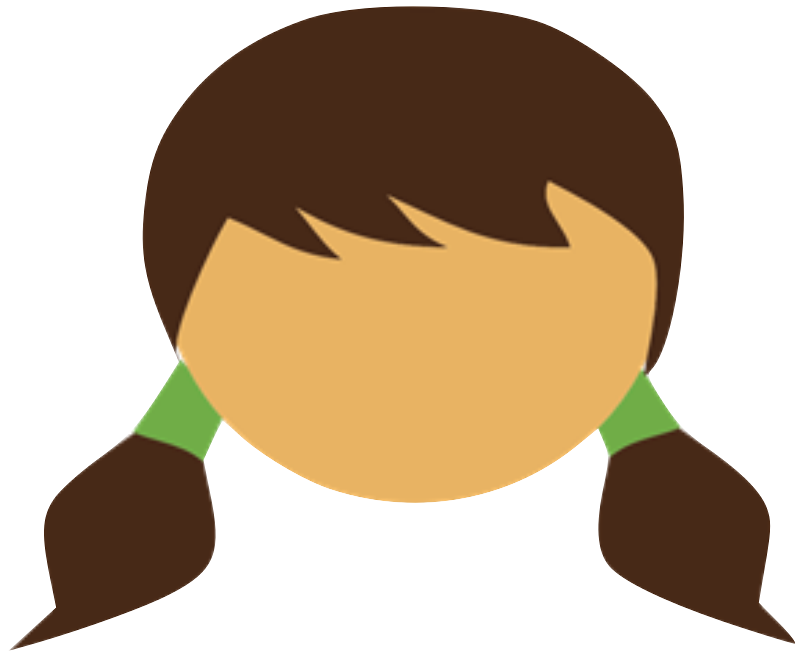
Private Information Retrieval

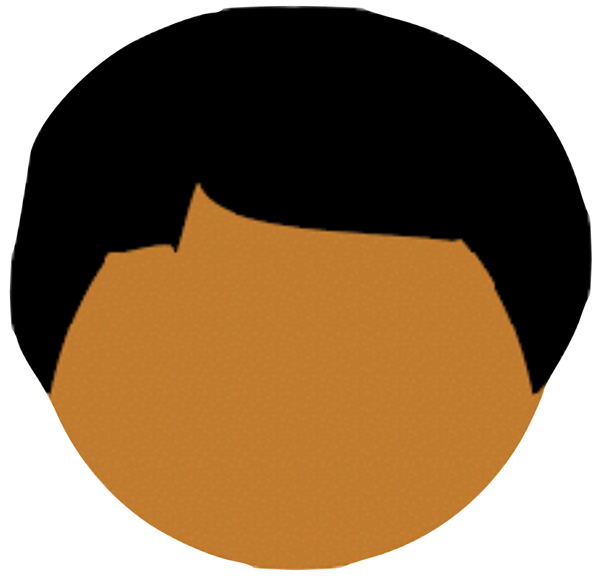


Client wishes to privately query one element from a large database


$\approx \lambda \cdot \log n$ bits
1 record
$\mathrm{point}_0(i, \cdot)$
$\approx \lambda \cdot \log n$ bits
$\mathrm{point}_1(i, \cdot)$
1 record

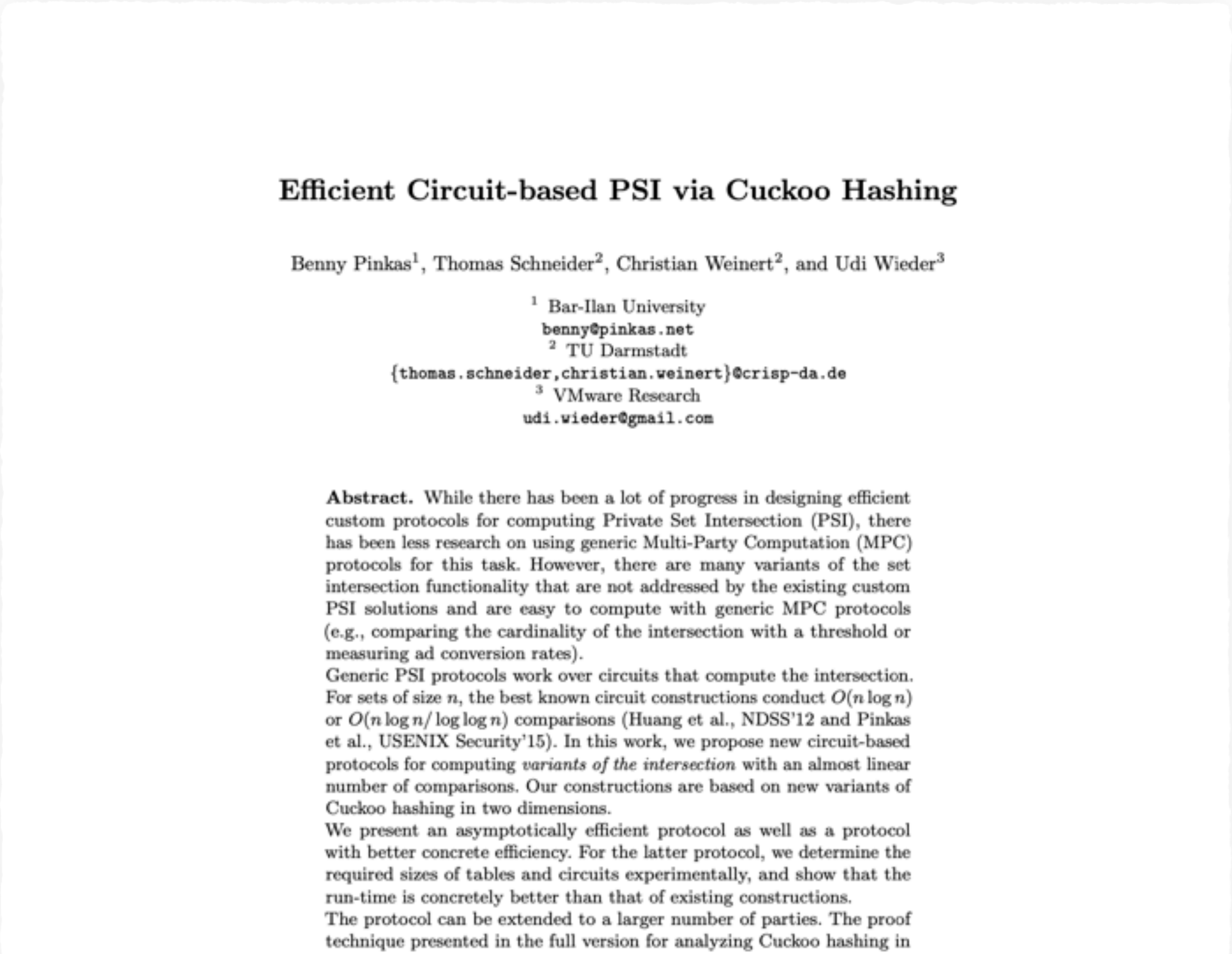
PSI





{13, **17**, 25, **45**, 52, 101}

{1, 4, **17**, 19, 21, **45**, 100}



Efficient Circuit-based PSI via Cuckoo Hashing

Benny Pinkas[1], Thomas Schneider[2], Christian Weinert[2], and Udi Wieder[3]

[1] Bar-Ilan University
benny@pinkas.net
[2] TU Darmstadt
{thomas.schneider,christian.weinert}@crisp-da.de
[3] VMware Research
udi.wieder@gmail.com

Abstract. While there has been a lot of progress in designing efficient custom protocols for computing Private Set Intersection (PSI), there has been less research on using generic Multi-Party Computation (MPC) protocols for this task. However, there are many variants of the set intersection functionality that are not addressed by the existing custom PSI solutions and are easy to compute with generic MPC protocols (e.g., comparing the cardinality of the intersection with a threshold or measuring ad conversion rates).

Generic PSI protocols work over circuits that compute the intersection. For sets of size n, the best known circuit constructions conduct $O(n \log n)$ or $O(n \log n / \log \log n)$ comparisons (Huang et al., NDSS'12 and Pinkas et al., USENIX Security'15). In this work, we propose new circuit-based protocols for computing *variants of the intersection* with an almost linear number of comparisons. Our constructions are based on new variants of Cuckoo hashing in two dimensions.

We present an asymptotically efficient protocol as well as a protocol with better concrete efficiency. For the latter protocol, we determine the required sizes of tables and circuits experimentally, and show that the run-time is concretely better than that of existing constructions.

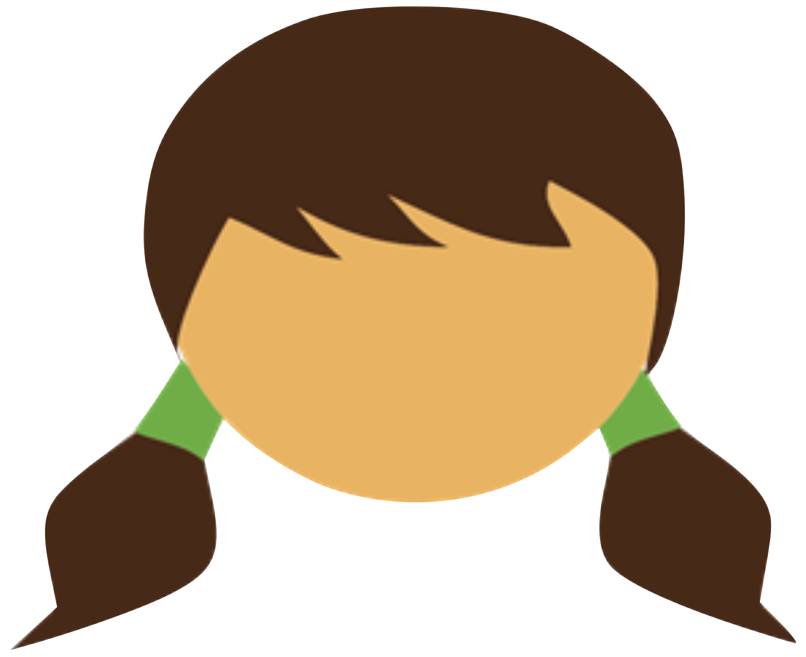
The protocol can be extended to a larger number of parties. The proof technique presented in the full version for analyzing Cuckoo hashing in

{17,45}

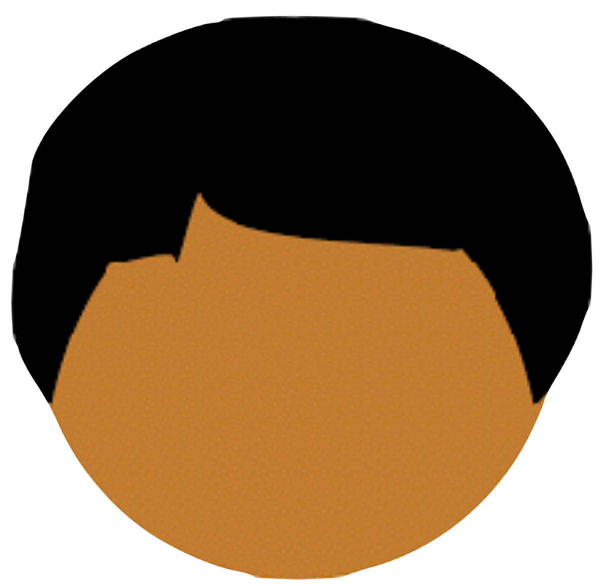
Special case of MPC

“Just use MPC”

Because it is a special case, we can hope for much more efficiency

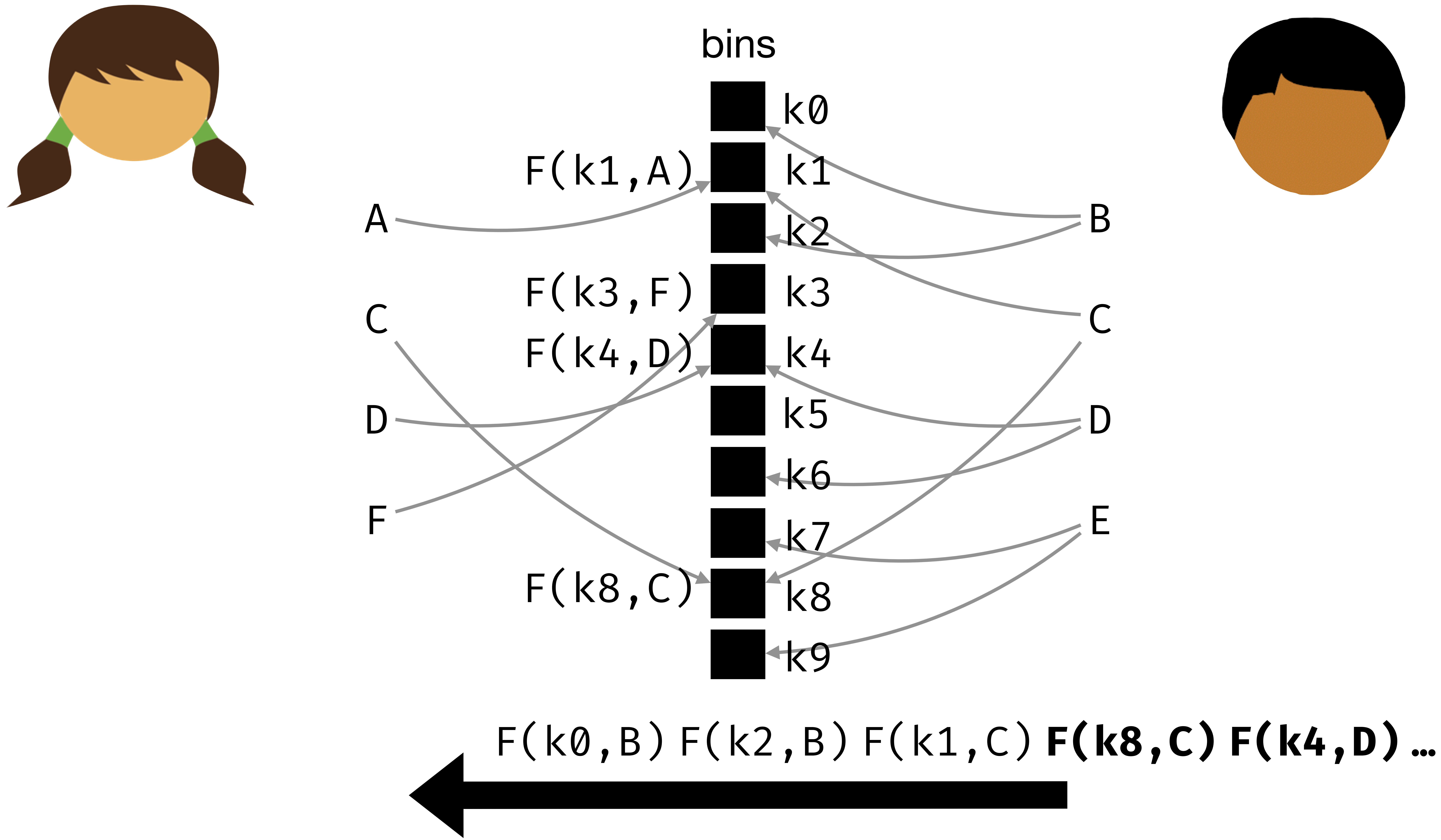


Batched Oblivious PRF



Essentially batched 1-out-of-N OT

x0	F(k0, x0)	0	k0
x1	F(k1, x1)	1	k1
x2	F(k2, x2)	2	k2
x3	F(k3, x3)	3	k3
x4	F(k4, x4)	4	k4
x5	F(k5, x5)	5	k5


bins
k0
F(k1,A)
k1
A
k2
B
F(k3,F)
k3
C
C
F(k4,D)
k4
D
k5
D
k6
F
k7
E
F(k8,C)
k8
k9
F(k0,B) F(k2,B) F(k1,C) **F(k8,C) F(k4,D)** …

Setting

Semi-honest Security

Malicious Security

Zero Knowledge

General-Purpose Tools

GMW Protocol

- Multi-party
- Multi-round

Garbled Circuit

- Constant Round
- Two Party

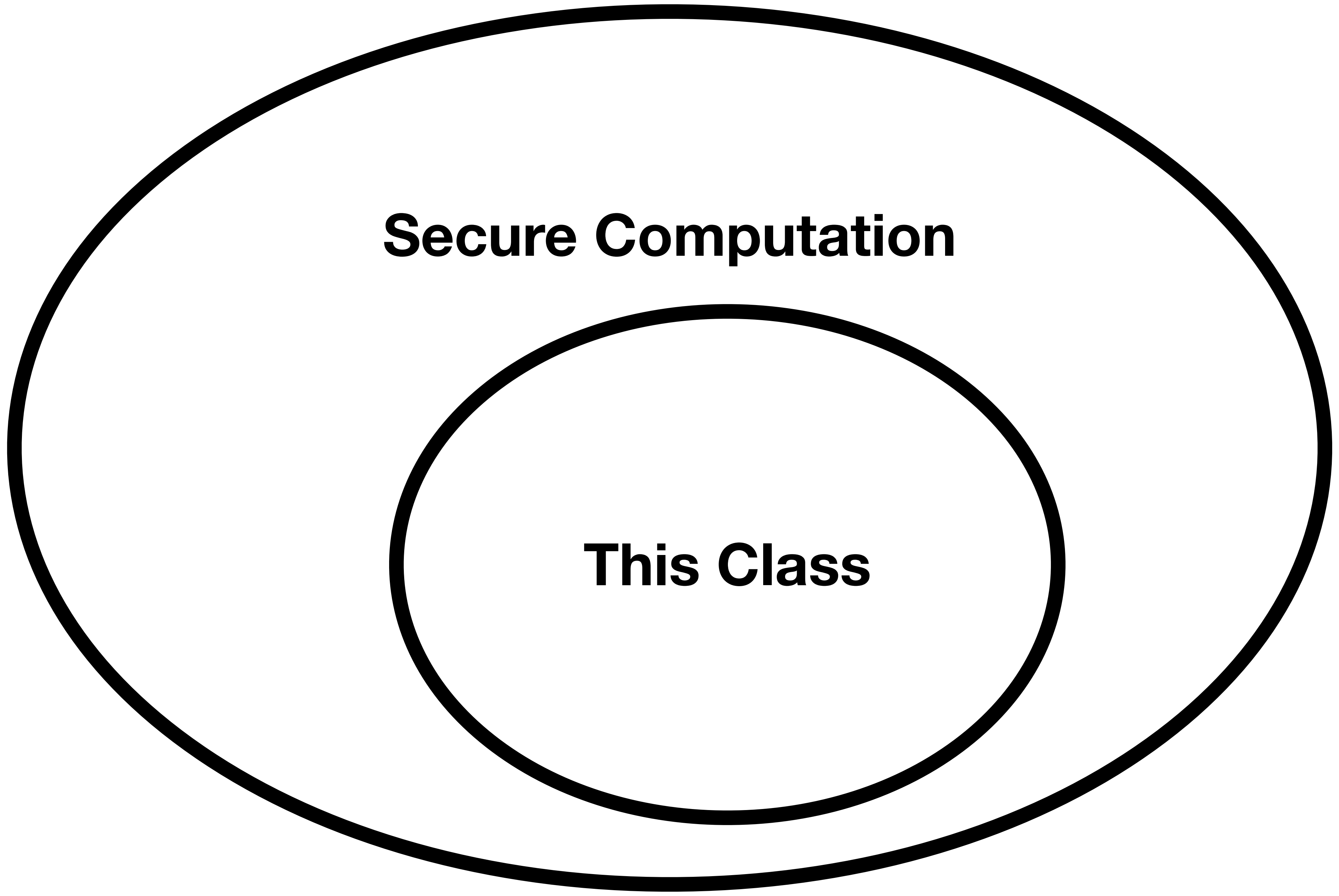
Primitives

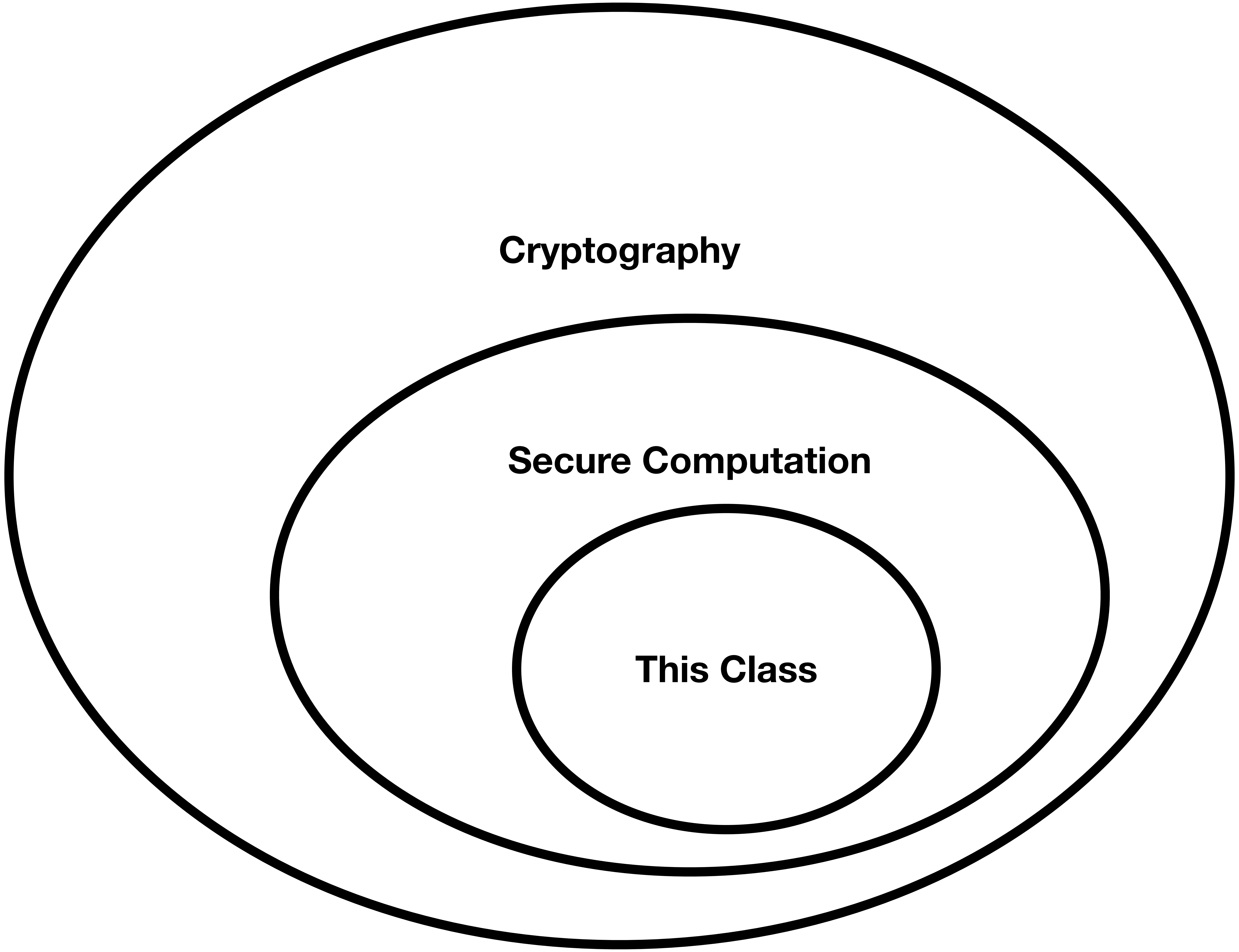
Oblivious Transfer

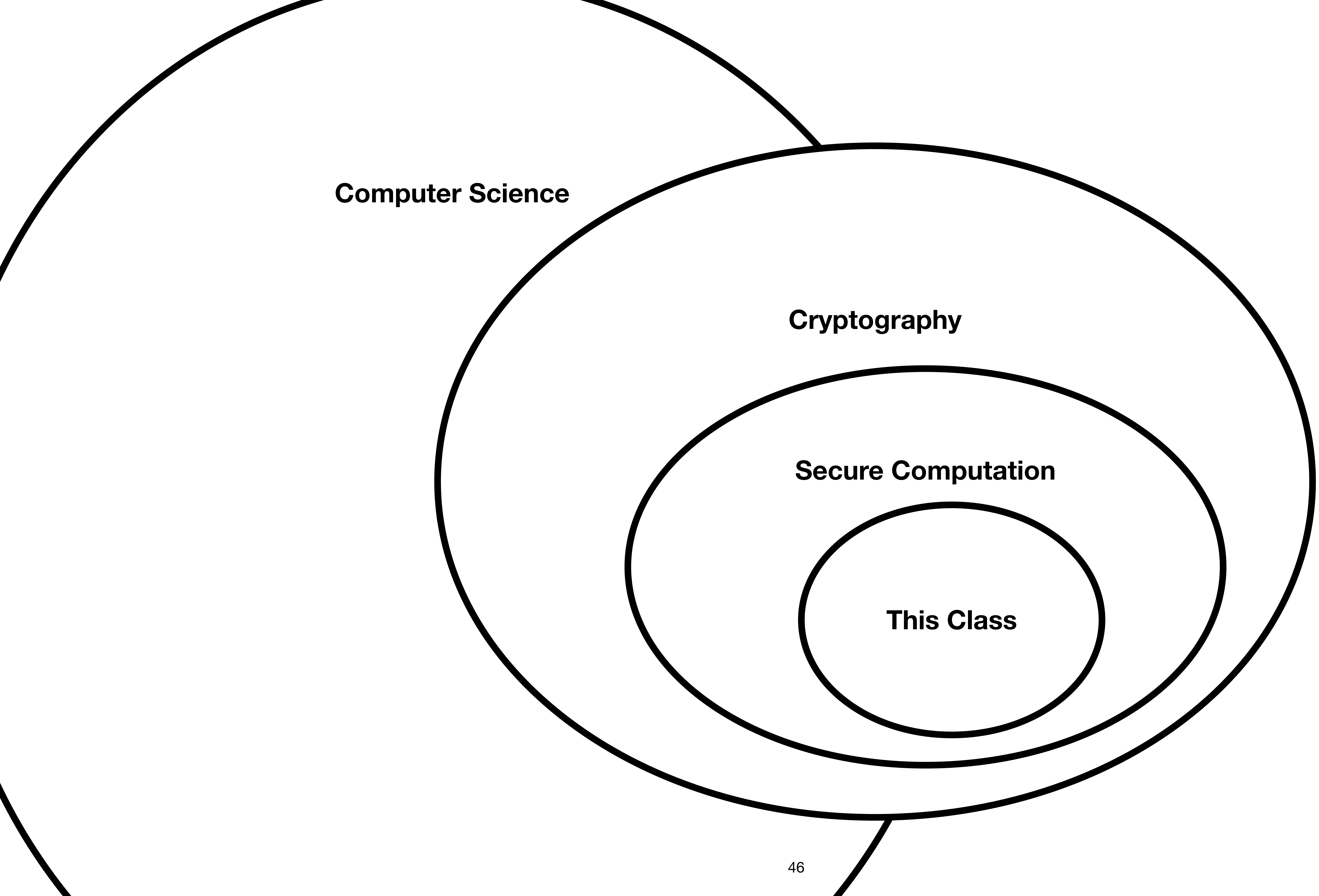
Pseudorandom generators/functions/encryption

Commitments

ORAM


Secure Computation
This Class


Cryptography
Secure Computation
This Class


Computer Science
Cryptography
Secure Computation
This Class