How To Avoid Obfuscation Using Witness PRFs

Mark Zhandry – Stanford University

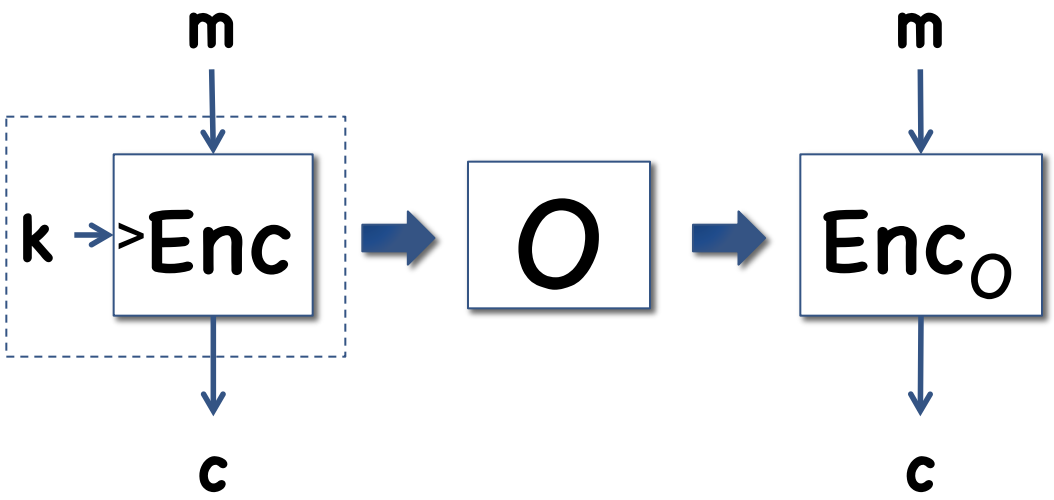
What is Obfuscation?

"Scramble" a program:

- Maintain functionality
- Hide internal details

Typical crypto application: hide embedded secrets

- Example: symmetric key to public key encryption



The Field of Obfuscation

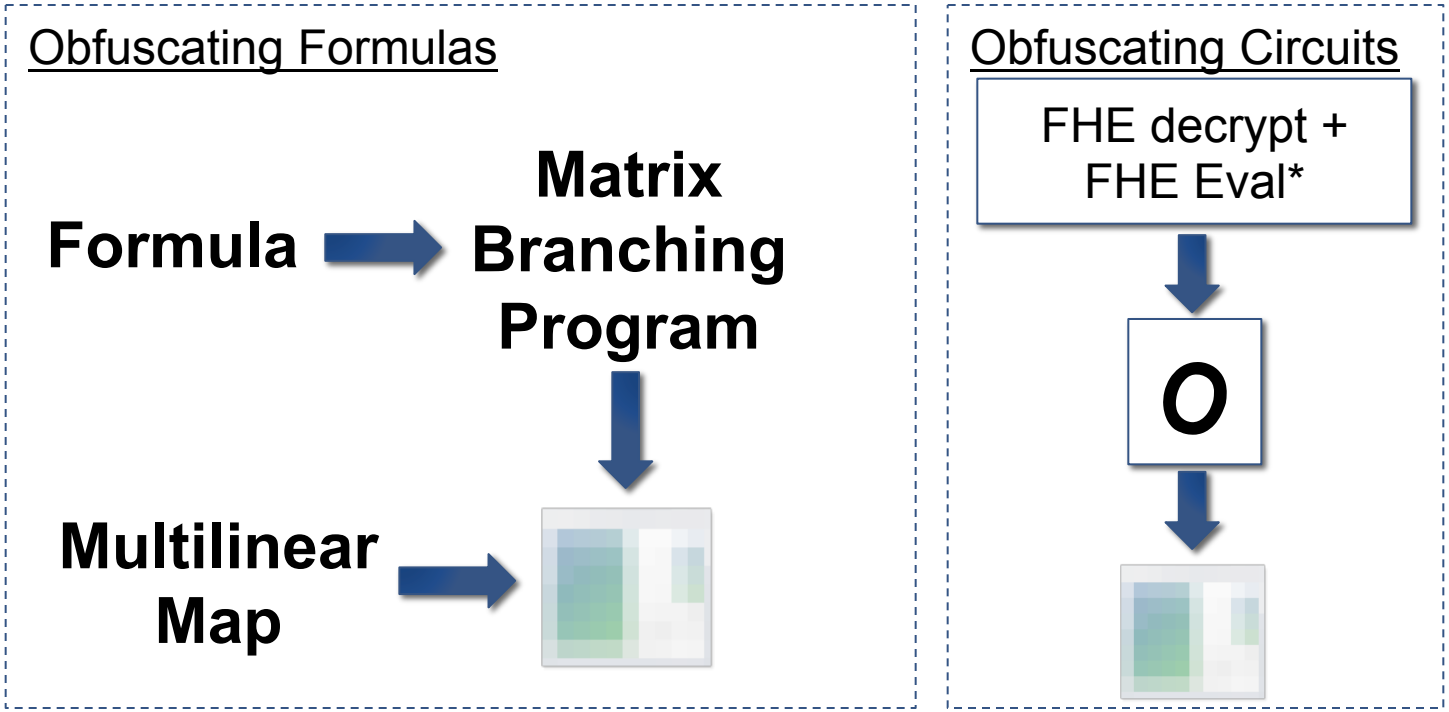
- [Hada'00, BGIRSVY'01] Investigation, definitions, impossibilities
- [GK'05, GR'07] Further investigation
- **[GGHRSW'13] First candidate obfuscator**
  - Functional encryption
- [BR'13, BGKPS'13, …] Additional constructions
- [SW'13, GGHR'13, BZ'13, ABGSZ'13, …] Uses
  - Public key encryption, signatures, deniable encryption, multiparty key exchange, MPC, …
- [BCPR'13, BCP'13, KMNPRY'14 …] Further Investigation

Good news: Obf is solving everyone's open problems!

Bad news: Obf is far from practical

How Obfuscation (Currently) Works

[GGHRSW'13, BR'13, BGKPS'13, PST'13, AGIS'14]



Caveat:

- For some schemes, obfuscating formulas enough
- However, many times requires obfuscating lattice primitives
  ⇒ still obfuscating large program

Outline of Talk

- Motivating example:

  Non-interactive multiparty key exchange w/o setup

- Obfuscation-based solution [BZ'14]

- New primitive: Witness PRF
  - Abstracts features needed from obfuscation

- Construction from multilinear maps
  - Comparable efficiency to witness encryption

MULTIPARTY KEY EXCHANGE

Multiparty (Non-Interactive) Key Exchange

Public bulletin board

s_A

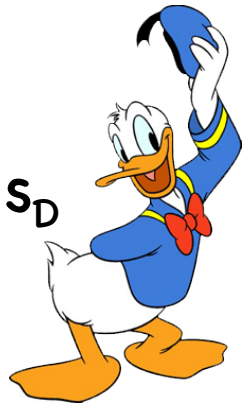
K_{ABCD}

s_B

K_{ABCD}

s_C

K_{ABCD}



s_D

K_{ABCD}

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

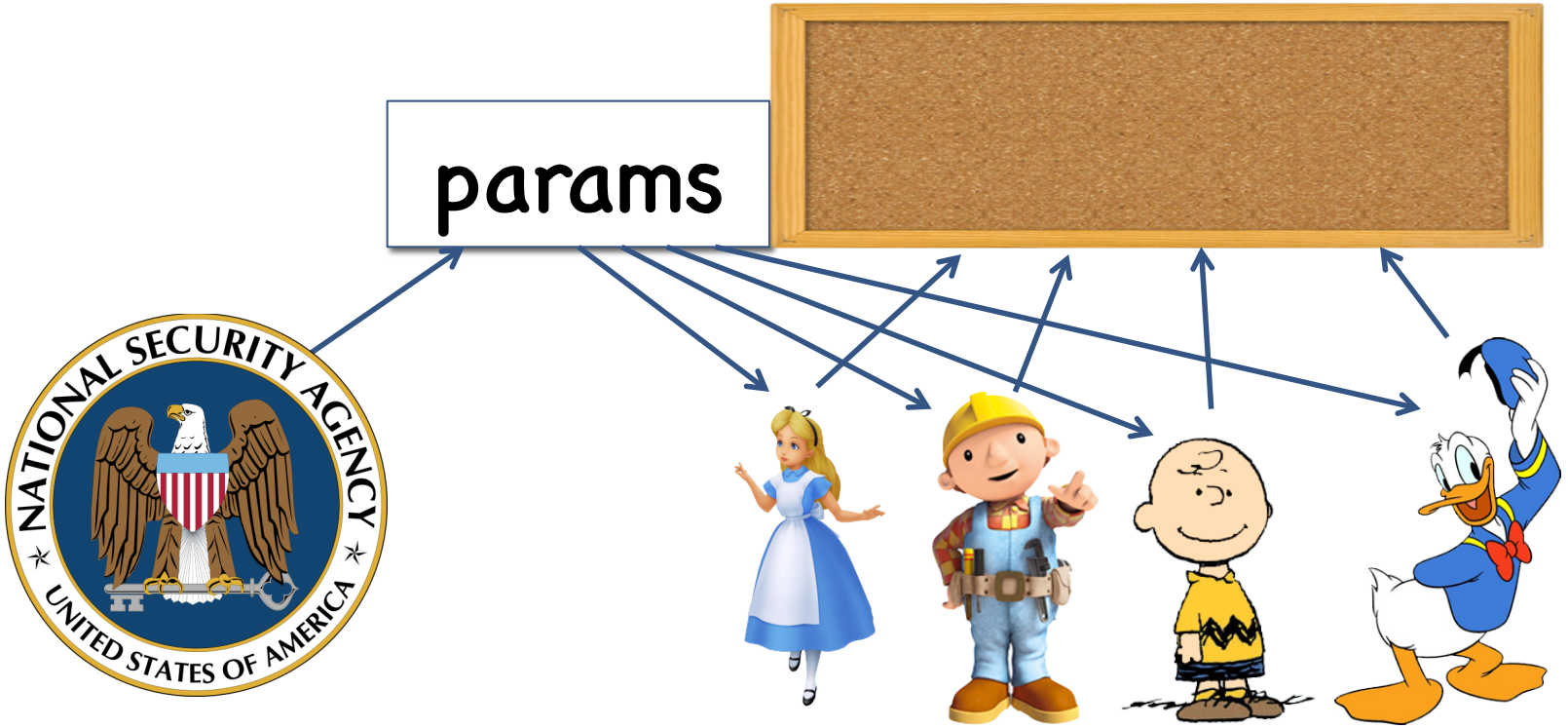
Obfuscation [BZ'14]

No setup at all

Map-based 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

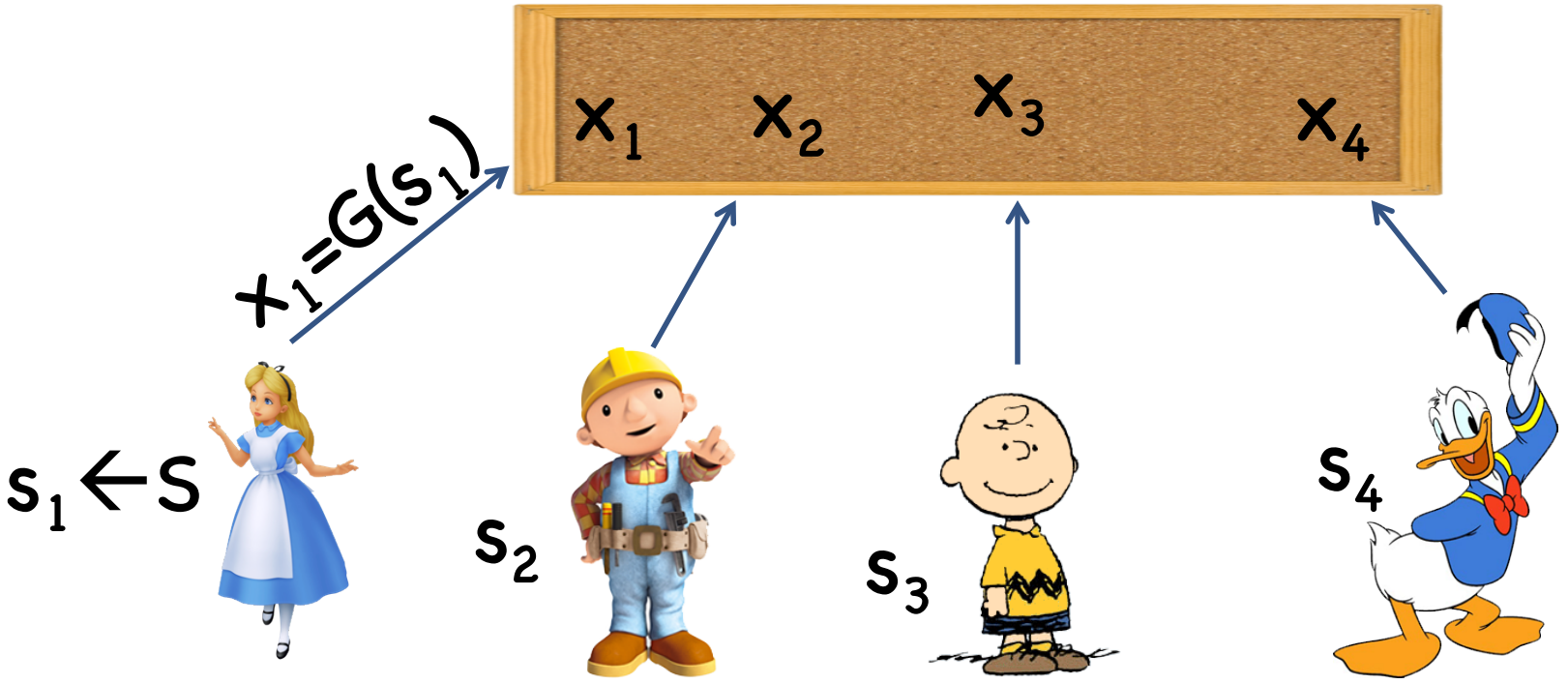


OBFUSCATION-BASED SOLUTION

Starting Point

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)

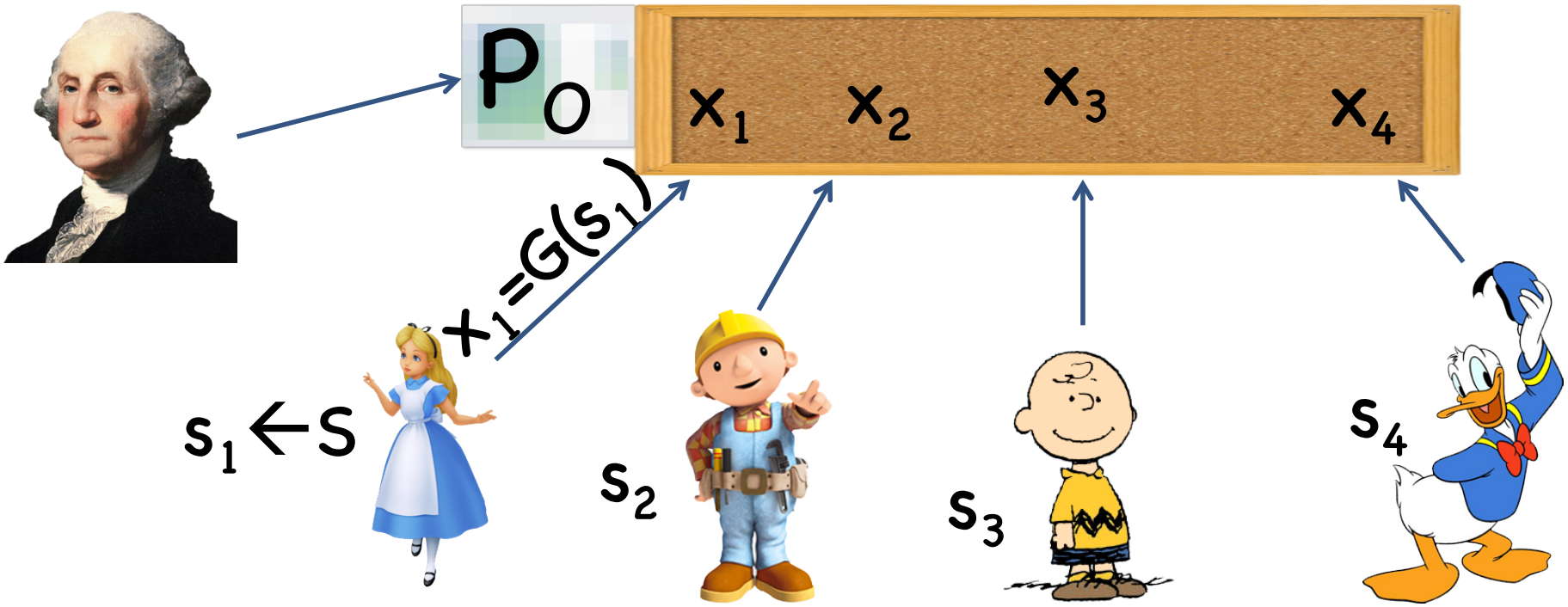
k

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

O

P_O

First attempt



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

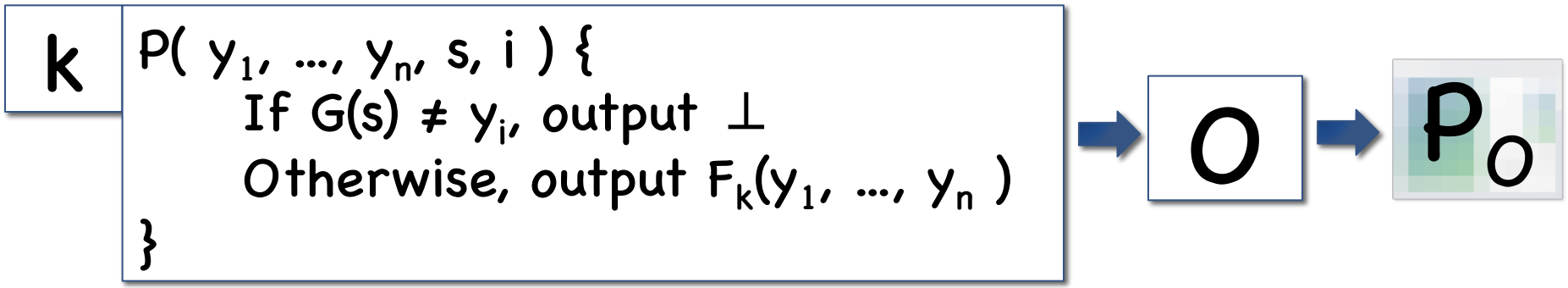
Problems:

- **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 $\mathbf{P_O}$, sends with $\mathbf{x_1}$

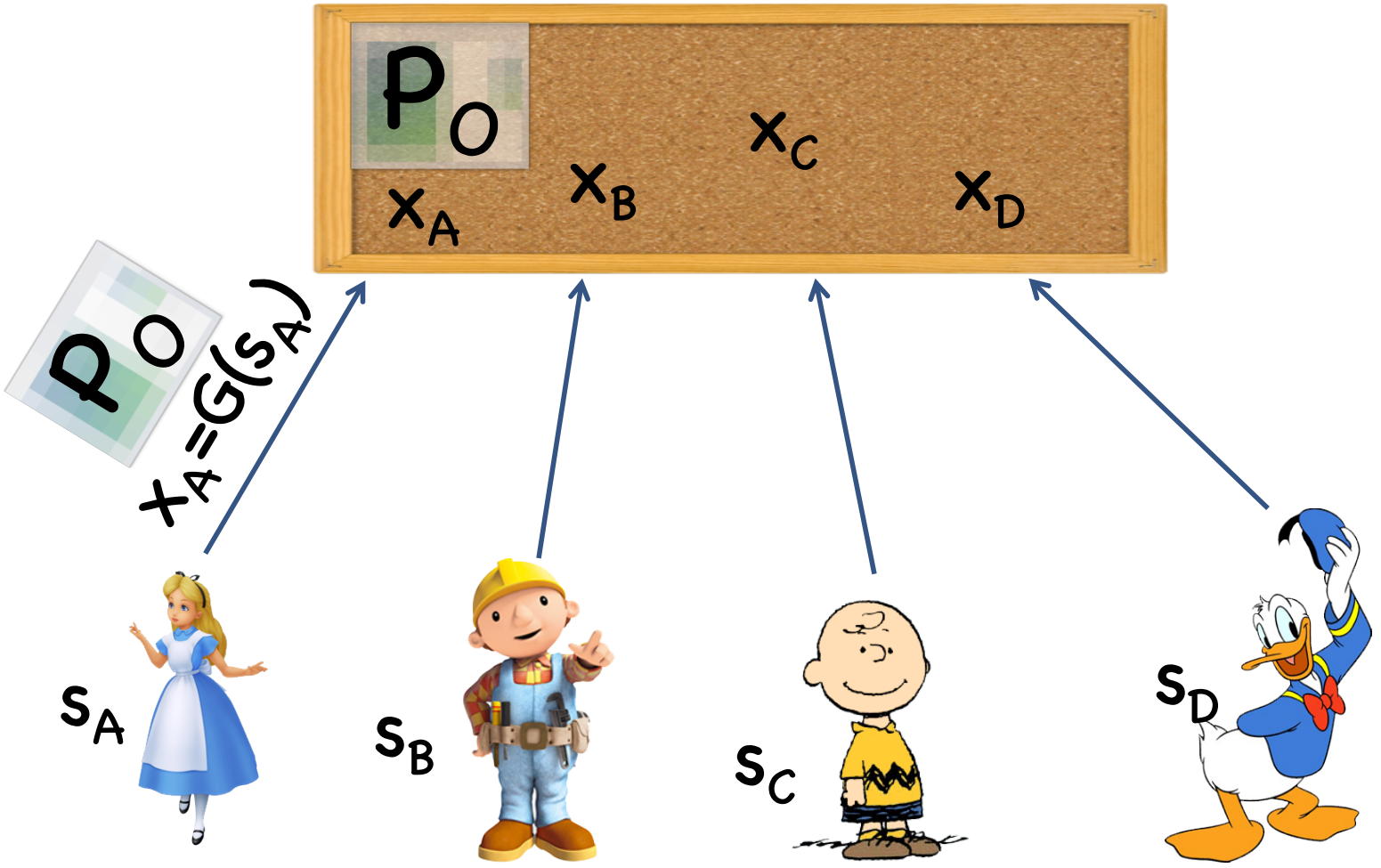
Hiding **k**

Enhance primitives:

- OWF → PRG
- PRF → "puncturable PRF" (e.g. [GGM'84])

THM([BZ'14]):
iO + PRG + puncturable PRF = multiparty NIKE

Multiparty Key Exchange Without Trusted Setup



WITNESS PRFs

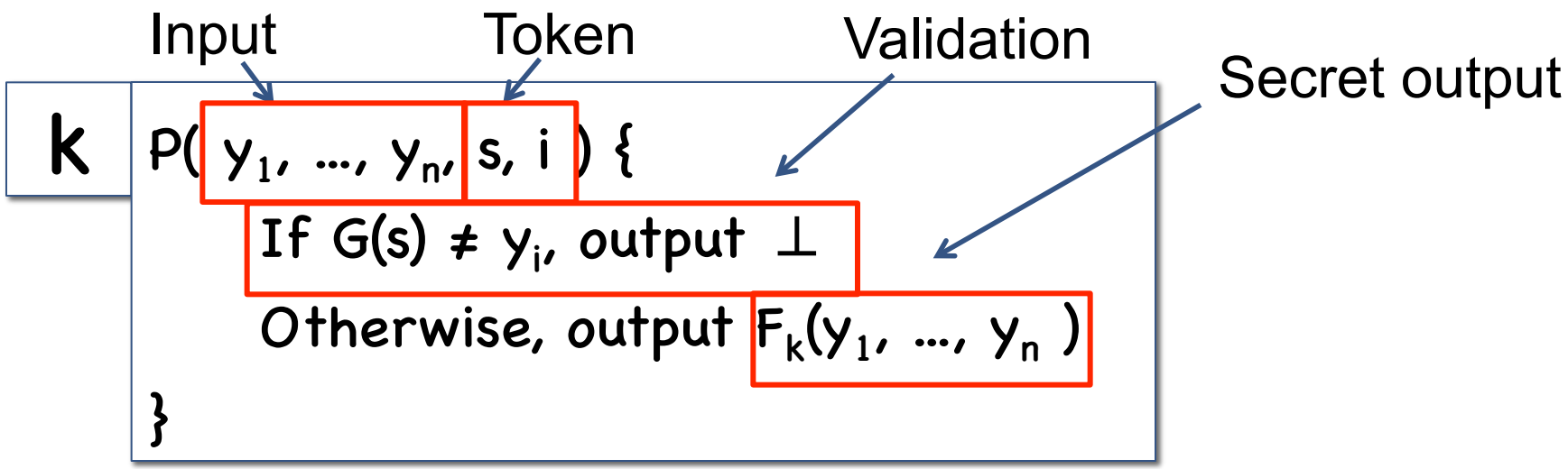
Are we working too hard?

Don't need full power of obfuscation for key exchange

- Don't care if computation leaked, *except* for PRF step

Obfuscation as access control:

- Can only learn output if given token for input



New Abstraction: Witness PRFs

NP Language **L**

ek

Instance: **x∈L**
Witness: **w**

(fk,ek) ← Gen(L)
fk defines a PRF: **F(fk,·)**
domain = instances

Eval(ek,x,w) = F(fk,x)
if **w** is witness that **x∈L**

Security: if **x∉L**, **F(fk,x)** pseudorandom given **ek**

- Even if **x** chosen by adv.
- Even if adv. has **F(fk,·)** oracle

Similar to smooth projective hash functions (SPHFs)

Relation to Witness Encryption [GGSW'13]

(witness key encapsulation)

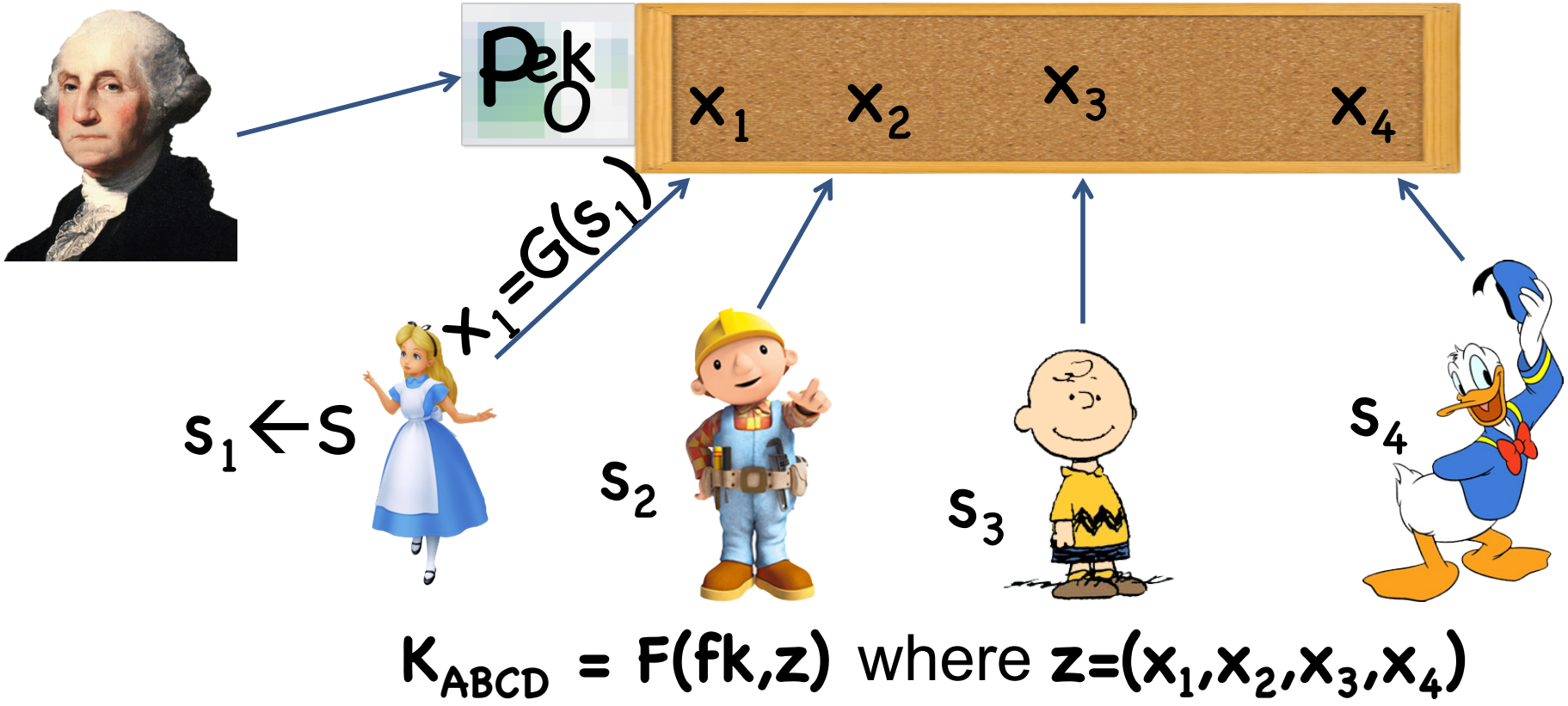
NP Language **L**

Security: if $x \notin L$, k pseudorandom given c

Main difference:

- Witness encryption: Alice needs x during setup
- Witness PRF: Alice does not need x

Witness PRFs → Multiparty Key Exchange



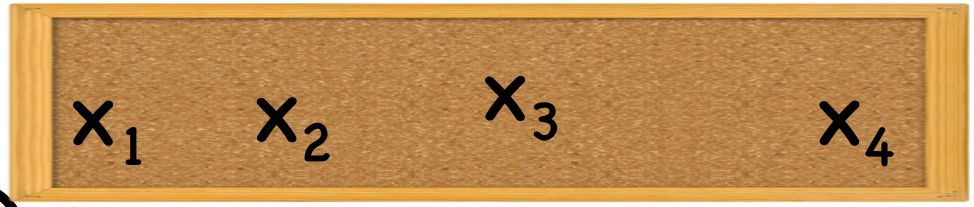
$K_{ABCD} = F(fk,z)$ where $z=(x_1,x_2,x_3,x_4)$

Language **L**?

- Need all parties to have witness for **z** (so $z \in L$)
- Eavesdropper can't compute witness (can't even tell if $z \in L$)

Witness PRFs → Multiparty Key Exchange

ek



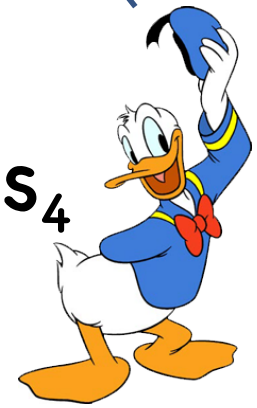
$x_1 = G(s_1)$

$s_1 \leftarrow S$

s_2

s_3

s_4



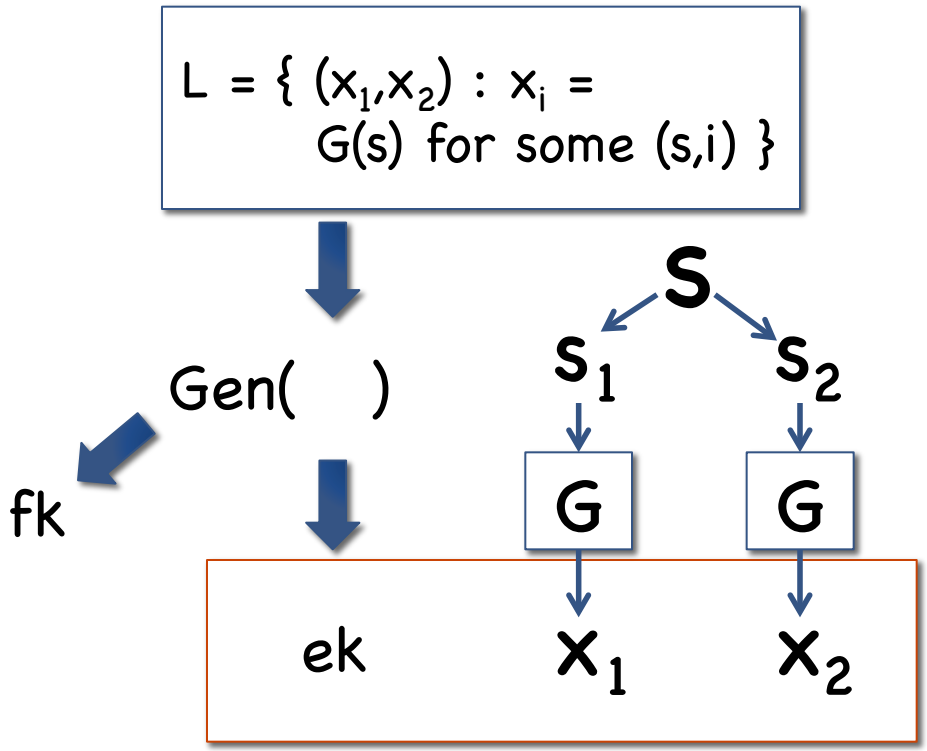
$K_{ABCD} = F(fk,z)$ where $z=(x_1,x_2,x_3,x_4)$

$L = \{ z=(x_1,x_2,x_3,x_4) : x_i = G(s)$ for some $s,i \}$

Alice's witness: $(s_1,1)$, Bob's witness: $(s_2,2)$, …

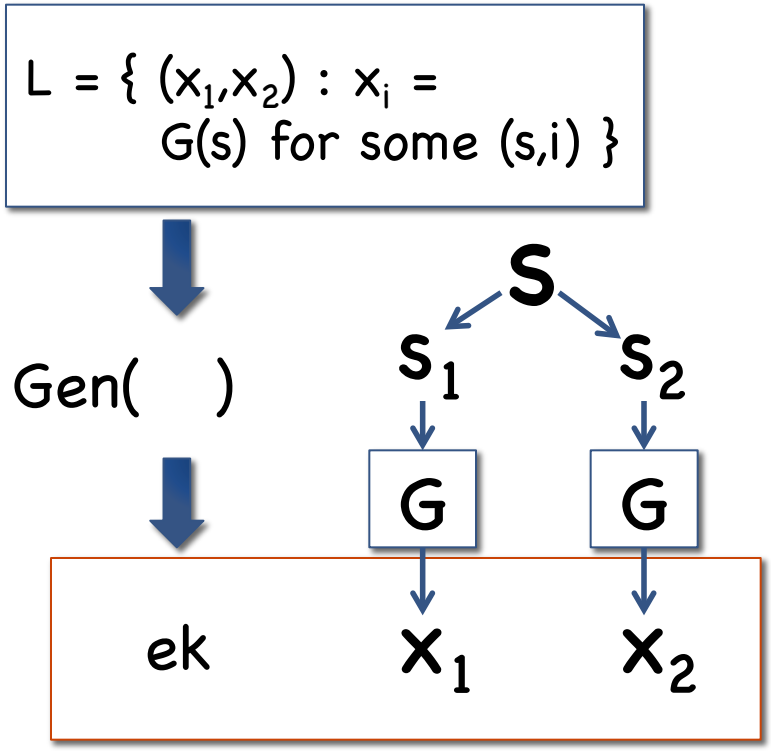
Alice's computation: $K_{ABCD} = Eval(ek,z,(s_1,1))$

Security Proof (2 users)

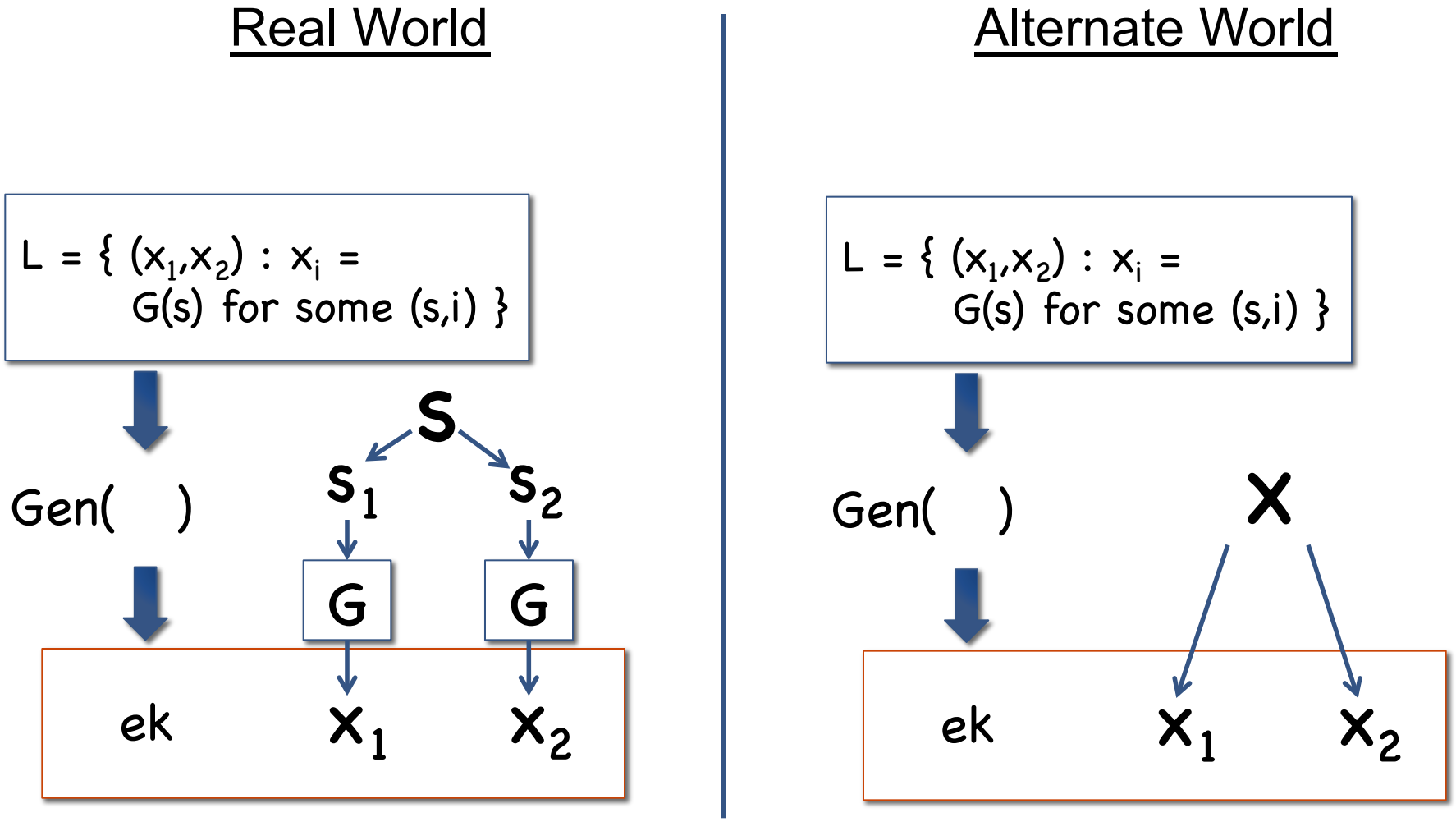


Adversary's goal:
Learn **F(fk,(x$_1$, x$_2$))**

Security Proof



Step 1: Replace x_i



Security of **G** ⇒ worlds indistinguishable

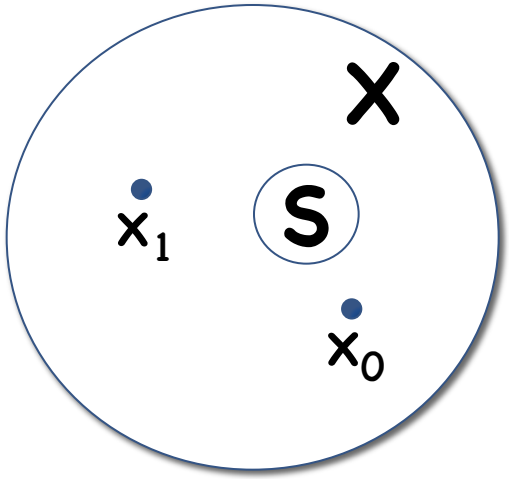
Step 1: Replace x_i

Alternate World

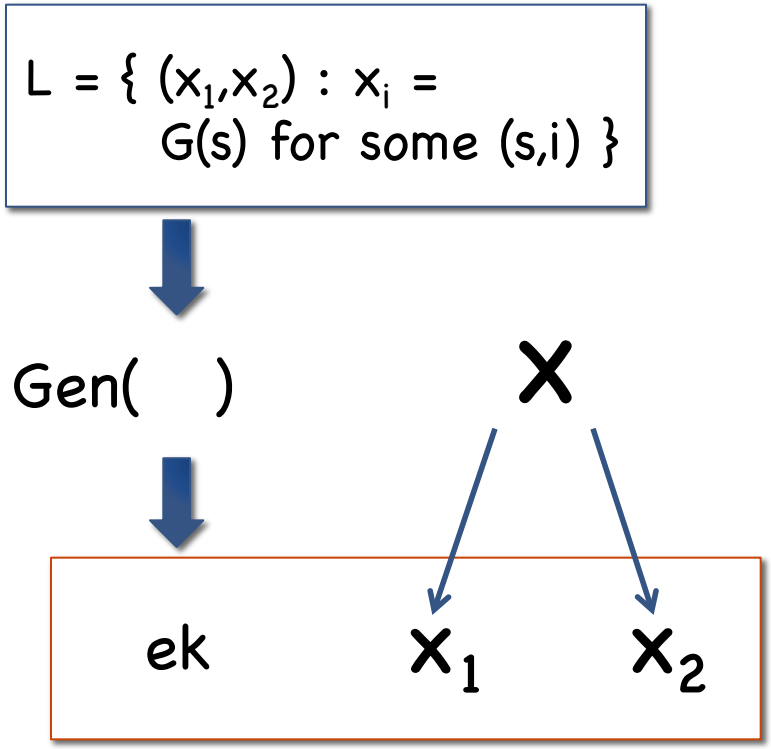
Observation:

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

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



$\Rightarrow (x_1, x_2) \notin L$



Step 2: Apply witness PRF

$(x_1, x_2) \notin L$

$F(fk, (x_1, x_2))$ indist. from random

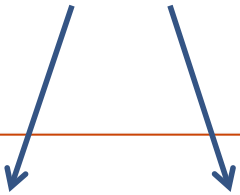
Secure in alternate world

Secure in real world ✓

Alternate World

$L = \{ (x_1, x_2) : x_i = G(s) \text{ for some } (s,i) \}$

Gen( )



ek x_1 x_2

What can WPRFs be used for?

- CCA Secure PKE [iO: SW'13]
- (Reusable) witness encryption [iO: GGHRSW'13]
- (Reusable) secret sharing for monotone NP [WE: KNY'14]
- Multiparty key exchange w/o trusted setup [iO: BZ'14]
- Poly-many hardcore bits for any one-way function [diO: BST'14]*
- Fully distributed broadcast encryption with small parameters*

● = new in this work * = needs extractability notion

CONSTRUCTING WITNESS PRFs

Starting point: WE for Subset Sum

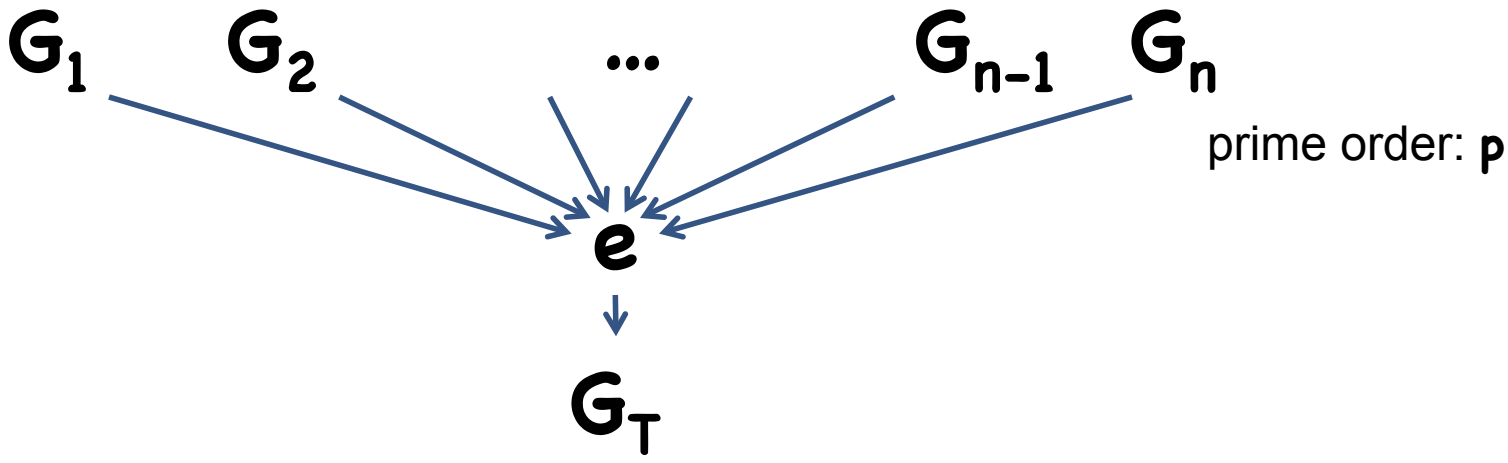
$\mathbf{SubSums(A \in Z^n) = \{ A \cdot x : x \in \{0,1\}^n \}}$

(i.e. subset-sums of elements of **A**)

Subset-sum problem: given **$\mathbf{(A \in Z^n , t \in Z)}$**, determine if **$\mathbf{t \in SubSums(A)}$**

Tool: Multilinear Maps

Groups:



Map:

Multilinearity: $e(g_1^a, g_2^b, \ldots, g_n^z) = g_T^{ab\ldots z}$

Hopefully hard: anything but group operations, map

- Ex: undo exponentiation (DLOG), CDH, MDDH, etc.

WE for Subset Sum

Key observation: **c** independent of **t**

Enc (A, t): $\alpha \leftarrow Z_p$, $V_i = g_i^{\alpha A_i}$

$c = \{V_i\}$ $\quad k = g_T^{\alpha t}$

Dec ($\{V_i\}$, w) : Let $W_i = V_i$ if $w_i = 1$, g_i if $w_i = 0$

$$k = e(W_1, \ldots, W_n) = g_T^{\alpha w_1 A_1 + \ldots + w_n A_n} = g_T^{\alpha A \cdot w}$$

Can prove security* in generic multilinear map model
(i.e. need new assumptions on maps)

*not quite secure: e.g. $t = p-1$

Witness PRF for **L = SubSums(A)**

Gen (A): $\alpha \leftarrow Z_p$, $V_i = g_i^{\alpha A_i}$

$ek = \{V_i\}$ $fk = \alpha$

$F(\alpha, t) = g_T^{\alpha^t}$

Eval $(\{V_i\}, w)$: Let $W_i = V_i$ if $w_i = 1$, g_i if $w_i = 0$

Output: $e(W_1, \ldots, W_n) = g_T^{\alpha^{w_1 A_1 + \ldots + w_n A_n}} = g_T^{\alpha^{A \cdot w}}$ ✓

Can prove security* in generic multilinear map model
(i.e. need new assumptions on maps)

Final Step: Reduce NP to SubSums(A)

Need reduction from any language $L \in NP$ to SubsetSum where:

- A only depends language (not instance)
- t depends on language and instance
- Few extra minor requirements (to block trivial attacks)

We give solution:

- n linear in relation size
- Similar to [GOS'06] ZK proofs of knowledge for Circuit SAT

Proving Computation as Subset-Sum

Let $C: \{0,1\}^n \rightarrow \{0,1\}^m$ be a circuit

Goal: derive matrix V and target b such that:

- If $C(x)=y$, then is possible to compute a proof $\pi \in \{0,1\}^k$ where:

$$V \cdot \begin{pmatrix} x \\ \pi \\ y \end{pmatrix} = b$$

- If $C(x) \neq y$, then no such proof exists

Proving Computation as Subset-Sum

Simple case: $C(x) = \neg x$

$$V = (1 \quad 1) \qquad b = (1)$$

(No proof)

$$V \cdot \begin{pmatrix} x \\ y \end{pmatrix} = b \iff x + y = 1 \iff y = \neg x$$

Proving Computation as Subset-Sum

Simple case: $C(x_1,x_2) = x_1 \wedge x_2$

$$V = (1 \quad 1 \quad -1 \quad -2) \qquad b = (0)$$

On input x_1,x_2, prove that $y = x_1 \wedge x_2 = x_1x_2$:

$$\pi = x_1 \oplus x_2 = x_1 + x_2 - 2x_1x_2$$

$$V \cdot \begin{pmatrix} x_1 \\ x_2 \\ \pi \\ y \end{pmatrix} = x_1+x_2-(x_1+x_2-2x_1x_2)-2x_1x_2 = 0$$

Proving Computation as Subset-Sum

Simple case: $C(x_1,x_2) = x_1 \wedge x_2$

$$V = (1 \quad 1 \quad -1 \quad -2) \qquad b = (0)$$

On input x_1,x_2, $y = \neg(x_1 \wedge x_2) = 1-x_1x_2$,
prove that $y = x_1 \wedge x_2$?

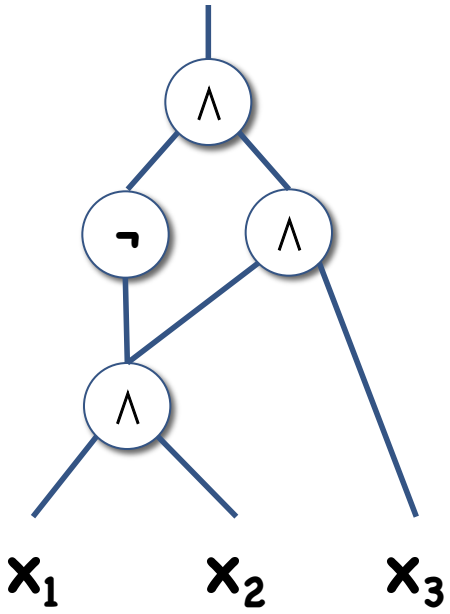
$$V \cdot \begin{pmatrix} x_1 \\ x_2 \\ \pi \\ y \end{pmatrix} = x_1+x_2-\pi-2(1-x_1x_2)$$
$$= -2+4x_1x_2+(x_1 \oplus x_2)-\pi \neq 0$$

Proving Computation as Subset-Sum

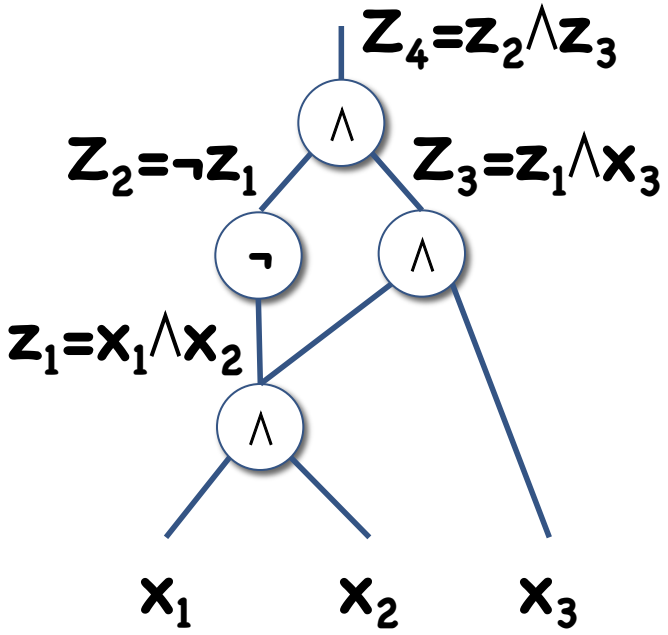
For general circuits, verify gate-by-gate

- Each gate gets two columns: value and proof
- Each gate gets one row: verification matrix

Ex: $C(x_1,x_2,x_3) =$



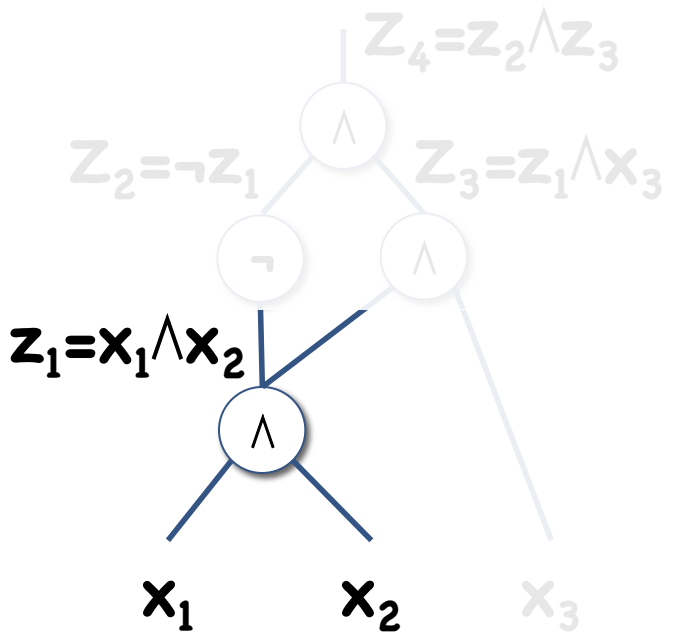
Proving Computation as Subset-Sum



$$V=\begin{pmatrix} 1 & 1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -2 \end{pmatrix} \quad b=\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

x_1 x_2 x_3 π_1 z_1 z_2 π_3 z_3 π_4 z_4

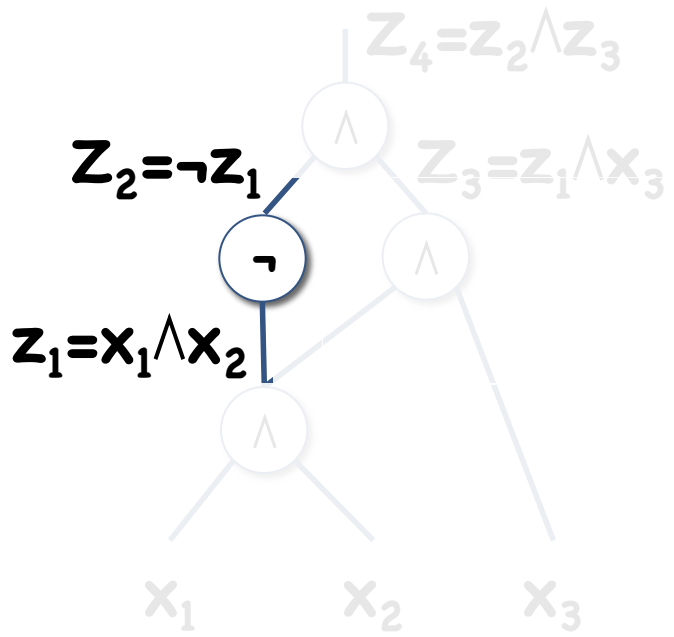
Proving Computation as Subset-Sum



$$V=\begin{pmatrix} 1 & 1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -2 \end{pmatrix} \quad b=\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

x_1 x_2 x_3 π_1 z_1 z_2 π_3 z_3 π_4 z_4

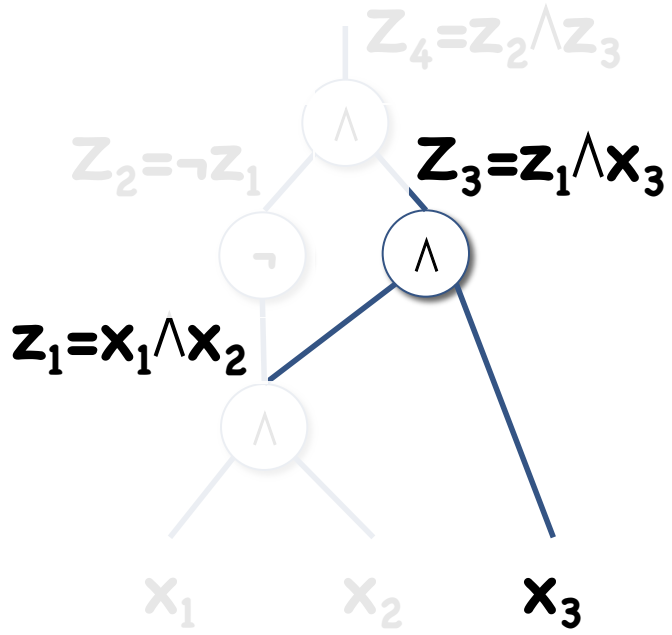
Proving Computation as Subset-Sum



$$V=\begin{pmatrix} 1 & 1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -2 \end{pmatrix} \qquad b=\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

x_1 x_2 x_3 π_1 z_1 z_2 π_3 z_3 π_4 z_4

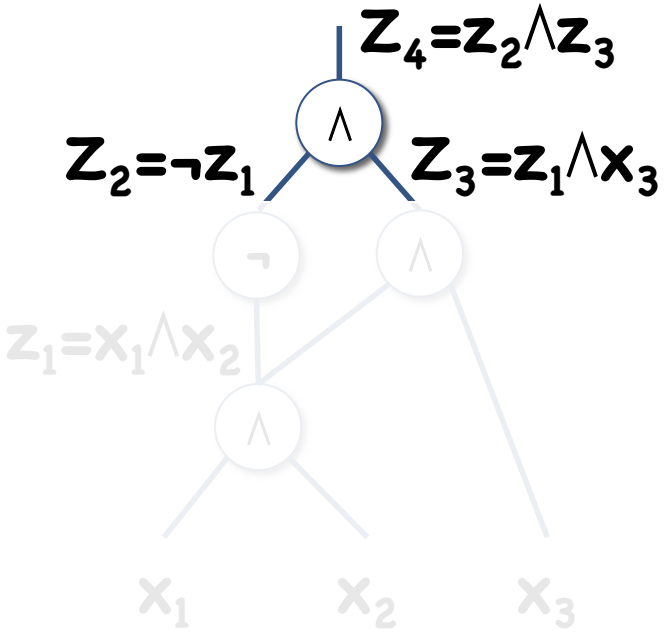
Proving Computation as Subset-Sum



$$V=\begin{pmatrix} 1 & 1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{-1} & \mathbf{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -2 \end{pmatrix} \quad b=\begin{pmatrix} 0 \\ 1 \\ \mathbf{0} \\ 0 \end{pmatrix}$$

x_1 x_2 $\mathbf{x_3}$ π_1 $\mathbf{z_1}$ z_2 $\mathbf{\pi_3}$ $\mathbf{z_3}$ π_4 z_4

Proving Computation as Subset-Sum



$$V=\begin{pmatrix} 1 & 1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -2 \end{pmatrix} \quad b=\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

x_1 x_2 x_3 π_1 z_1 z_2 π_3 z_3 π_4 z_4

Reduction

Given NP relation $R:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}$, construct V,b:

$$\exists\, \pi \quad \text{s.t.} \quad V \cdot \begin{pmatrix} x \\ w \\ \pi \\ 1 \end{pmatrix} = b \quad \Leftrightarrow \quad R(x,w)=1$$

Reduction

Write

$$V = \begin{pmatrix} B & C & D & E \end{pmatrix}$$

$$A \cdot \begin{pmatrix} x \\ w \\ \pi \\ 1 \end{pmatrix} = B \cdot x + C \cdot w + D \cdot \pi + E = b$$

$$C \cdot w + D \cdot \pi = b - E - B \cdot x$$

Reduction

Define

$$A = \begin{pmatrix} C & D \end{pmatrix}$$

On input x, let $t = b - E - B \cdot x$

$$t \in \text{SubSums}(A) \iff \exists w \text{ s.t. } R(x,w)=1$$

$$C \cdot w + D \cdot \pi = b - E - B \cdot x$$

*Not quite there: A consists of vectors, but scalars. Multiple ways to fix

Witness PRFs vs Obfuscation

Witness PRF steps:

- Build witness PRF for circuit relations directly
  - NP reduction to subset-sum
  - Place subset-sum "in the exponent" of multilinear map

- Efficiency depends on reduction
- Comparable to WE

Obfuscation steps:

- Step 1: Build obfuscator for formulas
  - Transform formula into branching program
  - Put branching program "in the exponent" of multilinear map
- Step 2: Boost to general circuits
  - Universal circuits + proof of FHE eval + FHE decryption
  - Obfuscate it all

Main inefficiency of witness PRFs: multilinear maps
- Lots of research ⇒ likely to improve in near future

Witness PRFs vs Obfuscation

Witness PRF advantages

- More efficient
- Simpler construction
- Simpler assumptions on multilinear maps

Obfuscation advantages

- More versatile
  - Intermediate computations hidden
  - Hide more general secrets
- More applications
  - Short signatures
  - Functional encryption
  - Deniable encryption
  - …

Conclusion

For many obfuscation applications, use witness PRFs instead

Open questions:

- From standard assumptions on multilinear maps?
- From LWE?
- Stronger notions with similar efficiency?

Thanks!