LOCKABLE OBFUSCATION

[VBB OBFUSCATION FOR COMPUTE-AND-COMPARE PROGRAMS]

VENKATA KOPPULA

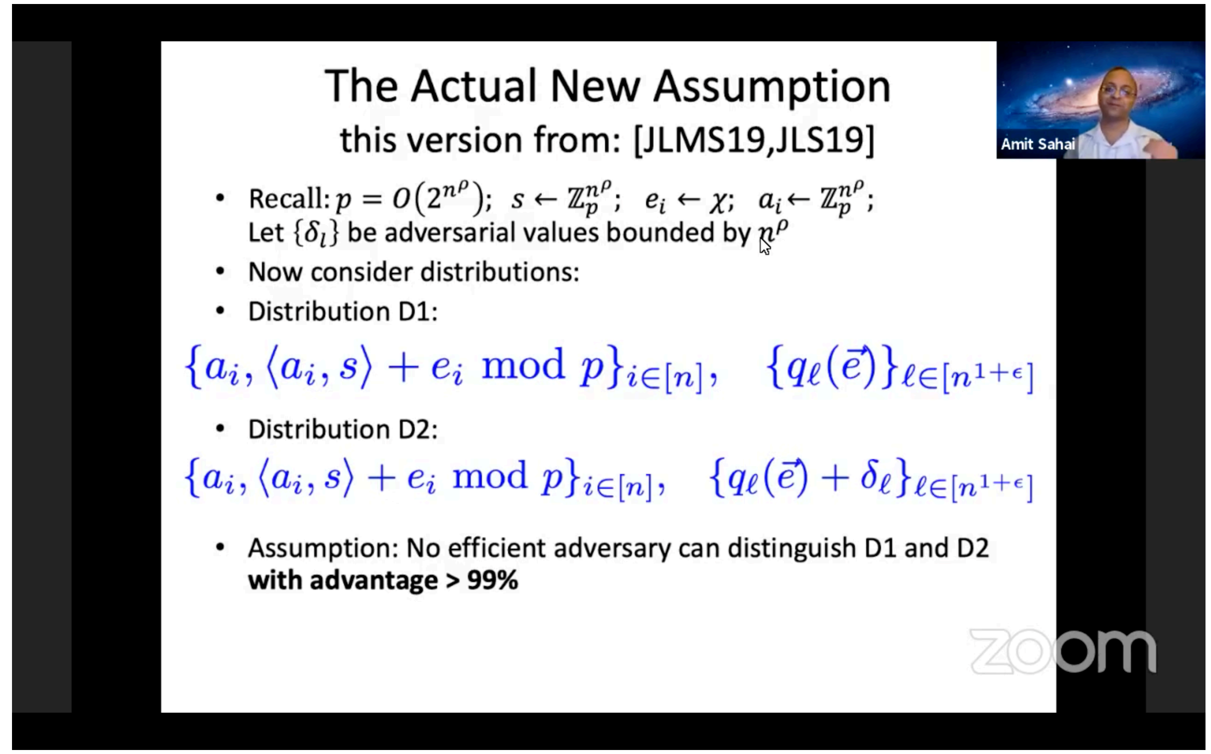
(Talk based on concurrent works: [Goyal, K, Waters], [Wichs, Zirdelis])

Code Obfuscation: compiling code to hide the implementation

Code Obfuscation: compiling code to hide the implementation

Timeline not drawn to scale

CODE OBFUSCATION: COMPILING CODE TO HIDE THE IMPLEMENTATION



GGH13 + GGHRSW

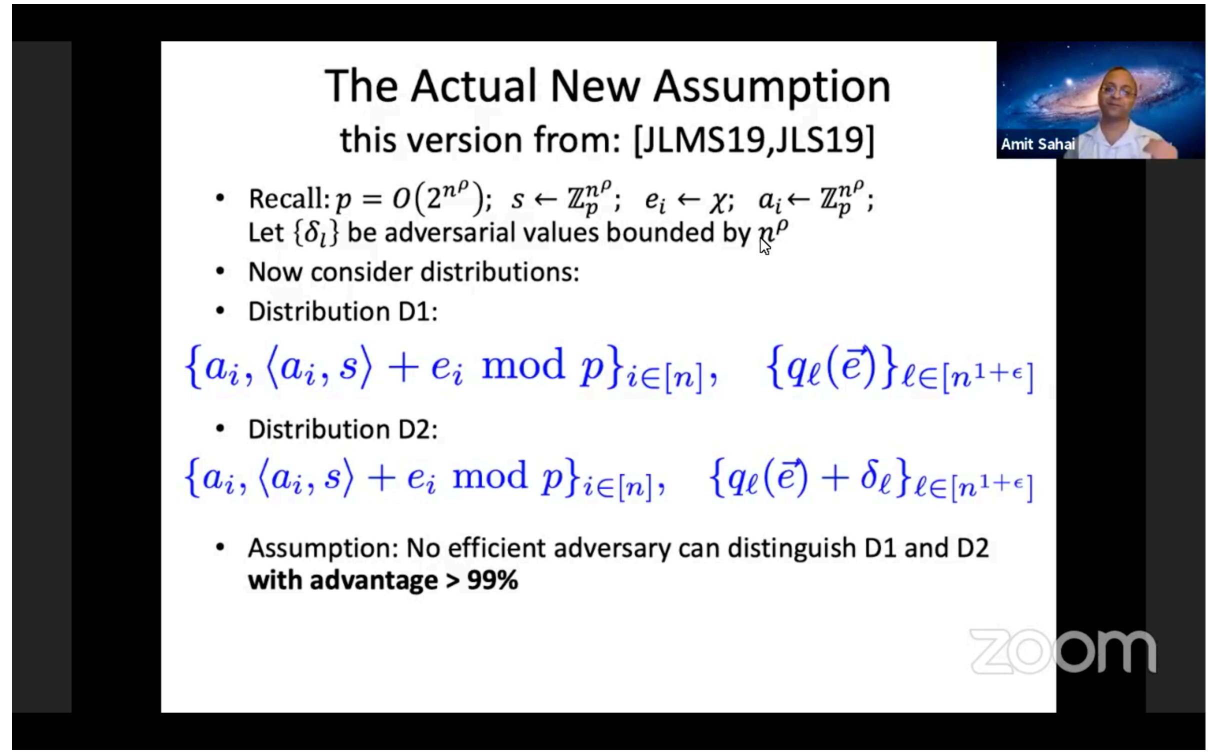
2013

2020

Timeline not drawn to scale

Code Obfuscation: compiling code to hide the implementation

GGH13 + GGHRSW



2013

2020

2020 + T

IO from standard assumptions

Timeline not drawn to scale

Code Obfuscation: compiling code to hide the implementation

What functions can be obfuscated using standard assumptions (e.g. LWE) ?

Code Obfuscation: compiling code to hide the implementation

What functions can be obfuscated using standard assumptions (e.g. LWE) ?

Prior Works: VBB Obfuscation for simple function classes.

- Point Functions [C97, …]
- Hyperplanes [CRV09]
- Conjunctions [BVWW16, …]

CODE OBFUSCATION: COMPILING CODE TO HIDE THE IMPLEMENTATION

What functions can be obfuscated using standard assumptions (e.g. LWE) ?

Prior Works: VBB Obfuscation for simple function classes.

- Point Functions [C97, …]
- Hyperplanes [CRV09]
- Conjunctions [BVWW16, …]

- **Today: Compute-and-compare programs**

The Function Class: Compute-and-Compare Programs

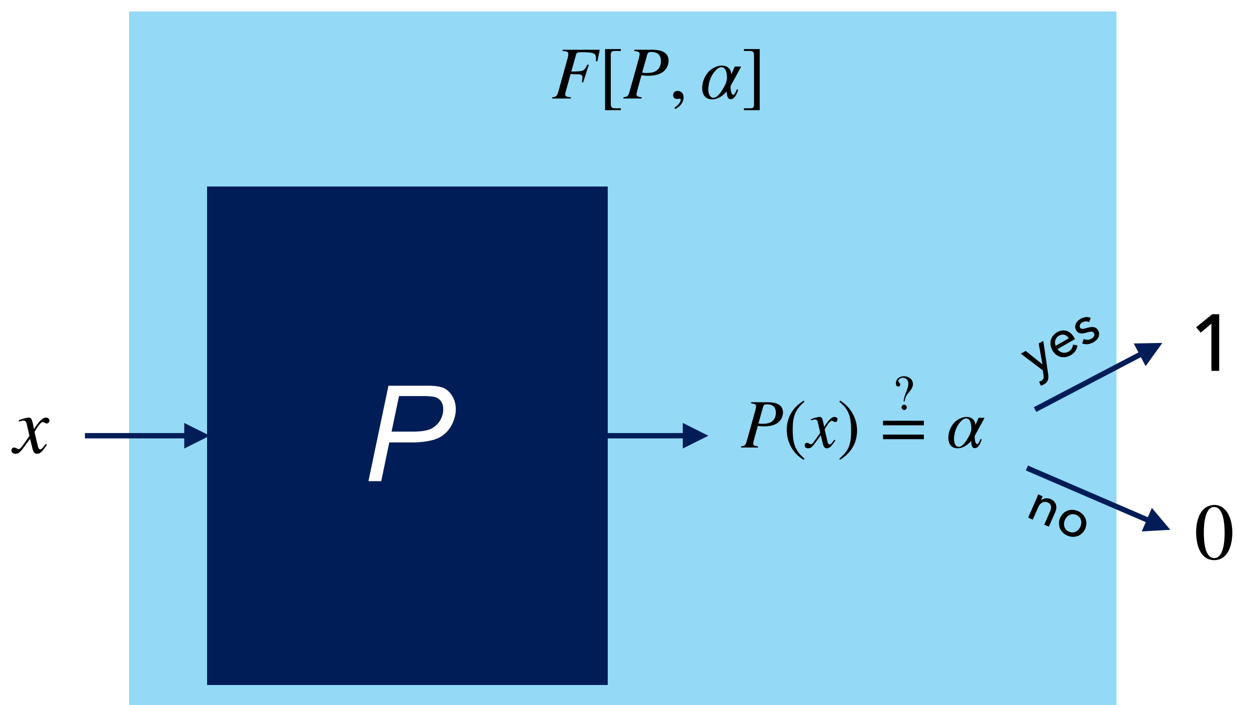
The Function Class: Compute-and-Compare Programs

Every function parameterized by program P and string α

$F[P, \alpha]$

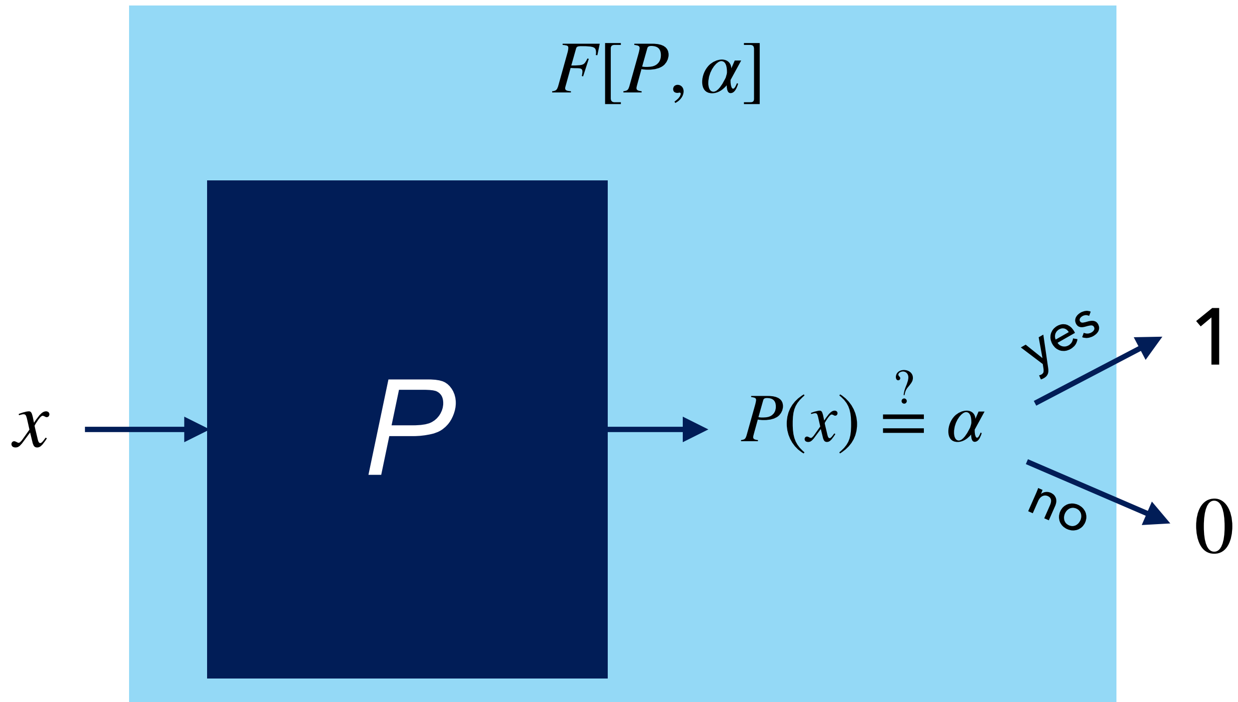
The Function Class: Compute-and-Compare Programs

Every function parameterized by program P and string α



The Function Class: Compute-and-Compare Programs

Every function parameterized by program P and string α



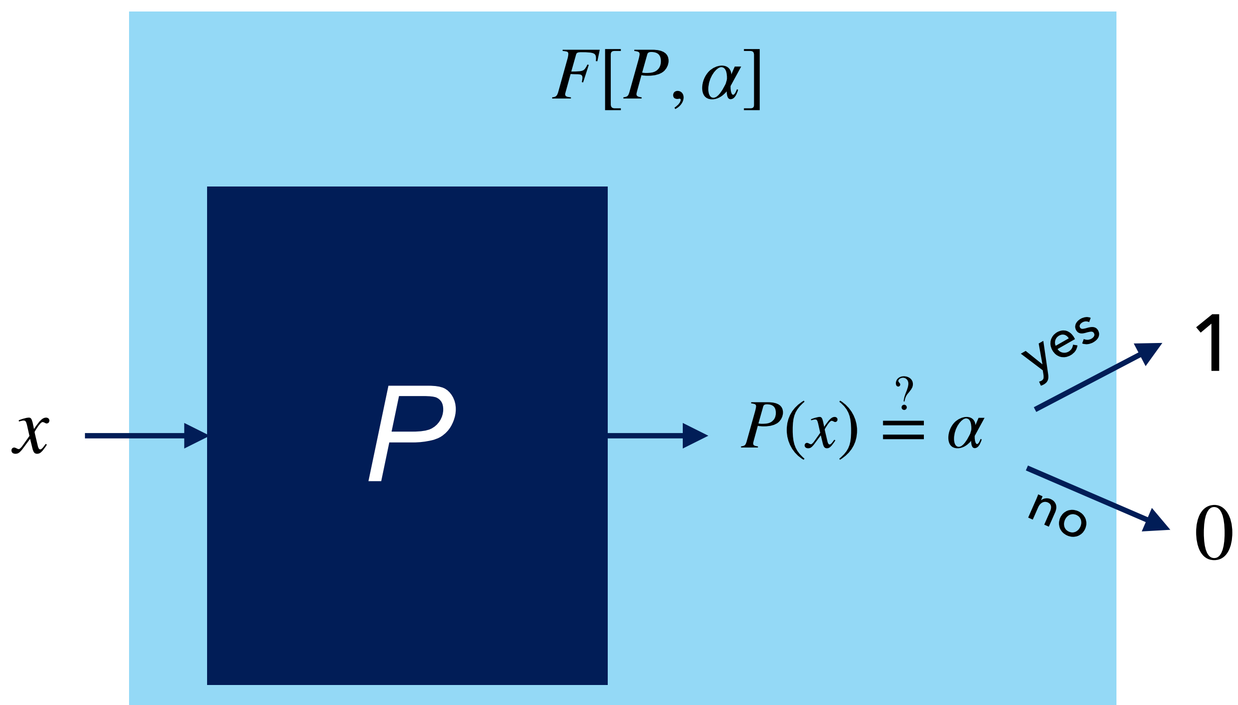
POINT FUNCTIONS

$$\mathsf{PF}[\alpha](x) = \begin{cases} 1 \text{ if } x = \alpha \\ 0 \text{ otherwise} \end{cases}$$

$$\mathsf{PF}[\alpha] \equiv F[\mathsf{Id}, \alpha]$$

The Function Class: Compute-and-Compare Programs

Every function parameterized by program P and string α



Point Functions

$$\mathsf{PF}[\alpha](x) = \begin{cases} 1 \text{ if } x = \alpha \\ 0 \text{ otherwise} \end{cases}$$

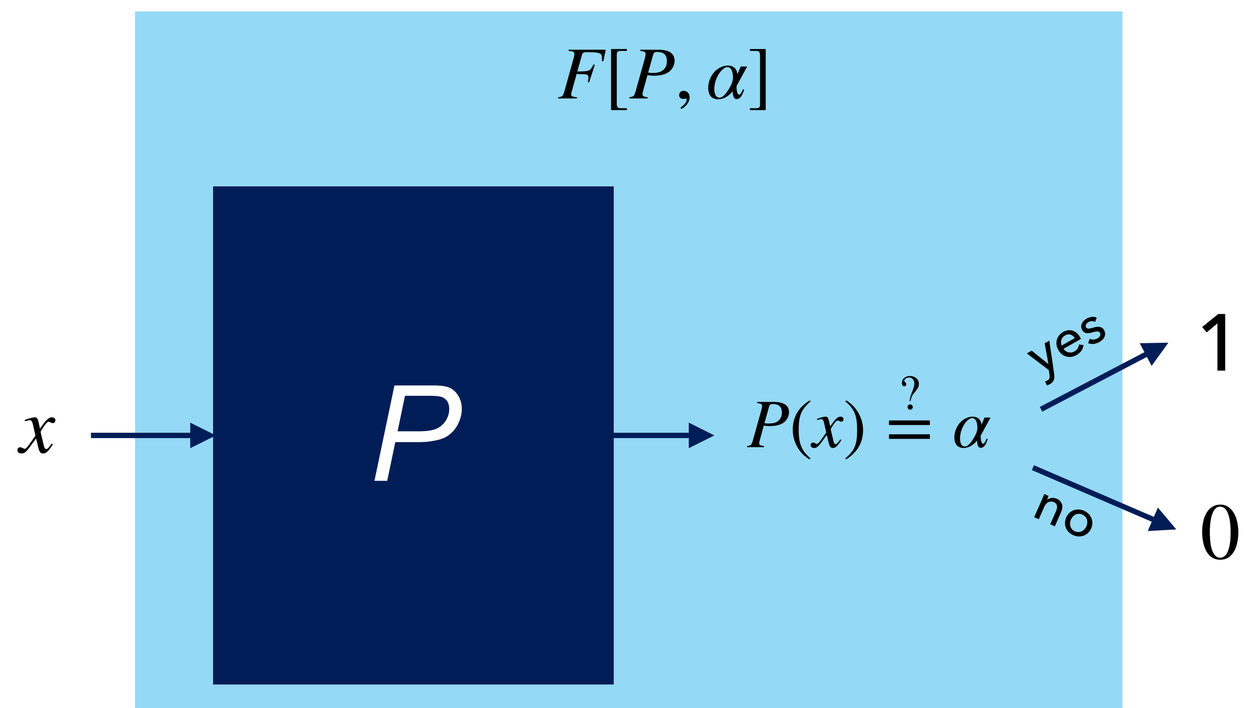
$$\mathsf{PF}[\alpha] \equiv F[\mathsf{Id}, \alpha]$$

Conjunctions

$$f(x_1, \ldots, x_n) = x_1 \wedge x_5 \wedge \overline{x_7}$$

The Function Class: Compute-and-Compare Programs

Every function parameterized by program P and string α



POINT FUNCTIONS

$$\mathsf{PF}[\alpha](x) = \begin{cases} 1 \text{ if } x = \alpha \\ 0 \text{ otherwise} \end{cases}$$

$$\mathsf{PF}[\alpha] \equiv F[\mathsf{Id}, \alpha]$$

CONJUNCTIONS

$$f(x_1, \ldots, x_n) = x_1 \wedge x_5 \wedge \overline{x_7}$$

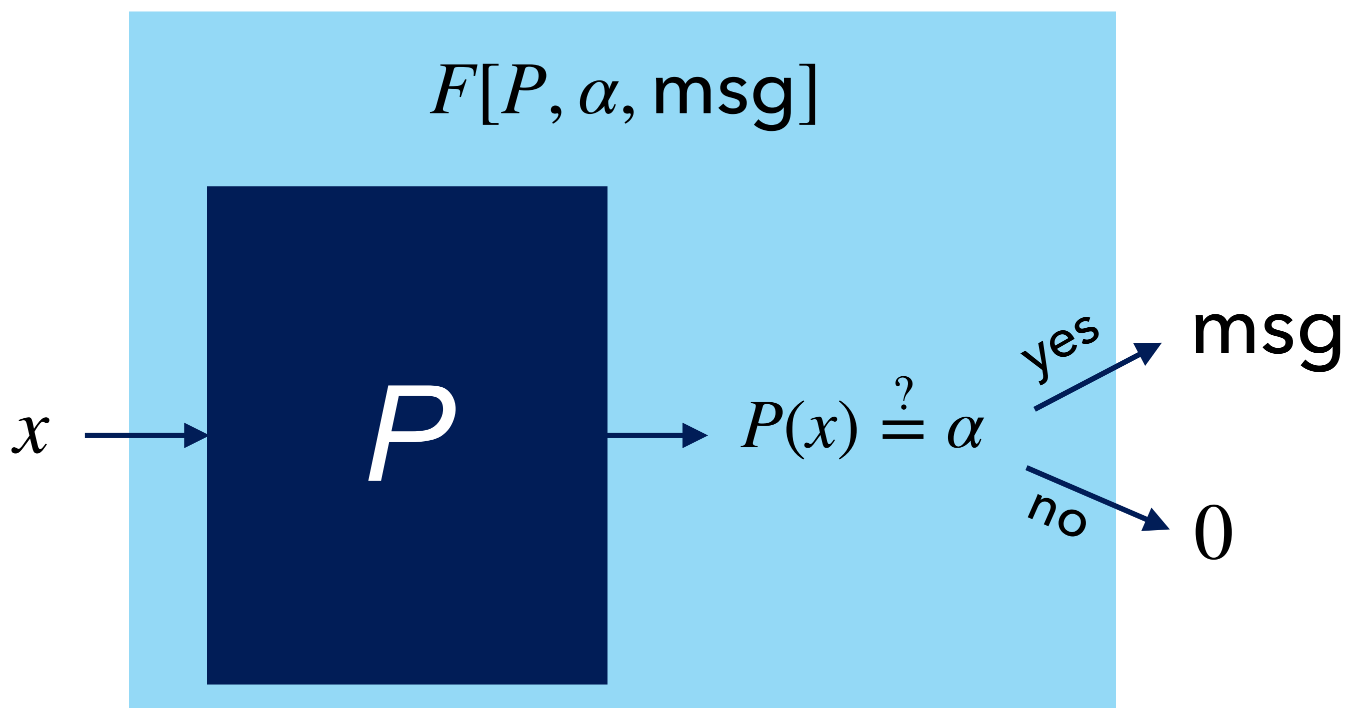
$$P(x_1, \ldots, x_n) = (x_1, x_5, x_7)$$

$$\alpha = 110$$

$$f \equiv F[P, \alpha]$$

The Function Class: Compute-and-Compare Programs

Every function parameterized by program P and strings α, msg



Security: (Average) VBB Obfuscation

Security: (Average) VBB Obfuscation

SECURITY

Security: (Average) VBB Obfuscation

SECURITY

For randomly chosen α

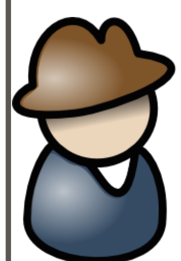
Obf$[P, \alpha, \text{msg}]$ hides P, msg

Security: (Average) VBB Obfuscation

SECURITY

For randomly chosen α

Obf$[P, \alpha, \text{msg}]$ hides P, msg



P, msg ←

α : random string

Obf$[P, \alpha, \text{msg}]$ →
Obf[#, #, #]

Guess ←

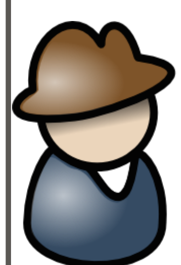
SECURITY: (AVERAGE) VBB OBFUSCATION

SECURITY

lock

For randomly chosen α

Obf[P, α, msg] hides P, msg



P, msg ←

α : random string

Obf[P, α, msg] →

Obf[#, #, #]

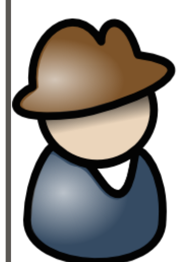
Guess ←

LOCKABLE OBFUSCATION (LO)

SECURITY

For randomly chosen α ← lock

Obf$[P, \alpha, \text{msg}]$ hides P, msg



P, msg ←

α : random string

Obf$[P, \alpha, \text{msg}]$ →

Obf[#, #, #]

Guess ←

Lockable Obfuscation (LO)

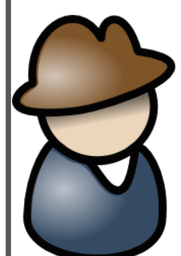
SECURITY

lock

For randomly chosen α

Obf[P, α, msg] hides P, msg

lock must be long enough



P, msg ←

α : random string

Obf[P, α, msg] →

Obf[#, #, #]

Guess ←

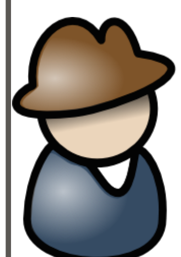
Lockable Obfuscation (LO)

SECURITY

lock

For randomly chosen α

Obf[P, α, msg] hides P, msg



P, msg

α : random string

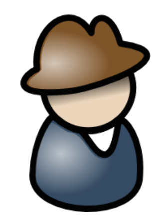
Obf[P, α, msg]

Obf[#, #, #]

Guess

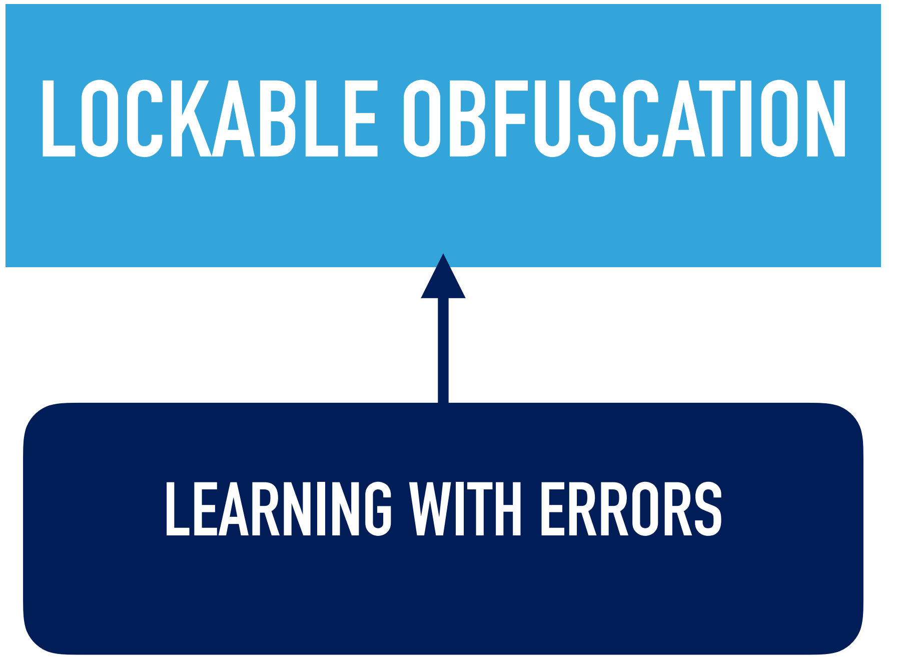
lock must be long enough

Single bit lock?

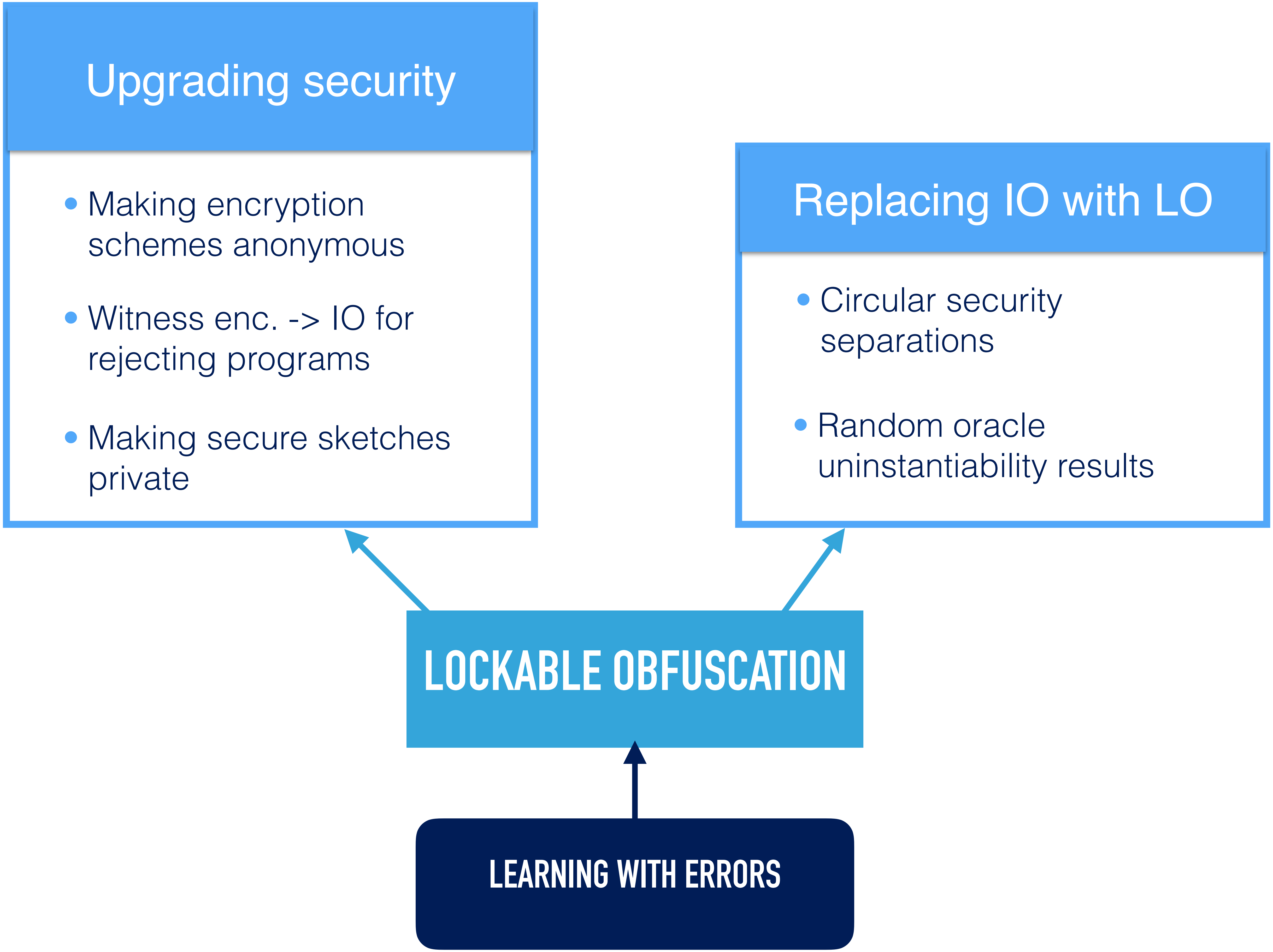


P = all accepting prog.

If $\alpha = 1$, adversary can distinguish


LOCKABLE OBFUSCATION
LEARNING WITH ERRORS

Upgrading security
Making encryption schemes anonymous
Witness enc. -> IO for rejecting programs
Making secure sketches private
LOCKABLE OBFUSCATION
LEARNING WITH ERRORS


Upgrading security
Making encryption schemes anonymous
Witness enc. -> IO for rejecting programs
Making secure sketches private
Replacing IO with LO
Circular security separations
Random oracle uninstantiability results
LOCKABLE OBFUSCATION
LEARNING WITH ERRORS

Lockable Obfuscation: Applications

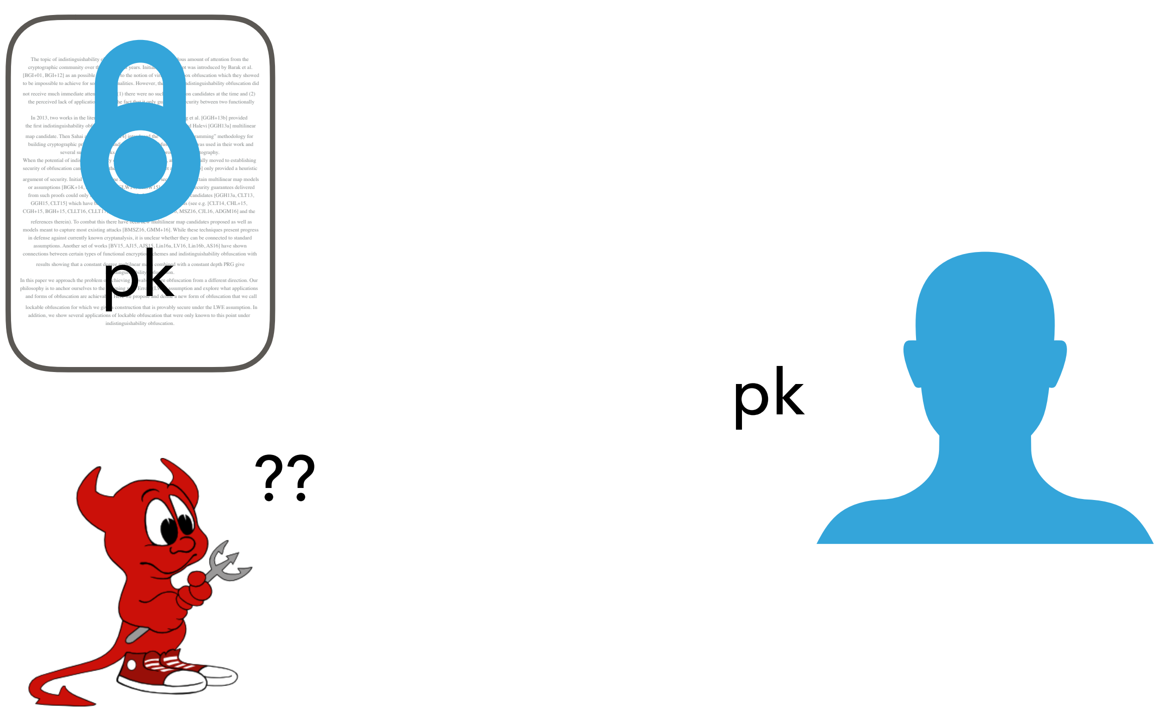
Anonymous encryption schemes

[Bellare, Boldyreva, Desai, Pointcheval 01]

Lockable Obfuscation: Applications

Anonymous encryption schemes

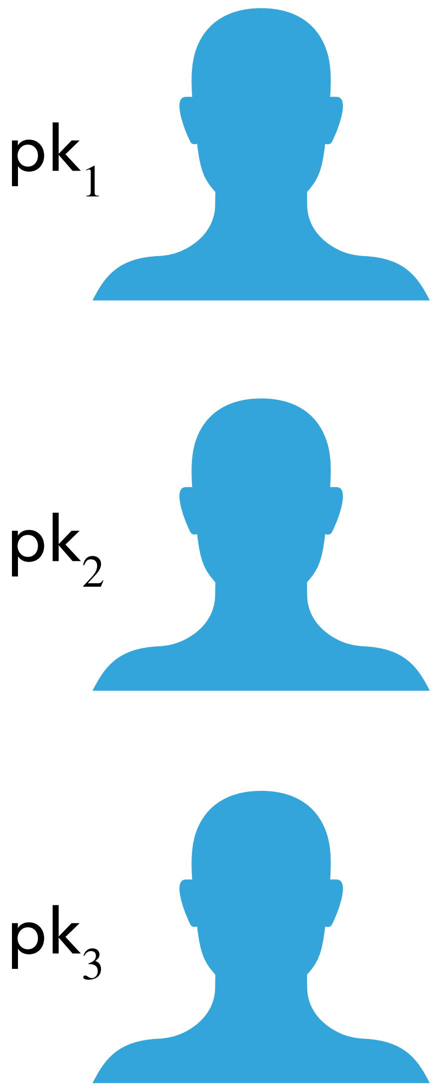
[Bellare, Boldyreva, Desai, Pointcheval 01]



Lockable Obfuscation: Applications

Anonymous encryption schemes

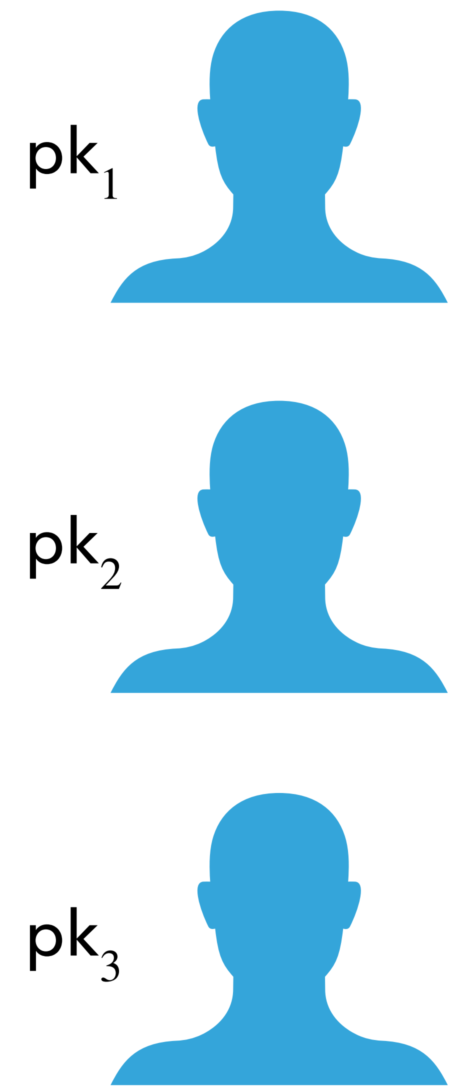
[Bellare, Boldyreva, Desai, Pointcheval 01]



Lockable Obfuscation: Applications

Anonymous encryption schemes

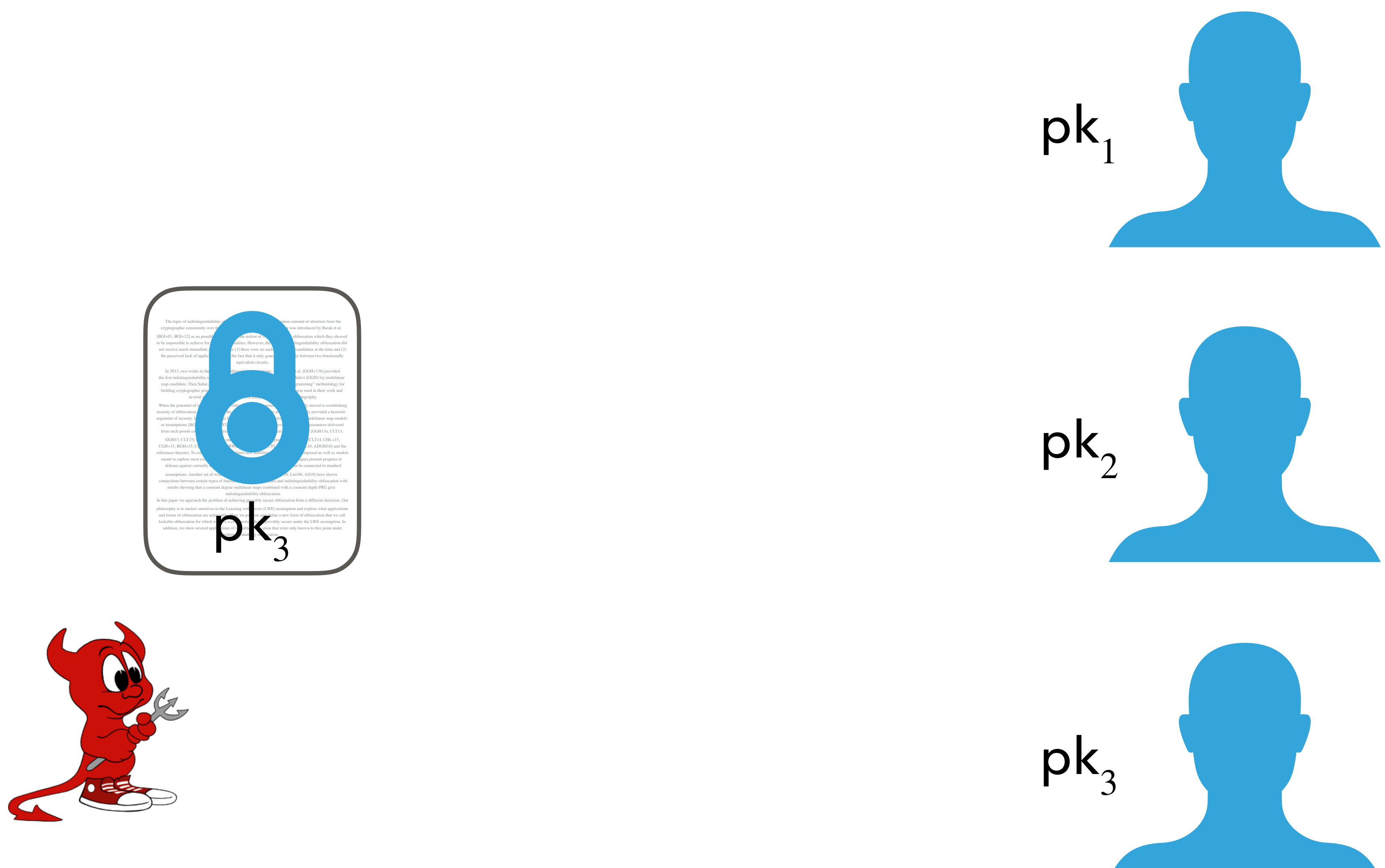
[Bellare, Boldyreva, Desai, Pointcheval 01]



Lockable Obfuscation: Applications

Anonymous encryption schemes

[Bellare, Boldyreva, Desai, Pointcheval 01]



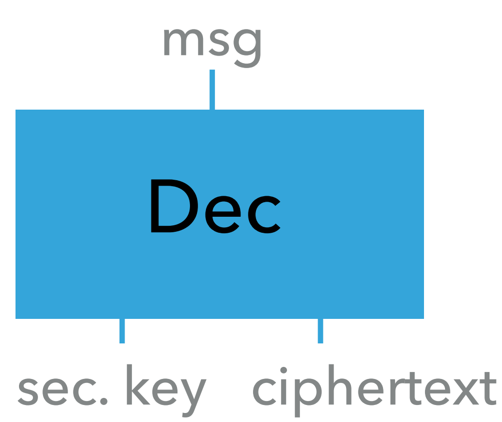
Does adversary learn this message is for pk_3?

Any encryption scheme can be made anonymous using LO.

$(\text{Enc}, \text{Dec}) \rightarrow (\text{Enc-anon}, \text{Dec-anon})$

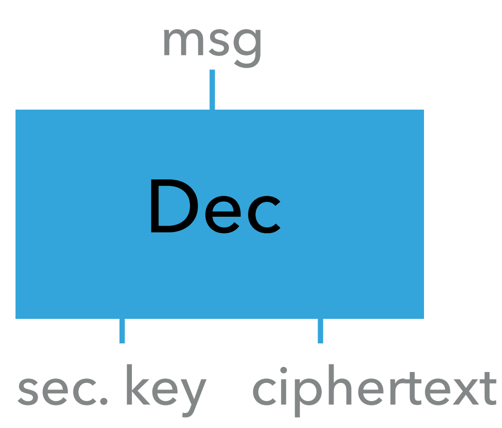
Any encryption scheme can be made anonymous using LO.

(Enc, Dec) → (Enc-anon, Dec-anon)



Any encryption scheme can be made anonymous using LO.

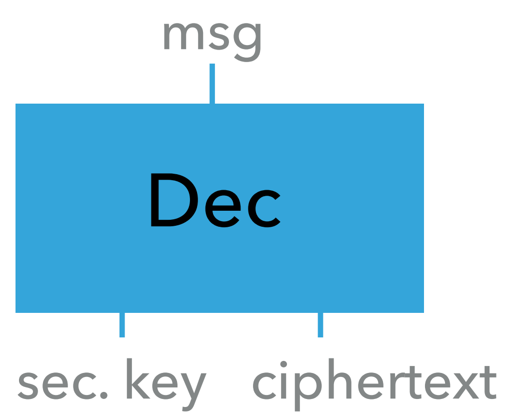
(Enc, Dec) → (Enc-anon, Dec-anon)



Enc-anon(m, pk)

Any encryption scheme can be made anonymous using LO.

(Enc, Dec) → (Enc-anon, Dec-anon)

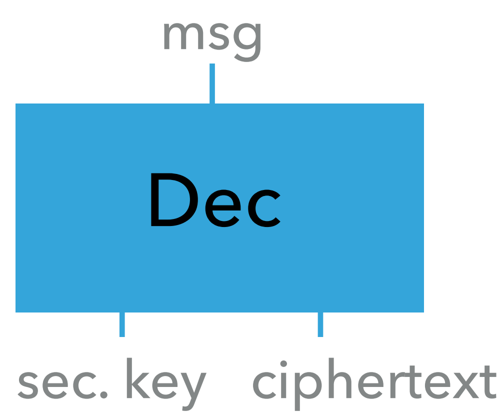


Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

ANY ENCRYPTION SCHEME CAN BE MADE ANONYMOUS USING LO.

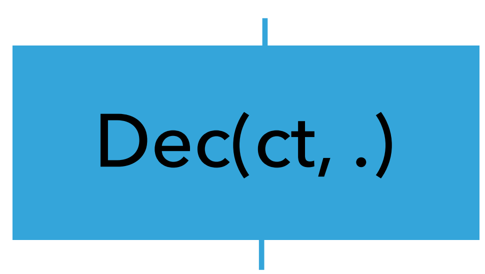
(Enc, Dec) → (Enc-anon, Dec-anon)



Enc-anon(m, pk)

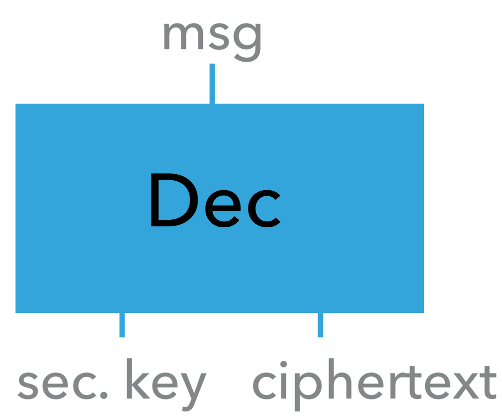
Choose random string α. ct = Enc(α, pk)

P :



Any encryption scheme can be made anonymous using LO.

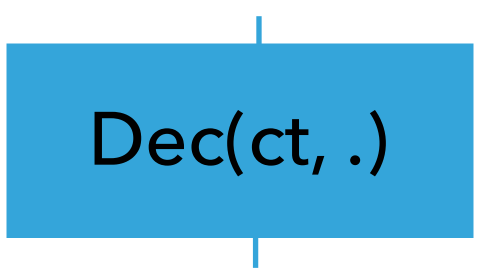
(Enc, Dec) → (Enc-anon, Dec-anon)



Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

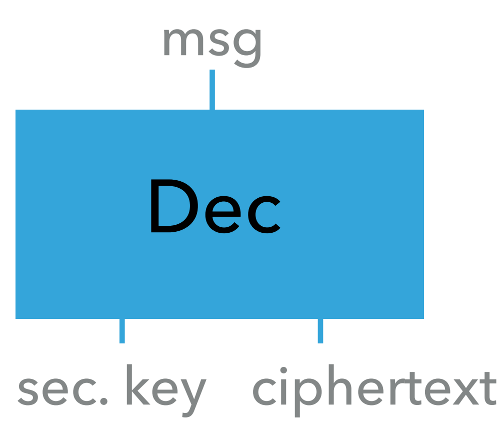
P :



Output Obf[P, α, m] as anon. ciphertext

ANY ENCRYPTION SCHEME CAN BE MADE ANONYMOUS USING LO.

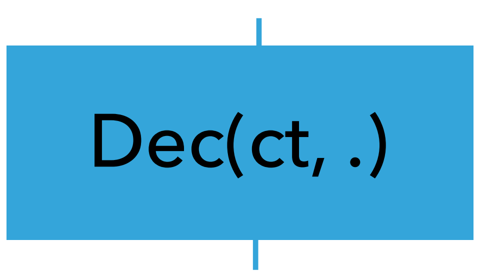
(Enc, Dec) → (Enc-anon, Dec-anon)



Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

P :



Output Obf[P, α, m] as anon. ciphertext

Dec-anon using sk

Run program with input = sk

Any encryption scheme can be made anonymous using LO.

(Enc, Dec) → (Enc-anon, Dec-anon)

Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

P: Dec(ct, .)

Anon. ct: Obf[P, α, m]

Any encryption scheme can be made anonymous using LO.

(Enc, Dec) → (Enc-anon, Dec-anon)

Why decryption works

P : Dec(ct, .)

Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

P: Dec(ct, .)

Anon. ct: Obf[P, α, m]

Any encryption scheme can be made anonymous using LO.

(Enc, Dec) → (Enc-anon, Dec-anon)

Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

P: Dec(ct, .)

Anon. ct: Obf[P, α, m]

<u>Why decryption works</u>

P : Dec(ct, .)

P(sk) = α

ANY ENCRYPTION SCHEME CAN BE MADE ANONYMOUS USING LO.

(Enc, Dec) → (Enc-anon, Dec-anon)

Enc-anon(m, pk)

Choose random string α. ct = Enc(α, pk)

P: Dec(ct, .)

Anon. ct: Obf[P, α, m]

Why decryption works

P : Dec(ct, .)

P(sk) = α

Obf[P, α, m](sk) = m

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Enc-anon(m, pk)

$\mathsf{ct} = \mathsf{Enc}(\alpha, \mathsf{pk})$

P : Dec(ct, .)

Anon. ct: Obf[P, α, m]

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Can guess pk from Obf(P, α, m)

Enc-anon(m, pk)

$\mathsf{ct} = \mathsf{Enc}(\alpha, \mathsf{pk})$

P : Dec(ct, .)

Anon. ct: Obf[P, α, m]

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Can guess pk from Obf(P, α, m)

Enc-anon(m, pk)

ct = Enc(α, pk)

P : Dec(ct, .)

Anon. ct: Obf[P, α, m]

Enc-anon'(m, pk)

ct' = Enc(0, pk)

P' : Dec(ct', .)

Anon. ct': Obf[P', α, m]

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Enc-anon(m, pk)

ct $=$ Enc$(\alpha,$ pk$)$

P : Dec(ct, .)

Anon. ct: Obf[P, α, m]

$\approx$

Security of Enc

Enc-anon'(m, pk)

ct' $=$ Enc$(0,$ pk$)$

P' : Dec(ct', .)

Anon. ct': Obf[P', α, m]

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Can guess pk from Obf(P', α, m)

Enc-anon'(m, pk)

ct' = Enc(0, pk)

P' : Dec(ct', .)

Anon. ct': Obf[P', α, m]

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Can guess pk from Obf(P', α, m)

Enc-anon'(m, pk)

ct' = Enc(0, pk)

P' : Dec(ct', .)

Anon. ct': Obf[P', α, m]

$\approx$

Security of LO

Enc-anon"(m, pk)

Anon. ct": Obf[#, #, #]

Any encryption scheme can be made anonymous using LO.

Security proof: sketch

Can guess pk from Obf(#, #, #)

Enc-anon"(m, pk)

Anon. ct": Obf[#, #, #]

Even advanced encryption schemes can be made anonymous using LO.

Broadcast Encryption

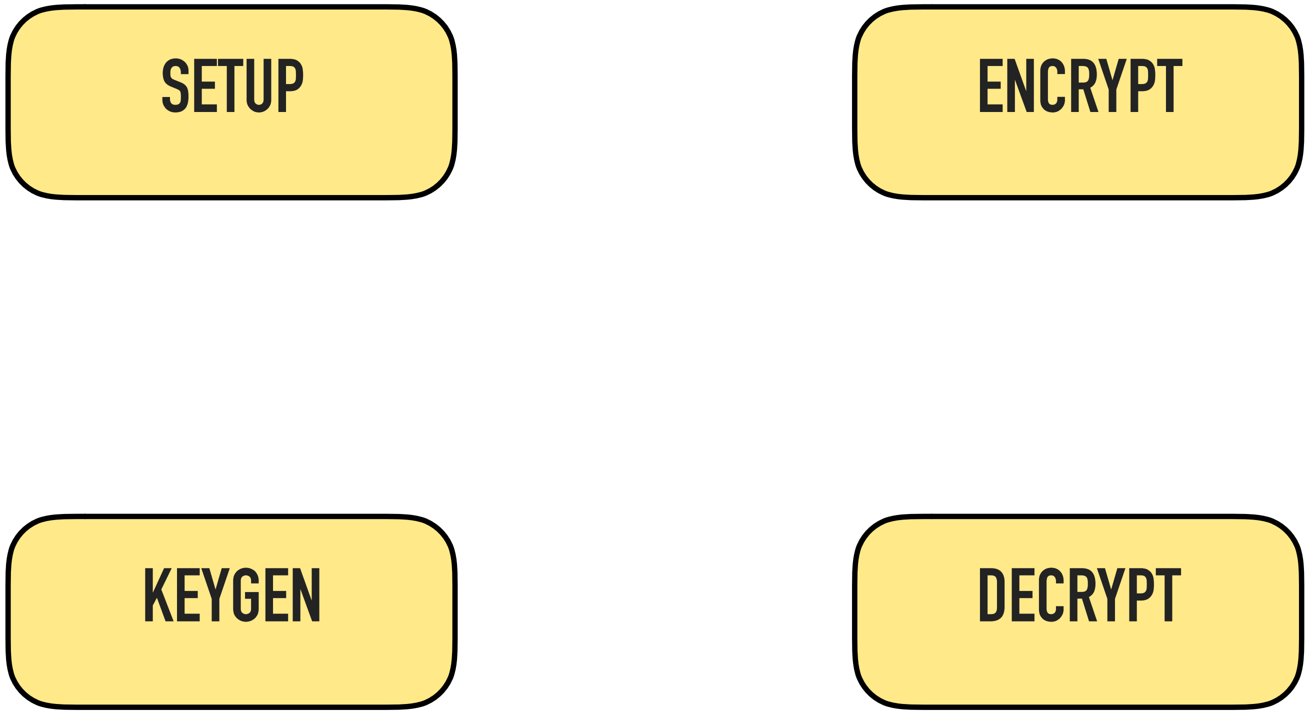
[Fiat, Naor 94]

Attribute Based Encryption

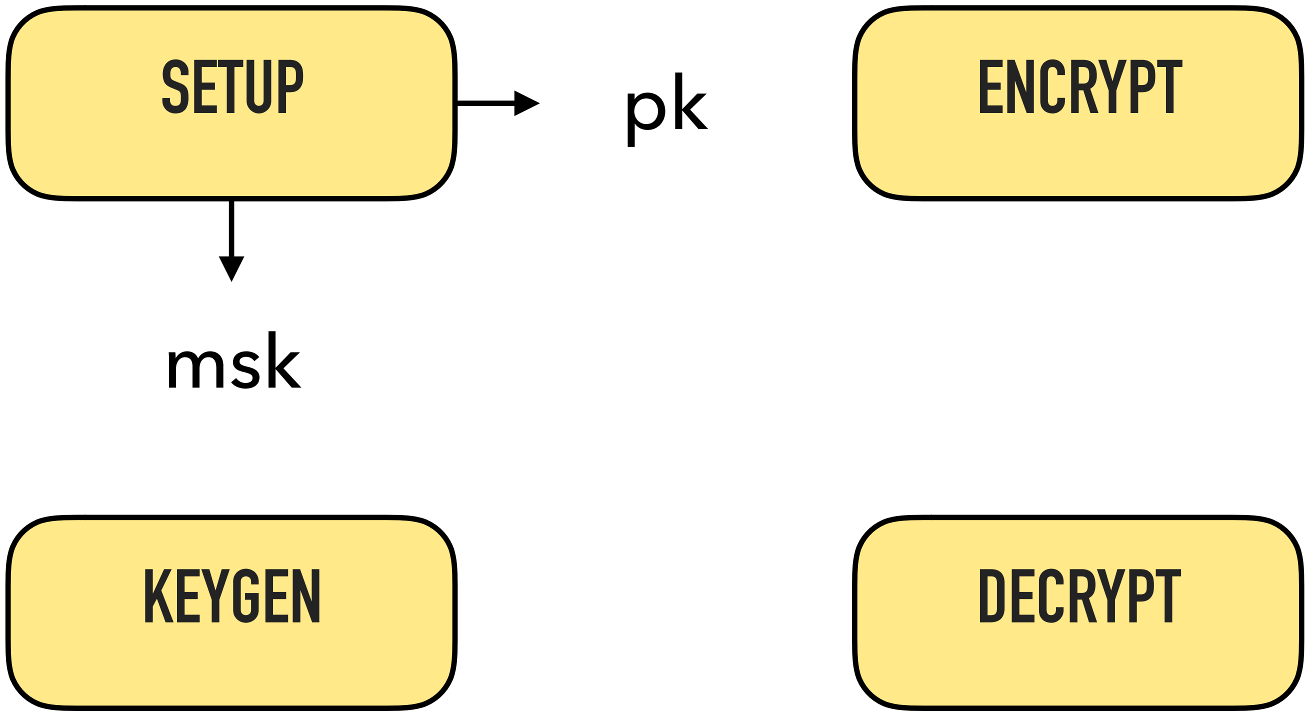
[Sahai,Waters 05]

ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]

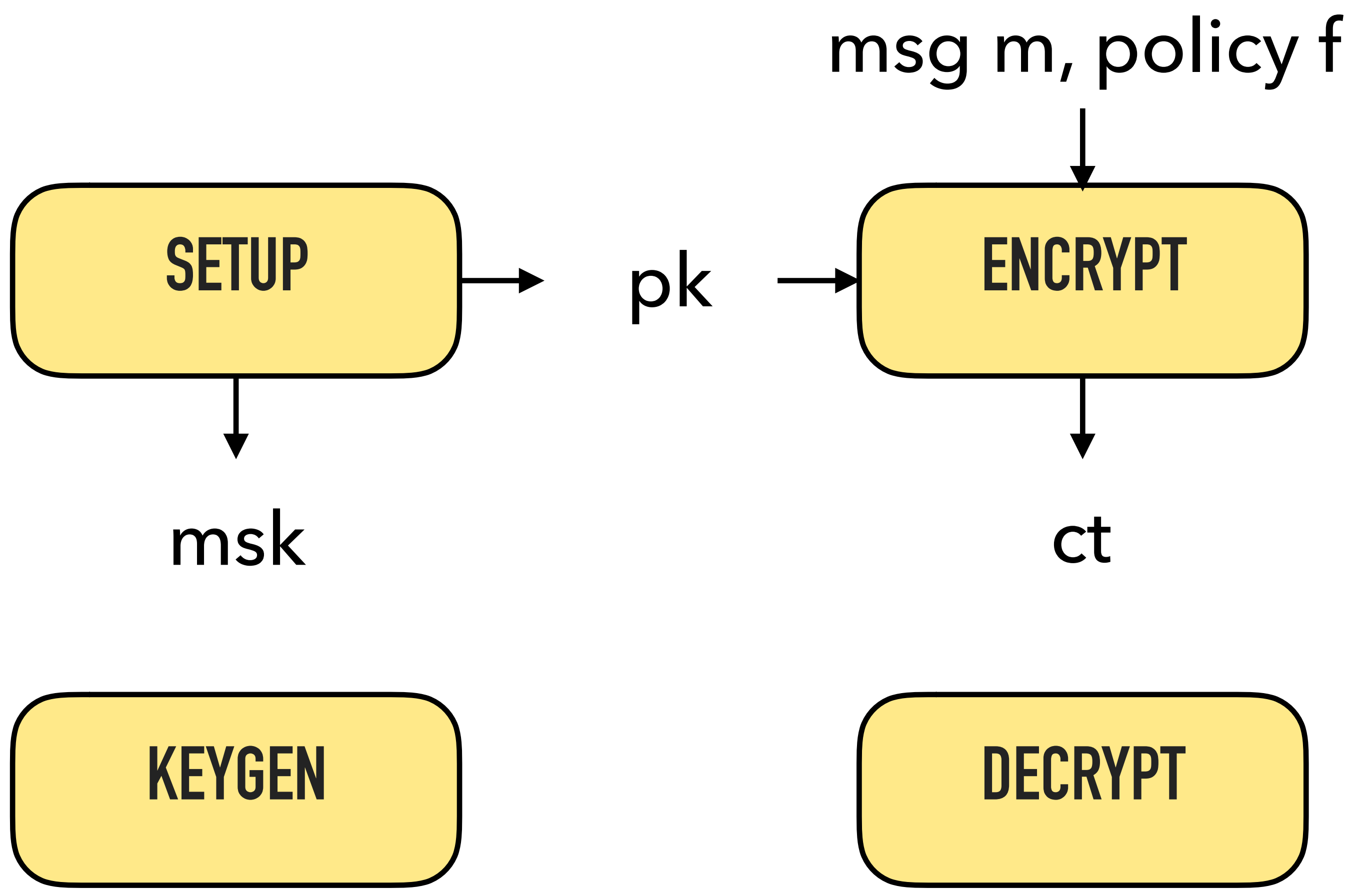
ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]



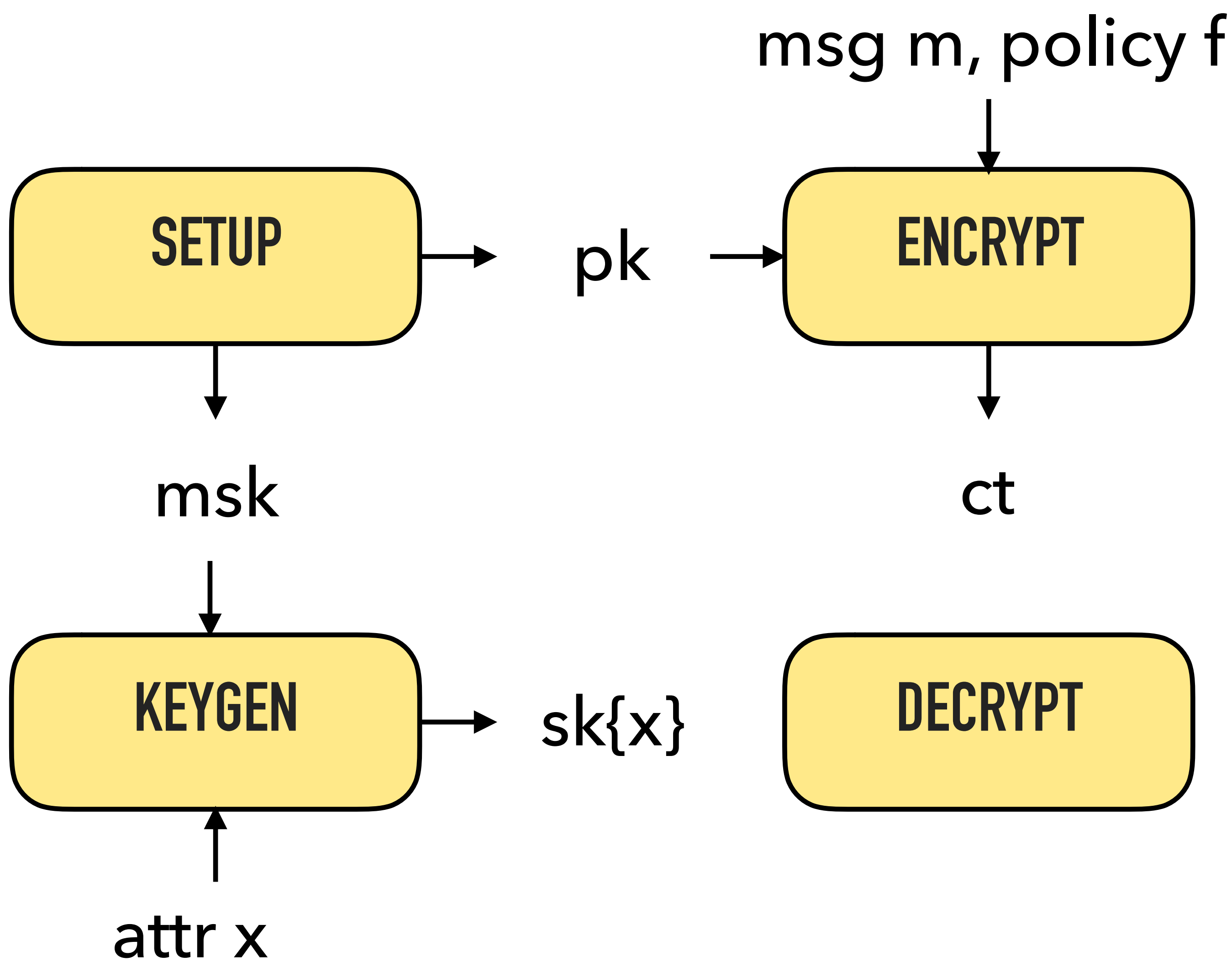
ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]



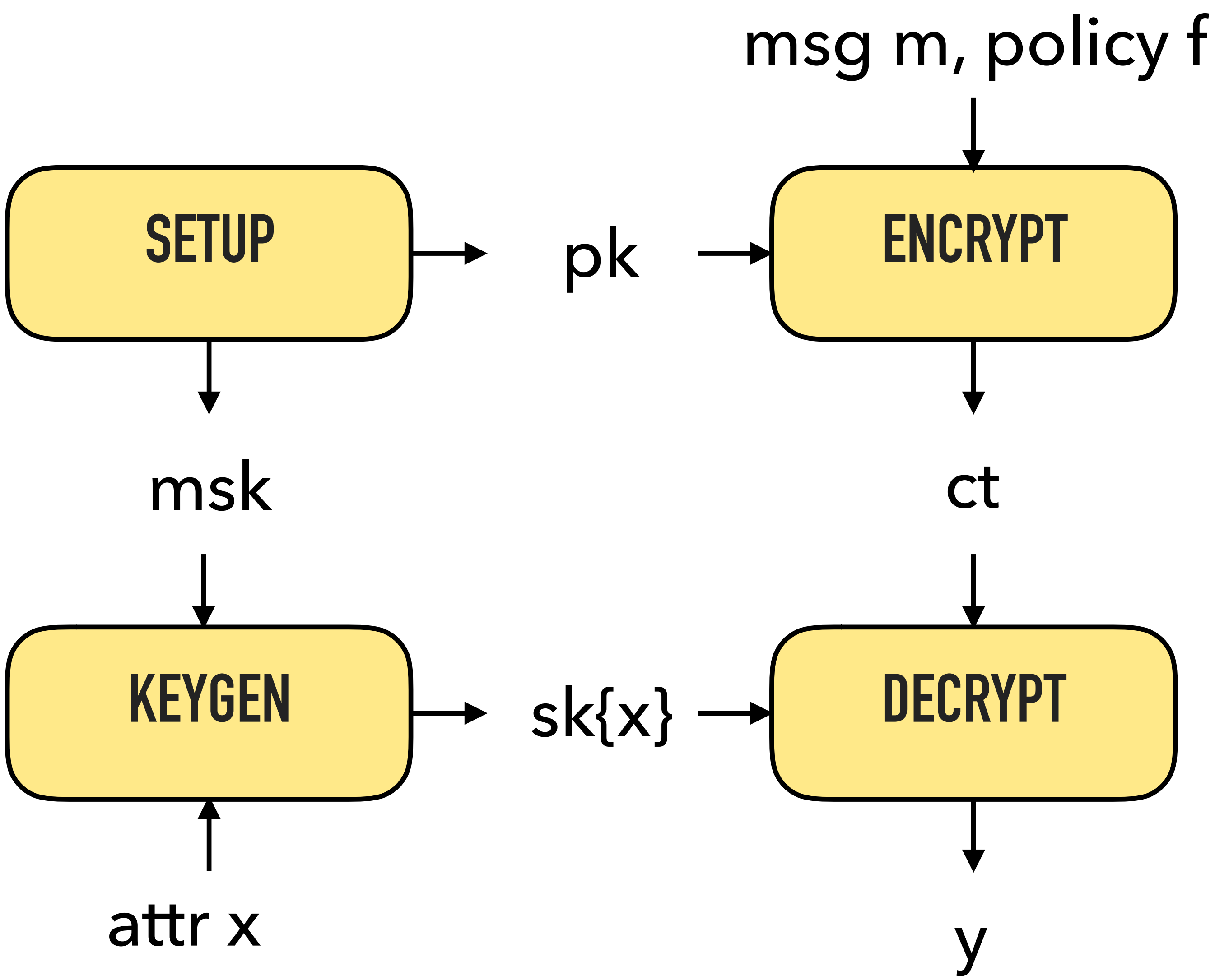
ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]



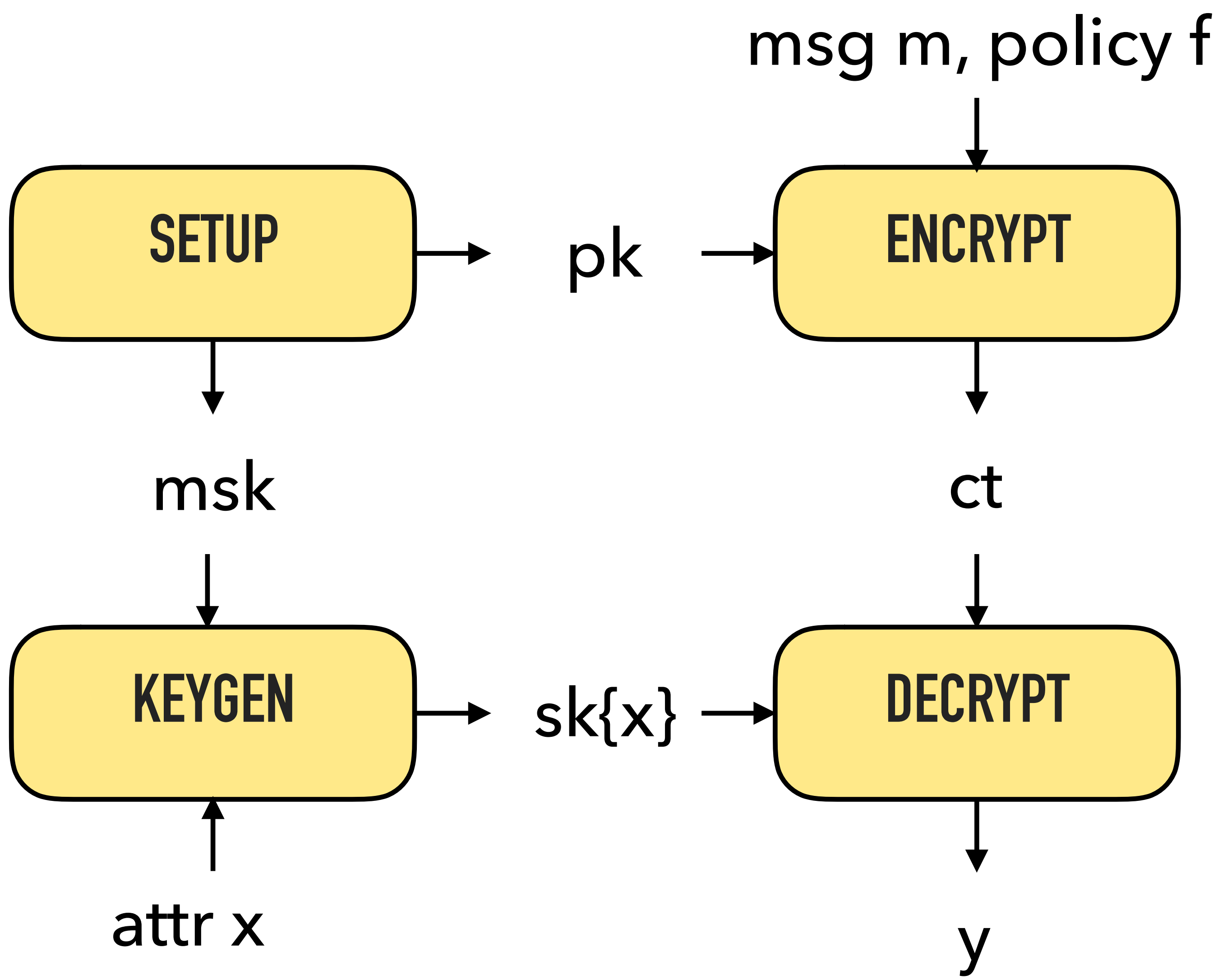
ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]



ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]



ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]



If f(x) = 1, then y = m

ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]

ATTRIBUTE BASED ENCRYPTION [Sahai-Waters 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ ... $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = m | policy f

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

PREDICATE ENCRYPTION

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = m | policy f

sk{x_1} sk{x_2} . . . sk{x_q}

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f is hidden
(even if $f(x_k) = 1$ for some k)

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f is hidden
(even if $f(x_k) = 1$ for some k)

IMPLIES OBFUSCATION!

ATTRIBUTE BASED ENCRYPTION [Sahai–Waters 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

ONE-SIDED PREDICATE ENCRYPTION

PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f is hidden
(even if $f(x_k) = 1$ for some k)

IMPLIES OBFUSCATION!

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

ONE-SIDED PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

If $f(x_k) = 0$ for all k, f is hidden

PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f is hidden
(even if $f(x_k) = 1$ for some k)

IMPLIES OBFUSCATION!

ATTRIBUTE BASED ENCRYPTION [SAHAI-WATERS 05]

ct = | m | policy f |

$sk\{x_1\}$ $sk\{x_2\}$ . . . $sk\{x_q\}$

ATTRIBUTE BASED ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f may not be hidden

ONE-SIDED PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

If $f(x_k) = 0$ for all k, f is hidden

If $f(x_k) = 1$ for some k, f may not be hidden

PREDICATE ENCRYPTION

If $f(x_k) = 0$ for all k, then m is hidden

f is hidden
(even if $f(x_k) = 1$ for some k)

IMPLIES OBFUSCATION!

ATTRIBUTE BASED ENC.

ATTRIBUTE BASED ENC.

Key-Policy
[GPSW 06]

Ciphertext-Policy
[BSW 07]

Multi-Authority
[C 07; CC 09]

Regular Languages
[W 12]

Circuits
[GVW 13]

Circuits : Short keys
[BGGHNSV 13]

NFA
[AMY 19]

ATTRIBUTE BASED ENC.

Key-Policy
[GPSW 06]

Ciphertext-Policy
[BSW 07]

Multi-Authority
[C 07; CC 09]

Regular Languages
[W 12]

Circuits
[GVW 13]

Circuits : Short keys
[BGGHNSV 13]

NFA
[AMY 19]

(ONE-SIDED) PREDICATE ENC.

Circuits : Short keys
[GVW 15]

ATTRIBUTE BASED ENC.

Key-Policy
[GPSW 06]

Ciphertext-Policy
[BSW 07]

Multi-Authority
[C 07; CC 09]

Regular Languages
[W 12]

Circuits
[GVW 13]

Circuits : Short keys
[BGGHNSV 13]

NFA
[AMY 19]

(ONE-SIDED) PREDICATE ENC.

Key-Policy

Ciphertext-Policy

Multi-Authority

Regular Languages

Circuits

Circuits : Short keys
[GVW 15]

NFA

LOCKABLE OBFUSCATION:

CONSTRUCTION

Lockable Obfuscation: Behind the scenes

Lockable Obfuscation: Behind the scenes

2-Circular Security Separations

[Bishop, Hohenberger, Waters 15]

Lockable Obfuscation: Behind the scenes

2-Circular Security Separations

[Bishop, Hohenberger, Waters 15]

n-Circular Security Separations

[Alamati, Peikert 16; K, Waters 16]

Lockable Obfuscation: Behind the scenes

2-Circular Security Separations

[Bishop, Hohenberger, Waters 15]

n-Circular Security Separations

[Alamati, Peikert 16; K, Waters 16]

Bit Circular Security Separations

[Goyal, K, Waters 17a]

Lockable Obfuscation: Behind the scenes

2-Circular Security Separations

[Bishop, Hohenberger, Waters 15]

n-Circular Security Separations

[Alamati, Peikert 16; K, Waters 16]

Bit Circular Security Separations

[Goyal, K, Waters 17a]

Lockable Obfuscation

[Goyal, K, Waters 17b; Wichs, Zirdelis 17]

Lockable Obfuscation: Construction

STEP 1: Lockable Obf. for NC^1

STEP 2: Bootstrapping

LO for NC^1 + FHE ⟶ LO for P/poly

* FHE with NC^1 decryption

Step 2: Bootstrapping Lockable Obfuscation

Obf() : Obfuscator for NC^1

(Enc, Dec, Eval) : FHE scheme

Obfuscation of Ckt. C with lock α :

Step 2: Bootstrapping Lockable Obfuscation

Obf() : Obfuscator for NC^1

(Enc, Dec, Eval) : FHE scheme

Obfuscation of Ckt. C with lock α:

Enc (C, sk)

Step 2: Bootstrapping Lockable Obfuscation

Obf() : Obfuscator for NC^1

(Enc, Dec, Eval) : FHE scheme

Obfuscation of Ckt. C with lock α:

Enc (C, sk) Obf(Dec (· , sk) α)

Obf. of Dec ckt with lock α

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: $\textsc{Enc}(C, \textsc{sk})$ $\textsc{Obf}\left(\textsc{Dec}(\cdot, \textsc{sk}) \quad \alpha \right)$

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: Enc (C, sk) $\text{Obf}\left(\text{Dec}(\cdot, \text{sk}) \quad \alpha \right)$

Evaluation on input x:

STEP 2: BOOTSTRAPPING LOCKABLE OBFUSCATION

Obfuscation of C: ENC (C, SK) OBF(DEC (· , SK) α)

Evaluation on input x:

ENC (C, SK) —Hom. Eval.→ ENC (C(x), SK)

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: Enc (C, sk) Obf(Dec (· , sk) α)

Evaluation on input x:

Enc (C, sk) —Hom. Eval.→ Enc (C(x), sk) → Obf(Dec (· , sk) α)

↓ $1 \text{ if } C(x) = \alpha$
0 otherwise

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: Enc (C, sk) $\text{Obf}\left(\text{Dec}(\cdot, \text{sk}) \quad \alpha \right)$

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: $\textsc{Enc}(C, \textsc{sk})$ $\textsc{Obf}\left(\textsc{Dec}(\cdot\,, \textsc{sk}) \quad \alpha \right)$

Security:

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: $\textsc{Enc}(C, \textsc{sk})$ $\quad \textsc{Obf}\left(\textsc{Dec}(\cdot\,, \textsc{sk}) \quad \alpha \right)$

$\approx$

Security of $\textsc{Obf}$

Security:

$\textsc{Enc}(C, \textsc{sk})$ $\quad \textsc{Obf}\left(\# \quad \# \right)$

Step 2: Bootstrapping Lockable Obfuscation

Obfuscation of C: Enc (C, sk) Obf(Dec (· , sk) α)

$\approx$

Security of Obf

Security:

Enc (C, sk) Obf(# #)

$\approx$

Security of FHE

Enc (#, sk) Obf(# #)

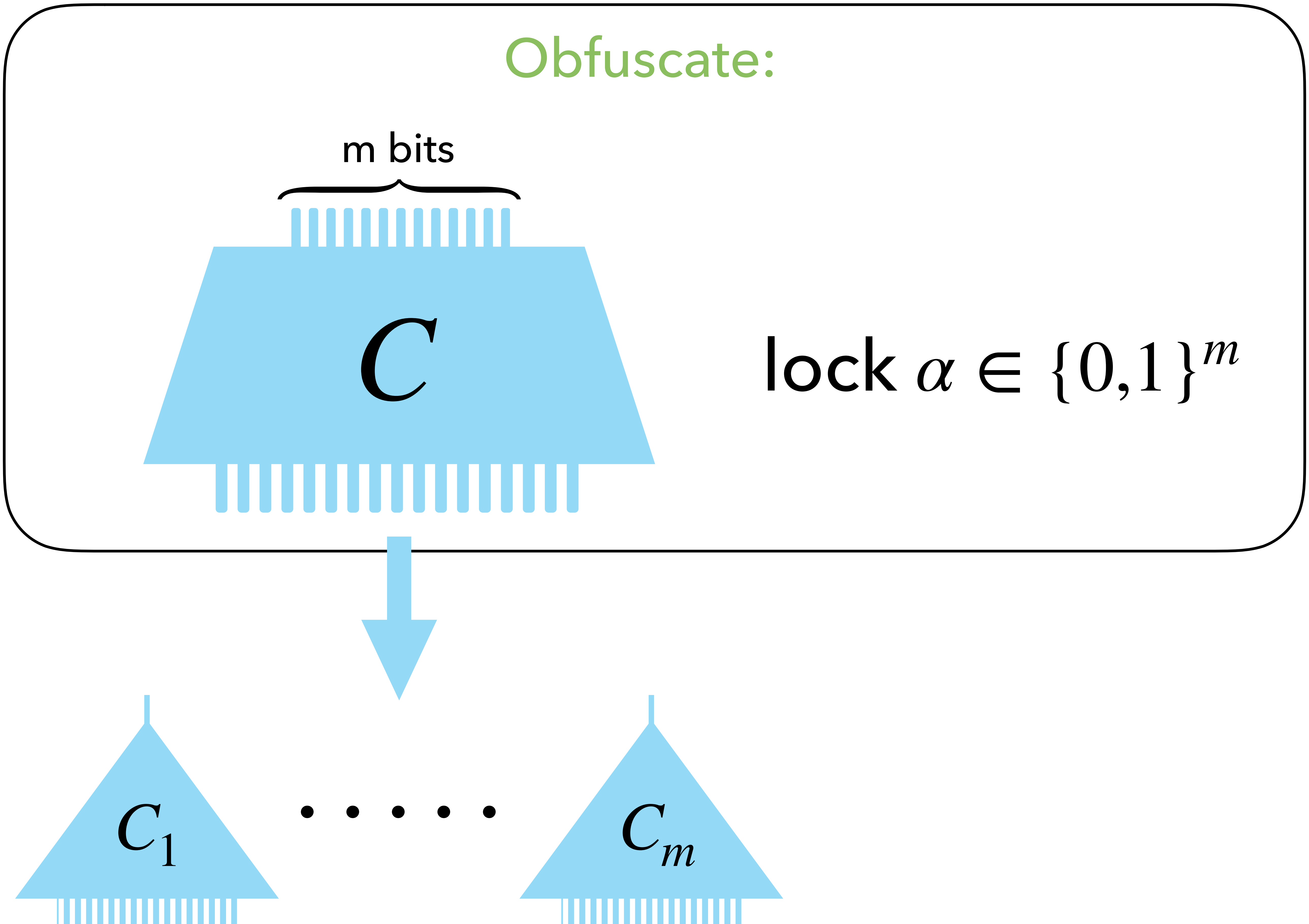
Lockable Obfuscation: Construction

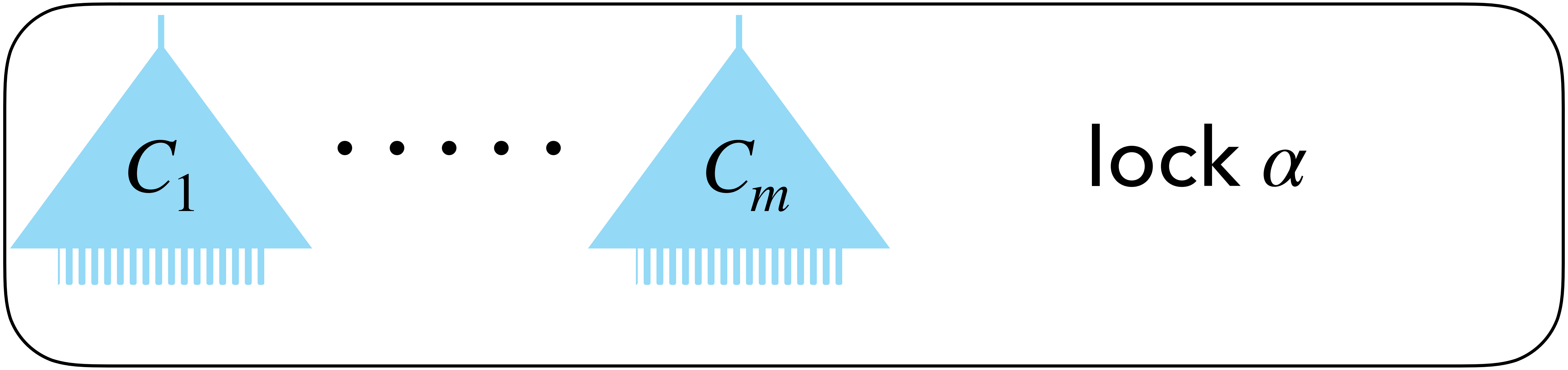
STEP 1: Lockable Obf. for NC^1

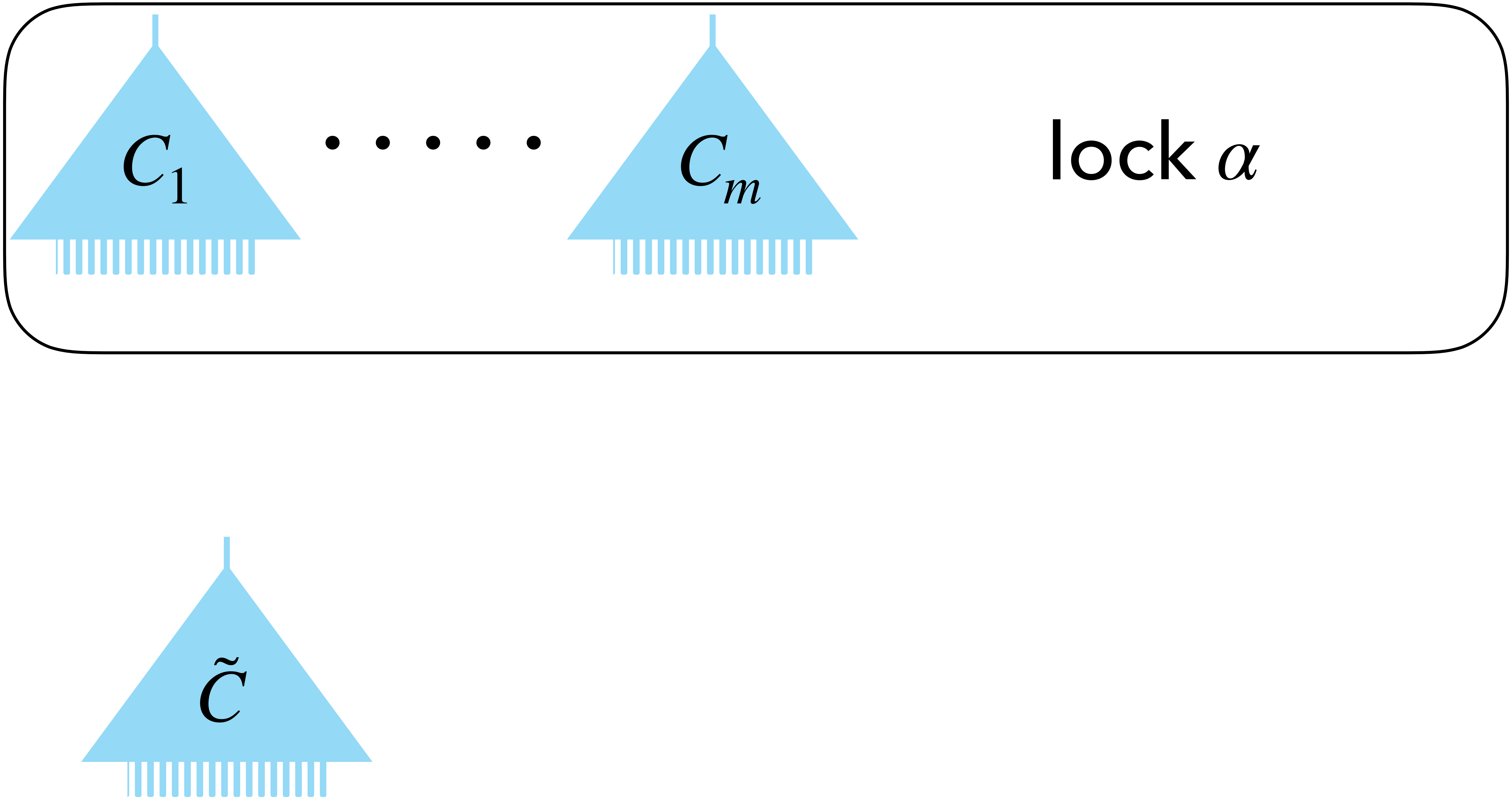
STEP 2: Bootstrapping

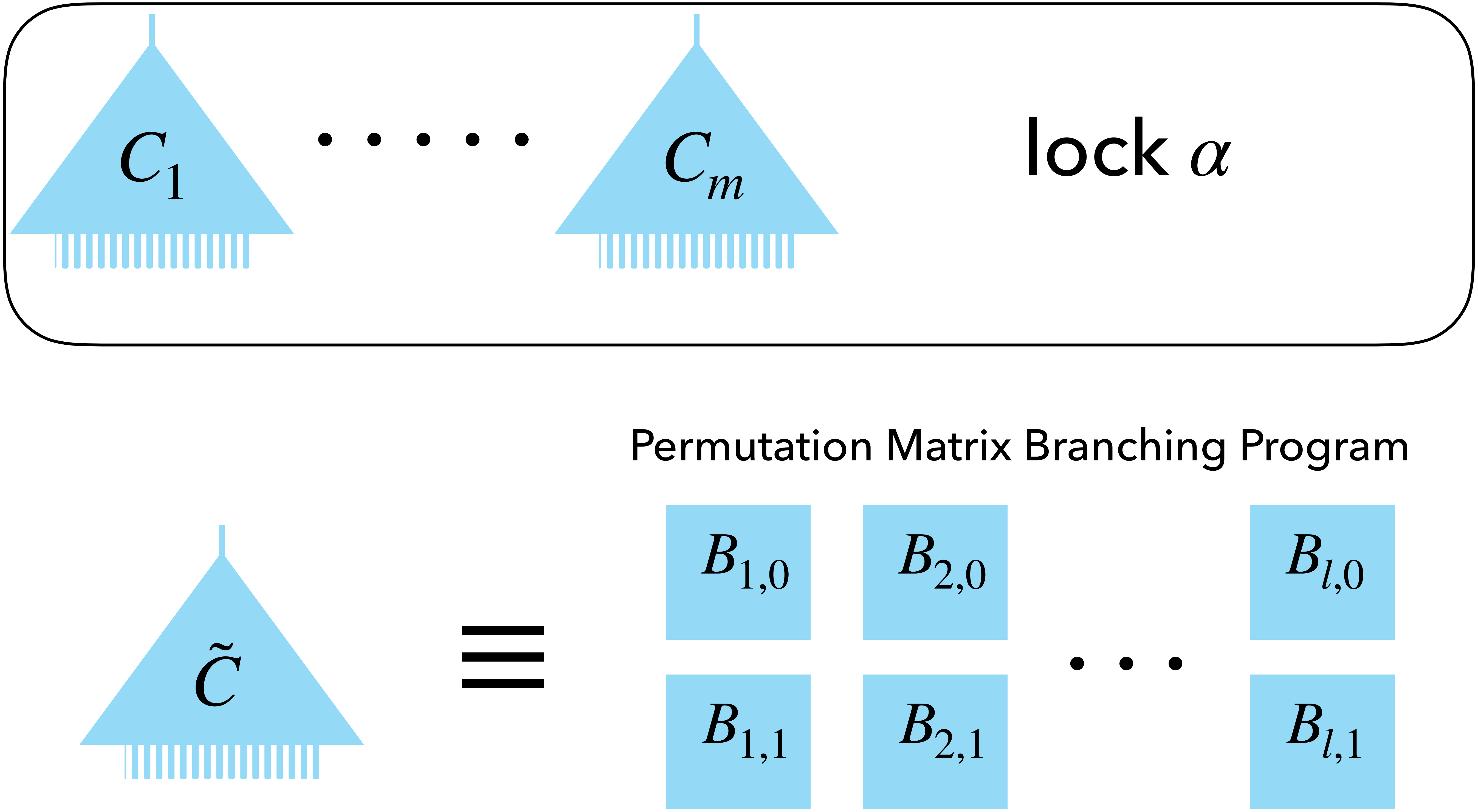
LO for NC^1 + FHE ⟶ LO for P/poly

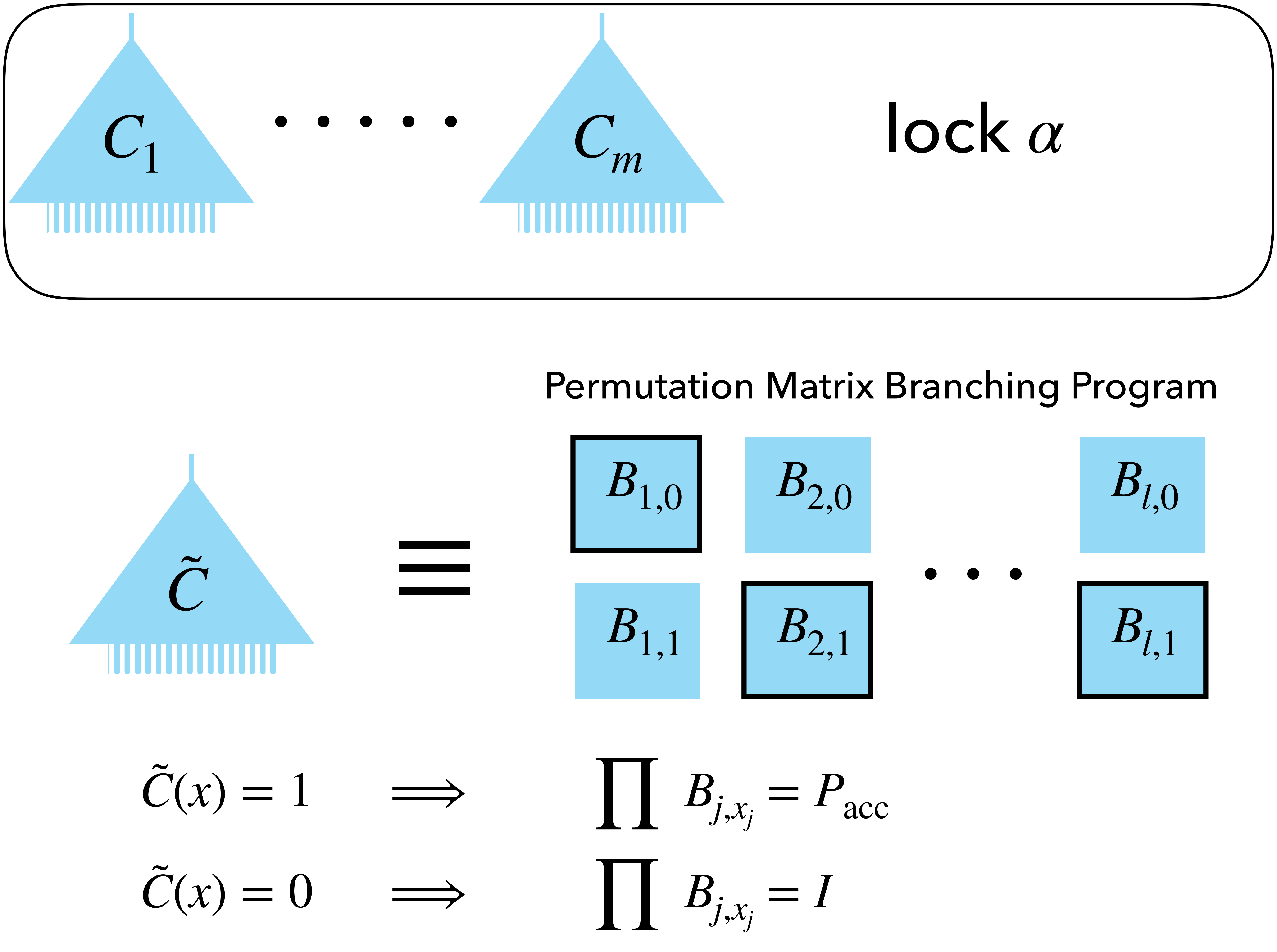
* FHE with NC^1 decryption


Obfuscate:
m bits
C
lock $\alpha \in \{0,1\}^m$
C_1
C_m

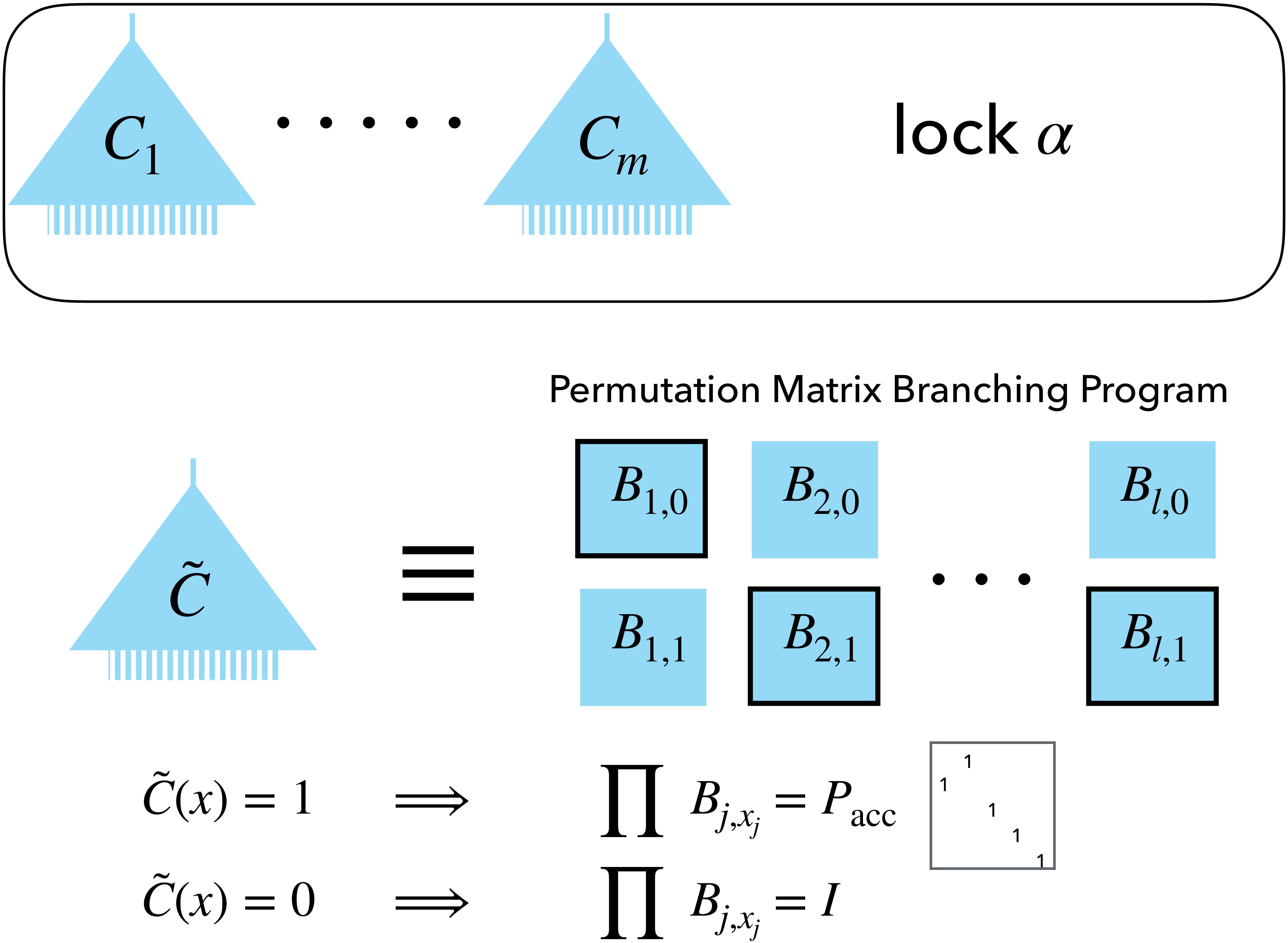

C_1
C_m
lock α


C_1
C_m
lock α
$\tilde{C}$

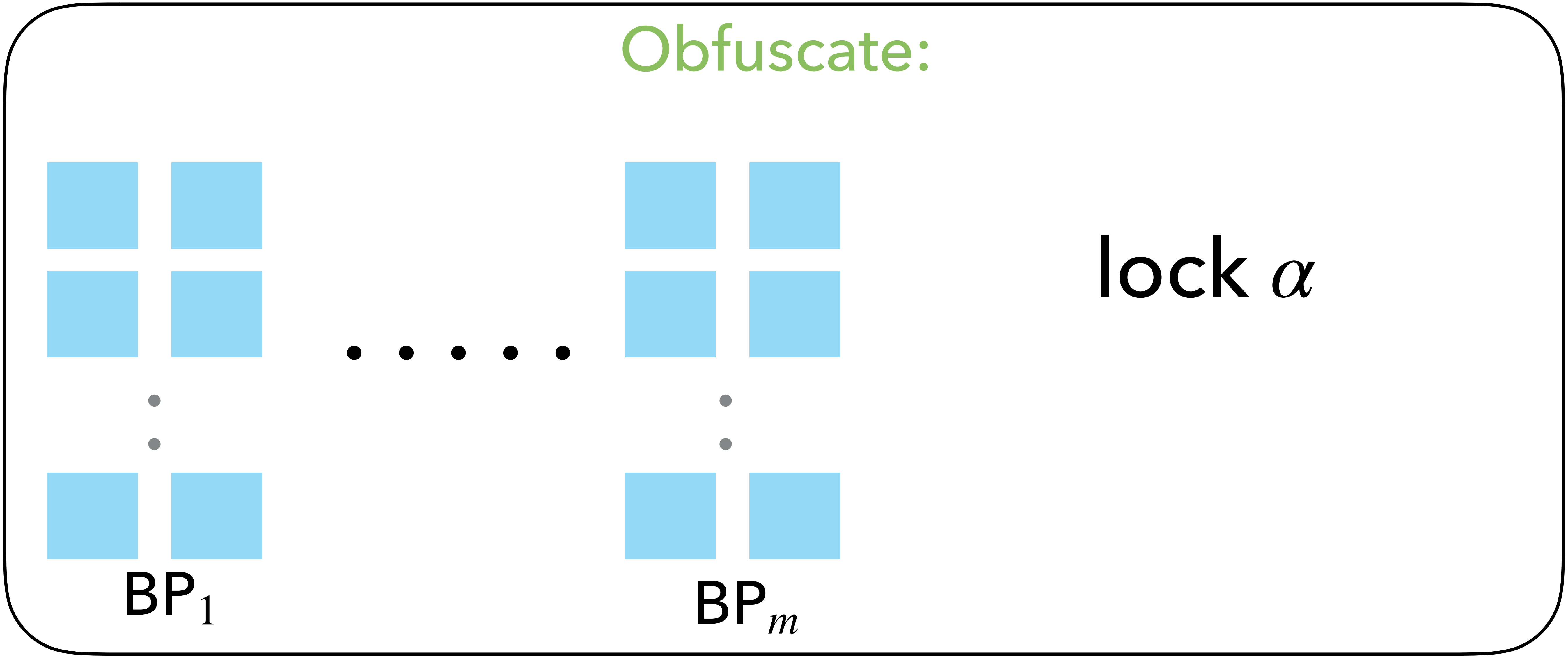

C_1
C_m
lock α
Permutation Matrix Branching Program
$\tilde{C}$
$\equiv$
$B_{1,0}$
$B_{2,0}$
$B_{l,0}$
$B_{1,1}$
$B_{2,1}$
$B_{l,1}$



C_1
C_m
lock α
Permutation Matrix Branching Program
$\tilde{C}$
$\equiv$
$B_{1,0}$
$B_{2,0}$
$B_{l,0}$
$B_{1,1}$
$B_{2,1}$
$B_{l,1}$
$\tilde{C}(x) = 1 \implies \prod B_{j,x_j} = P_{\text{acc}}$
$\tilde{C}(x) = 0 \implies \prod B_{j,x_j} = I$



C_1
C_m
lock α
Permutation Matrix Branching Program
$\tilde{C}$
$\equiv$
$B_{1,0}$
$B_{2,0}$
$B_{l,0}$
$B_{1,1}$
$B_{2,1}$
$B_{l,1}$
$\tilde{C}(x) = 1 \implies \prod B_{j,x_j} = P_{acc}$
1
1
1
1
1
$\tilde{C}(x) = 0 \implies \prod B_{j,x_j} = I$



Evaluation on x : Output 1 iff $\mathrm{BP}_i(x) = \alpha_i$ for all i .

Security : Obfuscation hides all BP_i

3-Step Recipe for Obfuscating

$$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$$

3-Step Recipe for Obfuscating $(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0$.

3-Step Recipe for Obfuscating
$(\text{BP}_1, \ldots, \text{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0$.

For each i, 'encode' $\text{BP}_i : M_{i,0} : M_{i,1}$.

Encoding$_i$ hides BP_i

Evaluation of encoding$_i$ on x outputs $\approx M_{i,\text{BP}_i(x)}$

3-Step Recipe for Obfuscating

$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0$.

For each i, 'encode' $\mathsf{BP}_i : M_{i,0} : M_{i,1}$.

Encoding$_i$ hides BP_i

Evaluation of encoding$_i$ on x outputs $\approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

3-Step Recipe for Obfuscating

$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0$.

For each i, 'encode' $\mathsf{BP}_i : M_{i,0} : M_{i,1}$.

Encoding$_i$ hides BP$_i$

Evaluation of encoding$_i$ on x outputs $\approx R_x \cdot M_{i,\mathsf{BP}_i(x)}$

Output all encodings.

3-Step Recipe for Obfuscating

$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

$$\sum_i M_{i,\alpha_i} = 0.$$

Eval of encoding$_i$ on $x \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

3-Step Recipe for Obfuscating
$(\text{BP}_1, \ldots, \text{BP}_m, \alpha)$

$$\sum_i M_{i,\alpha_i} = 0.$$

Eval of encoding$_i$ on $x \approx R_x \cdot M_{i,\text{BP}_i(x)}$

Evaluation on x :

Encoding$_i$ $\longrightarrow \approx R_x \cdot M_{i,\text{BP}_i(x)}$

Output 1 if the sum is small.

3-Step Recipe for Obfuscating

(BP$_1, \ldots,$ BP$_m, \alpha)$

$$\sum_i M_{i,\alpha_i} = 0.$$

Eval of encoding$_i$ on $x \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

Evaluation on x :

if BP$_i(x) = \alpha_i$ for all i

Encoding$_i$ $\longrightarrow \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

Output 1 if the sum is small.

3-Step Recipe for Obfuscating
$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

$$\sum_i M_{i,\alpha_i} = 0.$$

Eval of encoding$_i$ on $x \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

Evaluation on x :

if $\mathsf{BP}_i(x) = \alpha_i$ for all i

Encoding$_i \longrightarrow \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$ $\approx R_x \cdot M_{i,\alpha_i}$

Output 1 if the sum is small.

3-Step Recipe for Obfuscating
$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

$$\sum_i M_{i,\alpha_i} = 0.$$

Eval of encoding$_i$ on $x \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

Evaluation on x :

if $\mathsf{BP}_i(x) = \alpha_i$ for all i

Encoding$_i \longrightarrow \approx R_x \cdot M_{i,\mathrm{BP}_i(x)}$

Output 1 if the sum is small.

$$\approx R_x \cdot M_{i,\alpha_i}$$

$$\approx \sum R_x \cdot M_{i,\alpha_i}$$

$$= R_x \cdot \left(\sum M_{i,\alpha_i} \right)$$

$$= 0$$

'Encode' BP : M_0 : M_1 .

WANT:

Encoding hides BP

Eval of encoding on $x \approx R_x \cdot M_{\mathrm{BP}(x)}$

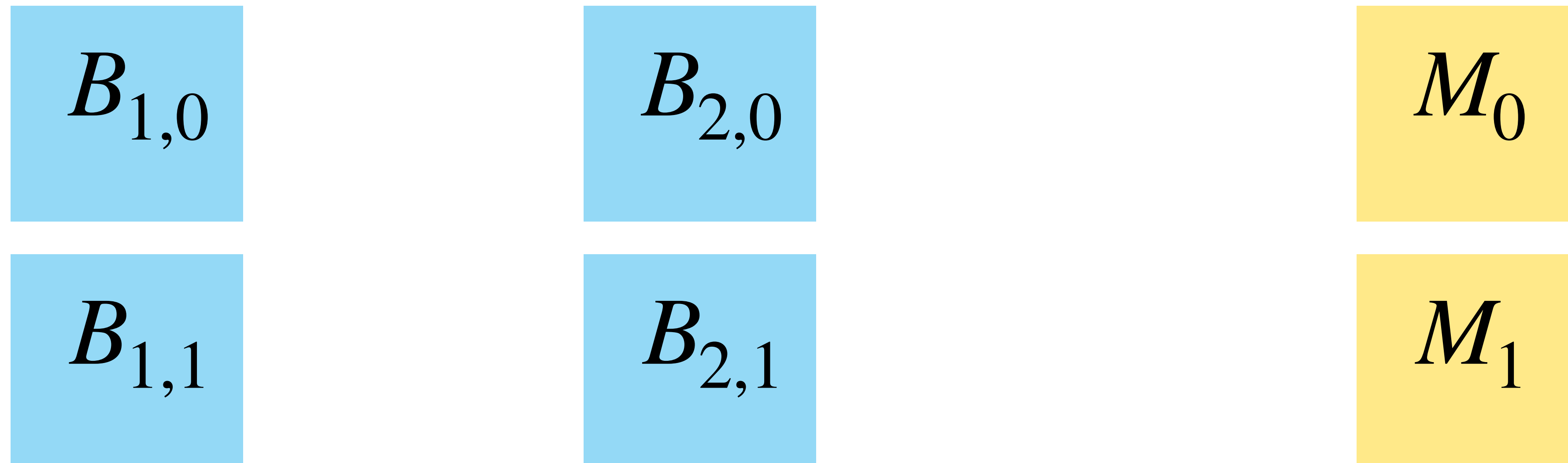
$B_{1,0}$	$B_{2,0}$	M_0
$B_{1,1}$	$B_{2,1}$	M_1

'Encode' BP : $M_0 : M_1$.

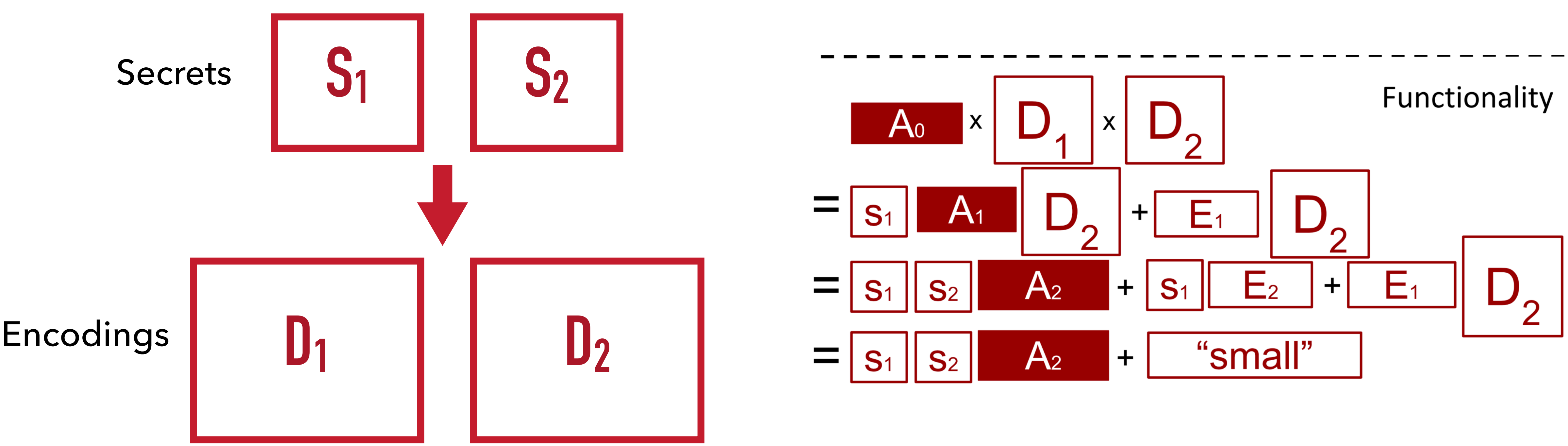
WANT:

Encoding hides BP

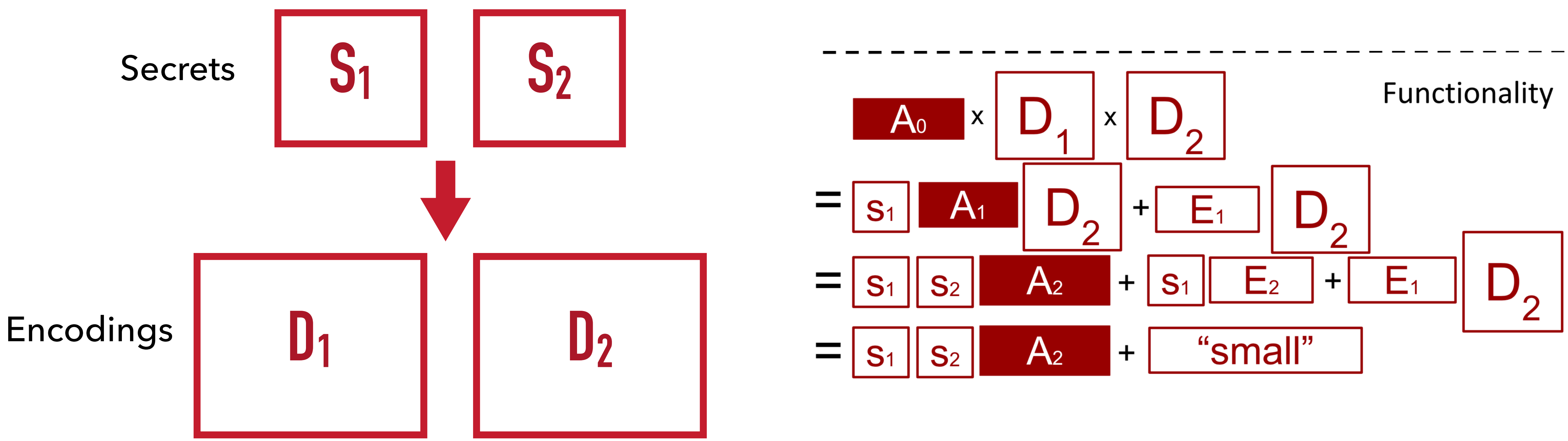
Eval of encoding on $x \approx R_x \cdot M_{\mathrm{BP}(x)}$



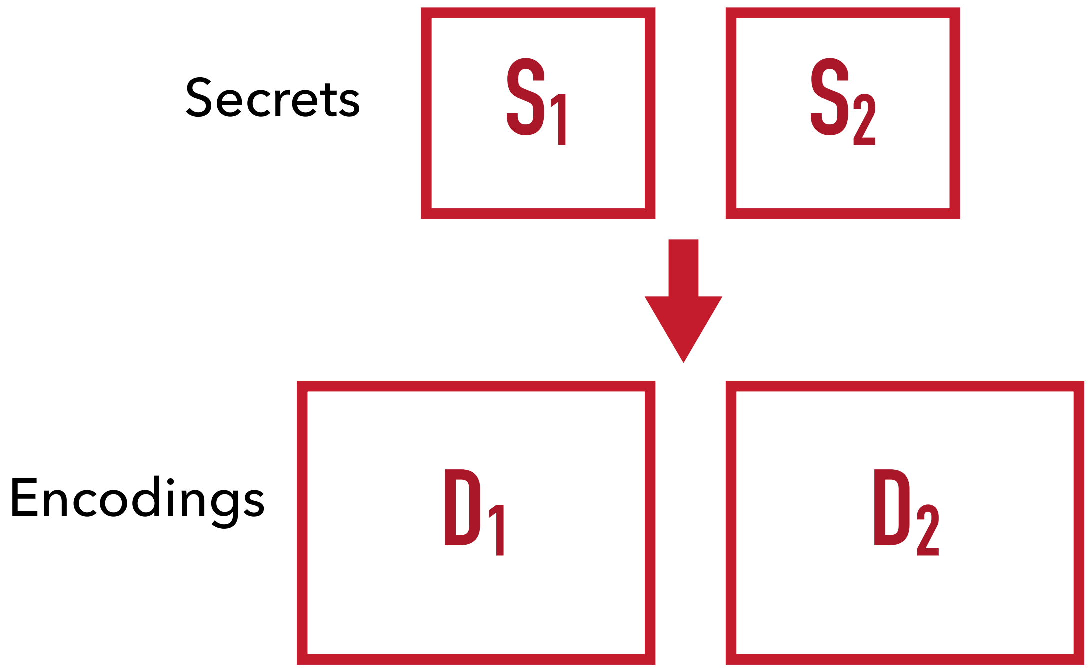
GGH15 ENCODING OF 'MATRIX' SECRETS:

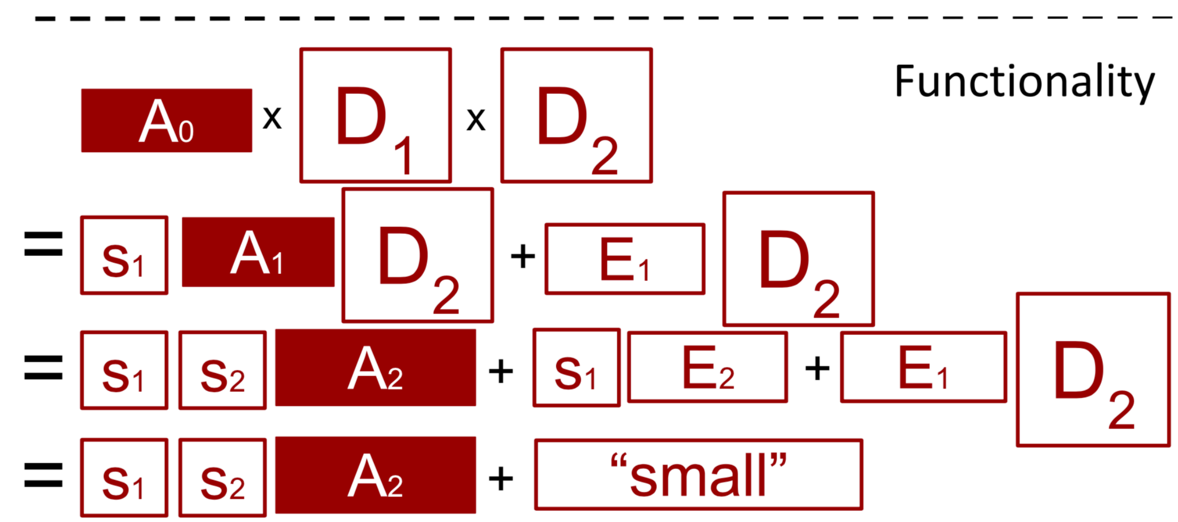


GGH15 Encoding of 'matrix' secrets:



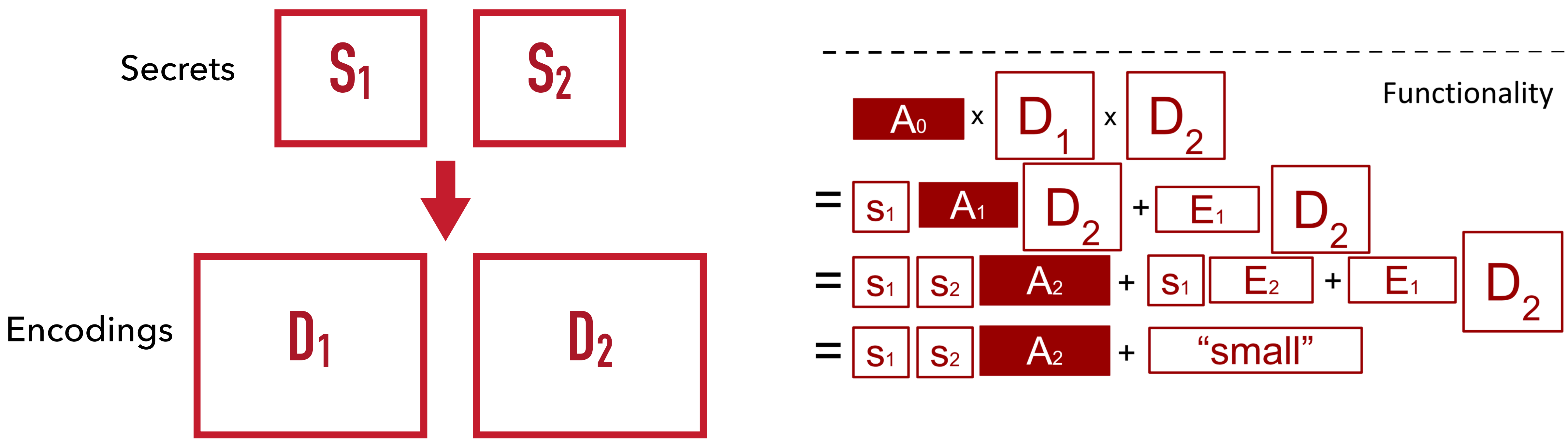
GGH15 Encoding of 'matrix' secrets:



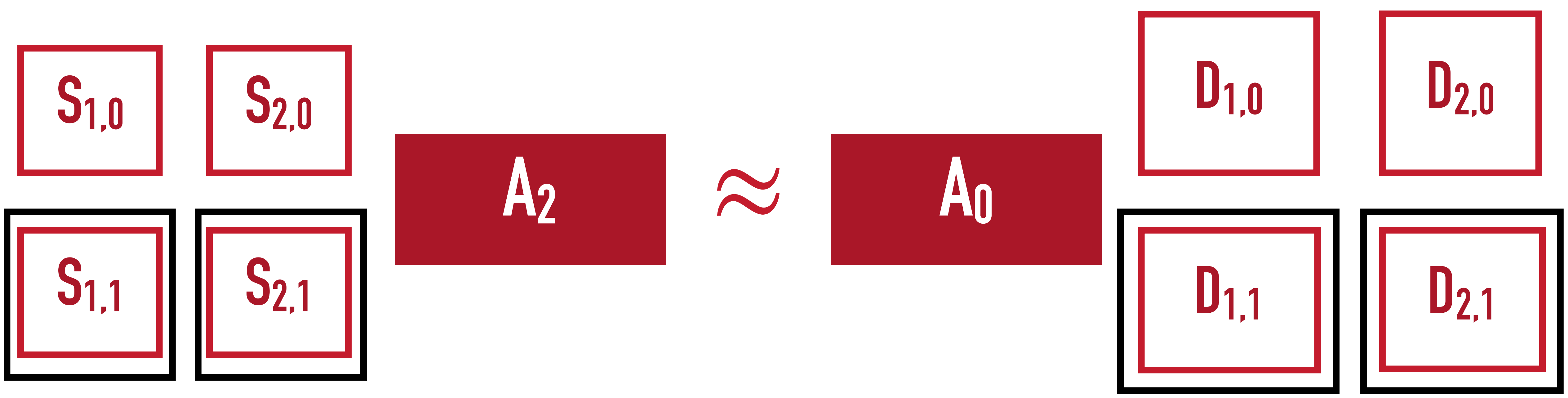


Subset Product of 'matrix' secrets -> Subset Product of Encodings

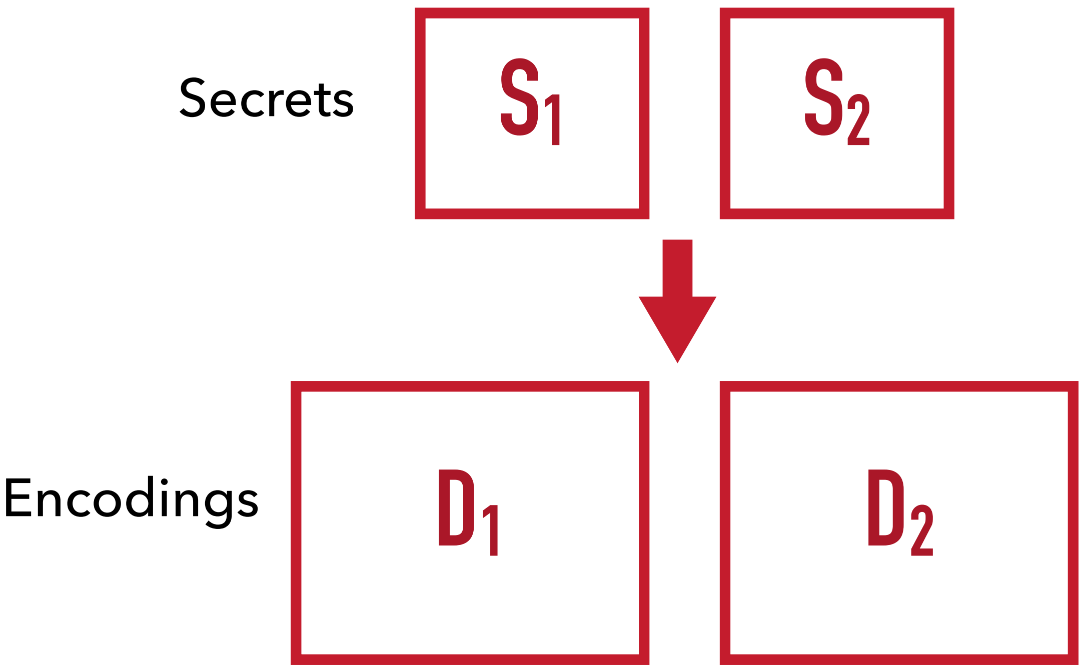
GGH15 Encoding of 'matrix' secrets:

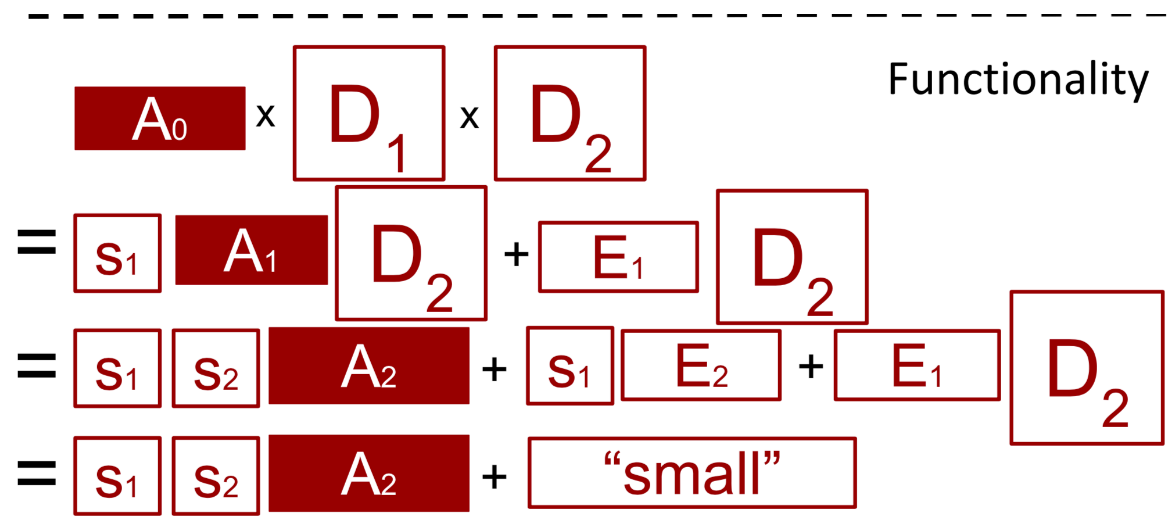


Subset Product of 'matrix' secrets -> Subset Product of Encodings

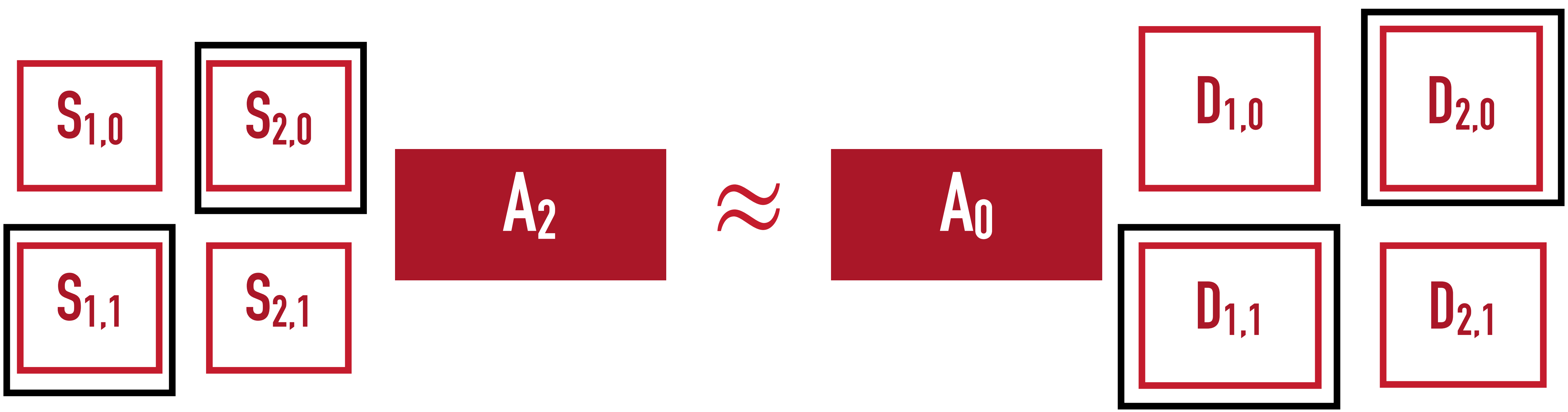


GGH15 Encoding of 'matrix' secrets:

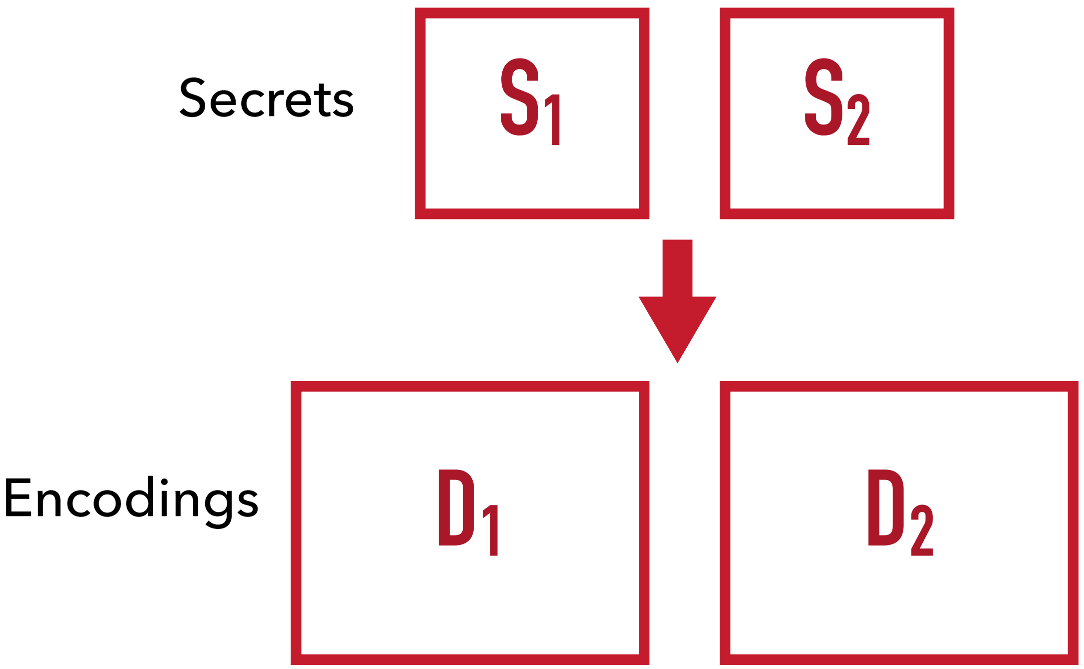


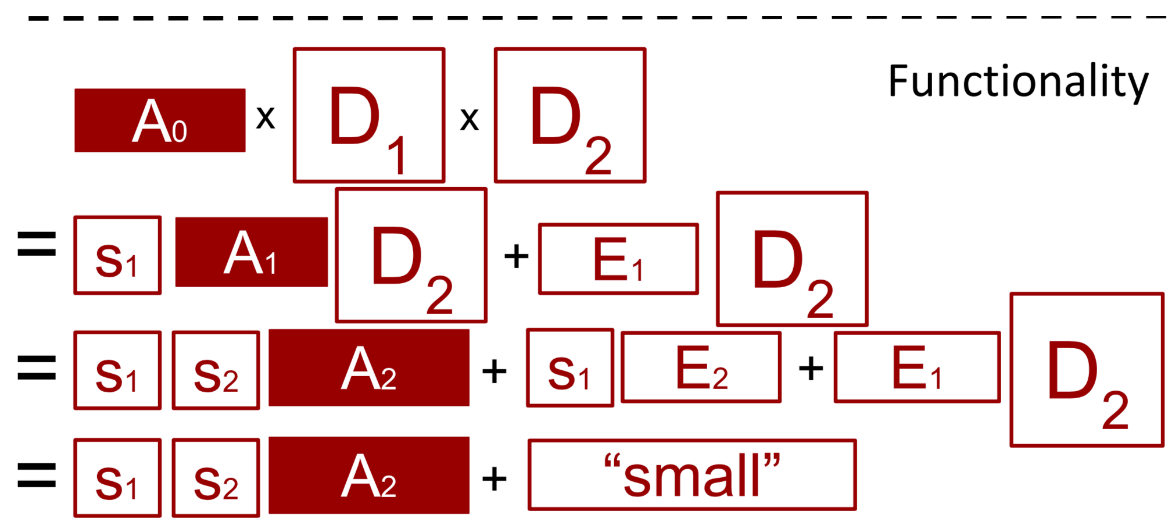


Subset Product of 'matrix' secrets -> Subset Product of Encodings

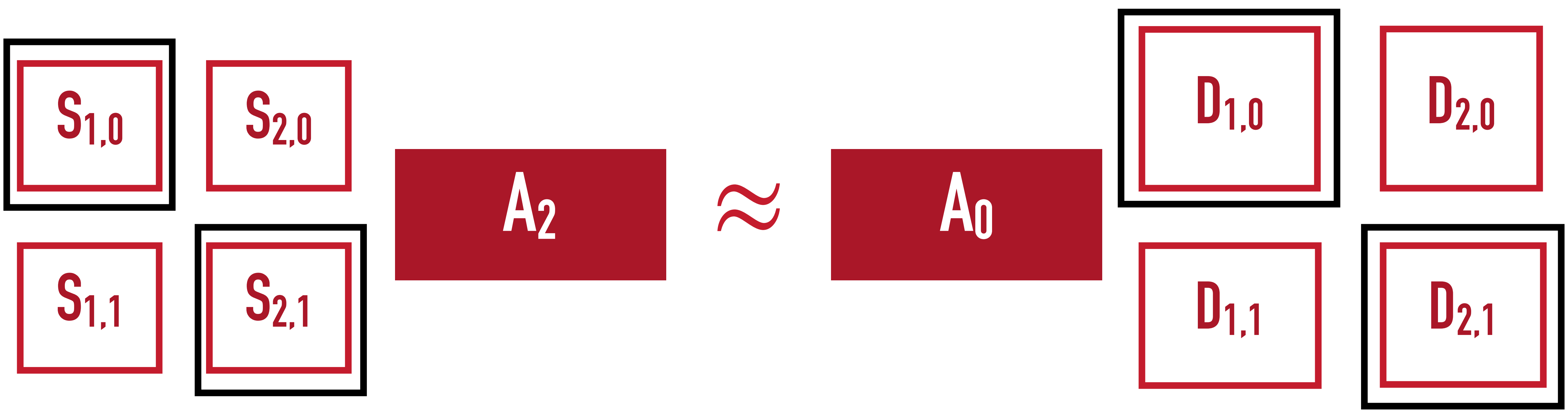


GGH15 Encoding of 'matrix' secrets:

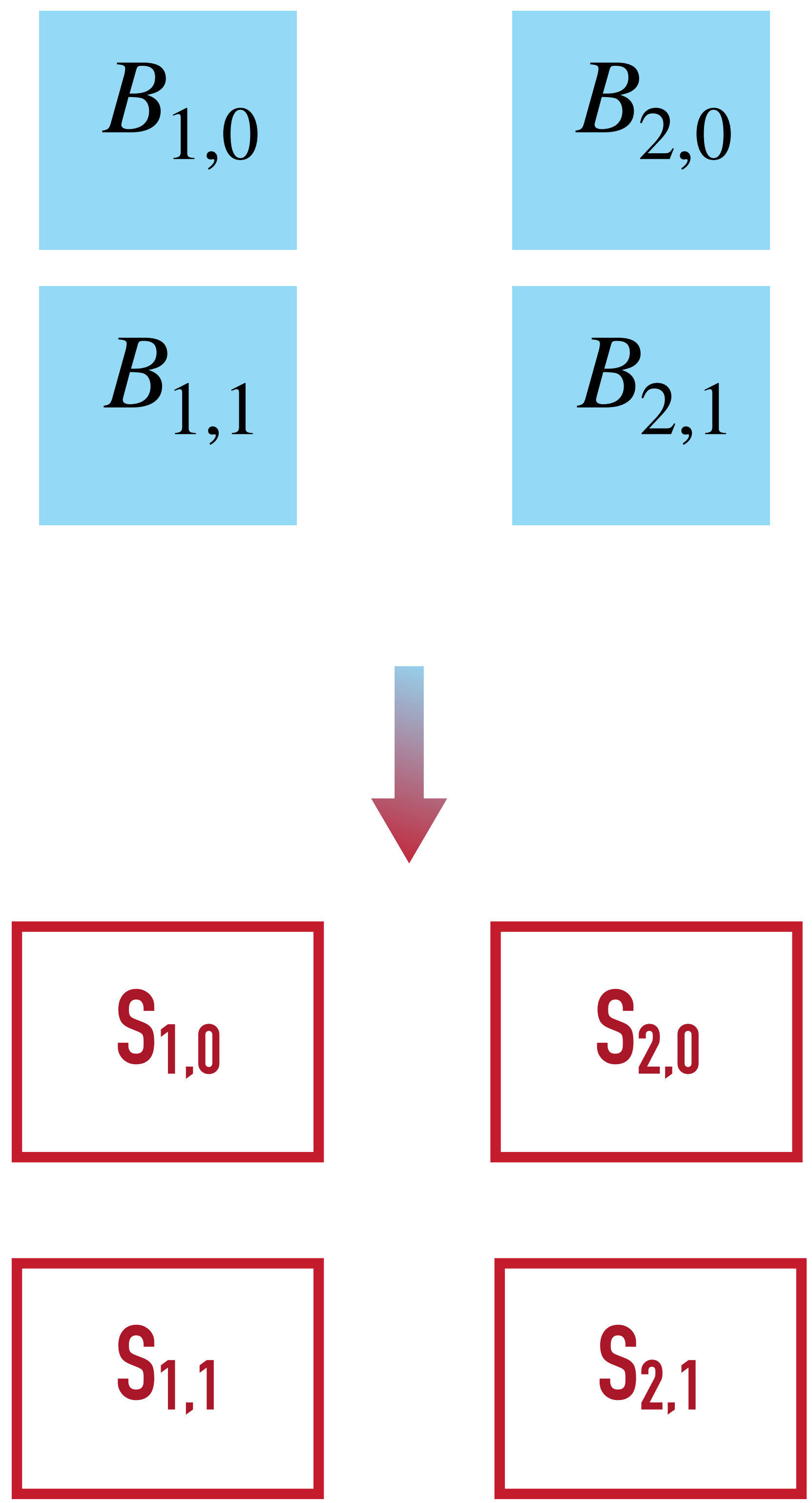




Subset Product of 'matrix' secrets -> Subset Product of Encodings



From Branching Programs to GGH15 Secrets:



From Branching Programs to GGH15 Secrets:

$$\left(\prod B_{i,x_i}\right) = \begin{bmatrix} & 1 & & & \\ 1 & & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \quad \textbf{or} \quad \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

From Branching Programs to GGH15 Secrets:

$$\left(\prod B_{i,x_i}\right)\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{bmatrix} & 1 & & & \\ 1 & & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} \text{ or } \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix}$$

FROM BRANCHING PROGRAMS TO GGH15 SECRETS:

$$\left(\prod B_{i,x_i}\right)\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{bmatrix} & 1 & & & \\ 1 & & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} \text{ or } \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix}$$

$$\text{first entry of } \left(\prod B_{i,x_i}\right)\begin{bmatrix} z_0 \\ z_1 \\ \vdots \end{bmatrix} = \begin{cases} z_1 \text{ if BP}(x) = 1 \\ z_0 \text{ if BP}(x) = 0 \end{cases}$$

From Branching Programs to GGH15 Secrets:

$$\left(\prod B_{i,x_i}\right)\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{bmatrix} & 1 & & & \\ 1 & & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} \text{ or } \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix}$$

$$\text{first entry of } \left(\prod B_{i,x_i}\right)\begin{bmatrix} z_0 \\ z_1 \\ \vdots \end{bmatrix} = \begin{cases} z_1 \text{ if } \mathrm{BP}(x) = 1 \\ z_0 \text{ if } \mathrm{BP}(x) = 0 \end{cases}$$

$$\text{first block of } \left(\prod B_{i,x_i} \otimes I\right)\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{cases} M_1 \text{ if } \mathrm{BP}(x) = 1 \\ M_0 \text{ if } \mathrm{BP}(x) = 0 \end{cases}$$

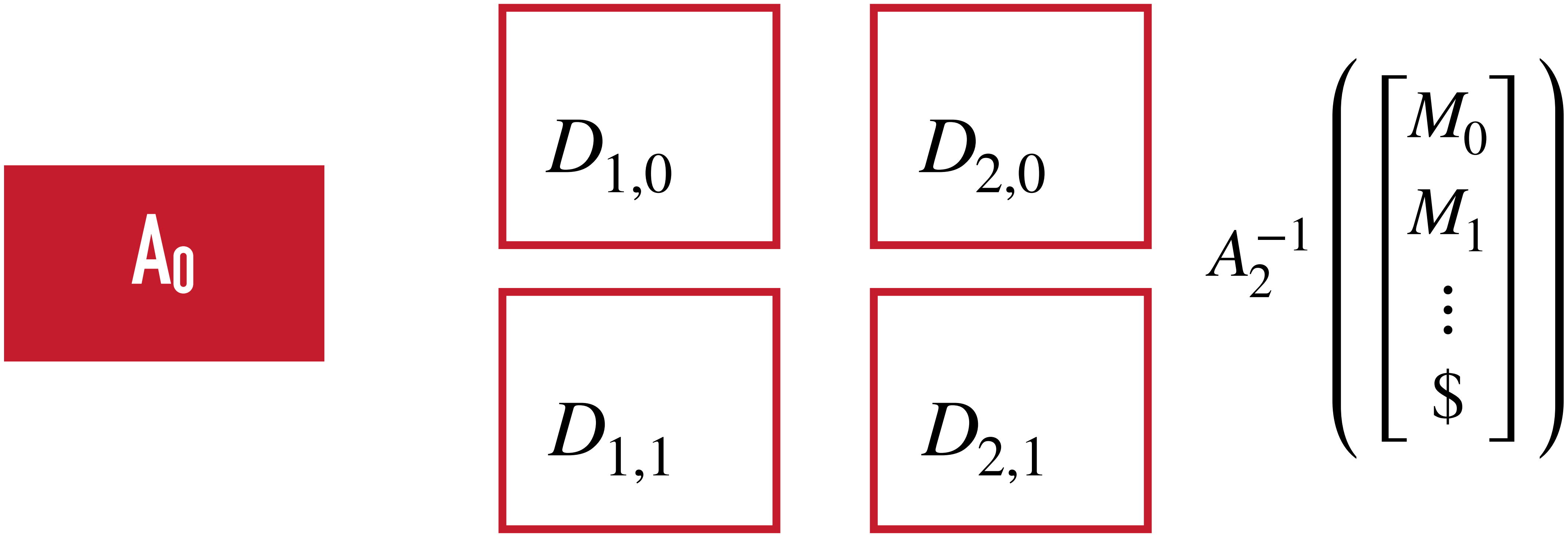
From Branching Programs to GGH15 Secrets:

$$\left(\prod B_{i,x_i}\right) \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{bmatrix} & 1 & & & \\ 1 & & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix} \text{ or } \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ \$ \end{bmatrix}$$

$$\text{first entry of } \left(\prod B_{i,x_i}\right) \begin{bmatrix} z_0 \\ z_1 \\ \vdots \end{bmatrix} = \begin{cases} z_1 \text{ if } \mathsf{BP}(x) = 1 \\ z_0 \text{ if } \mathsf{BP}(x) = 0 \end{cases}$$

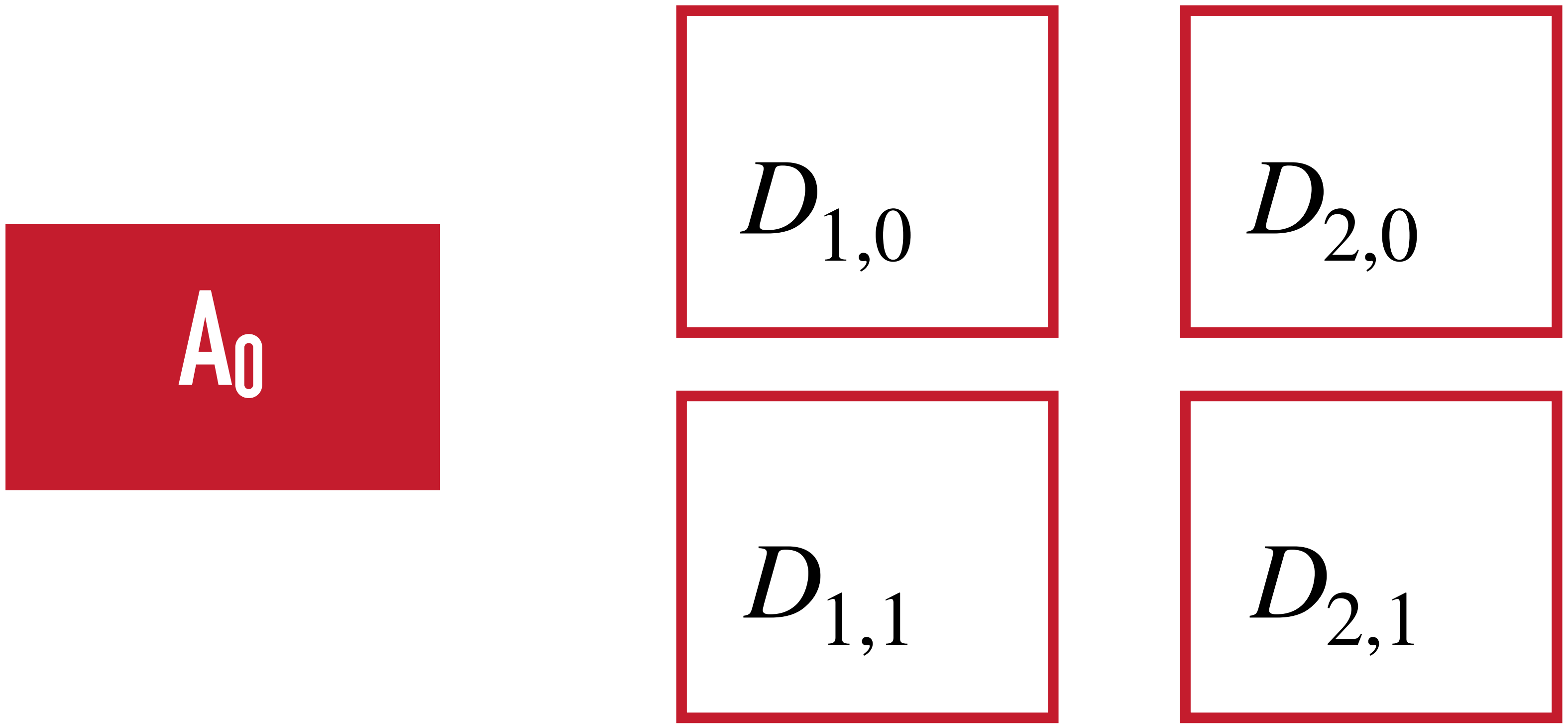
$$\text{first block of } \left(\prod \overbrace{B_{i,x_i} \otimes I}^{S_{i,x_i}}\right) \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{cases} M_1 \text{ if } \mathsf{BP}(x) = 1 \\ M_0 \text{ if } \mathsf{BP}(x) = 0 \end{cases}$$

From GGH15 Secrets to Encodings:



$$D_{i,b} = A_{i-1}^{-1}\left(\left(B_{i,b} \otimes I\right) A_i + \text{noise}\right)$$

From GGH15 Secrets to Encodings:



$$A_2^{-1}\left(\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}\right)$$

$$D_{i,b} = A_{i-1}^{-1}\left(\left(B_{i,b} \otimes I\right) A_i + \text{noise}\right)$$

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

From GGH15 Secrets to Encodings:

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

From GGH15 Secrets to Encodings:

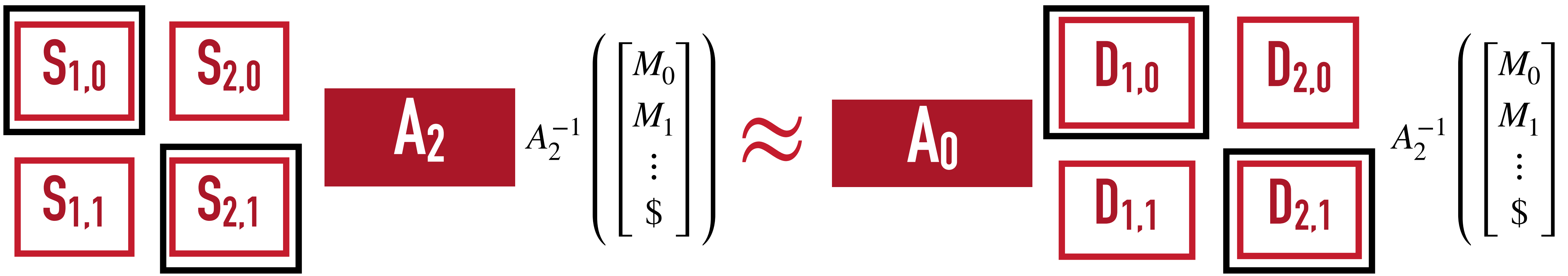
Eval of encoding on $x = M_{\mathrm{BP}(x)}$

Subset Product of 'matrix' secrets -> Subset Product of Encodings

From GGH15 Secrets to Encodings:

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

Subset Product of 'matrix' secrets -> Subset Product of Encodings



From GGH15 Secrets to Encodings:

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

Subset Product of 'matrix' secrets -> Subset Product of Encodings

$S_{1,0}$ $S_{2,0}$

$S_{1,1}$ $S_{2,1}$

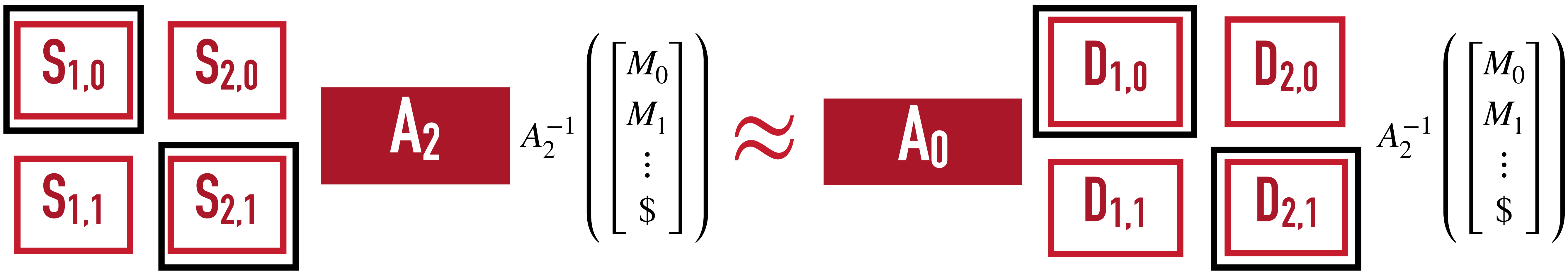
$$A_2 \; A_2^{-1}\left(\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}\right) \approx A_0 \; \begin{matrix} D_{1,0} & D_{2,0} \\ D_{1,1} & D_{2,1} \end{matrix} \; A_2^{-1}\left(\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}\right)$$

$$\left(\prod \overbrace{B_{i,x_i} \otimes I}^{S_{i,x_i}}\right) \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}$$

From GGH15 Secrets to Encodings:

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

Subset Product of 'matrix' secrets -> Subset Product of Encodings

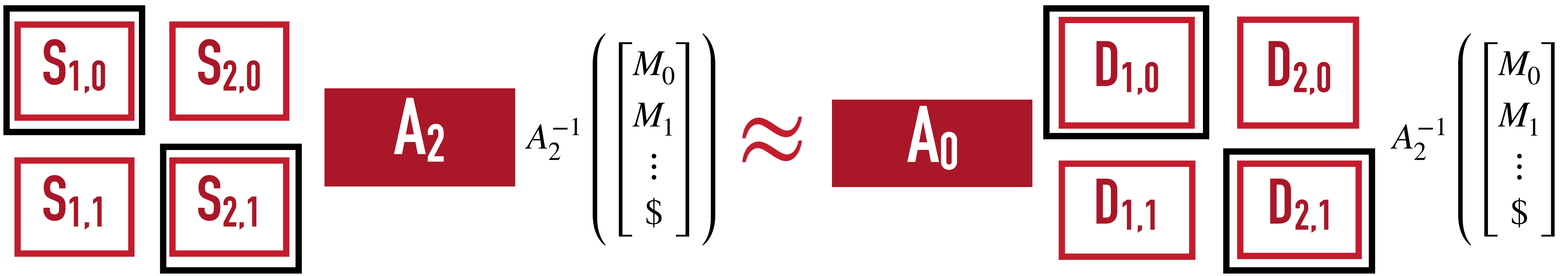


$$\text{first block of } \left(\prod \overbrace{B_{i,x_i} \otimes I}^{S_{i,x_i}}\right) \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{cases} M_1 \text{ if } \mathrm{BP}(x) = 1 \\ M_0 \text{ if } \mathrm{BP}(x) = 0 \end{cases}$$

FROM GGH15 SECRETS TO ENCODINGS:

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

SUBSET PRODUCT OF 'MATRIX' SECRETS -> SUBSET PRODUCT OF ENCODINGS



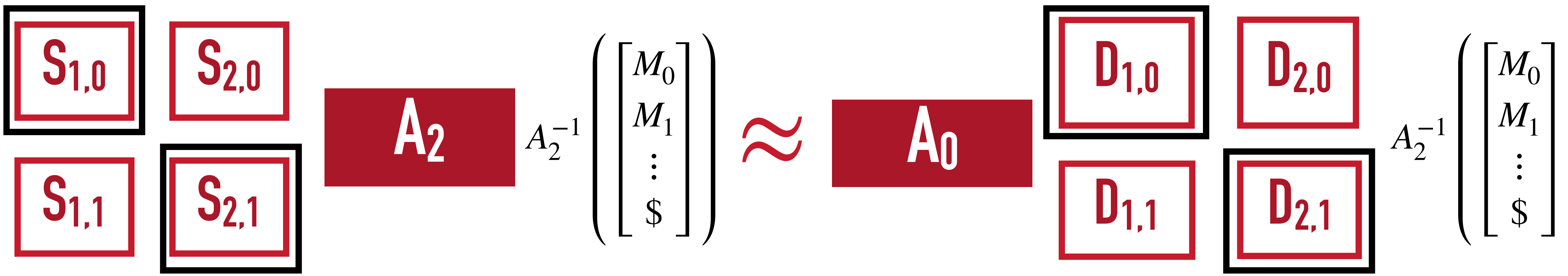
$$\text{first block of } \left(\prod \overbrace{B_{i,x_i} \otimes I}^{S_{i,x_i}} \right) \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{cases} M_1 \text{ if } \mathrm{BP}(x) = 1 \\ M_0 \text{ if } \mathrm{BP}(x) = 0 \end{cases}$$

Encoding$_i$ hides BP$_i$?

From GGH15 Secrets to Encodings:

Eval of encoding on $x = M_{\mathrm{BP}(x)}$

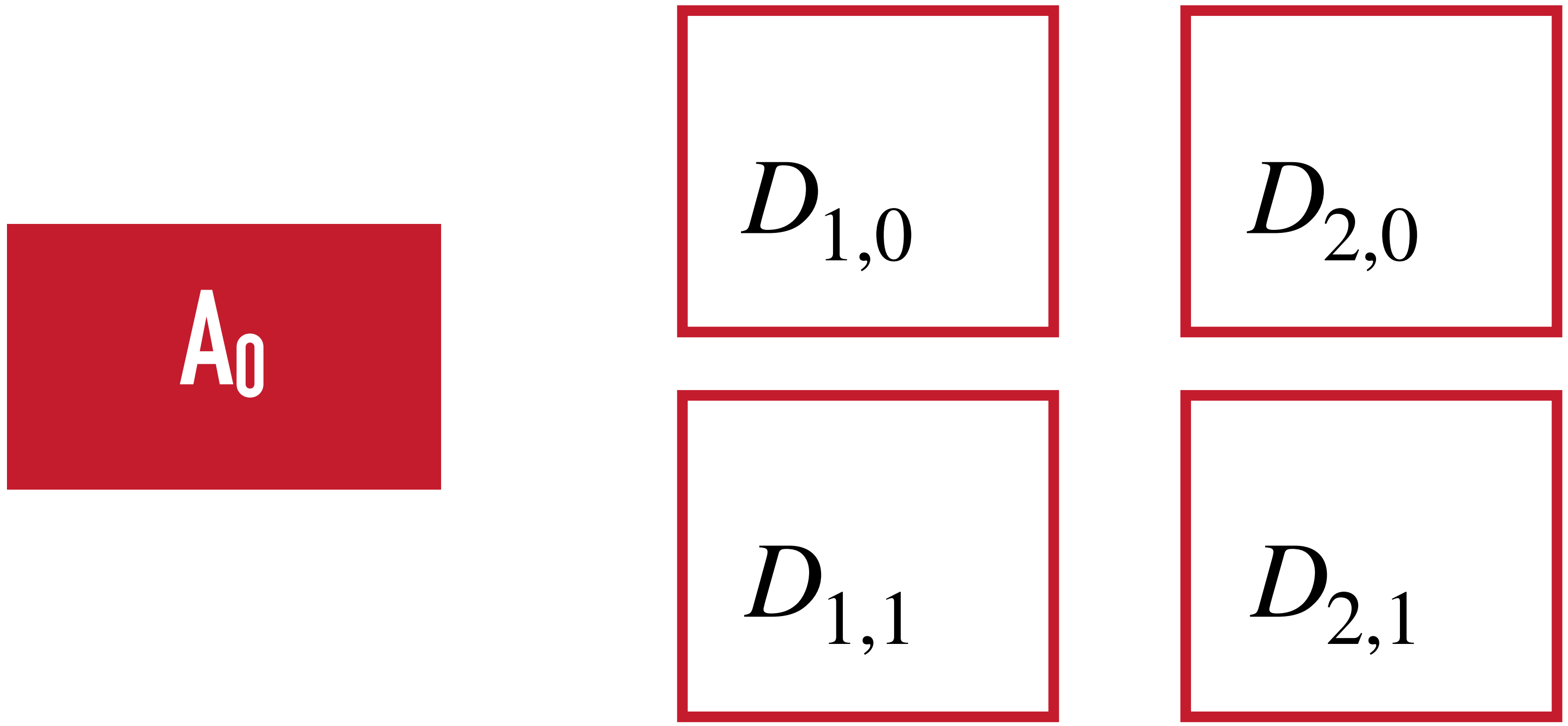
Subset Product of 'matrix' secrets -> Subset Product of Encodings



$$\text{first block of } \left(\prod \overbrace{B_{i,x_i} \otimes I}^{S_{i,x_i}} \right) \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix} = \begin{cases} M_1 \text{ if } \mathrm{BP}(x) = 1 \\ M_0 \text{ if } \mathrm{BP}(x) = 0 \end{cases}$$

Encoding$_i$ hides BP$_i$? ✗

SECURE CONSTRUCTION:



$$A_2^{-1}\left(\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}\right)$$

$$D_{i,b} = A_{i-1}^{-1}\left(\left(B_{i,b} \otimes R_{i,b}\right) A_i \; + \; \text{noise}\right)$$

$R_{i,b}$: random low norm

Secure Construction: Summary

Secure Construction: Summary

BP Matrices

$B_{1,0}$	$B_{2,0}$
$B_{1,1}$	$B_{2,1}$

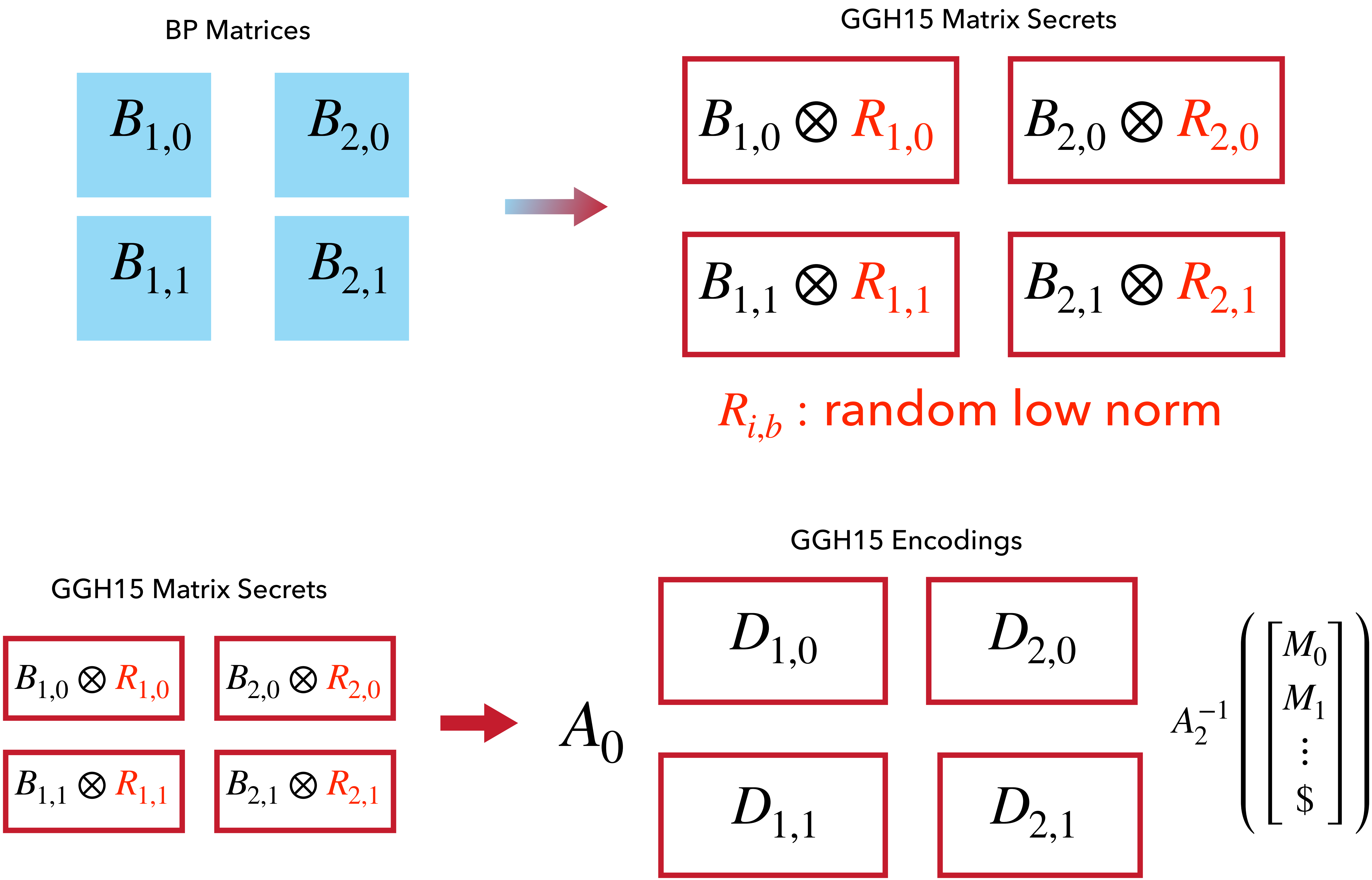
→

GGH15 Matrix Secrets

$B_{1,0} \otimes R_{1,0}$	$B_{2,0} \otimes R_{2,0}$
$B_{1,1} \otimes R_{1,1}$	$B_{2,1} \otimes R_{2,1}$

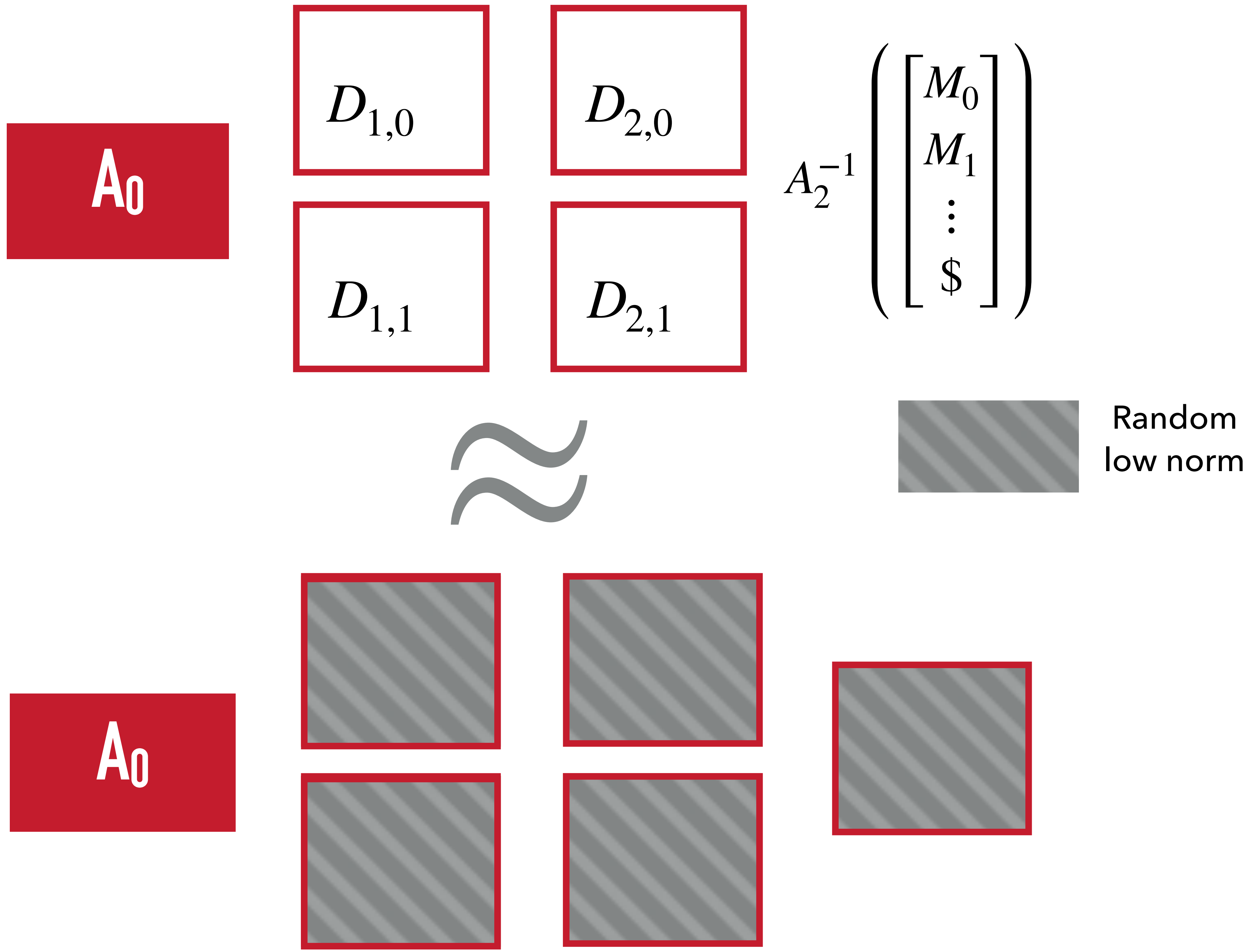
$R_{i,b}$: random low norm

SECURE CONSTRUCTION: SUMMARY

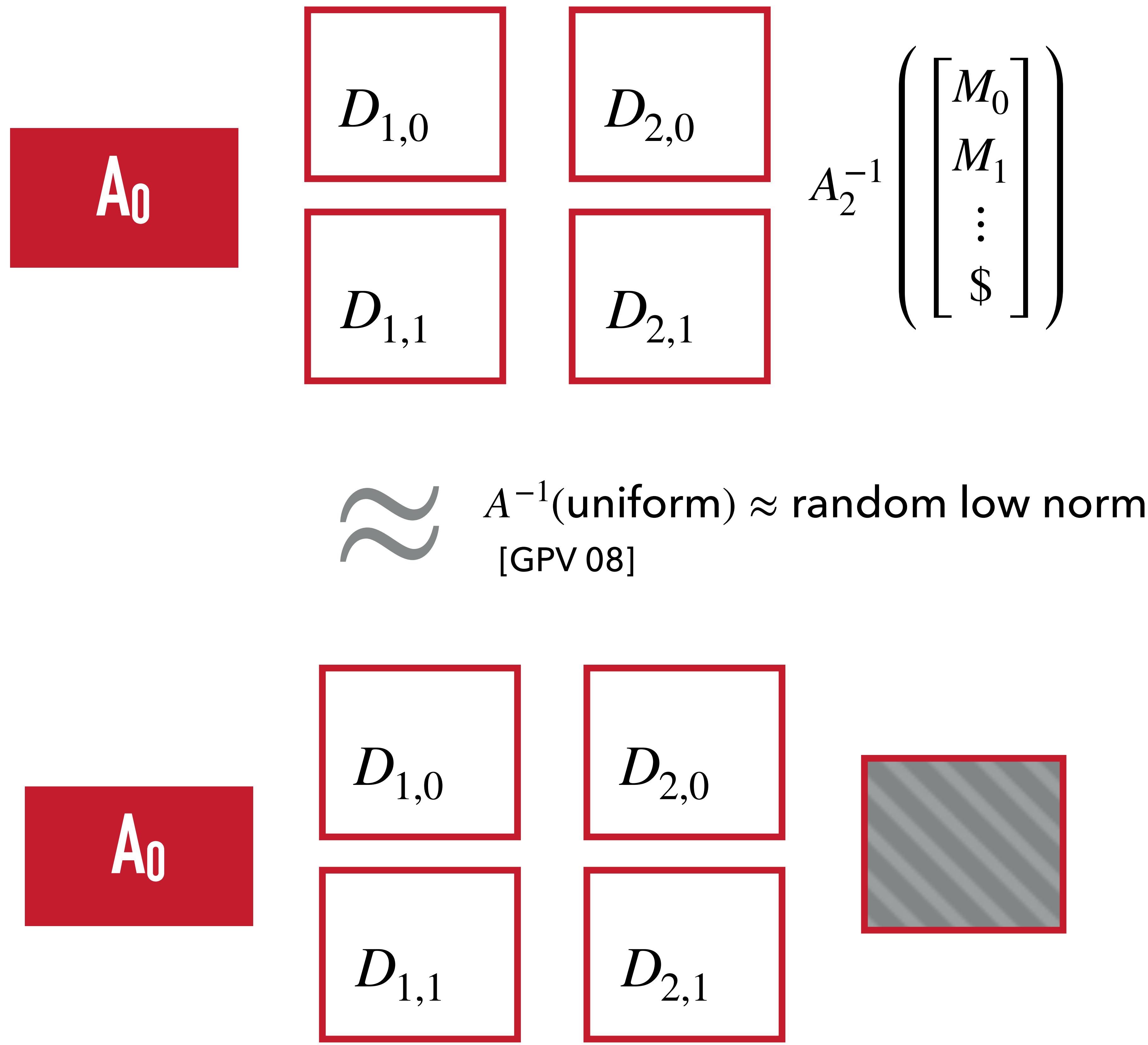


Security Lemma:

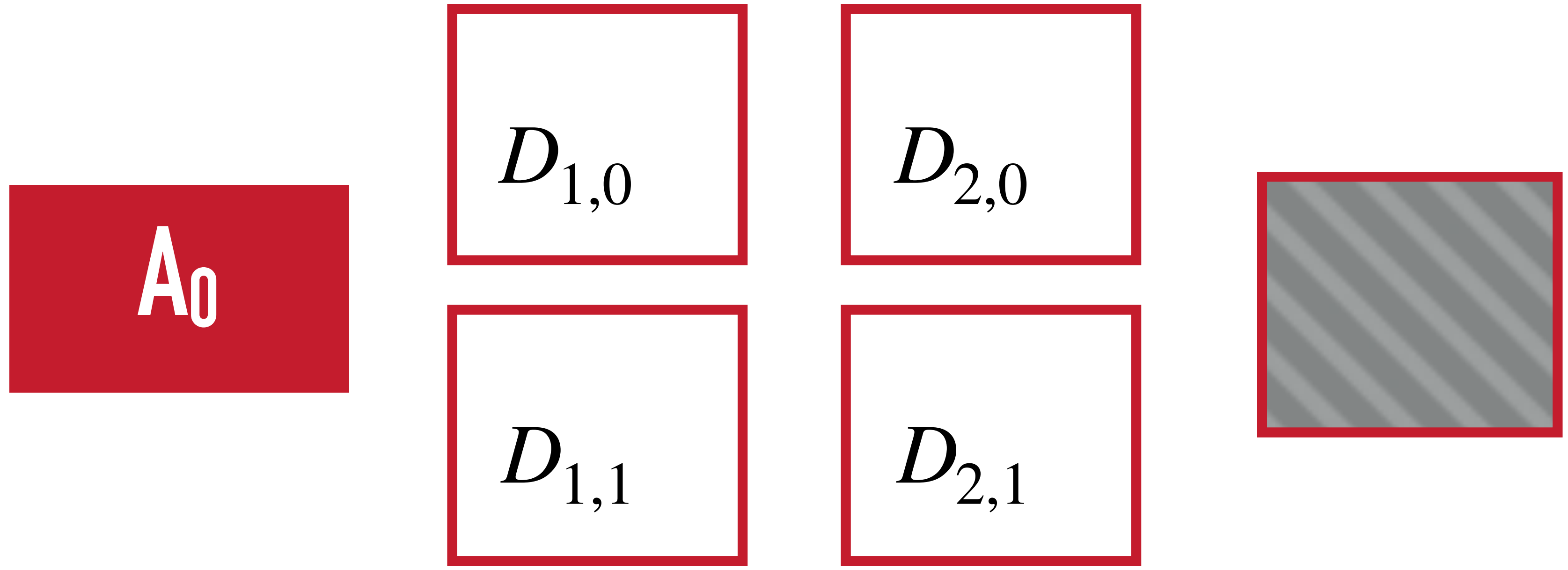
If M_0, M_1 are random, then



PROOF OF SECURITY LEMMA:



Proof of Security Lemma:



$$D_{2,b} = A_1^{-1}\left(\left(B_{2,b} \otimes R_{2,b}\right) A_2 \ + \ \text{noise}\right)$$

Proof of Security Lemma:

Proof of Security Lemma:

Permutation LWE

[Canetti, Chen 17]

P: Any Perm. Matrix

$$(A\ ,\ (P \otimes R) \cdot A\ +\ \text{noise})$$
$$\approx$$
$$(A\ ,\ U)$$

$A\ , U$: uniform

R : small entries

PROOF OF SECURITY LEMMA:

Permutation LWE

[Canetti, Chen 17]

P: Any Perm. Matrix

$$(A\ ,\ (P \otimes R) \cdot A\ +\ \text{noise})$$
$$\approx$$
$$(A\ , U)$$

A , U: uniform

R : small entries

$$\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes R \right)$$

A

Proof of Security Lemma:

Permutation LWE

[Canetti, Chen 17]

P: Any Perm. Matrix

$$(A\ ,\ (P \otimes R) \cdot A\ +\ \text{noise})$$
$$\approx$$
$$(A\ ,\ U)$$

A , U: uniform

R : small entries

$$\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes R \right) \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} = \begin{bmatrix} R \cdot A_3 & R \cdot A_4 \\ R \cdot A_1 & R \cdot A_2 \end{bmatrix}$$

Proof of Security Lemma:

Permutation LWE

[Canetti, Chen 17]

P: Any Perm. Matrix

$$(A\ ,\ (P \otimes R) \cdot A\ +\ \text{noise})$$
$$\approx$$
$$(A\ ,\ U)$$

$A\ ,\ U$: uniform

R : small entries

$$\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes R \right) \begin{array}{|c|c|} \hline A_1 & A_2 \\ \hline A_3 & A_4 \\ \hline \end{array} = \begin{bmatrix} R \cdot A_3 & R \cdot A_4 \\ R \cdot A_1 & R \cdot A_2 \end{bmatrix}$$

Proof of Security Lemma:

Permutation LWE

[Canetti, Chen 17]

P: Any Perm. Matrix

$$(A\ ,\ (P \otimes R) \cdot A\ +\ \text{noise})$$
$$\approx$$
$$(A\ ,\ U)$$

A , U: uniform

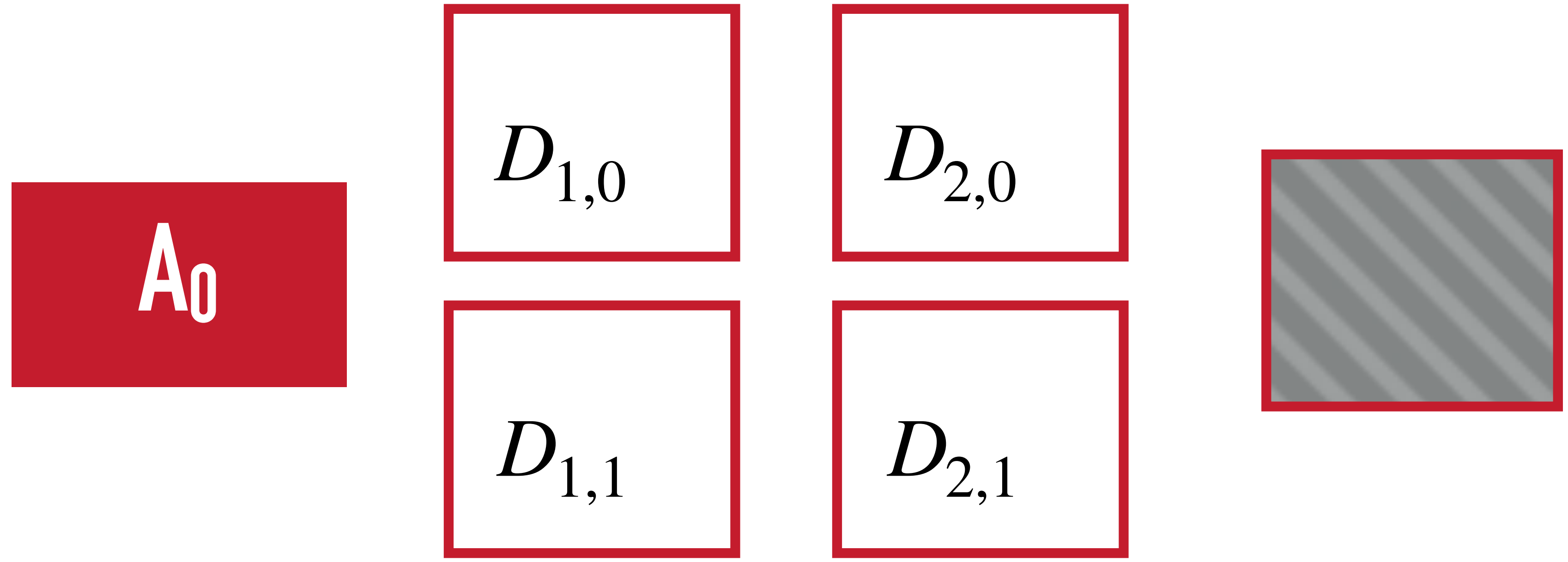
R : small entries

$$\left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \otimes R\right)$$

A_1	A_2
A_3	A_4

$$= \begin{bmatrix} R \cdot A_1 & R \cdot A_2 \\ R \cdot A_1 & R \cdot A_2 \end{bmatrix}$$

Security Lemma:

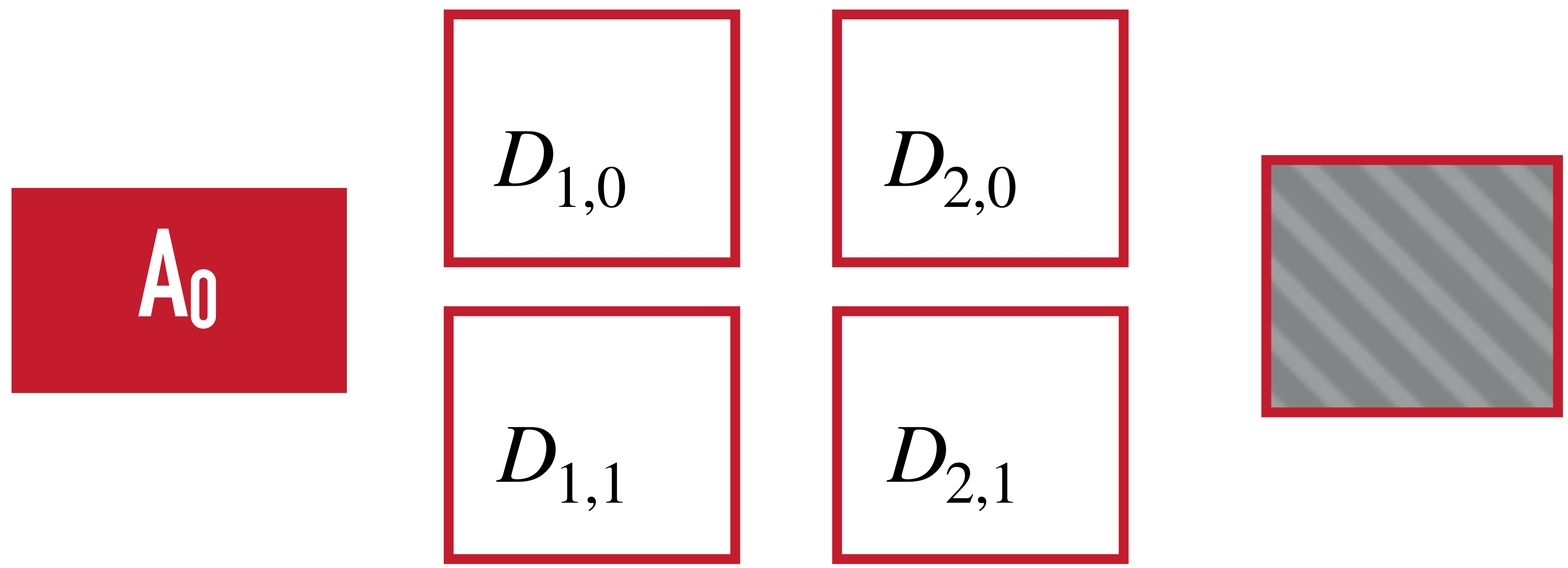


$$D_{2,b} = A_1^{-1}\left(\left(B_{2,b} \otimes R_{2,b}\right) A_2 \ + \ \text{noise} \right)$$

$\approx$ Perm. LWE

$$D_{2,b} = A_1^{-1}\left(\ \text{random} \right)$$

SECURITY LEMMA:



$$D_{2,b} = A_1^{-1}\left(\left(B_{2,b} \otimes R_{2,b}\right) A_2 \ + \ \text{noise}\right)$$

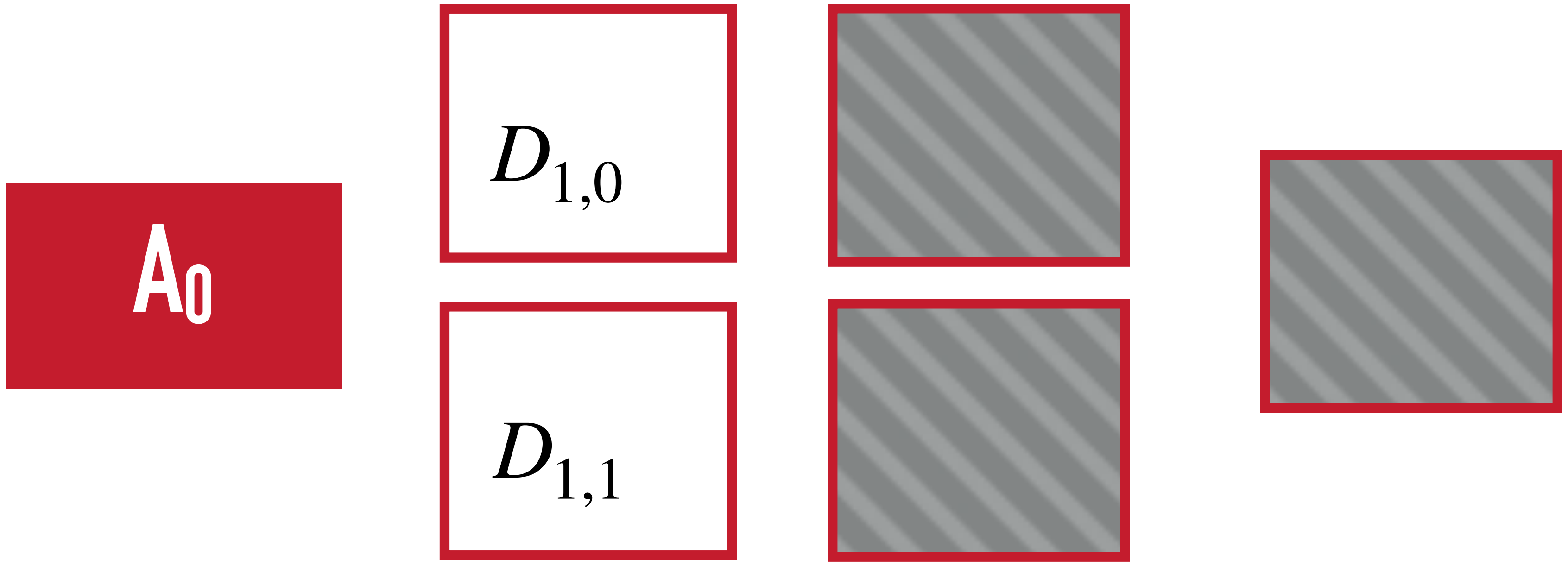
$\approx$ Perm. LWE

$$D_{2,b} = A_1^{-1}\left(\ \text{random}\right)$$

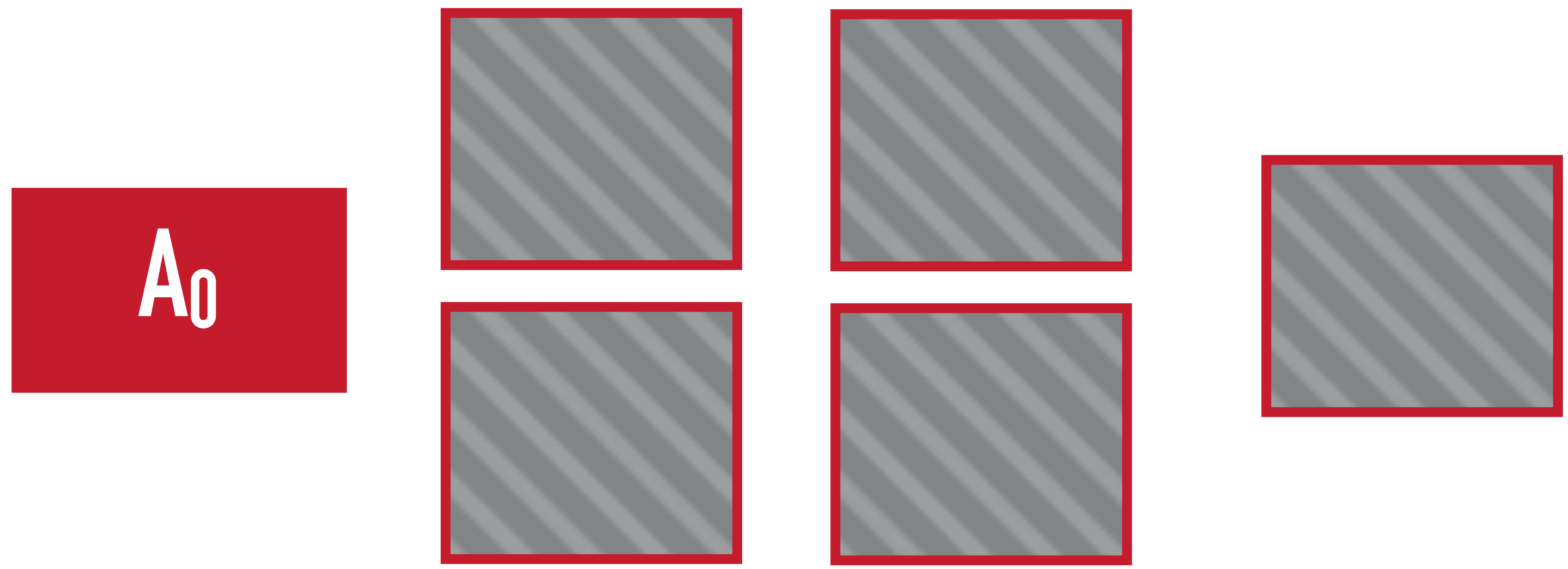
$\approx$ [GPV 08]

low norm random

Security Lemma:



Security Lemma:



3-Step Recipe for Obfuscating

$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0$.

For each i, 'encode' $\mathsf{BP}_i : M_{i,0} : M_{i,1}$.

Encoding$_i$ hides BP_i

Evaluation of encoding$_i$ on x outputs $\approx R_x \cdot M_{i,\mathsf{BP}_i(x)}$

Output all encodings.

3-Step Recipe for Obfuscating
$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0.$

For each i, 'encode' $\mathsf{BP}_i : M_{i,0} : M_{i,1}$.

needs random $M_{i,0}, M_{i,1}$

Encoding$_i$ hides BP_i

Evaluation of encoding$_i$ on x outputs $\approx R_x \cdot M_{i,\mathsf{BP}_i(x)}$

Output all encodings.

3-Step Recipe for Obfuscating

$(\mathsf{BP}_1, \ldots, \mathsf{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0$.

For each i, 'encode' $\mathsf{BP}_i : M_{i,0} : M_{i,1}$.

needs random $M_{i,0}, M_{i,1}$

Encoding$_i$ hides BP_i

Evaluation of encoding$_i$ on x outputs $\approx R_x \cdot M_{i,\mathsf{BP}_i(x)}$

Output all encodings.

3-Step Recipe for Obfuscating
$(\text{BP}_1, \ldots, \text{BP}_m, \alpha)$

Choose $2m$ random matrices $M_{i,b}$ s.t. $\sum_i M_{i,\alpha_i} = 0.$

Can use Leftover Hash Lemma if
$m >$ poly(input length, BP length)

For each i, 'encode' $\text{BP}_i : M_{i,0} : M_{i,1}$.

needs random $M_{i,0}, M_{i,1}$

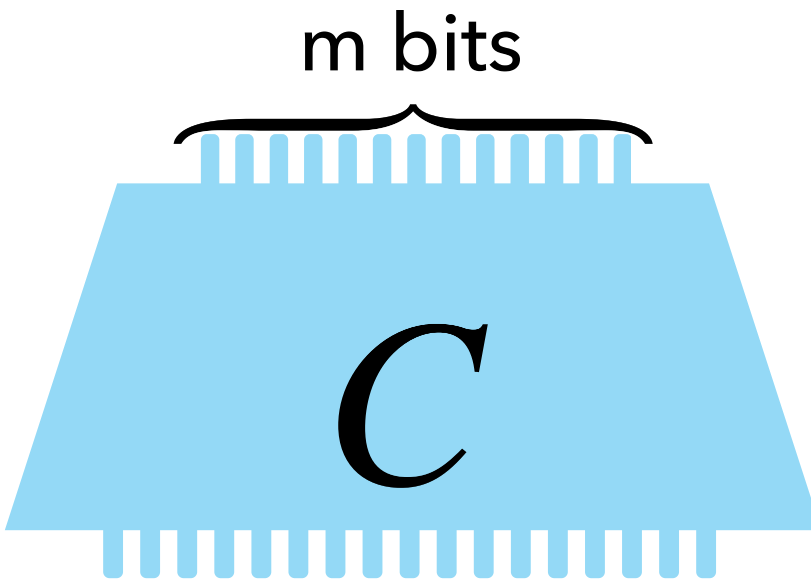
Encoding$_i$ hides BP_i

Evaluation of encoding$_i$ on x outputs $\approx R_x \cdot M_{i,\text{BP}_i(x)}$

Output all encodings.

Output length > poly(input length, BP length) ?

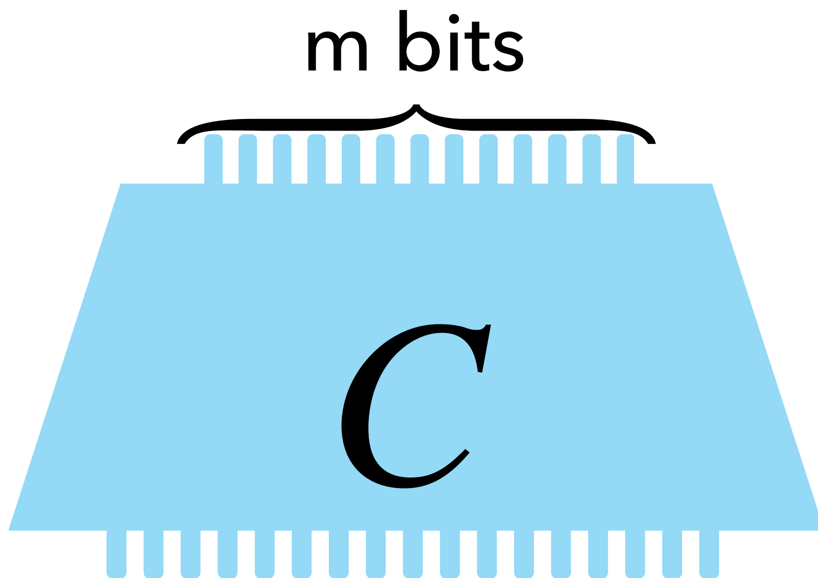
Obfuscate:



lock $\alpha \in \{0,1\}^m$

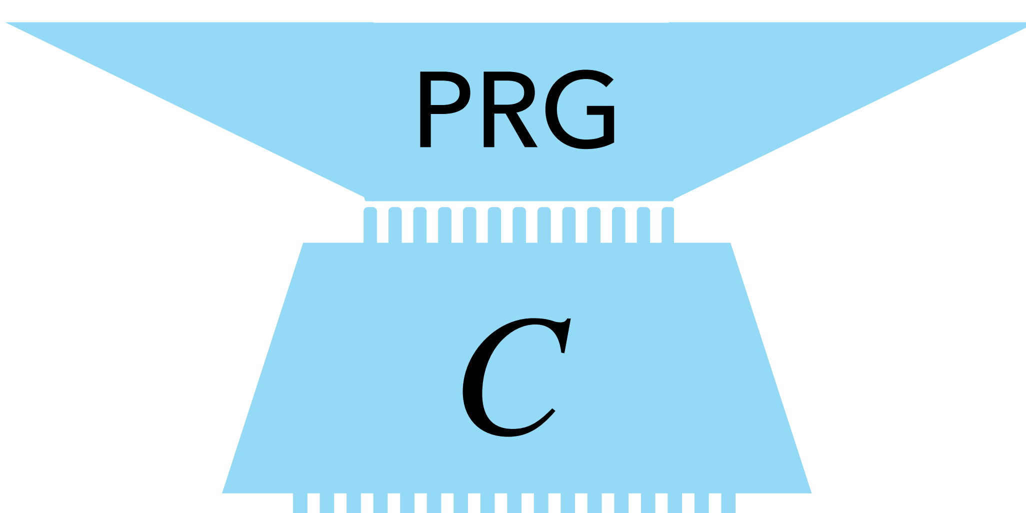
Output length > poly(input length, BP length) ?

Obfuscate:



lock $\alpha \in \{0,1\}^m$

Obfuscate:



lock PRG(α) $\in \{0,1\}^{m'}$

Lockable Obfuscation: Construction

STEP 1: Lockable Obf. for NC^1

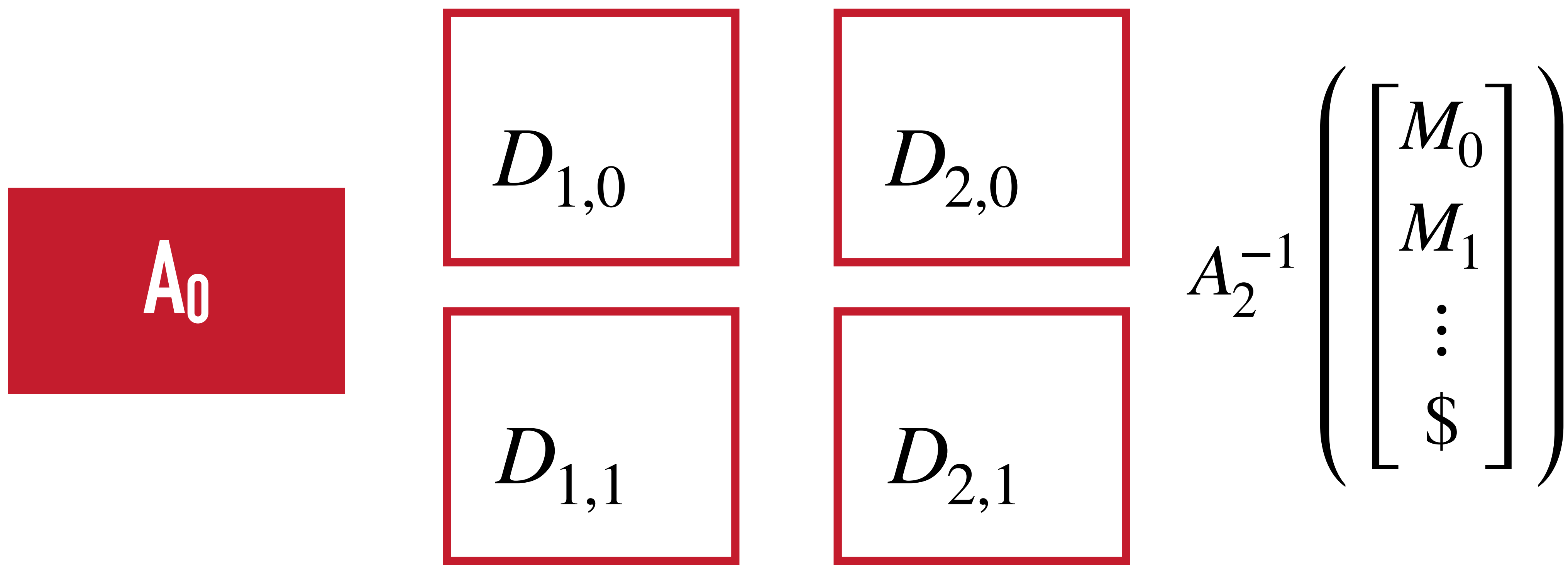
STEP 2: Bootstrapping

LO for NC^1 + FHE ⟶ LO for P/poly

* FHE with NC^1 decryption

Obfuscating Non-Permutation Branching Programs

[Chen, Vaikuntanathan, Wee 18]



$$D_{i,b} = A_{i-1}^{-1}\Big(\big(B_{i,b} \otimes R_{i,b}\big) A_i \; + \; \text{noise} \Big)$$

Obfuscating Non-Permutation Branching Programs

[Chen, Vaikuntanathan, Wee 18]

A_0

$D_{1,0}$ $D_{2,0}$

$D_{1,1}$ $D_{2,1}$

$$A_2^{-1}\left(\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}\right)$$

$$D_{i,b} = A_{i-1}^{-1}\Big(\big(B_{i,b} \otimes R_{i,b}\big) A_i \ + \ \text{noise} \Big)$$

$$D_{i,b} = A_{i-1}^{-1}\left(\begin{bmatrix} B_{i,b} & \\ & R_{i,b} \end{bmatrix} A_i \ + \ \text{noise} \right)$$

Obfuscating Non-Permutation Branching Programs

[Chen, Vaikuntanathan, Wee 18]

A_0

$D_{1,0}$ $D_{2,0}$

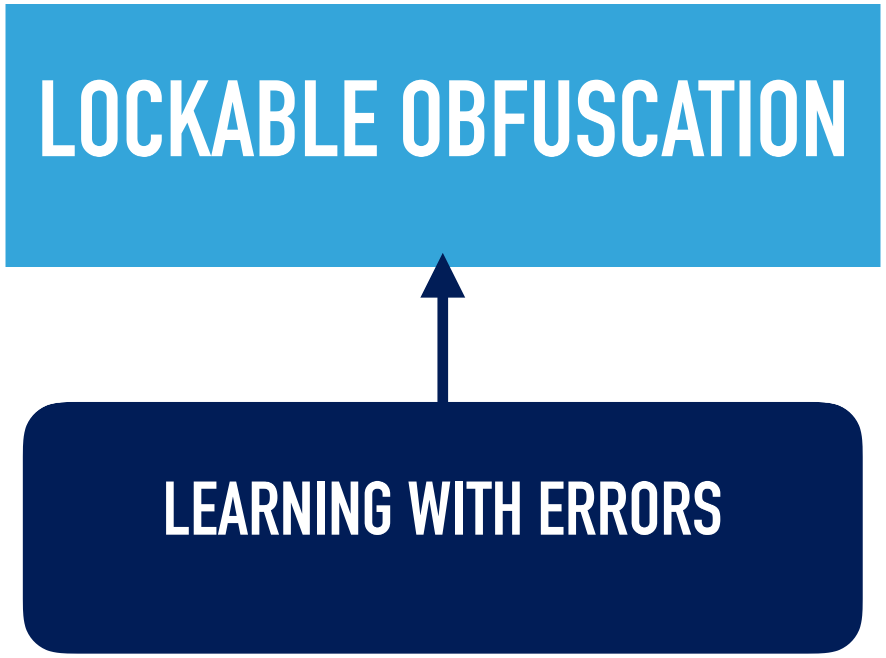
$D_{1,1}$ $D_{2,1}$

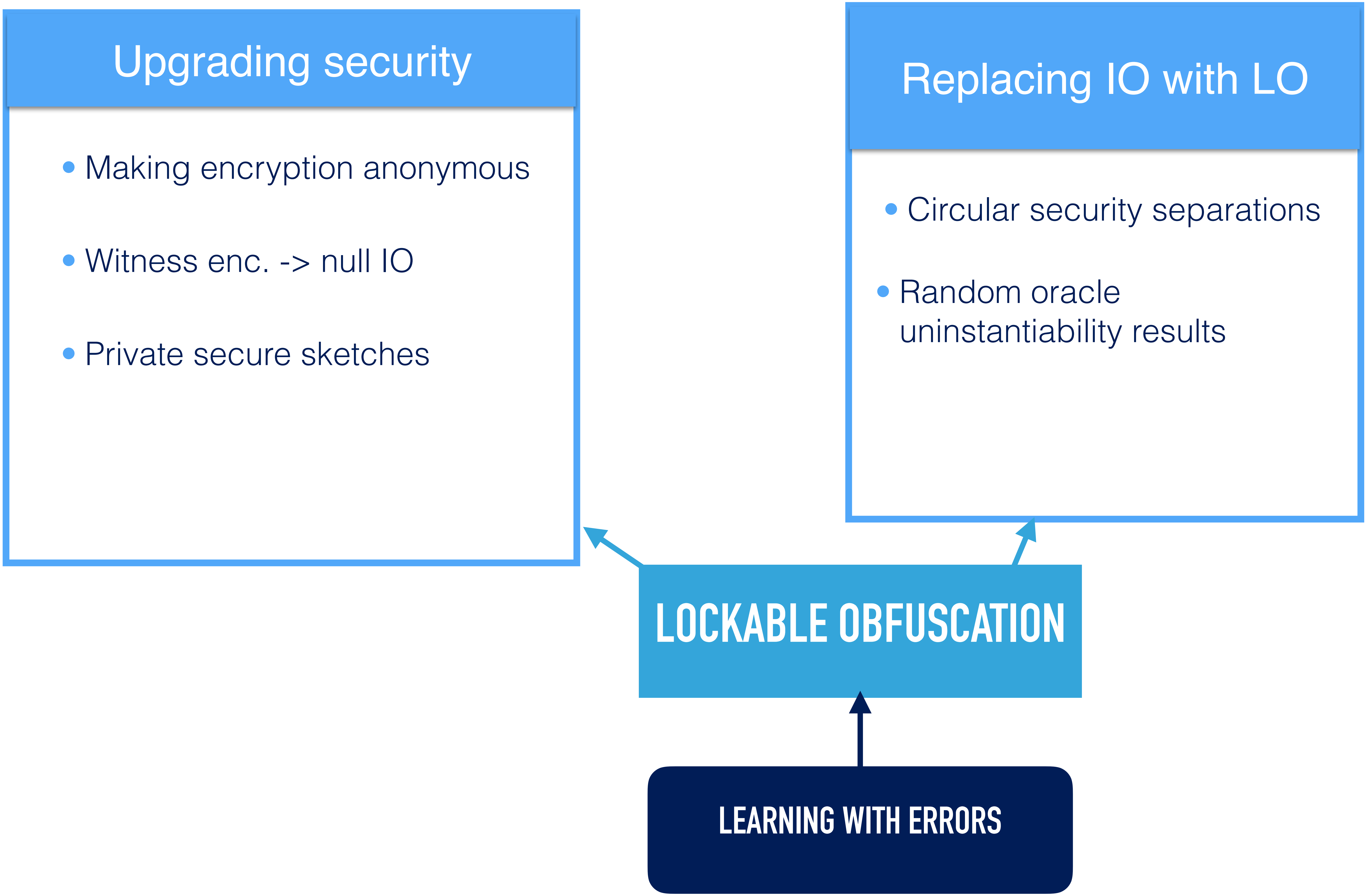
$$A_2^{-1}\left(\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \$ \end{bmatrix}\right)$$

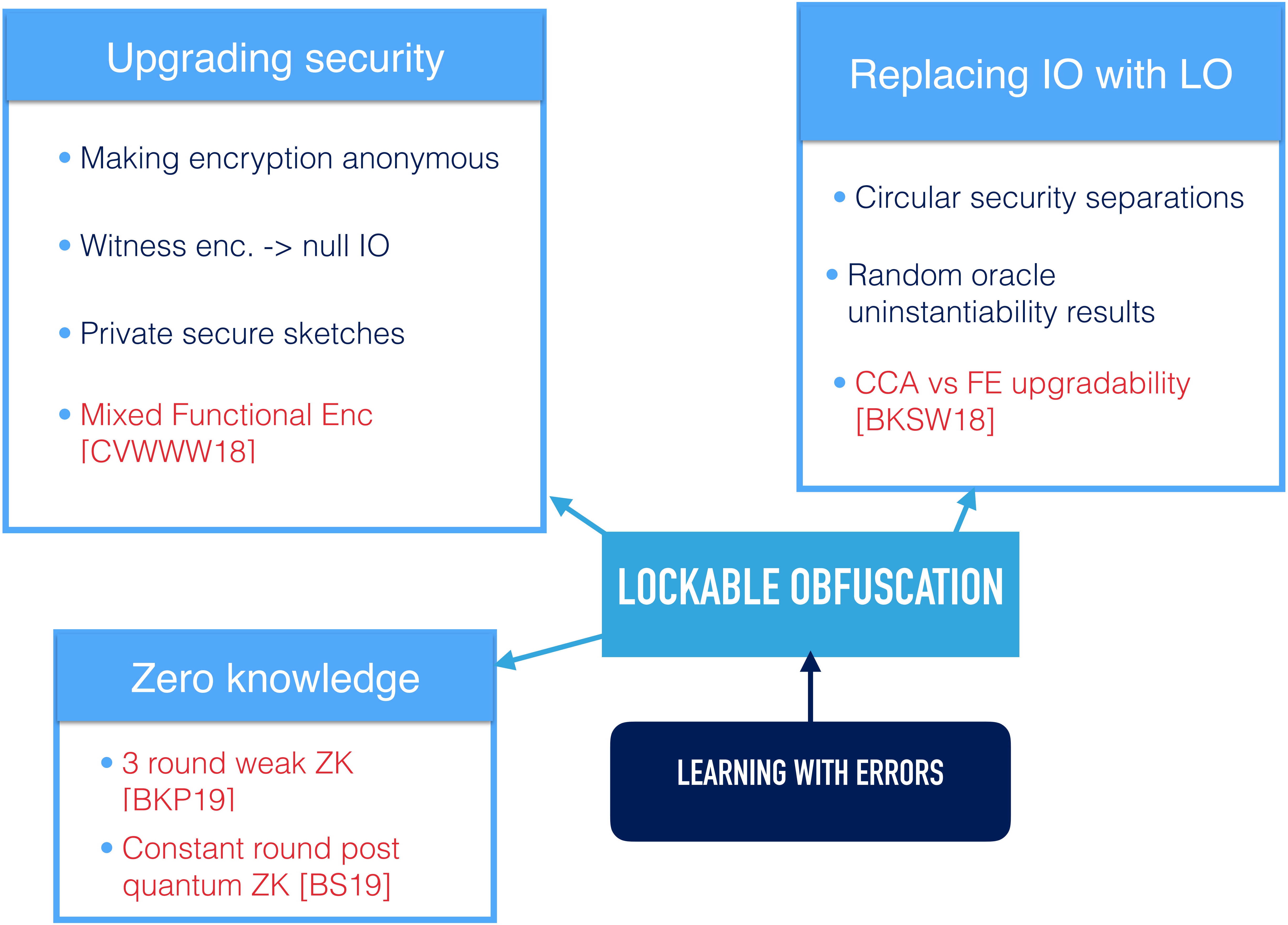
$$D_{i,b} = A_{i-1}^{-1}\left(\left(B_{i,b} \otimes R_{i,b}\right) A_i \;+\; \text{noise}\right)$$

RISHAB'S TALK

$$D_{i,b} = A_{i-1}^{-1}\left(\begin{bmatrix} B_{i,b} & \\ & R_{i,b} \end{bmatrix} A_i \;+\; \text{noise}\right) \overset{?}{\approx}$$


LOCKABLE OBFUSCATION
LEARNING WITH ERRORS


Upgrading security
Making encryption anonymous
Witness enc. -> null IO
Private secure sketches
Replacing IO with LO
Circular security separations
Random oracle uninstantiability results
LOCKABLE OBFUSCATION
LEARNING WITH ERRORS


Upgrading security
Making encryption anonymous
Witness enc. -> null IO
Private secure sketches
Mixed Functional Enc [CVWWW18]
Replacing IO with LO
Circular security separations
Random oracle uninstantiability results
CCA vs FE upgradability [BKSW18]
LOCKABLE OBFUSCATION
Zero knowledge
3 round weak ZK [BKP19]
Constant round post quantum ZK [BS19]
LEARNING WITH ERRORS

Open Problems

Open Problems

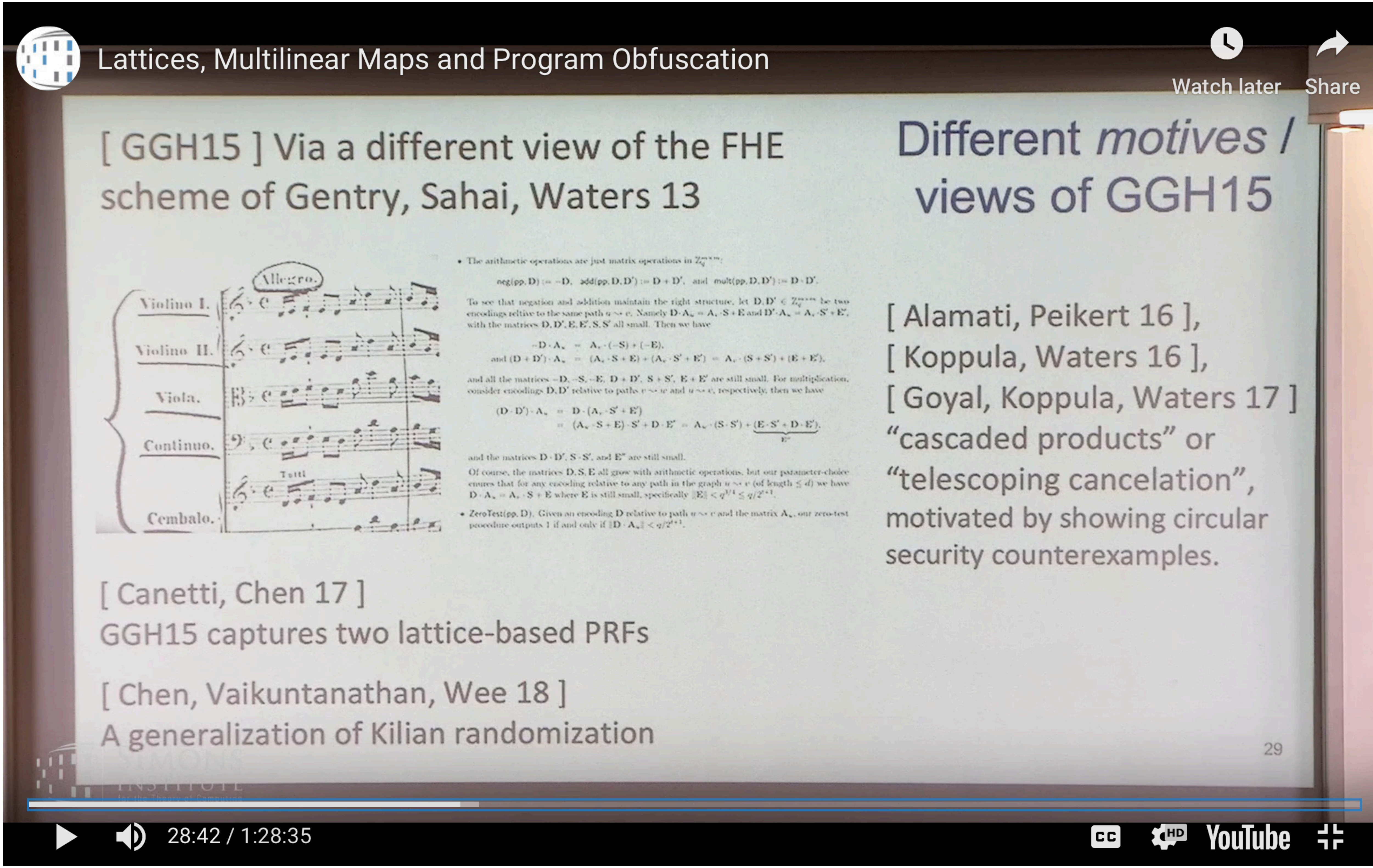
- More applications ?

Open Problems

- More applications ?
- Allowing some auxiliary information on lock/circuit/message?

Open Problems

- More applications ?
- Allowing some auxiliary information on lock/circuit/message?
- Obfuscation for other evasive circuits?



Yilei's talk on 'Lattices, Multilinear Maps and Program Obfuscation'

https://simons.berkeley.edu/talks/advanced-lattice-based-cryptography-fhe-abe-etc-0

Lockable Obfuscation: Behind the Scenes

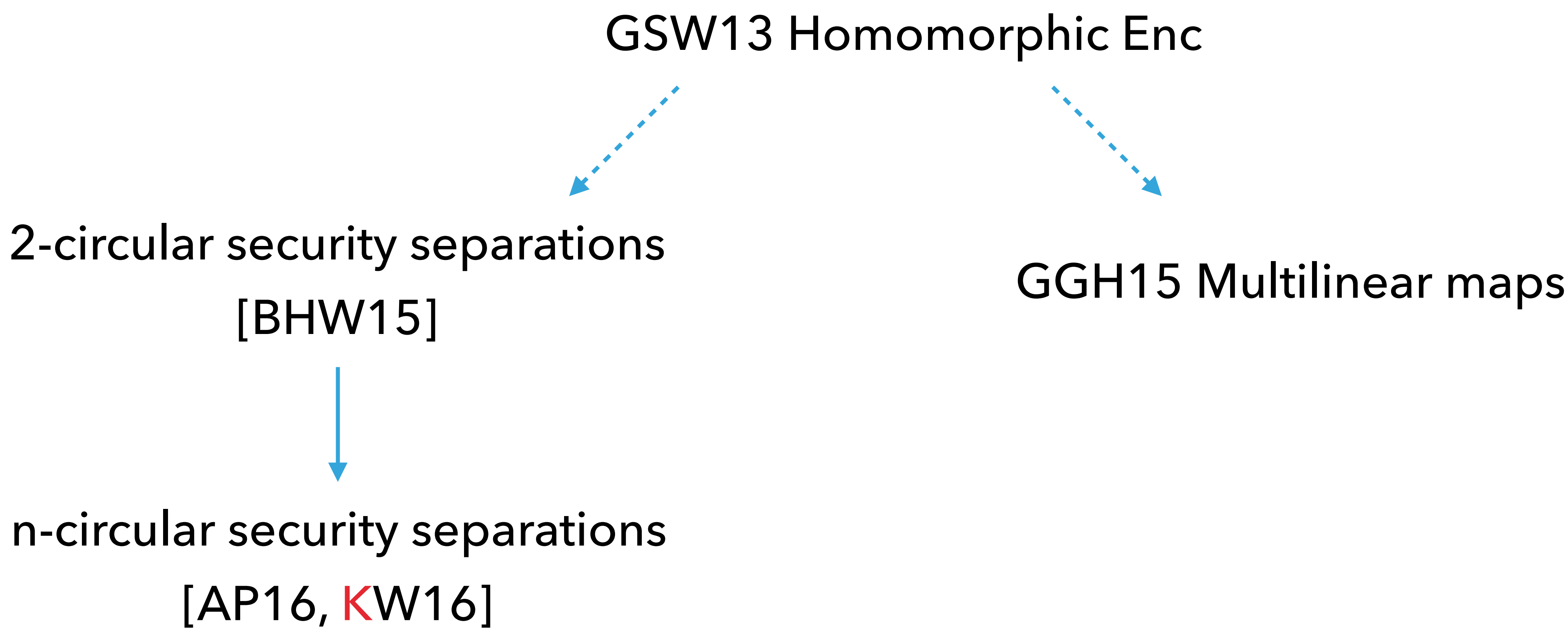
Lockable Obfuscation: Behind the Scenes

GSW13 Homomorphic Enc

2-circular security separations [BHW15]

GGH15 Multilinear maps

Lockable Obfuscation: Behind the Scenes



Lockable Obfuscation: Behind the Scenes

GSW13 Homomorphic Enc

2-circular security separations
[BHW15]

n-circular security separations
[AP16, KW16]

GGH15 Multilinear maps

Obfuscating Conjunctions
[BVWW16]

Lockable Obfuscation: Behind the Scenes

GSW13 Homomorphic Enc

2-circular security separations
[BHW15]

n-circular security separations
[AP16, KW16]

bit-circular security separations
[GKW17a]

GGH15 Multilinear maps

Obfuscating Conjunctions
[BVWW16]

Lockable Obfuscation: Behind the Scenes

GSW13 Homomorphic Enc

2-circular security separations
[BHW15]

n-circular security separations
[AP16, KW16]

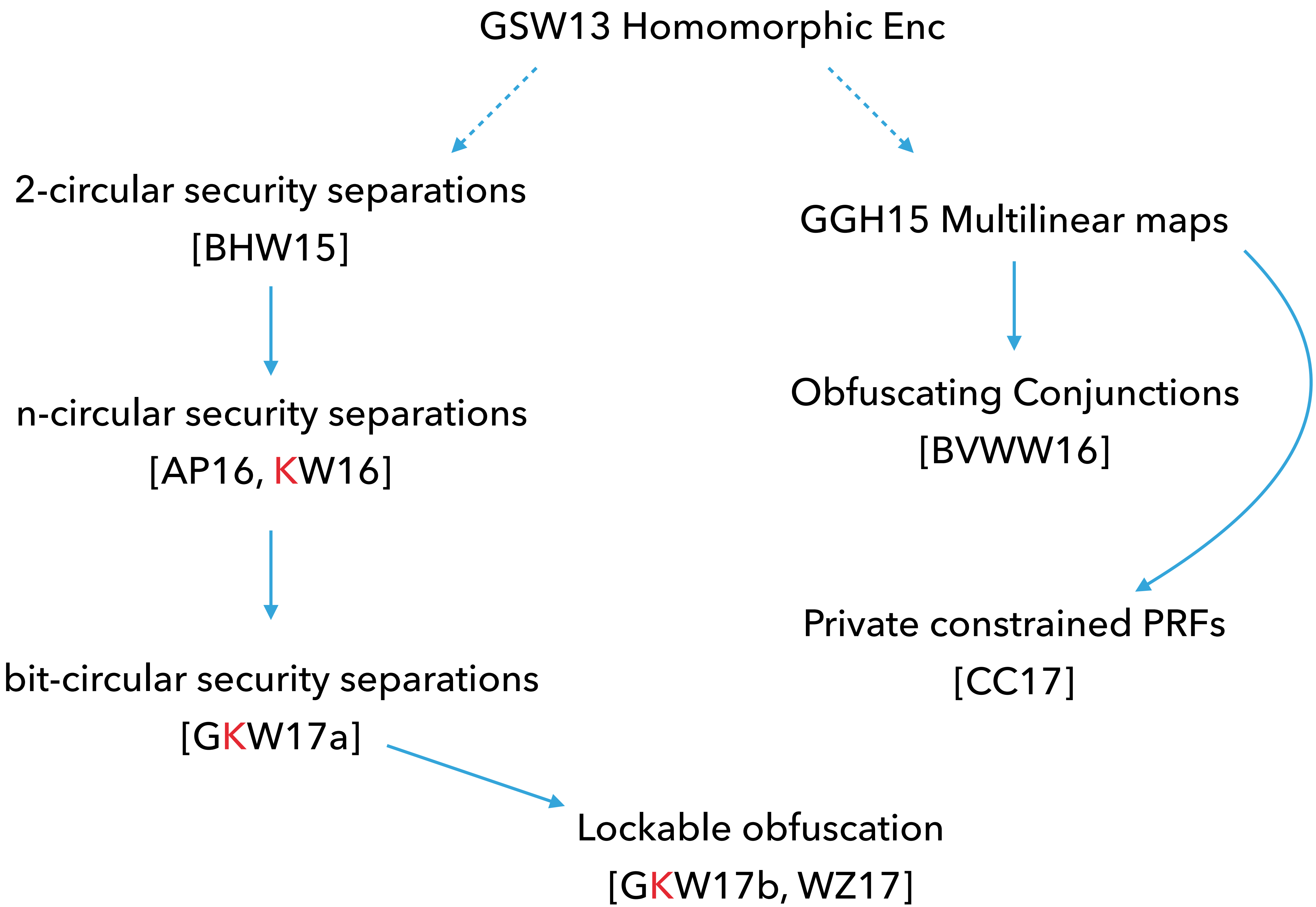
bit-circular security separations
[GKW17a]

GGH15 Multilinear maps

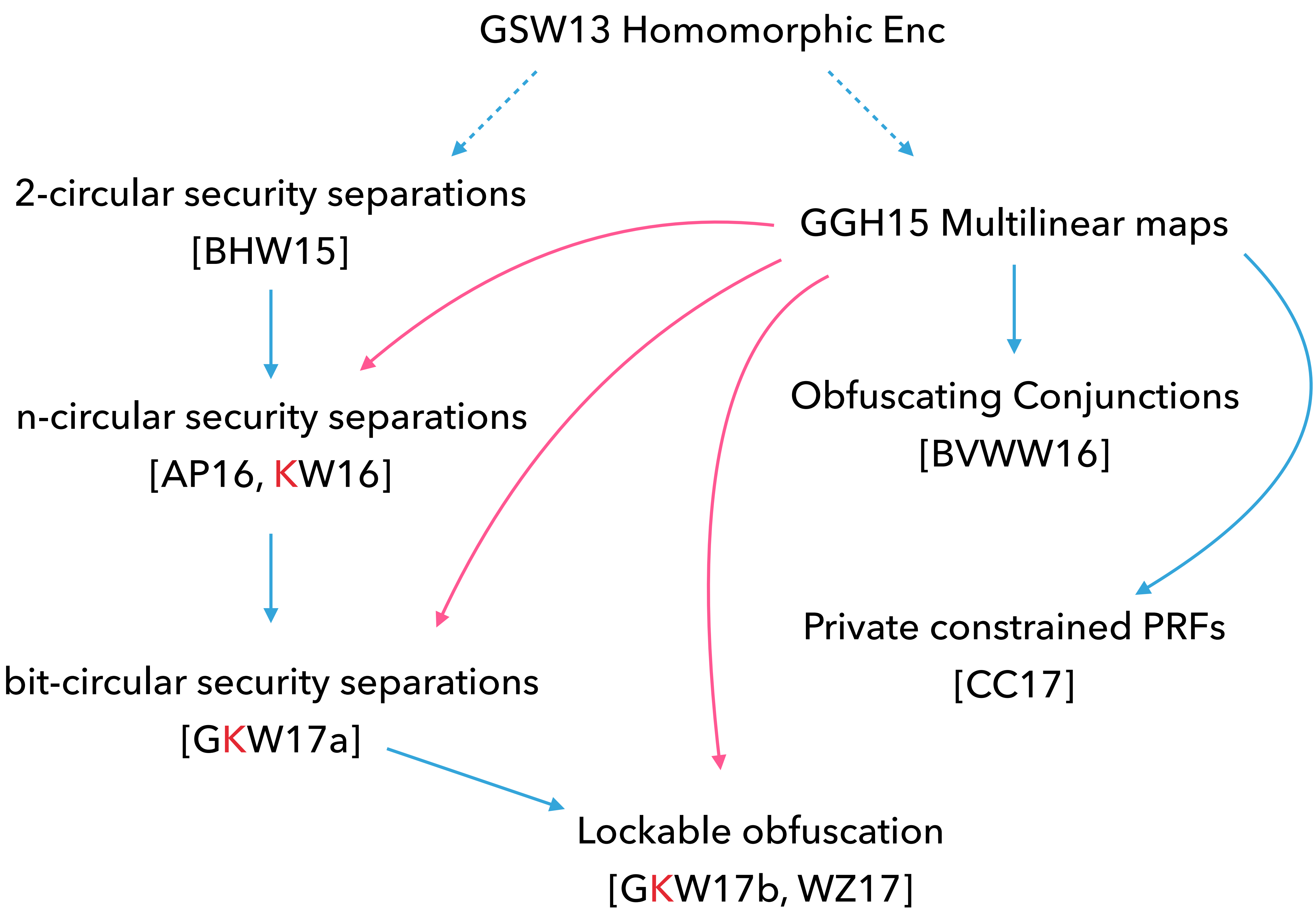
Obfuscating Conjunctions
[BVWW16]

Private constrained PRFs
[CC17]

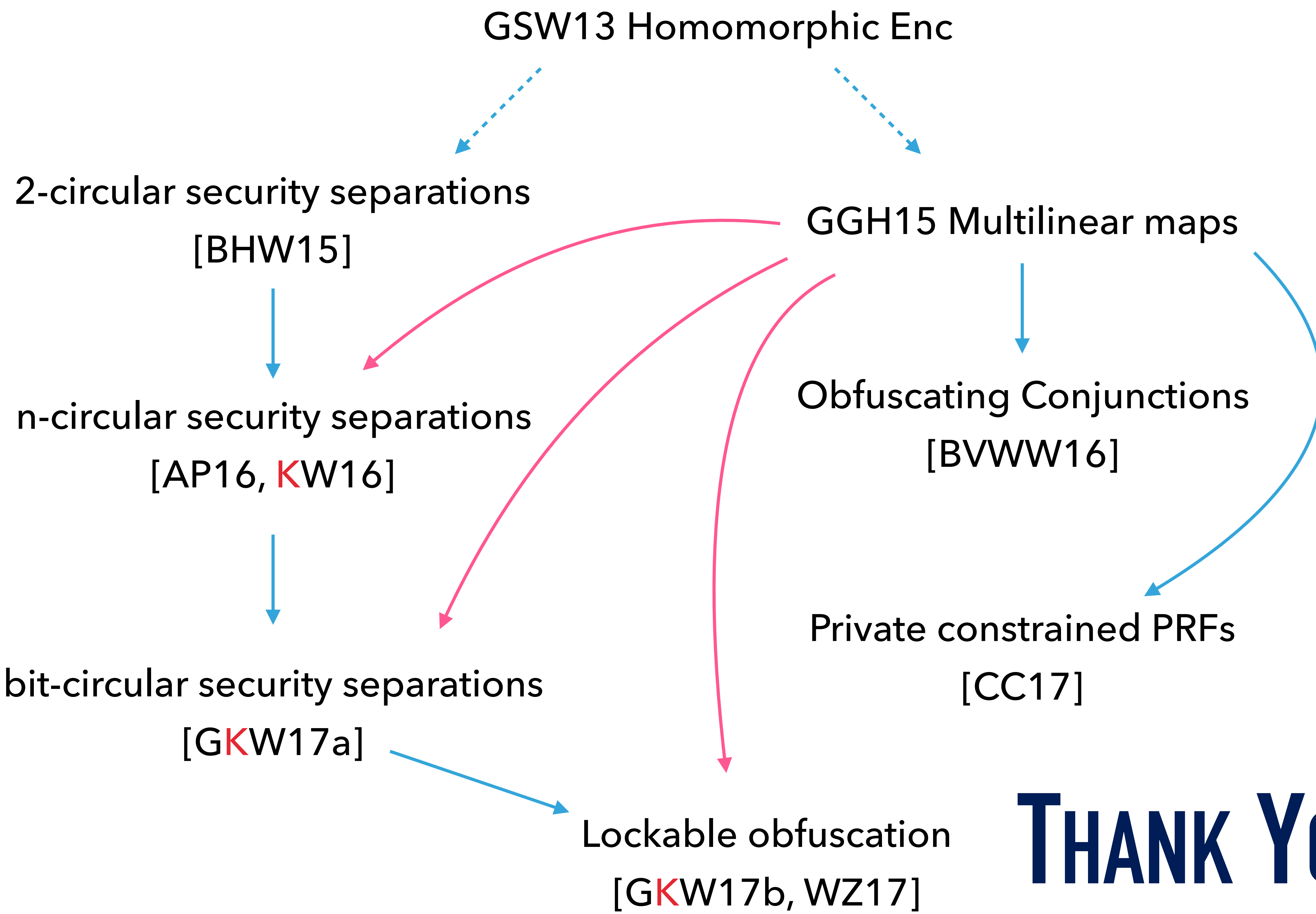
Lockable Obfuscation: Behind the Scenes



Lockable Obfuscation: Behind the Scenes



Lockable Obfuscation: Behind the Scenes



Thank You!!