Multiparty Key Exchange, Efficient Traitor Tracing, and More from IO

Dan Boneh **Mark Zhandry**
Stanford University

Program Obfuscation

Intuition: Scramble a program

- Preserve functionality
- Hide implementation details

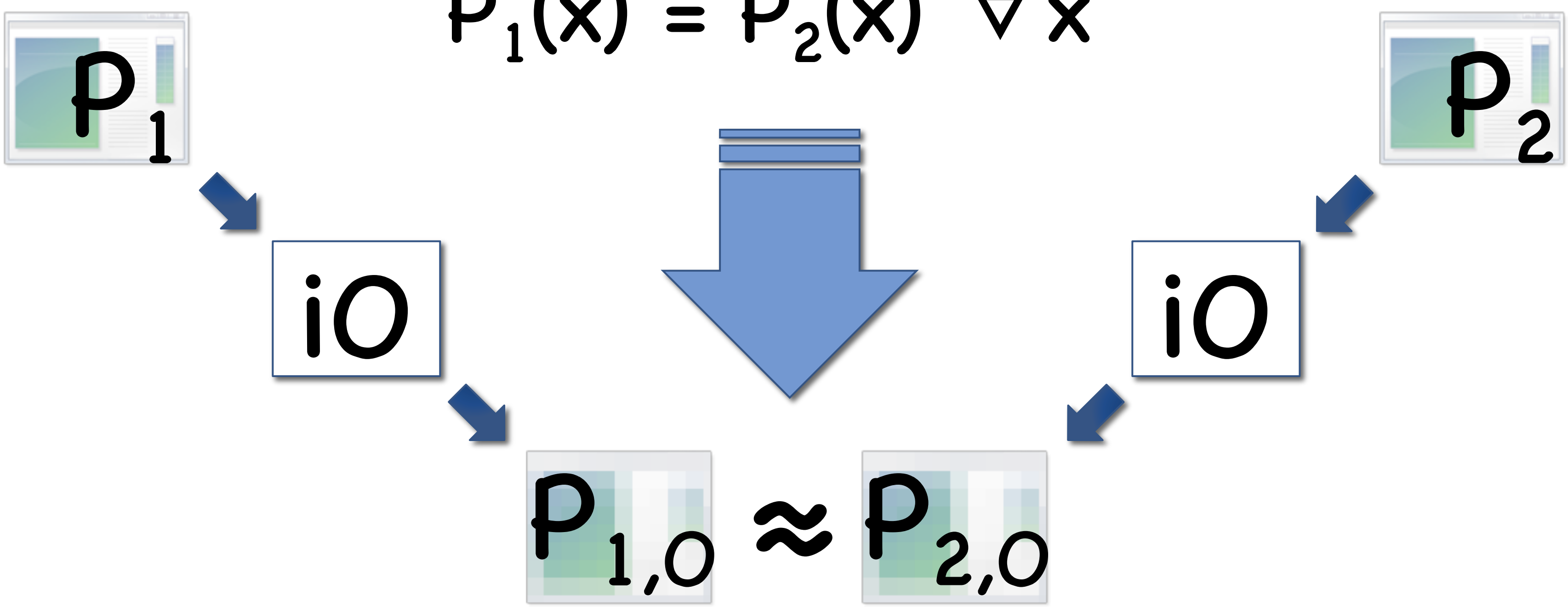
Applications:

- IP Protection
- Software Watermarking
- **Crypto**

Indist. Obfuscation (iO) [BGI+'01, GR'07]

If two programs have same functionality, obfuscations are indistinguishable

$$P_1(x) = P_2(x) \ \forall x$$



Big questions: How to build? How to use?

Indistinguishability Obfuscation (iO)

An exploding field:

- **[GGH+'13] First candidate iO construction**
  - Built from multilinear maps
  - First application: functional encryption
- [BR'13, BGK+'13, …] Additional constructions
- [SW'13, GGHR'13, **BZ'13**, ABGSZ'13, …] Uses
  - Public key encryption, signatures, deniable encryption, multiparty key exchange, MPC, …
- [BCPR'13, MR'13, BCP'13, …] Further Investigation

Our Results

Non-interactive multiparty key exchange

- First scheme without trusted setup

Efficient broadcast encryption

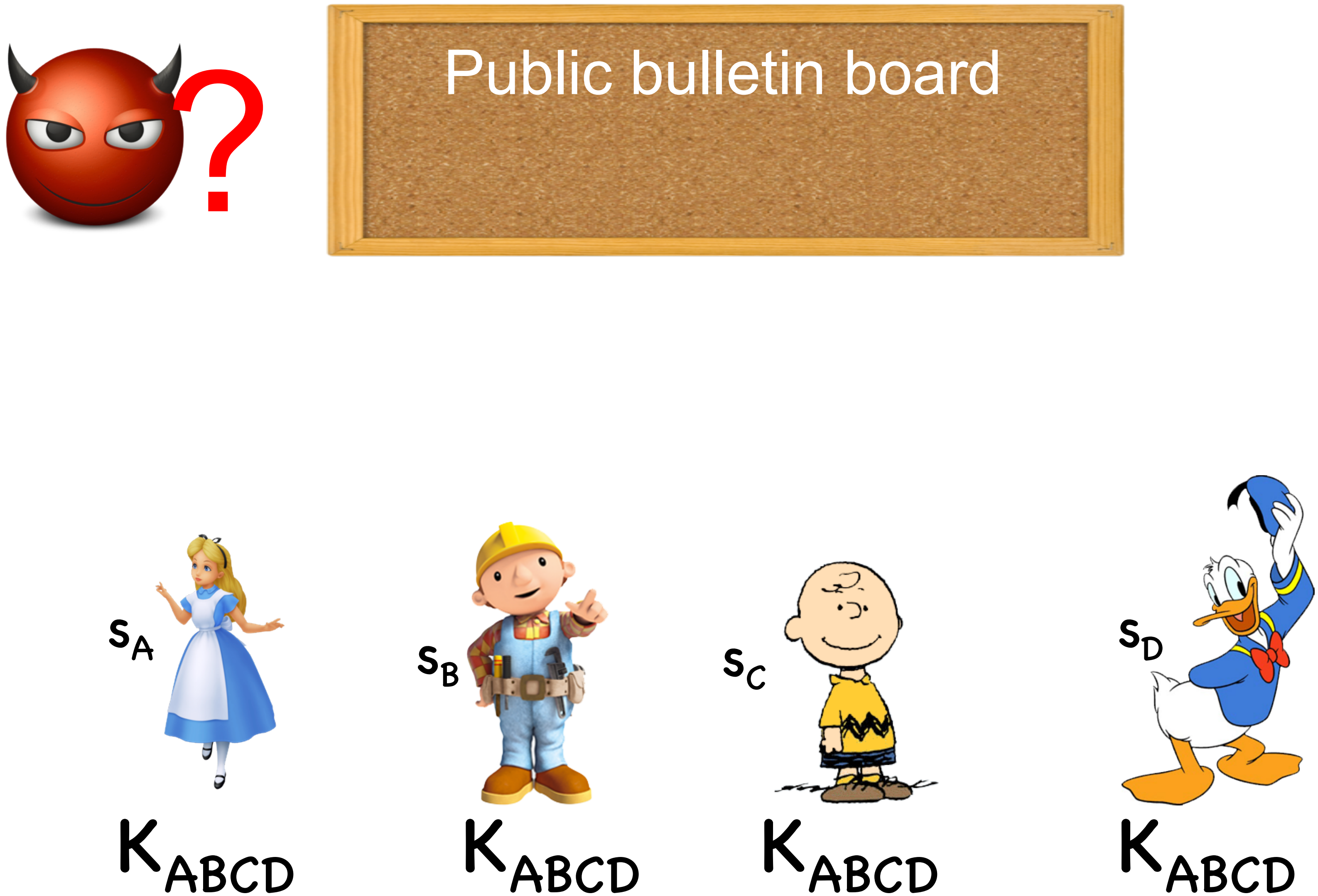
- Constant size ciphertext and secret keys
- First *distributed* system: users generate keys themselves

Efficient traitor tracing

- Shortest secret keys, ciphertexts, known
- Resolves open problem in Differential Privacy [DNR+09]

MULTIPARTY KEY EXCHANGE

(Non-Interactive) Multiparty Key Exchange



History

2 parties: Diffie Hellman Protocol [DH'76]

3 parties: Bilinear maps [Joux'2000]

n>3 parties: Multilinear maps [BS'03,GGH'13,CLT'13]

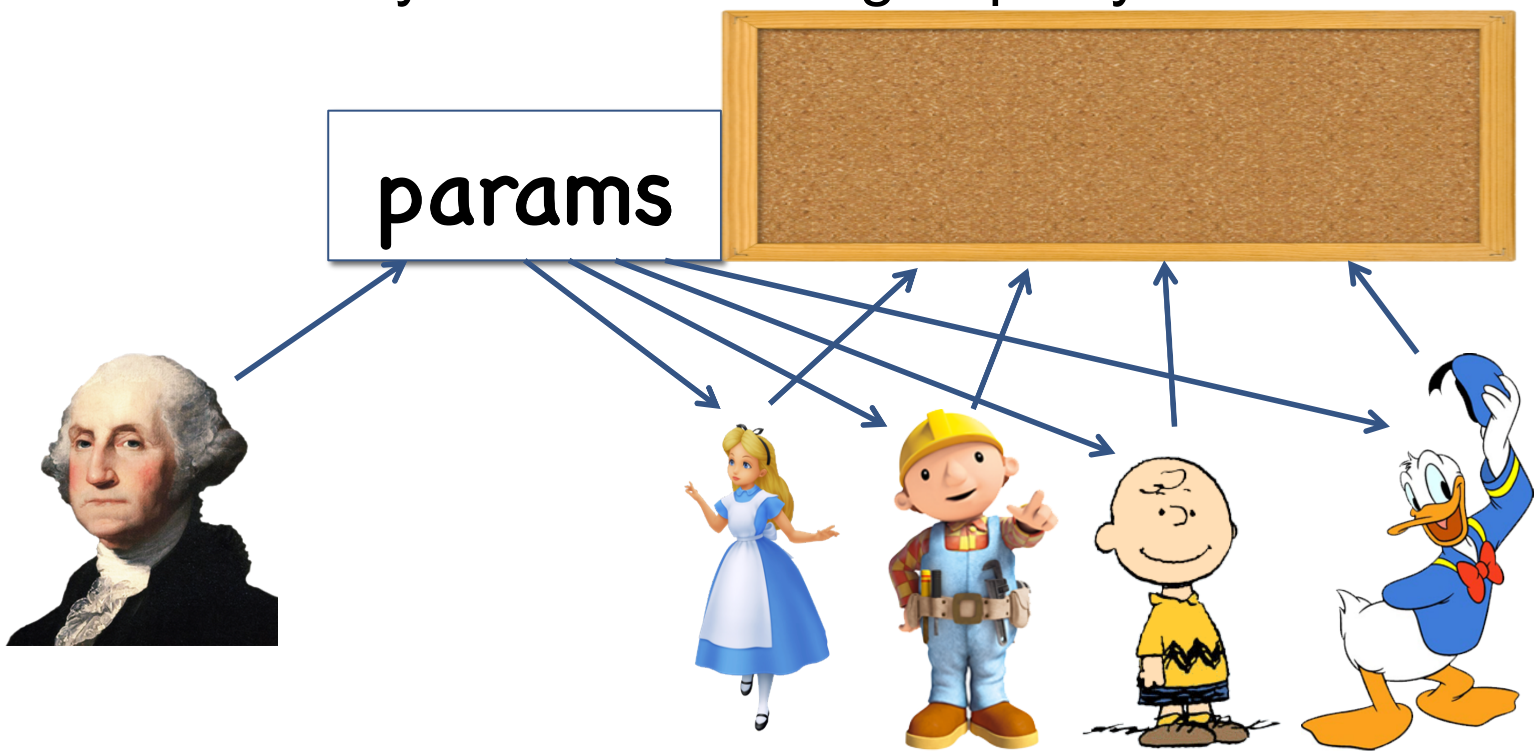
- Requires trusted setup phase

Our work: **n** parties, no trusted setup

Prior Constructions for **n>3**

First achieved using multilinear maps [GGH'13,CLT'13]

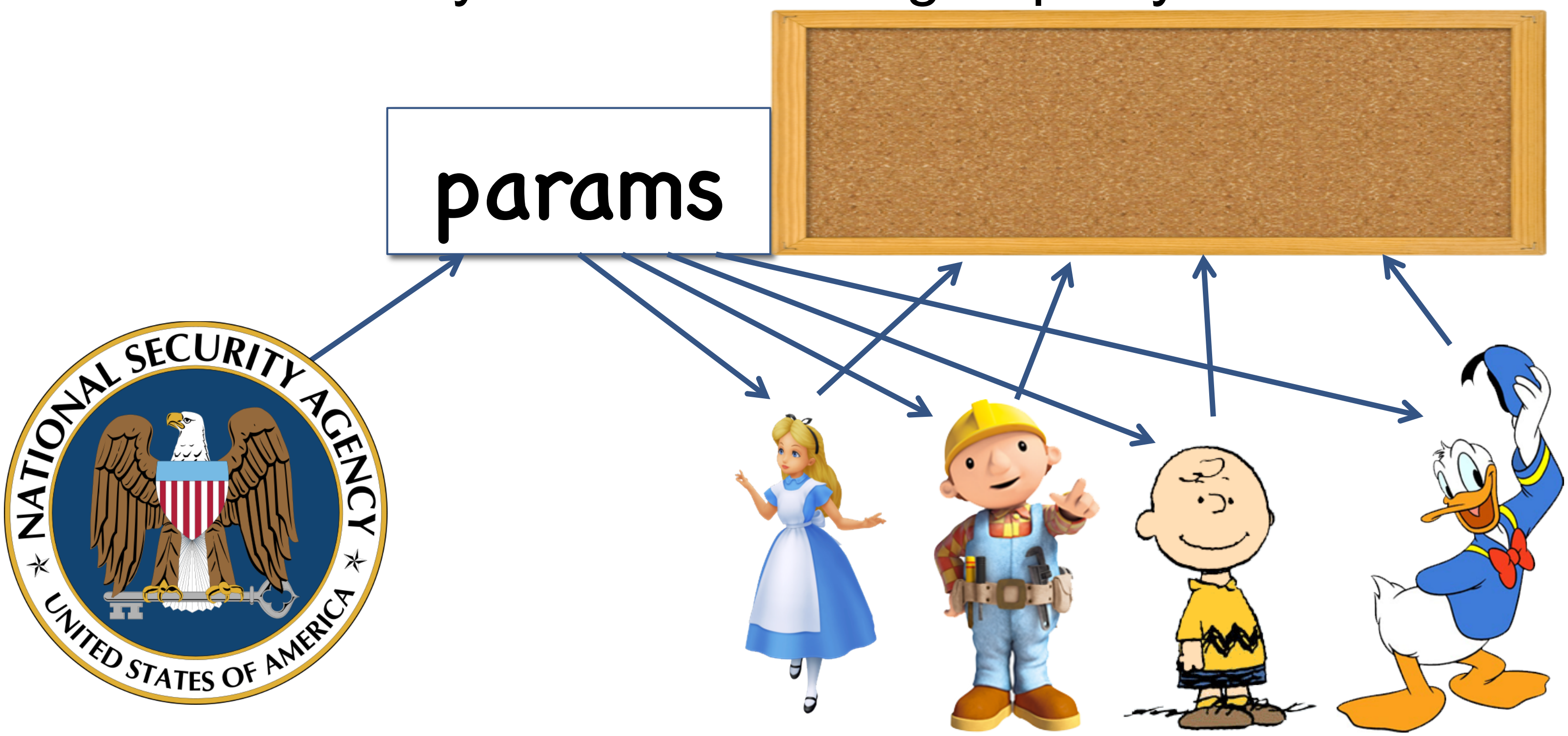
- These constructions all require trusted setup **before** protocol is run
- Trusted authority can also learn group key



Prior Constructions for **n>3**

First achieved using multilinear maps [GGH'13,CLT'13]

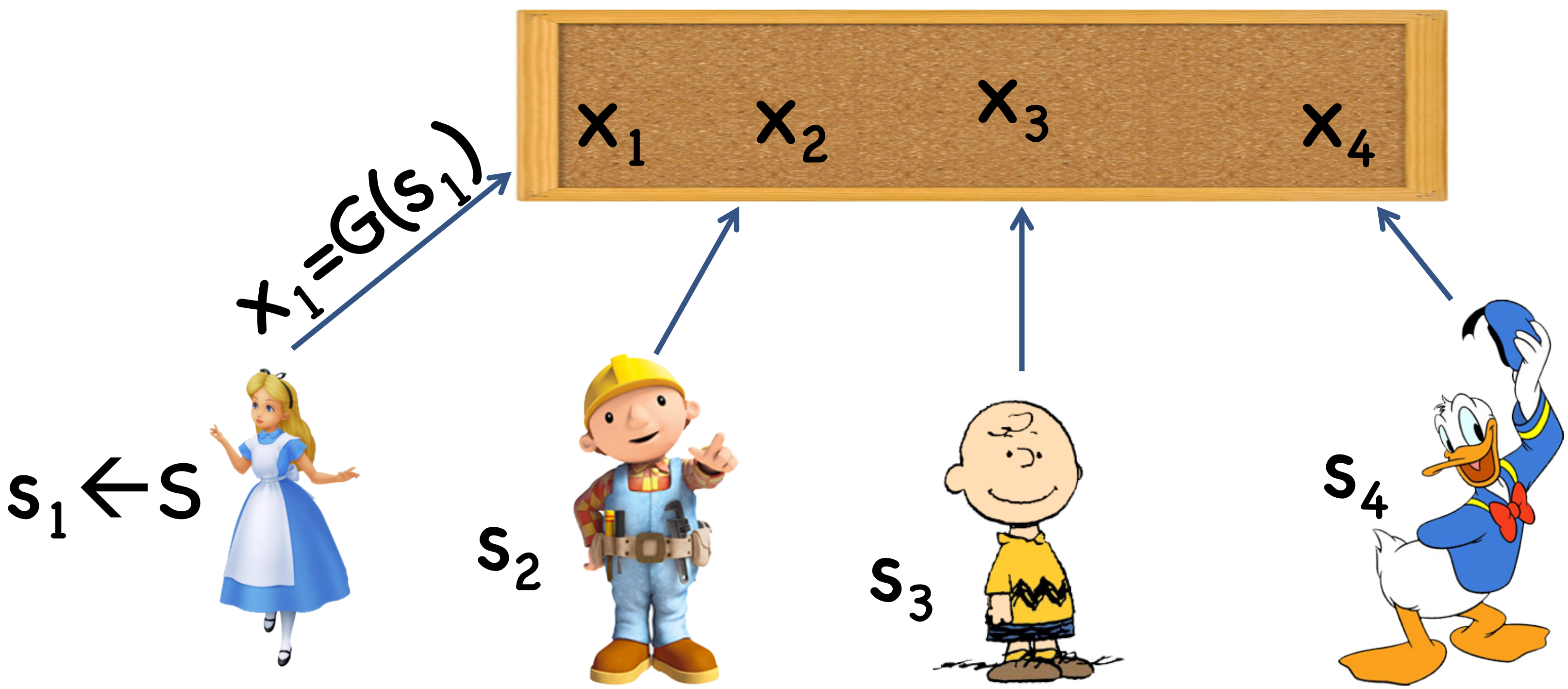
- These constructions all require trusted setup **before** protocol is run
- Trusted authority can also learn group key



Starting point for our construction

Building blocks:

- One-way function **G:S → X**
- Pseudorandom function (PRF) **F**



Shared key: $F_k(x_1, x_2, x_3, x_4)$ ← how to compute securely?

Introduce Trusted Authority (for now)

k

$P(y_1, \ldots, y_n, s, i)$ {
    If $G(s) \neq y_i$, output $\perp$
    Otherwise, output $F_k(y_1, \ldots, y_n)$
}

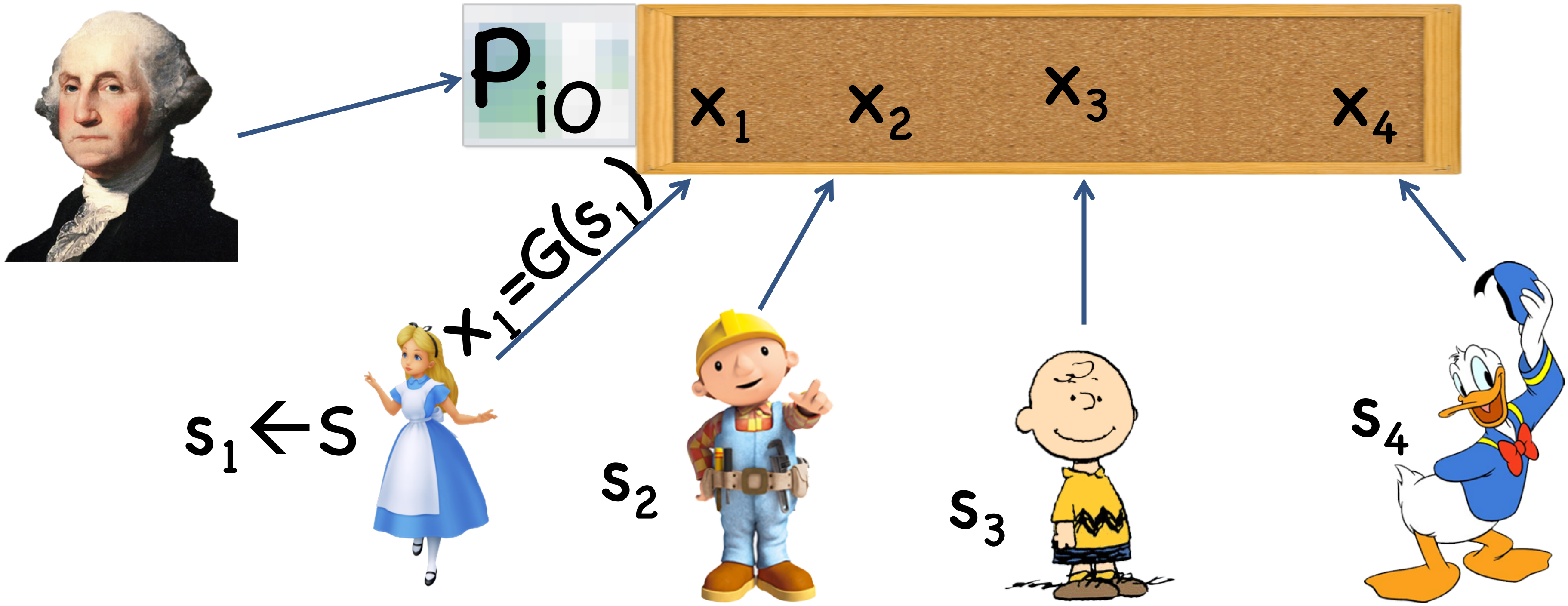
↓

iO

↓

P_{iO}

First attempt



$$K_{ABCD} = P_{iO}(x_1, x_2, x_3, x_4, s_1, 1)$$

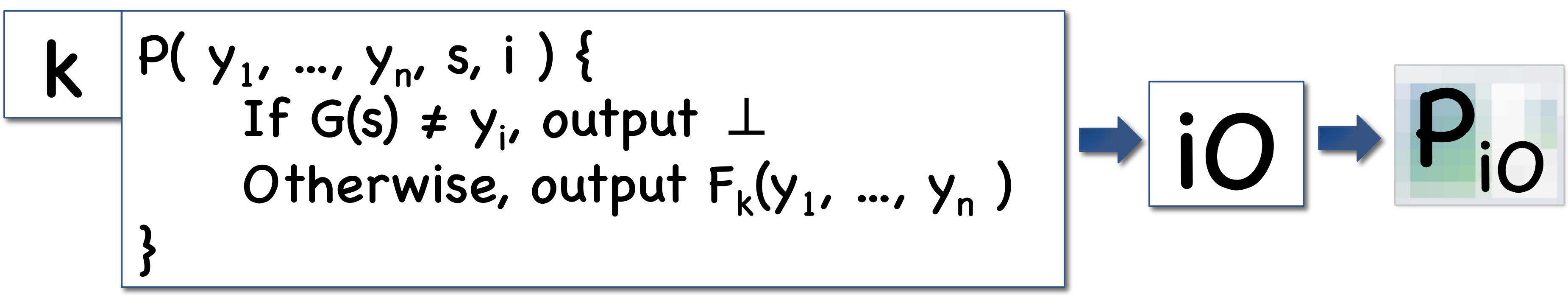
Problems:

- **k** not guaranteed to be hidden using iO
- Still have trusted authority

Removing Trusted Setup

As described, our scheme needs trusted setup

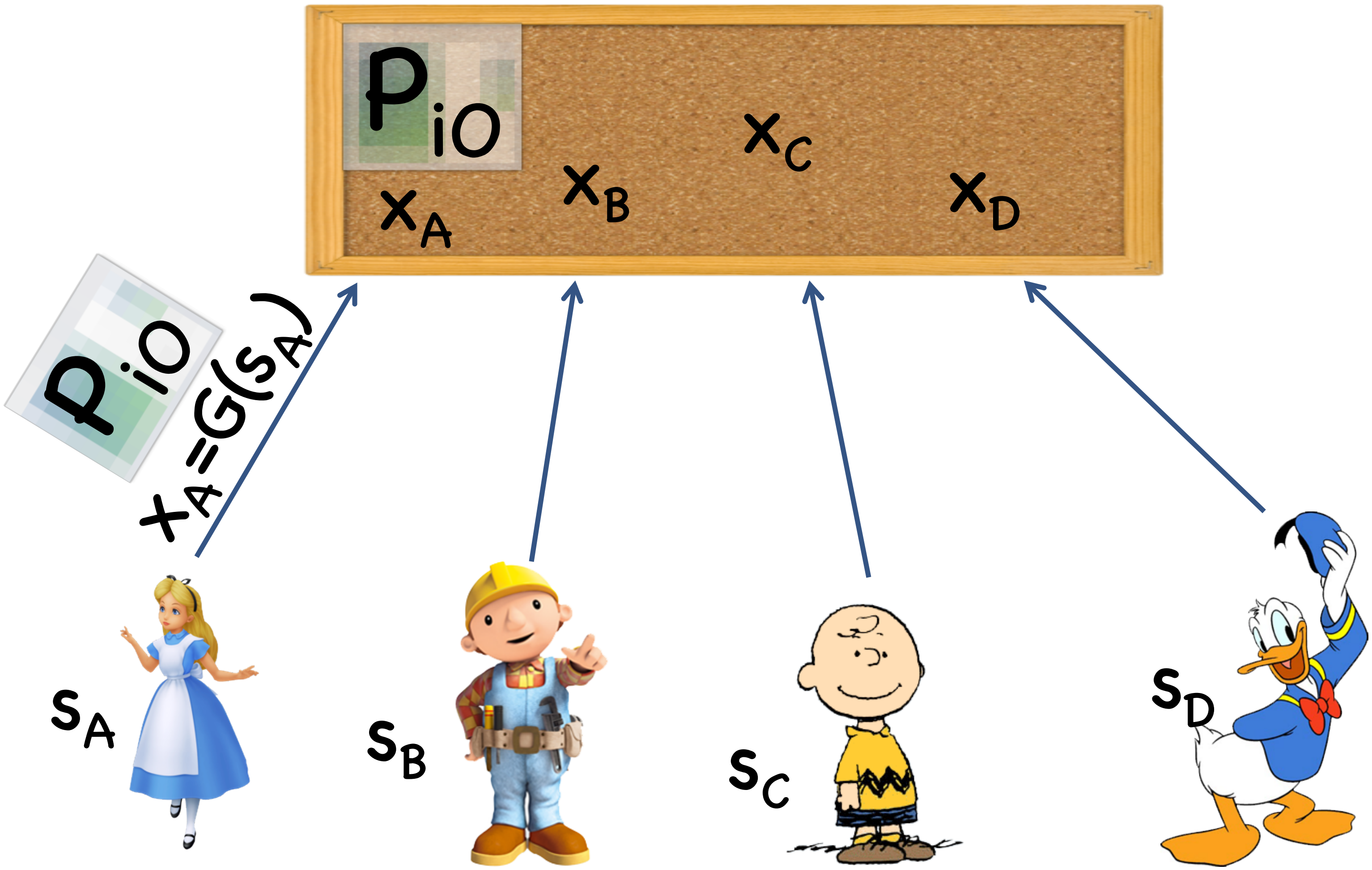
Observation: Obfuscated program can be generated independently of publishing step



Untrusted setup: designate user 1 as "master party"

- generates $\mathbf{P}_{iO}$, sends with $\mathbf{x}_1$

Multiparty Key Exchange Without Trusted Setup



Security equivalent to security of previous scheme

Hiding k

Follow "punctured program" paradigm of SW'13

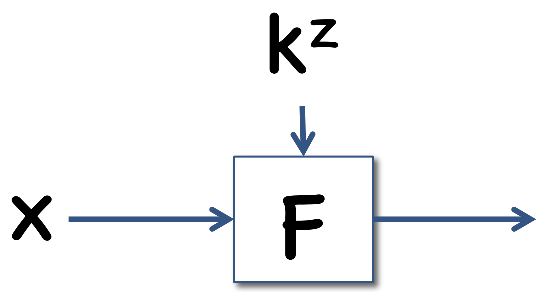
- Use pseudorandom generator for **G**

  $G: S \to X \quad |X| \gg |S|$

  $G(s),\ s \leftarrow S$ indist. from $x \leftarrow X$

- Use special "punctured PRF" for **F** [BW'13, KPTZ'13, BGI'13, SW'13]

  Punctured key $k^z \Rightarrow$ compute $F_k(\,\cdot\,)$ everywhere but z

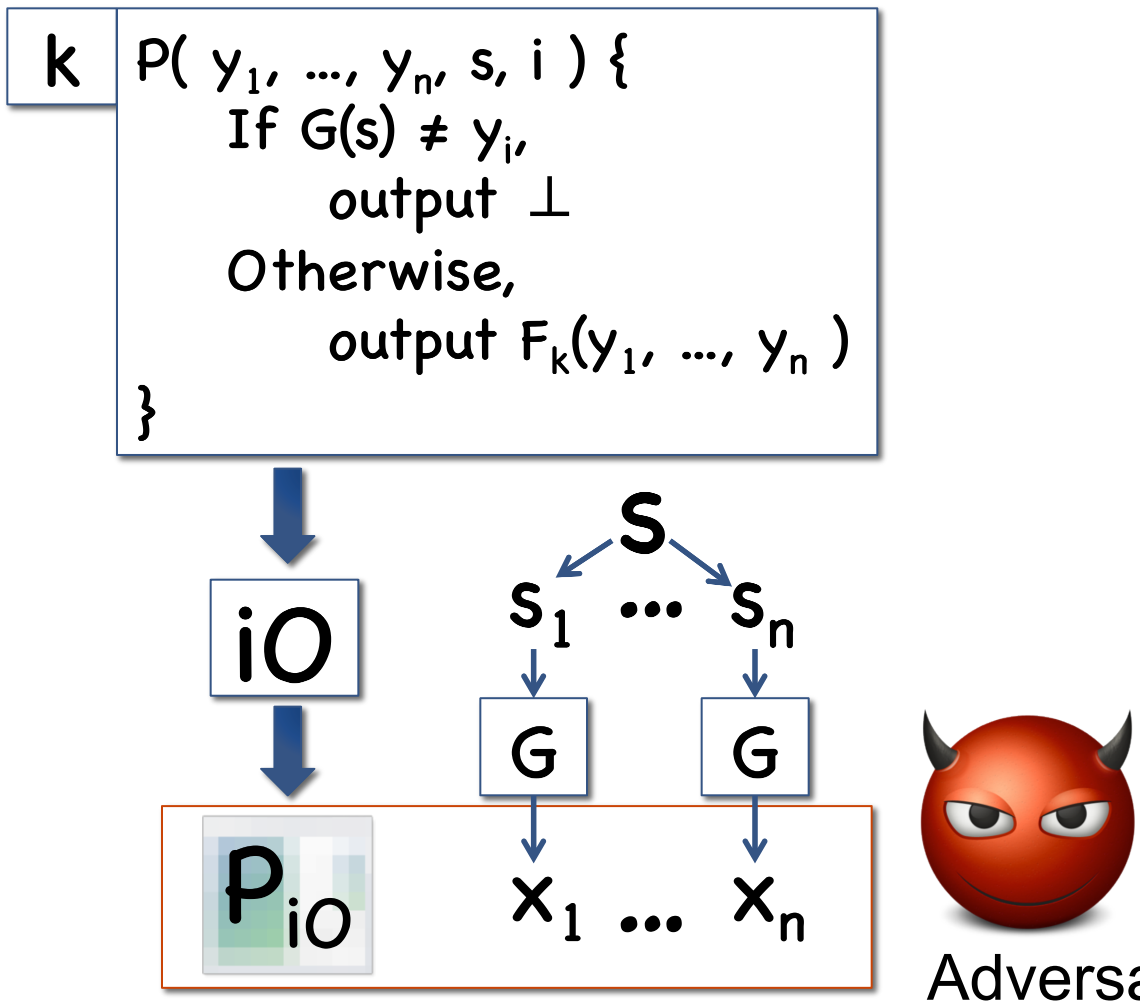


$$\begin{cases} F(k,x) & \text{if } x \neq z \\ \perp & \text{if } x = z \end{cases}$$

  Security: given k^z, cannot compute $t = F_k(z)$

  Construction: GGM'84

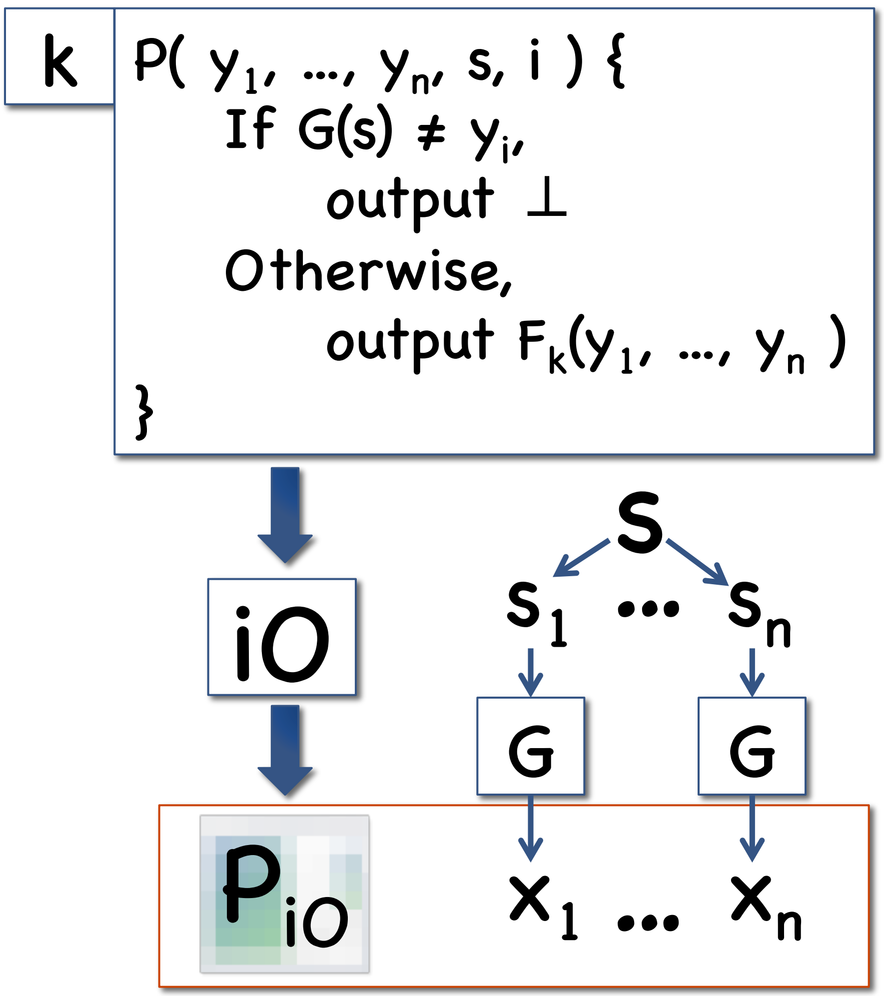
Security of Our Construction



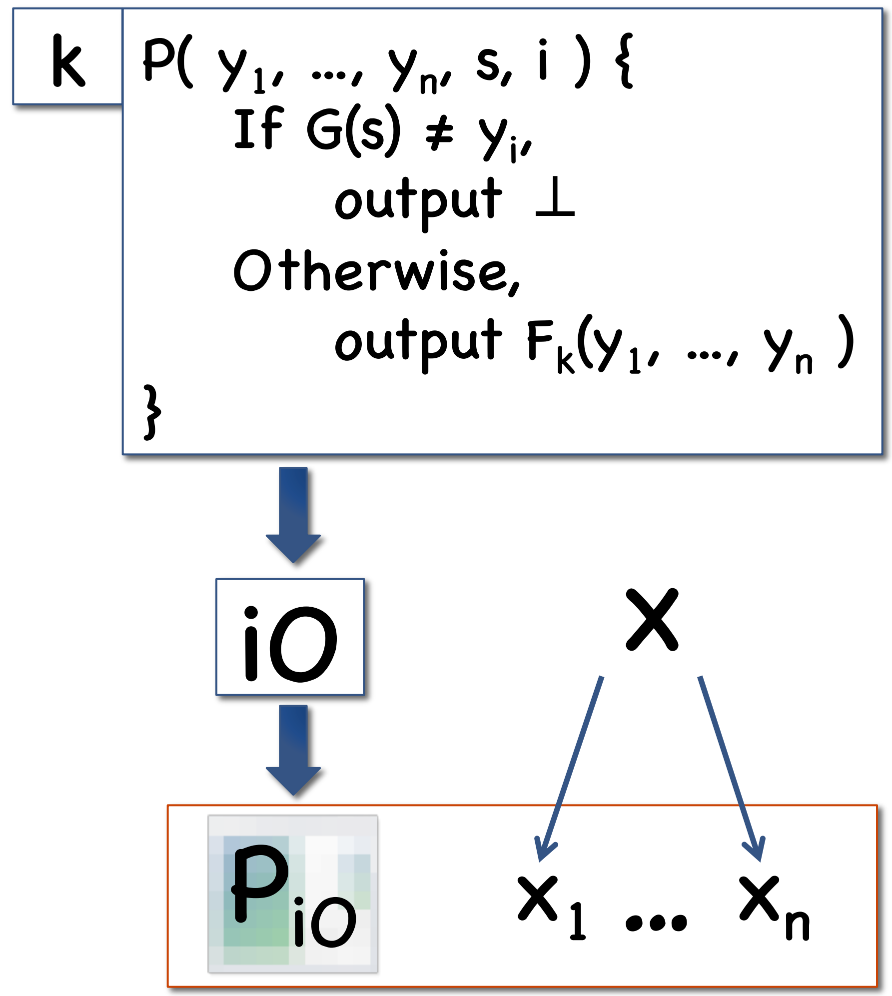
Adversary's goal:
Learn $F_k(x_1, \ldots, x_n)$

Step 1: Replace x_i

Real World



Alternate World 1



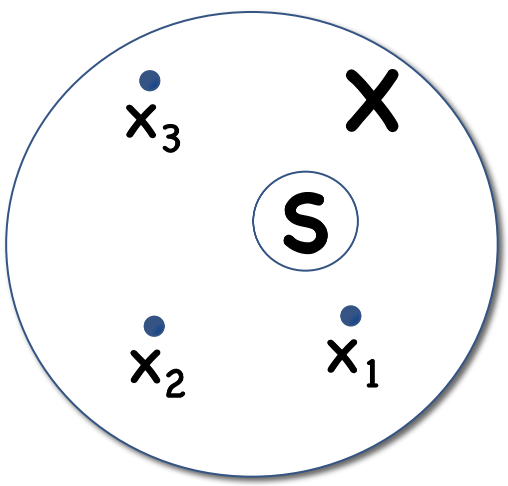
Security of **G** ⇒ words indistinguishable

Step 1: Replace x_i

Observation:

Since $|X| >> |S|$,

w.h.p. no **s,i** s.t. $G(s)=x_i$

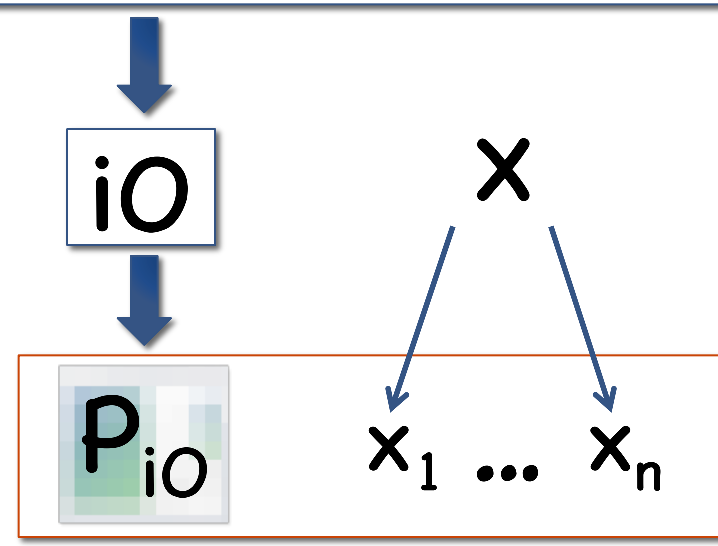


Never pass check when

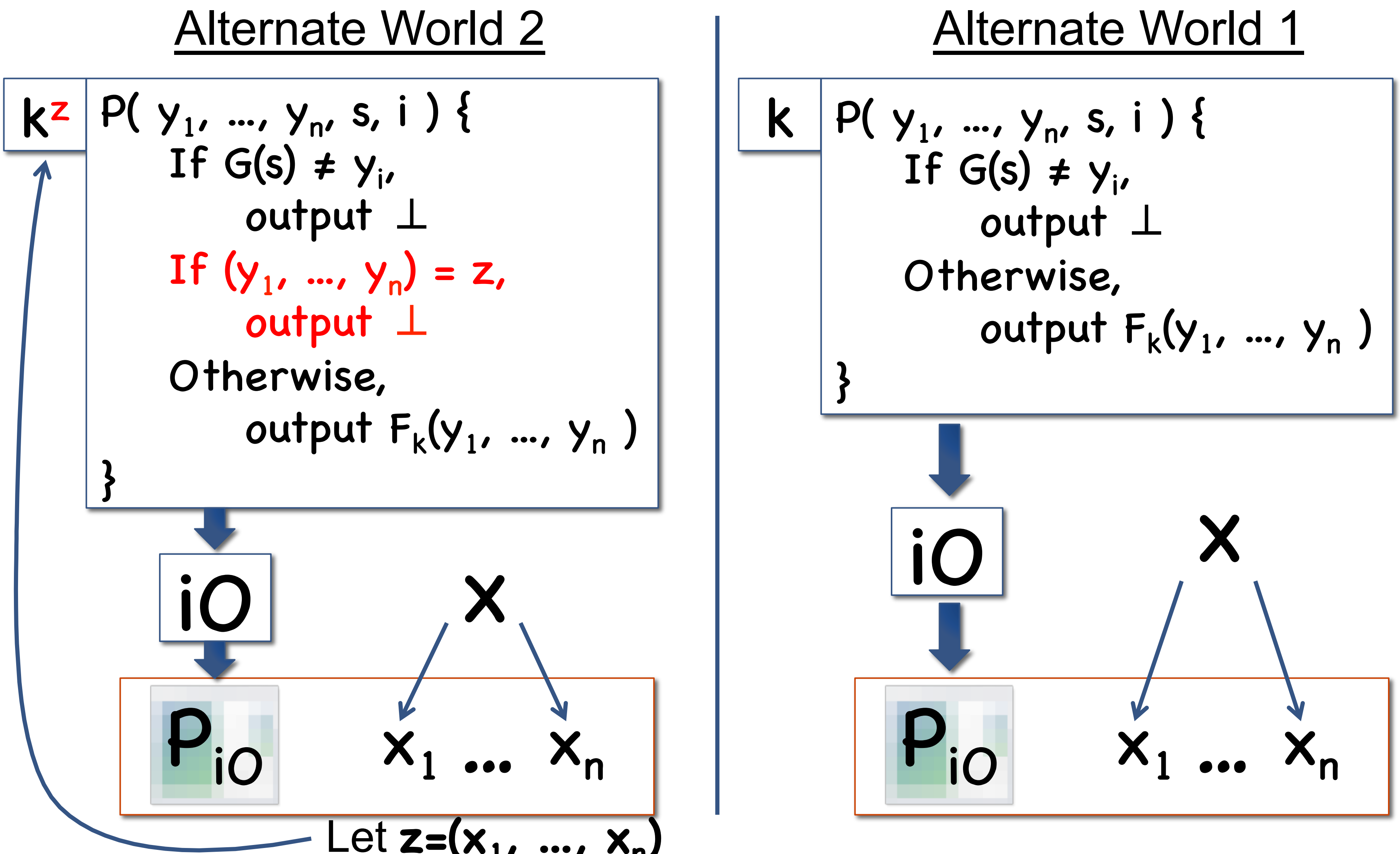
$y_1, \ldots, y_n = x_1, \ldots, x_n$

Alternate World 1

```
k | P( y_1, ..., y_n, s, i ) {
      If G(s) ≠ y_i,
          output ⊥
      Otherwise,
          output F_k(y_1, ..., y_n )
    }
```



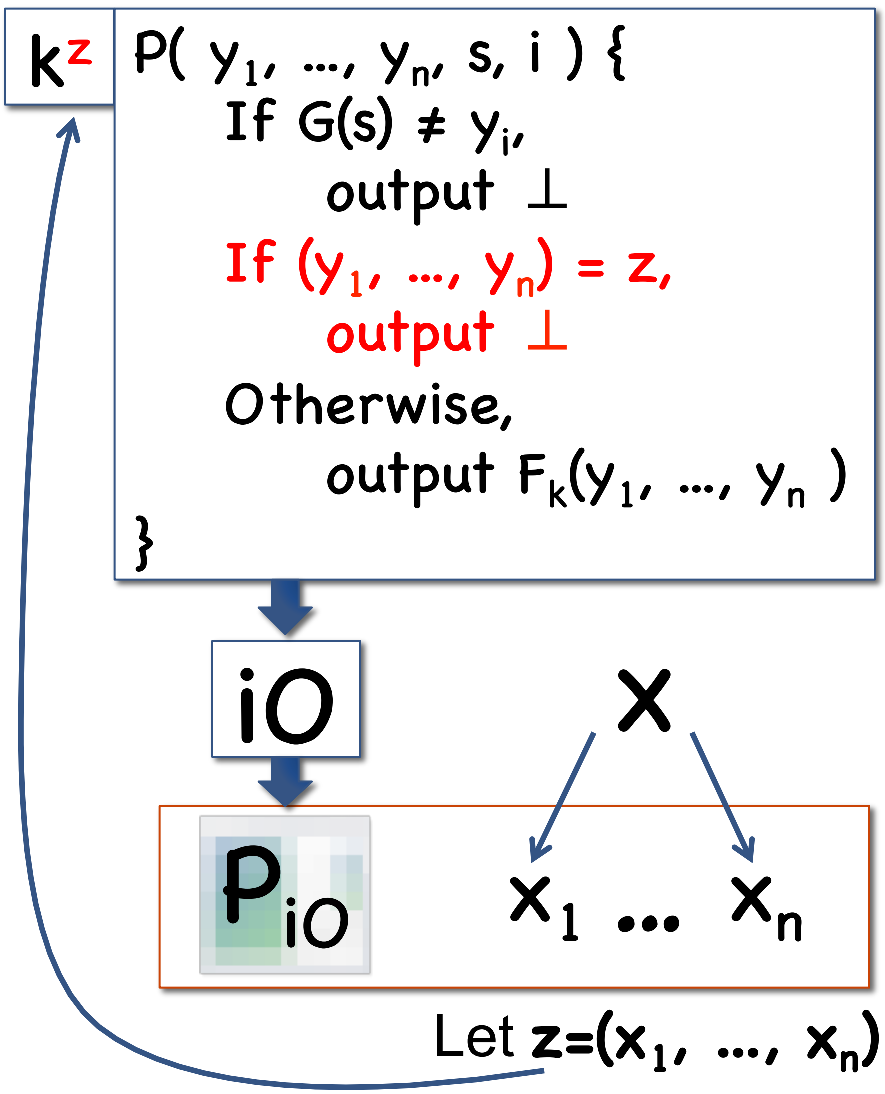
Step 2: Puncture



W.h.p. programs identical + iO ⇒ Worlds indistinguishable

Security

<u>Alternate World 2</u>



Adversary's goal: learn **F_k(z)**

Success in Real World

⇒ success in World 2

In World 2:

Adversary only sees **k^z**

⇒ cannot learn **F_k(z)**

✓

Future Work

Our work and others: iO is incredibly powerful

What else can we do with it? What can't we do?

Obfuscation is currently very inefficient

Can we make obfuscation practical?