GGH15 beyond permutation branching programs proofs, attacks, and candidates

Yilei Chen, Vinod Vaikuntanathan, Hoeteck Wee

BOSTON UNIVERSITY

VISA Research

MIT CSAIL

ENS
ÉCOLE NORMALE SUPÉRIEURE

> August 21, 2018, Palo Alto, heavy snow.

> August 21, 2018, Palo Alto, heavy snow.

> Alice finds a public-key encryption scheme based on Schrodinger's equation.

> August 21, 2018, Palo Alto, heavy snow.

> Alice finds a public-key encryption scheme based on Schrodinger’s equation.

> Alice missed the NIST PQC round one. But she find it cool to post it on the blockchain, and offers 100 Bitcoins to whoever breaks it.

> Not only does Alice post on the blockchain, she does it cool by encrypting the 100 Bitcoins using Witness encryption.

> Not only does Alice post on the blockchain, she does it cool by encrypting the 100 Bitcoins using Witness encryption.

> WitnessEnc(x, m), x = instance, m = message

Functionality: if x = SAT -----> can use the witness to decrypt the msg.

Security: if x = UNSAT -------> msg is hidden.

WitnessEnc(x = "there is an attack to Alice's PKE scheme", msg = 100 Bitcoins)

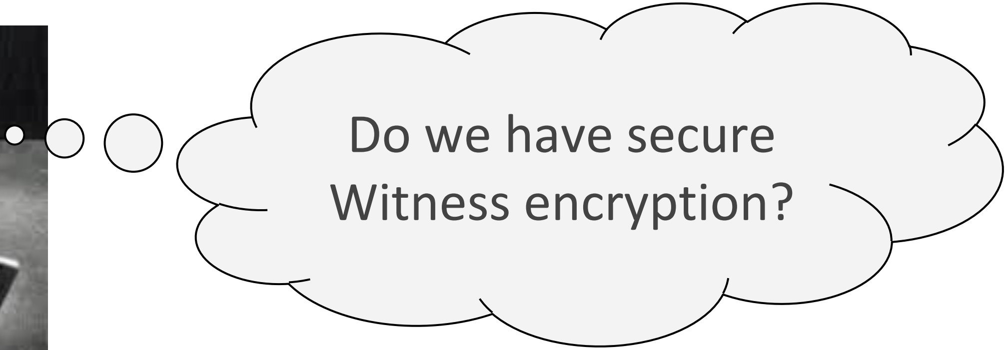
> Current status of witness encryption: there are several candidates (more-or-less based on multilinear maps); none of them are based on established cryptographic assumptions.

> [Garg et al. 13] candidate witness encryption based on GGH13.

> Broken by [Hu, Jia 16]

> [Gentry, Lewko, Waters 14] from multilinear subgroup decision assumption (which is also open)

> Null-iO candidates (there are many) => Witness encryption candidates



I am the title

GGH15 beyond permutation branching programs proofs, attacks, and candidates

Wait, what's the relation of witness encryption and the title??

A candidate multilinear map

GGH15 beyond permutation branching programs
proofs, attacks, and candidates

A candidate multilinear map

GGH15 beyond permutation branching programs
proofs, attacks, and candidates

applications

Private constrained PRFs

Multi party key agreement

Lockable obfuscation
(Compute-then-Compare obf.)

General purpose
Indistinguishability obfuscation

Security ????

$\boxed{\text{GGH15}}$ beyond permutation branching programs

proofs, attacks, and candidates

(As secure as LWE)

What we knew:

Private constrained PRFs

Multi party key agreement

Lockable obfuscation
(Compute-then-Compare obf.)

General purpose
Indistinguishability obfuscation

Motivation of this work: systematically study GGH15, discover more attacks and safe applications

GGH15 beyond permutation branching programs

proofs, attacks, and candidates

(As secure as LWE)

Private constrained PRFs

Lockable obfuscation
(Compute-then-Compare obf.)

Multi party key agreement

General purpose
Indistinguishability obfuscation

Motivation of this work: systematically study GGH15, discover more attacks and safe applications (maybe witness encryption?)

GGH15 beyond permutation branching programs

proofs, attacks, and candidates

(As secure as LWE)

Private constrained PRFs

Lockable obfuscation
(Compute-then-Compare obf.)

Witness encryption ???

Multi party key agreement

General purpose
Indistinguishability obfuscation

Summary of the results for GGH15 + non-perm branching programs:

- Proofs (focus of the talk):
  > Introduce new lattice toolkits;
  > New analysis techniques for GGH15.
  > Leads to PCPRFs and lockable obfuscation for general BPs.

- Attacks: New attacks on the iO candidates.

- Candidates: Witness encryption and iO.

Multilinear maps in a nutshell

> Multilinear maps: motivated in [Boneh, Silverberg 2003]

$$g, g^{s_1}, g^{s_2}, g^{s_3}, \ldots \rightarrow g^{\prod s}$$

Can be thought of as homomorphic encryption + public zero-test

Multilinear maps in a nutshell

> Multilinear maps: motivated in [Boneh, Silverberg 2003]

$$g, g^{S_1}, g^{S_2}, g^{S_3}, \ldots \rightarrow g^{\prod S}$$

Can be thought of as homomorphic encryption + public zero-test

> Bilinear maps from elliptic curves [Miller 1986]
> n-linear maps candidates: (all based on non-standard use of lattices)
>>>> Garg, Gentry, Halevi 2013 [GGH 13]
>>>> Coron, Lepoint, Tibouchi 2013 [CLT 13]
>>>> Gentry, Gorbunov, Halevi 2015 [GGH 15] (LWE-like)

*New: Trilinear maps from abelian varieties [Huang 2018], requires further investigation.

GGH15 in a nutshell

> Multilinear maps: motivated in [Boneh, Silverberg 2003]

$$g, g^{S_1}, g^{S_2}, g^{S_3}, \ldots \rightarrow g^{\prod S}$$

> (Ring)LWE analogy:

$$A, S_1 A + E_1, \ldots, S_k A + E_k \rightarrow \prod S A + E \mod q$$

> (Ring)LWE analogy:

$$A,\ S_1A+E_1, \ldots, S_kA+E_k \longrightarrow \prod S A+E \mod q$$

GGH15: "the blockchain in multilinear maps"

(also appear as "cascaded LWE" in [Koppula-Waters 16], [Alamati-Peikert 16])

GGH15 in a nutshell

> (Ring)LWE analogy:

$$A,\ S_1A+E_1, \ldots, S_kA+E_k \longrightarrow \prod SA+E \mod q$$

> GGH15: (also appear as “cascaded LWE” in [Koppula-Waters 16], [Alamati-Peikert 16])

$$A_0\ D_1 = S_1A_1+E_1, \quad A_1\ D_2 = S_2A_2+E_2 \mod q$$

GGH15 in a nutshell

> (Ring)LWE analogy:

$$A,\ S_1A+E_1, \dots, S_kA+E_k \longrightarrow \prod S A+E \mod q$$

> GGH15: (also appear as "cascaded LWE" in [Koppula-Waters 16], [Alamati-Peikert 16])

$$A_0\, D_1 = S_1A_1+E_1, \quad A_1\, D_2 = S_2A_2+E_2 \quad \mod q$$

D_i is sampled using the trapdoor of A_{i-1}

Lattice trapdoor 101
[Ajtai 99, Alwen, Peikert 09, Micciancio, Peikert 12]

A with trapdoor

Given Y find D s.t. A x D = Y

GGH15 in a nutshell

> (Ring)LWE analogy:

$$A,\ S_1A+E_1, \ldots, S_kA+E_k \longrightarrow \prod SA+E \mod q$$

> GGH15: (also appear as "cascaded LWE" in [Koppula-Waters 16], [Alamati-Peikert 16])

$$A_0 D_1 = S_1A_1+E_1, \quad A_1 D_2 = S_2A_2+E_2 \quad \mod q$$

D_i is sampled using the trapdoor of A_{i-1}

Publish A_0, D_1, D_2 as the encodings of S_1, S_2

GGH15 in a nutshell

> (Ring)LWE analogy:

$$A,\ S_1A+E_1, \ldots,\ S_kA+E_k \longrightarrow \prod SA+E \mod q$$

> GGH15: (also appear as “cascaded LWE” in [Koppula-Waters 16], [Alamati-Peikert 16])

$$A_0 D_1 = S_1A_1+E_1, \quad A_1 D_2 = S_2A_2+E_2 \quad \mod q$$

D_i is sampled using the trapdoor of A_{i-1}

Publish A_0, D_1, D_2 as the encodings of S_1, S_2

$$\text{Eval} = A_0 D_1 D_2 = (S_1A_1+E_1)D_2 = \underbrace{S_1S_2A_2}_{\text{functionality}} + \underbrace{E_1D_2+S_1E_2}_{\text{small}} \mod q$$

When witness encryption meets multilinear maps ...

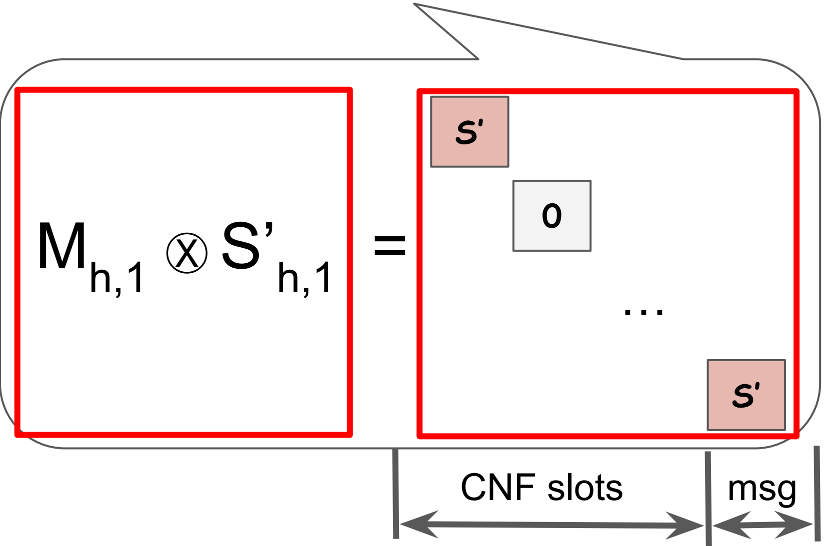
[Gentry, Lewko, Waters 14] witness encryption from mmaps subgroup decision assumption, which is instance independent.

[Gentry, Lewko, Waters 14] a special witness encryption from mmaps.

A strawman implementation of GLW14 in GGH15

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = S_{h,0} A_h + E_{h,0} \quad \bmod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \bmod q$$



- Low-rank matrices (bad news)
- Read-once BP (good news)

So far: A witness encryption with special structure that uses

GGH15 + low-rank matrix branching program.

So far: A witness encryption with special structure that uses

GGH15 + low-rank matrix branching program.

Q: Can we show anything secure for low-rank BP + GGH15?

So far: A witness encryption with special structure that uses

GGH15 + low-rank matrix branching program.

Q: Can we show anything secure for low-rank BP + GGH15?

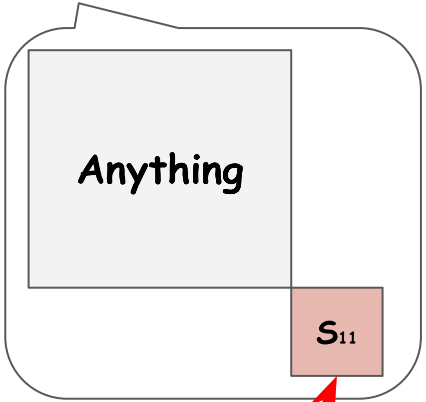
A: Yes! … In some limited cases

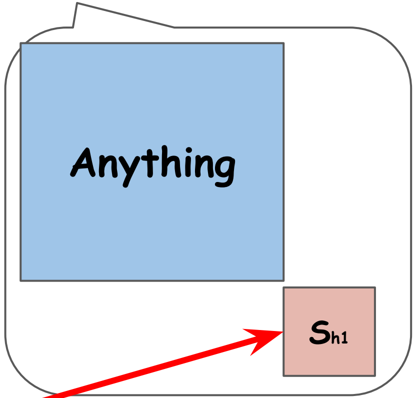
As secure as LWE:

When there is one “slot” that is always random in all the matrices.

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = S_{h,0} A_h + E_{h,0} \quad \mod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = \boxed{S_{h,1}} A_h + E_{h,1} \quad \mod q$$





The “always random” slot

Where can the special type of BP be useful?

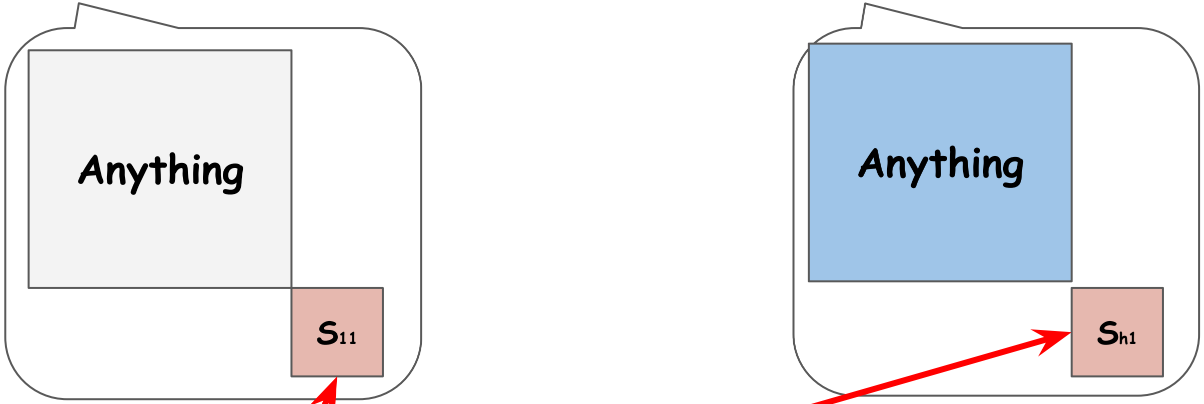
$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ \ldots, \quad A_{h-1} D_{h,1} = \boxed{S_{h,1}} A_h + E_{h,1} \mod q$$

The “always random” slot

Where can the special type of BP be useful?

We don’t know how to build a witness encryption or iO from this type of BP :(

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ \ldots, \quad A_{h-1} D_{h,1} = \boxed{S_{h,1}} A_h + E_{h,1} \quad \bmod q$$



The “always random” slot

Where can the special type of BP be useful?

We don’t know how to build a witness encryption or iO from this type of BP :(

We can simplify the private constrained PRF, Lockable obfuscation :)

E.g. Instantiate the private puncturable PRF from [Boneh, Lewi, Wu 17] described under the multilinear subgroup decision assumption:

Where can the special type of BP be useful?

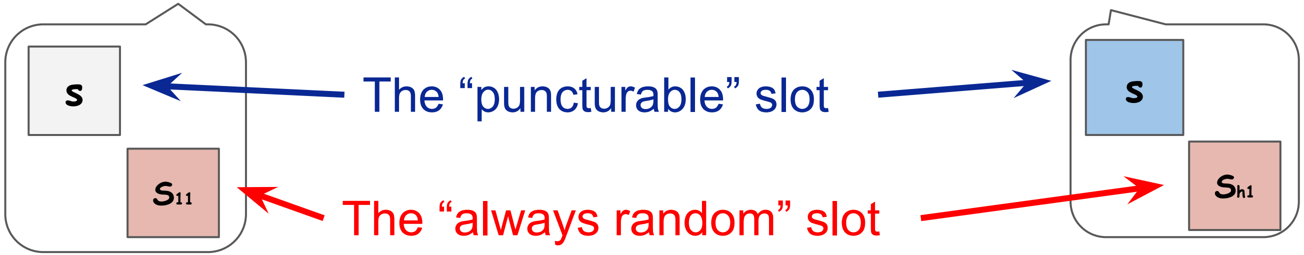
We don’t know how to build a witness encryption or iO from this type of BP :(

We can simplify the private constrained PRF, Lockable obfuscation :)

E.g. Instantiate the private puncturable PRF from [Boneh, Lewi, Wu 17] described under the multilinear subgroup decision assumption:

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = S_{h,0} A_h + E_{h,0} \quad \bmod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = \boxed{S_{h,1}} A_h + E_{h,1} \quad \bmod q$$



How to prove security for GGH15 + low-rank BPs?

What are you
trying to prove?

How to prove security for GGH15 + low-rank BPs?

Semantic security:

$$A_0\, D_{1,0} = S_{1,0} A_1 + E_{1,0},\ \ldots,\quad A_{h-1}\, D_{h,0} = S_{h,0} A_h + E_{h,0} \quad \bmod q$$

$$A_0\, D_{1,1} = S_{1,1} A_1 + E_{1,1},\ \ldots,\quad A_{h-1}\, D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \bmod q$$

$$\approx \text{ computational}$$

$$A_0\, D_{1,0} = \boxed{U_{1,0}},\ \ldots,\quad A_{h-1}\, D_{h,0} = \boxed{U_{h,0}} \quad \bmod q$$

$$A_0\, D_{1,1} = \boxed{U_{1,1}},\ \ldots,\quad A_{h-1}\, D_{h,1} = \boxed{U_{h,1}} \quad \bmod q$$

"A" matrices: using trapdoors; not using trapdoors

Replay: the proof for GGH15 + permutation BP

[Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]

Replay: the proof for GGH15 + permutation BP

[Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]

Goal: prove semantic security

For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = S_{h,0} A_h + E_{h,0} \quad \text{mod } q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = \boxed{S_{h,1}} A_h + E_{h,1} \quad \text{mod } q$$

$$\begin{pmatrix} S_{11} & & \\ & S_{11} & \\ & & S_{11} \end{pmatrix} \qquad \begin{pmatrix} S_{h1} & & \\ & & S_{h1} \\ & S_{h1} & \end{pmatrix}$$

LWE 101 [Regev 05]

$$A \,,\quad S \times A + E$$

$$\approx_{\text{computational}} \quad A \,,\quad U$$

LWE 101 [Regev 05]

A , S x A + E

$\approx$ computational A , U

Permutation - LWE:

$$\begin{pmatrix} A(1) \\ A(2) \\ A(3) \end{pmatrix}, \boxed{\begin{matrix} & S & \\ S & & \\ & & S \end{matrix}} \times \begin{pmatrix} A(1) \\ A(2) \\ A(3) \end{pmatrix} + E$$

$\approx$ computational

$$\begin{pmatrix} A(1) \\ A(2) \\ A(3) \end{pmatrix}, U$$

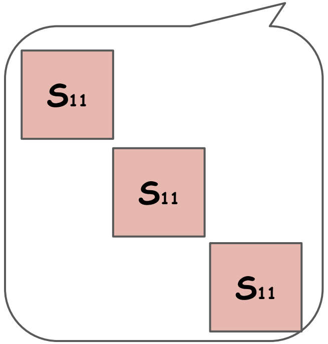
Goal: prove semantic security

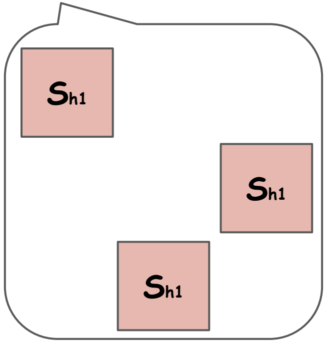
For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = S_{h,0} A_h + E_{h,0} \quad \bmod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = \boxed{S_{h,1}} A_h + E_{h,1} \quad \bmod q$$





[Step 1] LWE: $A_h, S_{h,0} A_h + E_{h,0}, S_{h,1} A_h + E_{h,1} \approx A_h, U_{h,0}, U_{h,1}$

Goal: prove semantic security

For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ \ldots, \quad A_{h-1} D_{h,0} = \; U_{h,0} \quad {}_0 \quad \bmod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ \ldots, \quad A_{h-1} D_{h,1} = \; U_{h,1} \quad {}_1 \quad \bmod q$$

$$\begin{pmatrix} s_{11} & & \\ & s_{11} & \\ & & s_{11} \end{pmatrix}$$

[Step 1] LWE: $A_h , S_{h,0} A_h + E_{h,0} , S_{h,1} A_h + E_{h,1} \approx A_h , U_{h,0} , U_{h,1}$

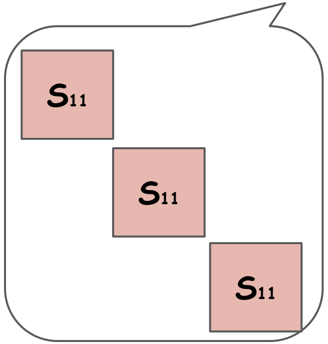
Goal: prove semantic security

For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = U_{h,0} \;\; 0 \quad \text{mod } q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = U_{h,1} \;\; 1 \quad \text{mod } q$$

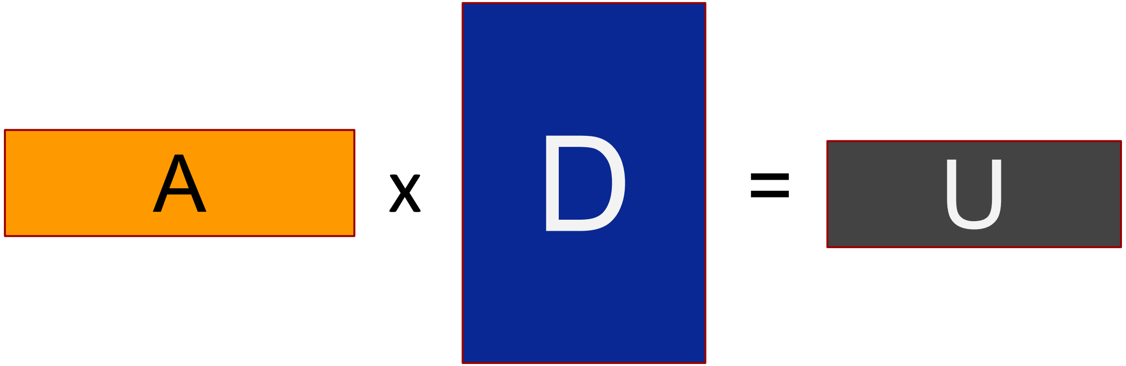


[Step 2] GPV: close the trapdoor of A_{h-1}

[Gentry, Peikert, Vaikuntanathan 08]

U is uniform

A trapdoor is used



[Gentry, Peikert, Vaikuntanathan 08]

U is uniform

A trapdoor is used

$$A \times D = U$$

$\approx$ statistical

$$A \times D = U$$

close the trapdoor of A

Goal: prove semantic security

For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ldots, \quad A_{h-1} D_{h,0} = U_{h,0} \quad {}_0 \quad \bmod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ldots, \quad A_{h-1} D_{h,1} = U_{h,1} \quad {}_1 \quad \bmod q$$

$$\begin{pmatrix} s_{11} & & \\ & s_{11} & \\ & & s_{11} \end{pmatrix}$$

[Step 2] GPV: close the trapdoor of A_{h-1}

Goal: prove semantic security

For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ \ldots, \quad A_{h-1} D_{h,0} = U_{h,0} \quad {}_0 \quad \bmod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ \ldots, \quad A_{h-1} D_{h,1} = U_{h,1} \quad {}_1 \quad \bmod q$$

$$\begin{pmatrix} s_{11} & & \\ & s_{11} & \\ & & s_{11} \end{pmatrix}$$

[Step 2] GPV: close the trapdoor of A_{h-1}

Goal: prove semantic security

For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = S_{1,0} A_1 + E_{1,0}, \ \ldots, \quad A_{h-1} D_{h,0} = U_{h,0} \quad {}_0 \quad \mod q$$

$$A_0 D_{1,1} = \boxed{S_{1,1}} A_1 + E_{1,1}, \ \ldots, \quad A_{h-1} D_{h,1} = U_{h,1} \quad {}_1 \quad \mod q$$

$$\begin{pmatrix} S_{11} & & \\ & S_{11} & \\ & & S_{11} \end{pmatrix}$$

...

[Step …] LWE .... GPV: close the trapdoor of A_1

Goal: prove semantic security

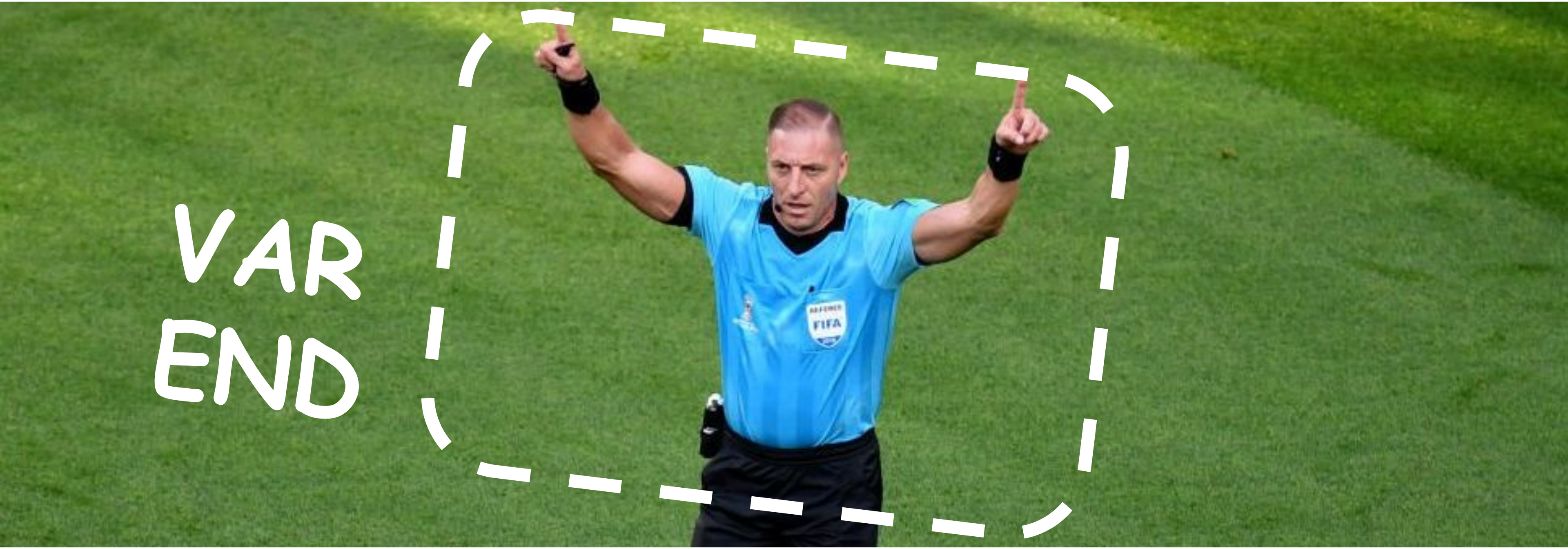
For permutation BP [Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]:

"A" matrices: using trapdoors; not using trapdoors

$$A_0 D_{1,0} = U_{1,0}, \ldots, A_{h-1} D_{h,0} = U_{h,0} \quad \text{mod } q$$

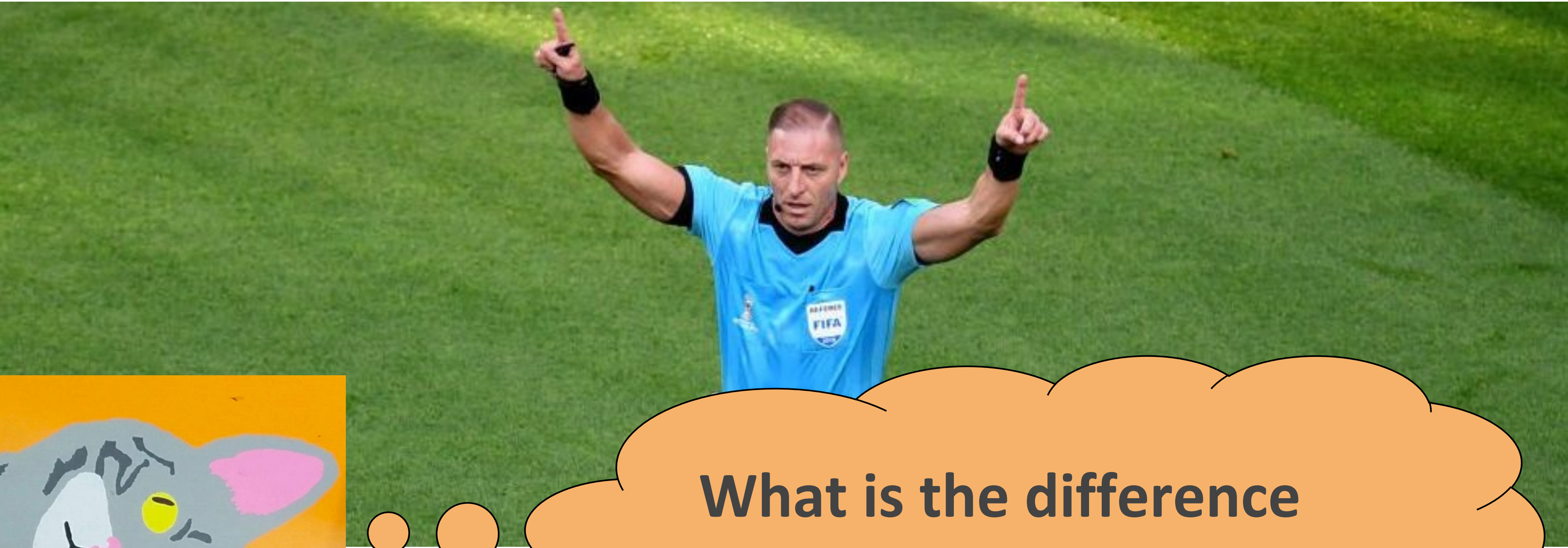
$$A_0 D_{1,1} = U_{1,1}, \ldots, A_{h-1} D_{h,1} = U_{h,1} \quad \text{mod } q$$

[Final Steps] Another LWE + GPV



Replay: the proof for GGH15 + permutation BP

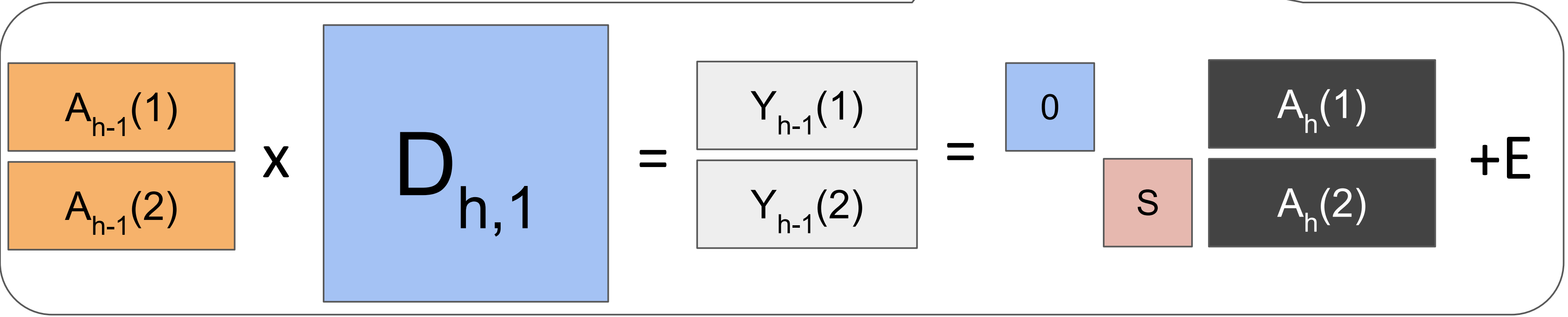
[Canetti, Chen 17], [Goyal, Koppula, Waters 17], [Wichs, Zirdelis 17]



What is the difference for low-rank matrices?

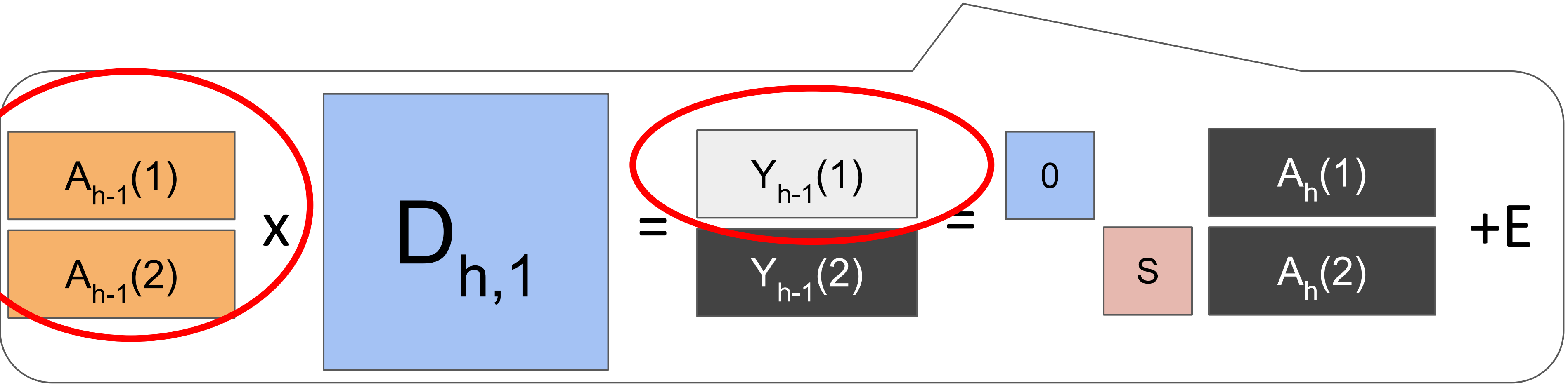
For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

$$A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}, \ldots, \boxed{A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1}} \quad \text{mod } q$$



For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

$$A_0 D_{1,1} = S_{1,1}A_1+E_{1,1}, \ldots, \boxed{A_{h-1} D_{h,1} = S_{h,1}A_h+E_{h,1}} \quad \bmod q$$



Observation: $Y_{h-1}(1)$ is not random

The problem: How to close the trapdoor of A_{h-1} ?

Lattice trapdoor Lemma 1:

Z is arbitrary

U is uniform

A trapdoor is used

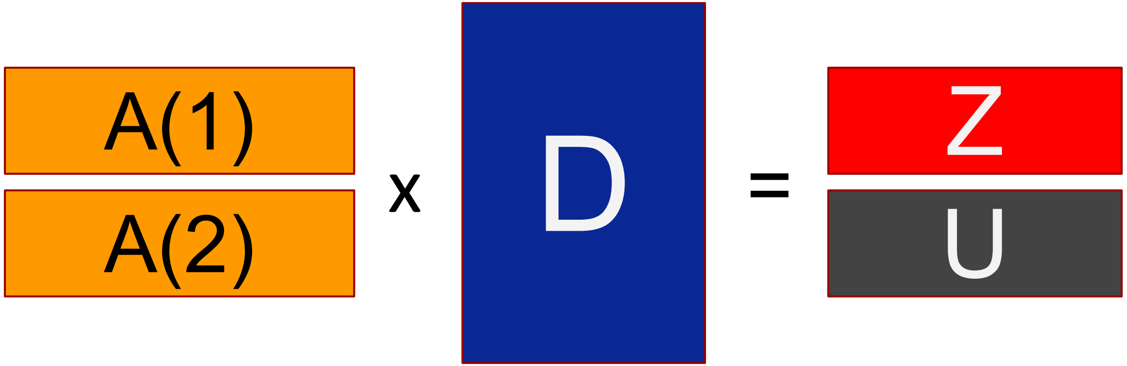
$$\begin{bmatrix} A(1) \\ A(2) \end{bmatrix} \times D = \begin{bmatrix} Z \\ U \end{bmatrix}$$

Lattice trapdoor Lemma 1:

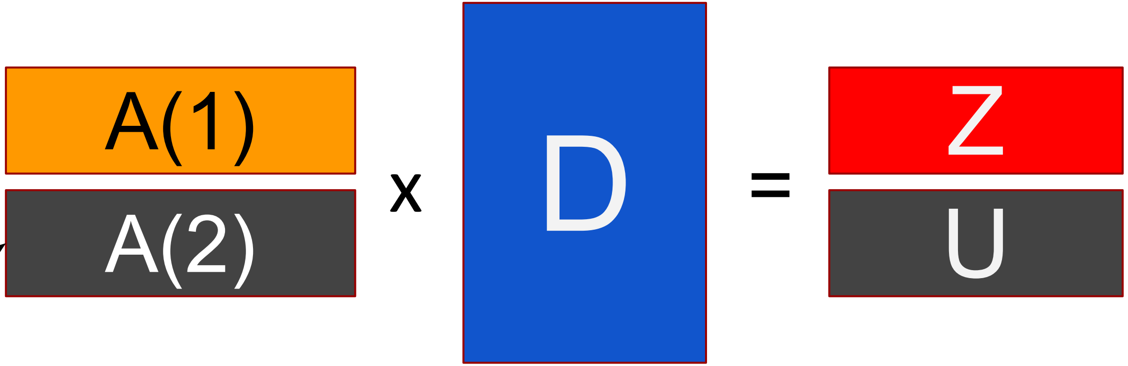
Z is arbitrary

U is uniform

A trapdoor is used



≈ statistical



close the trapdoor of A(2)

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

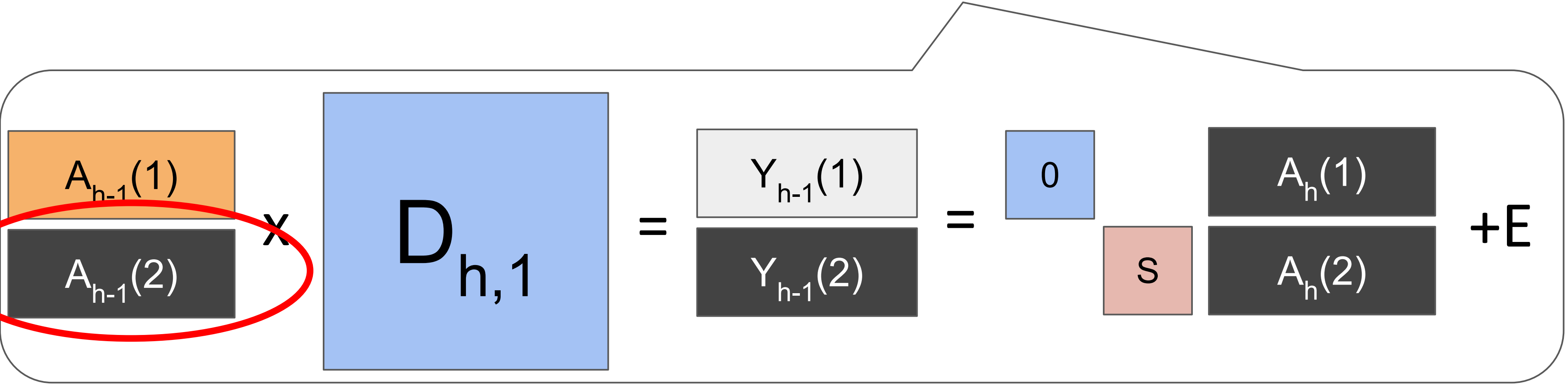
$$A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}, \ldots, \boxed{A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1}} \mod q$$

$$\begin{bmatrix} A_{h-1}(1) \\ A_{h-1}(2) \end{bmatrix} \times D_{h,1} = \begin{bmatrix} Y_{h-1}(1) \\ Y_{h-1}(2) \end{bmatrix} = \begin{bmatrix} 0 & \\ & S \end{bmatrix} \begin{bmatrix} A_h(1) \\ A_h(2) \end{bmatrix} + E$$

Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

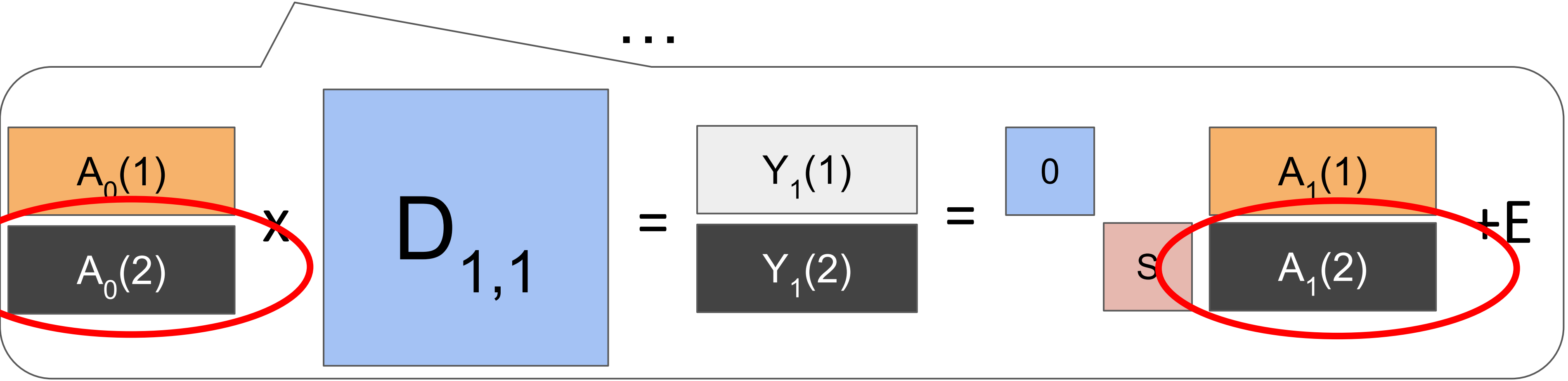
$$A_0 D_{1,1} = S_{1,1}A_1+E_{1,1}, \ldots, \boxed{A_{h-1} D_{h,1} = S_{h,1}A_h+E_{h,1}} \mod q$$



Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

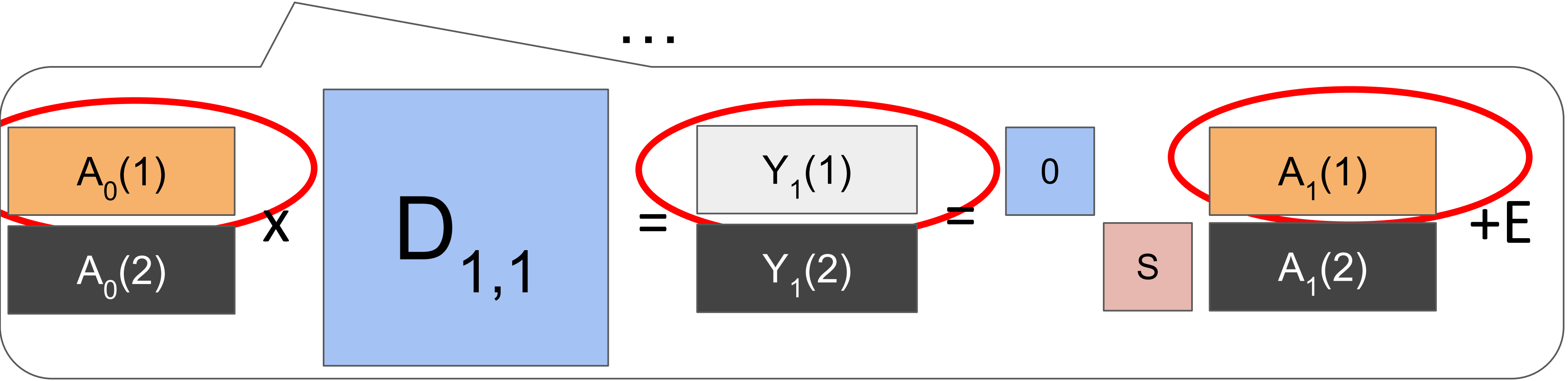
$\boxed{A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}}$, …, $A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1}$ mod q



Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

$$\boxed{A_0 D_{1,1} = S_{1,1}A_1+E_{1,1}}, \ldots, \quad A_{h-1} D_{h,1} = S_{h,1}A_h+E_{h,1} \quad \bmod q$$

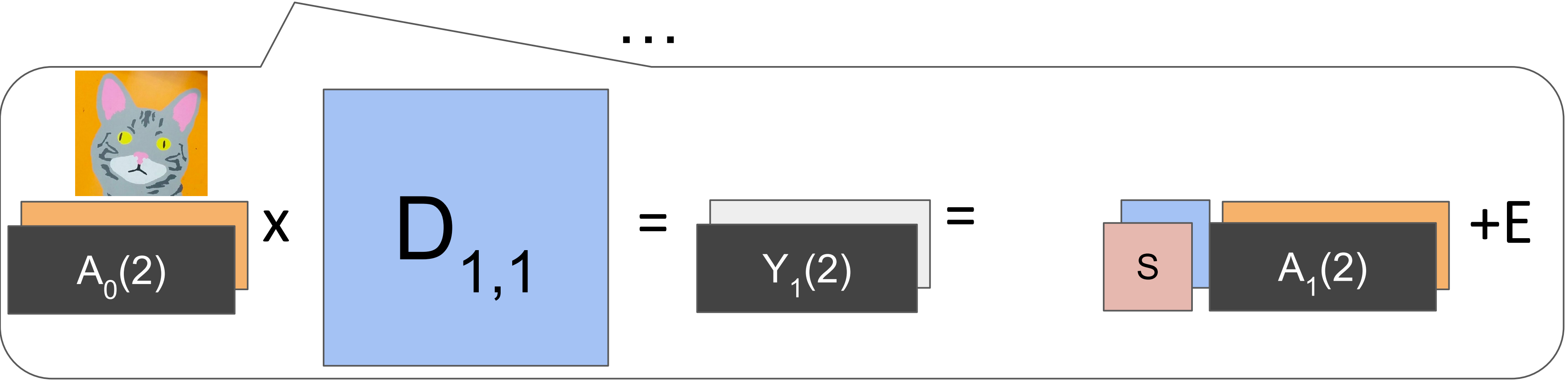


Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

Problem: Now how to deal with the upper matrices?

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

$$\boxed{A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}}, \ldots, \quad A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \text{mod } q$$

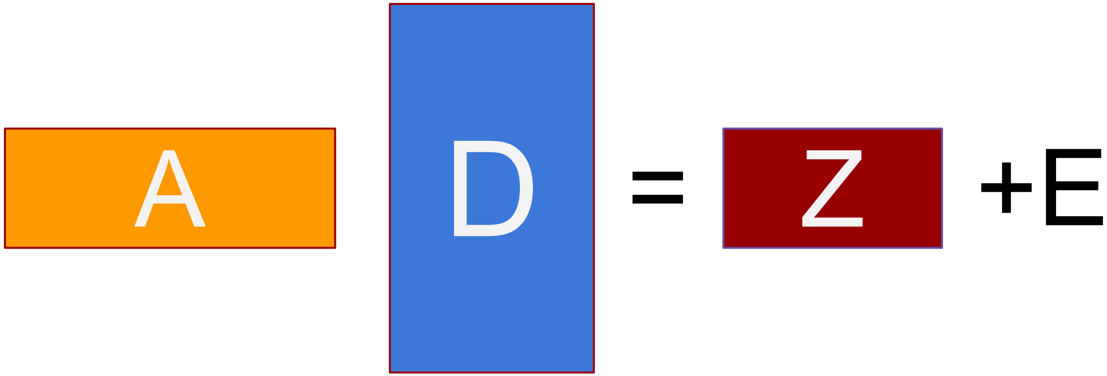


Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

Problem: Now how to deal with the upper matrices?

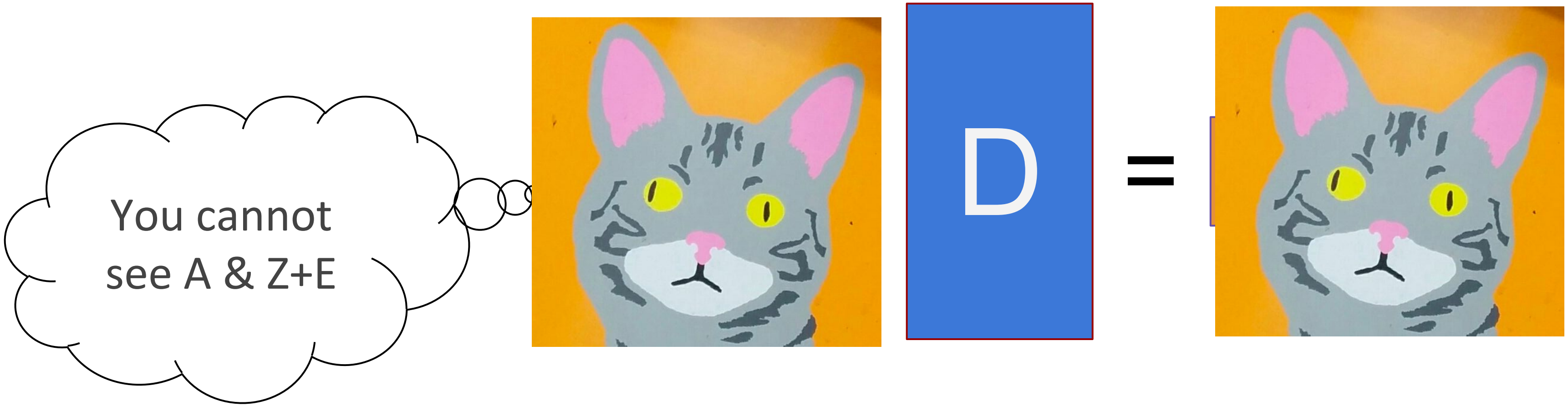
Solution: In the real construction, give out $A_0(1) + A_0(2)$.

Lattice trapdoor Lemma 2:



For any Z, for a uniformly random A, D is the preimage of Z+E.

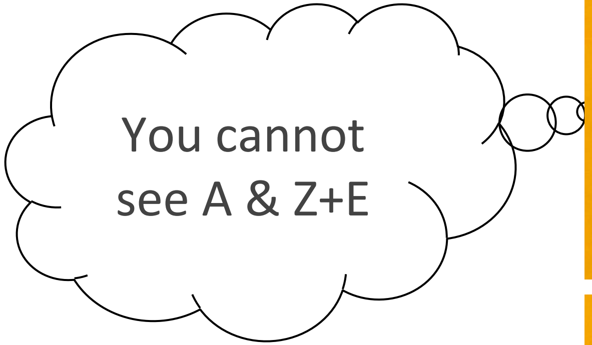
Lattice trapdoor Lemma 2:



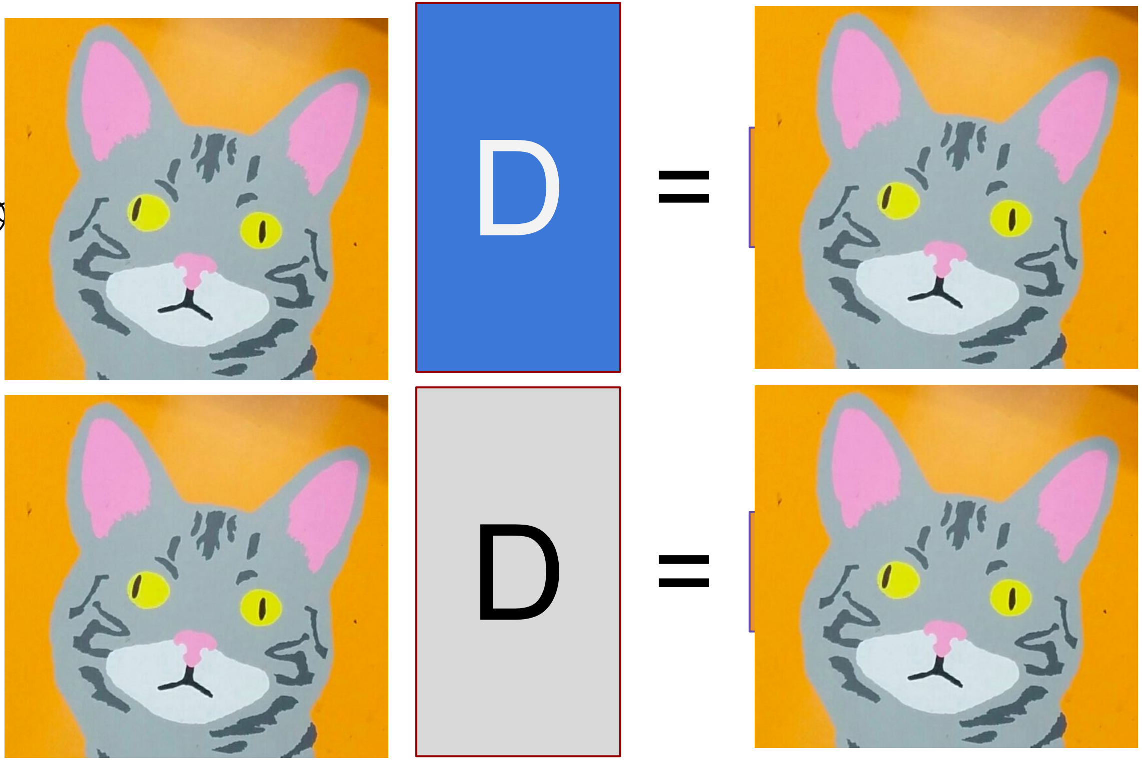
For any Z, for a uniformly random A, D is the preimage of Z+E.

If A & Z+ E is hidden,

Lattice trapdoor Lemma 2:



≈ computational

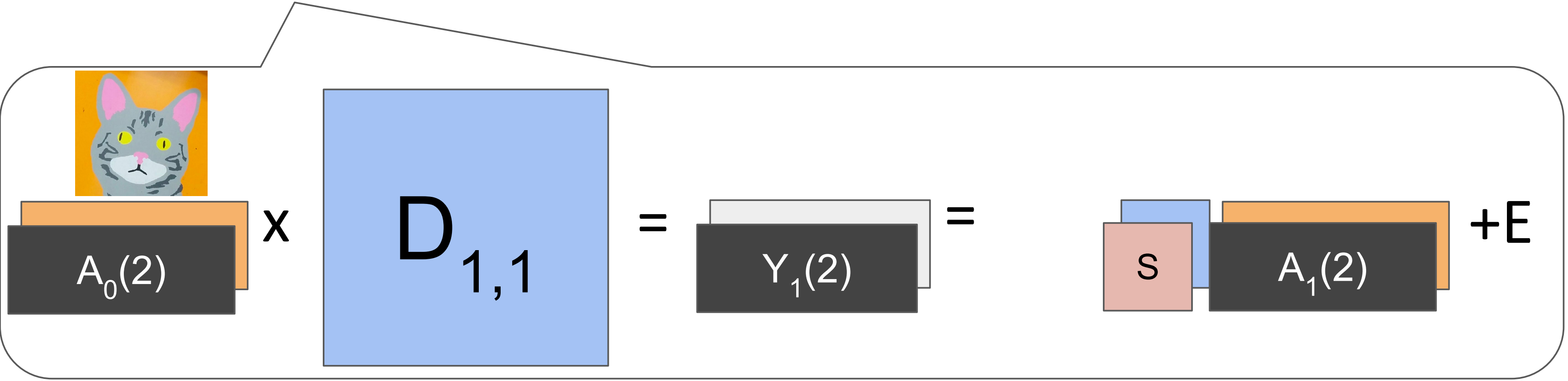


For any Z, for a uniformly random A, D is the preimage of Z+E.

If A & Z+ E is hidden, then D is indistinguishable from random Gaussian.

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

$$\boxed{A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}}, \ldots, \quad A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \text{mod } q$$



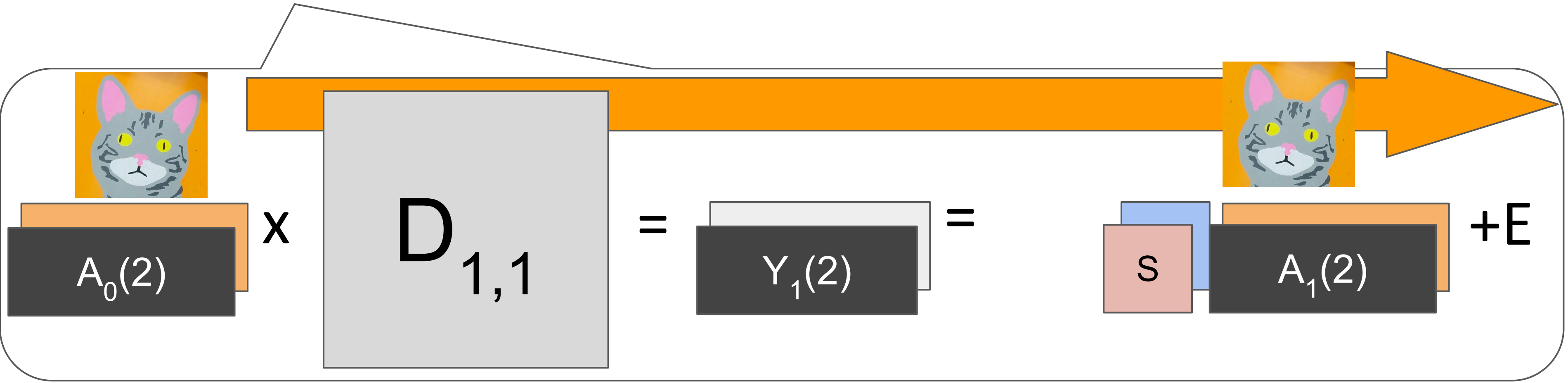
Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

Problem: Now how to deal with the upper matrices?

Solution: In the real construction, give out $A_0(1) + A_0(2)$, + Lemma 2

For possibly low-rank secret matrices: helpful to separate the matrices into (1) and (2)

$$\boxed{A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}},\ \ldots,\quad A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \text{mod } q$$



Use Lemma 1 + use S as public matrix: can close the lower trapdoor all the way back

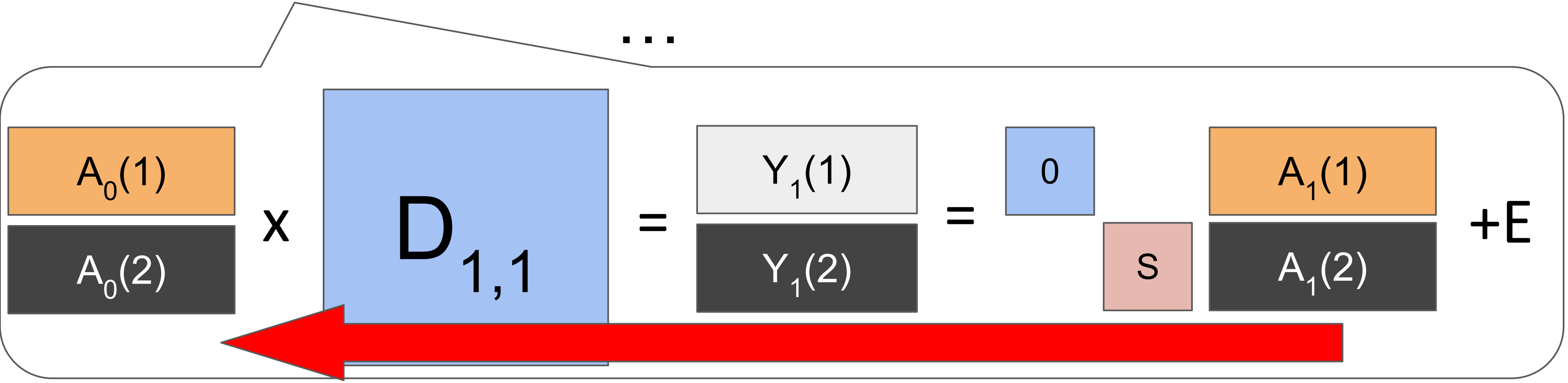
Problem: Now how to deal with the upper matrices?

Solution: In the real construction, give out $A_0(1) + A_0(2)$, + Lemma 2

Replay: the proof for GGH15 + low-rank BP

Replay: the proof for GGH15 + low-rank BP

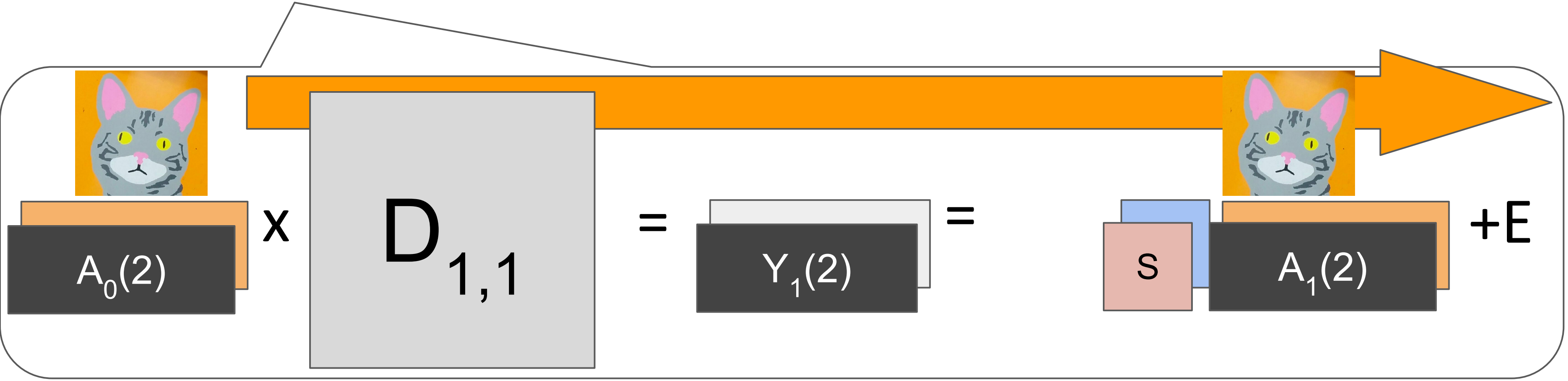
$$\boxed{A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}}, \ldots, \quad A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \text{mod } q$$



First use the lower level random matrices to come left (need new lemma 1)

Replay: the proof for GGH15 + low-rank BP

$$\boxed{A_0 D_{1,1} = S_{1,1} A_1 + E_{1,1}}, \ldots, \quad A_{h-1} D_{h,1} = S_{h,1} A_h + E_{h,1} \quad \text{mod } q$$



First use the lower level random matrices to come left (need new lemma 1)

Then use the upper level “hidden A at the left” to go right (need new lemma 2)

End of the proof for GGH15 + low-rank BP

BOSTON UNIVERSITY

VISA Research

BOSTON UNIVERSITY

VISA Research

BOSTON UNIVERSITY

VISA Research

Q: What about the other cases without a proof from LWE?

A: Hmm … some of them can be broken.

New attack on iO candidates based on GGH15.

With a very simple attack algorithm

New attack on iO candidates based on GGH15.

With a very simple attack algorithm:

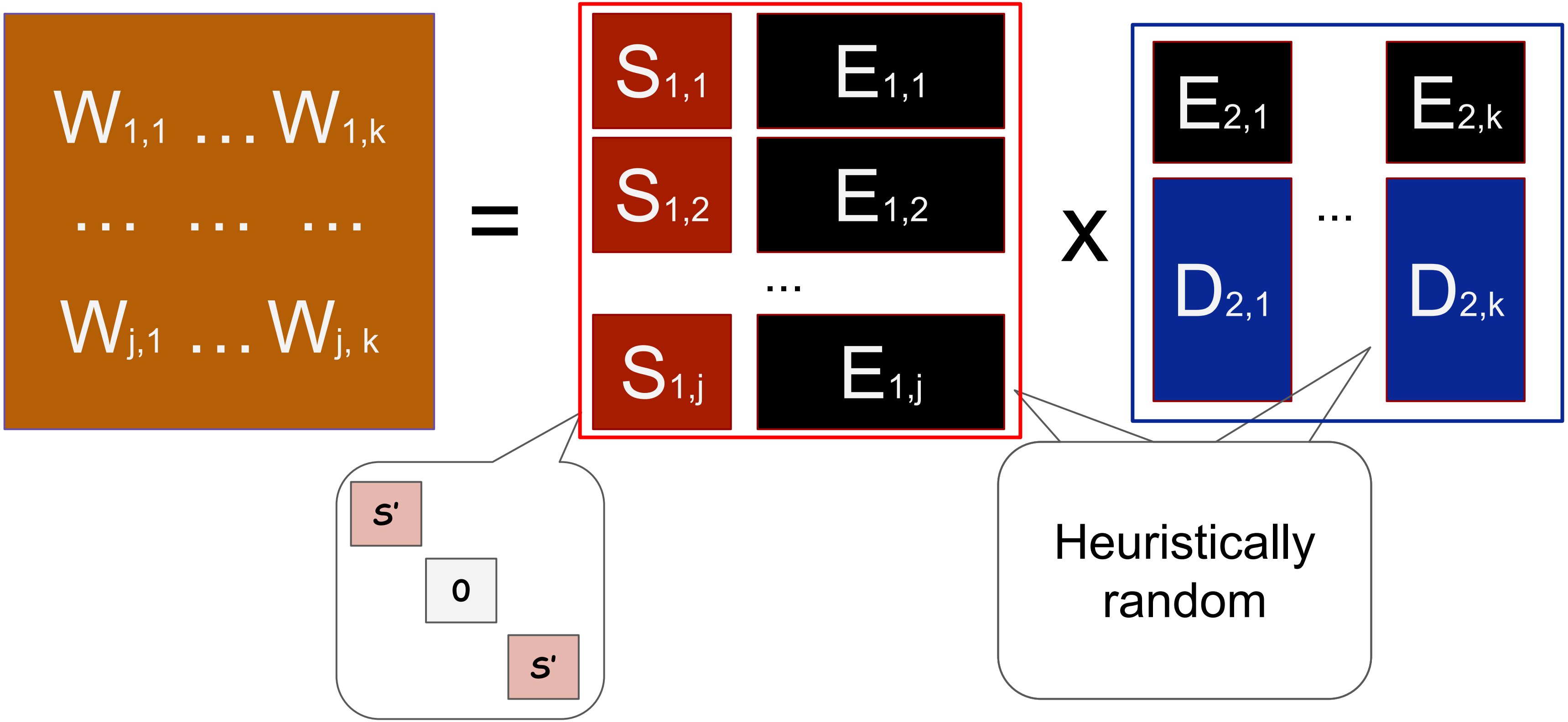
First compute a matrix,

$$\begin{matrix} W_{1,1} & \ldots & W_{1,k} \\ \ldots & \ldots & \ldots \\ W_{j,1} & \ldots & W_{j,k} \end{matrix} = \text{Results on many inputs that eval to small}$$

New attack on iO candidates based on GGH15.

With a very simple attack algorithm:

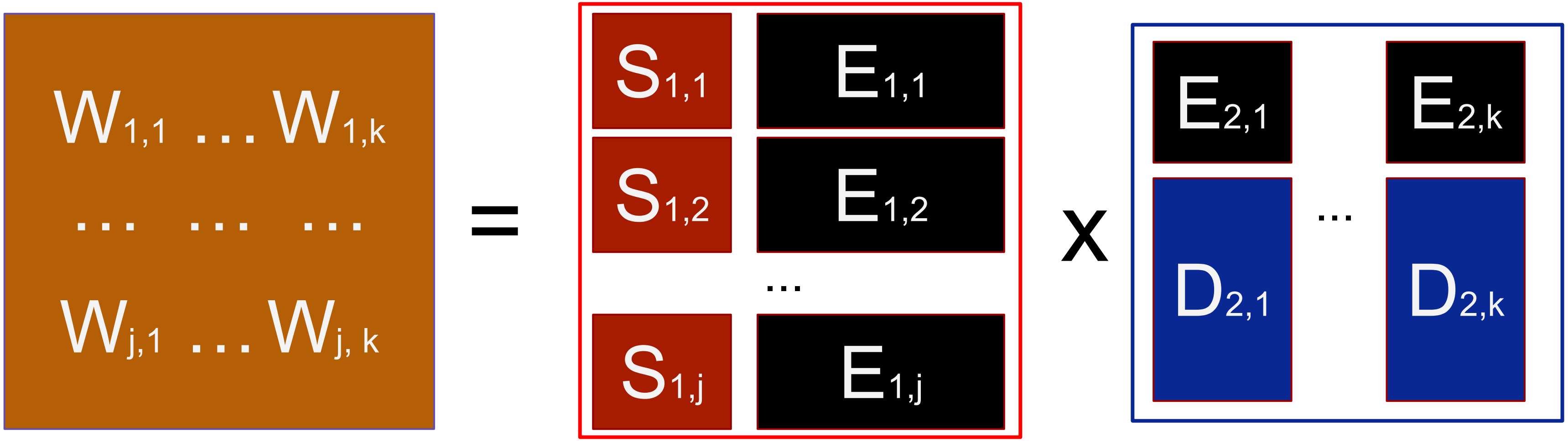
First compute a matrix, then compute the rank of the matrix.



New attack on iO candidates based on GGH15.

With a very simple attack algorithm:

First compute a matrix, then compute the rank of the matrix.



The analysis is quite involved, especially for the extension to non-input-partitioning BPs.

[code] https://github.com/wildstrawberry/cryptanalysesBPobfuscators/blob/master/ggh15analysis.sage

Almost done ...

- Proofs: Introducing new lattice toolkits;
  leads to new PCPRFs and lockable obfuscation for non-perm BPs.

- Attacks: New attacks on the iO candidates.

Wait, what about
witness encryption??

Almost done ...

- Proofs: Introducing new lattice toolkits;
  leads to new PCPRFs and lockable obfuscation for non-perm BPs.

- Attacks: New attacks on the iO candidates.

- Candidates:
  > Witness encryption: read-once BP, the simplest instantiation of GLW14 on GGH15 (removing all the unnecessary parts), *“a stone throw”* from the provable case.

Almost done ...

- Proofs: Introducing new lattice toolkits;
  leads to new PCPRFs and lockable obfuscation for non-perm BPs.

- Attacks: New attacks on the iO candidates.

- Candidates:
  > Witness encryption: read-once BP, the simplest instantiation of GLW14 on GGH15 (removing all the unnecessary parts), *"a stone throw"* from the provable case.
  > iO: read super-constant time BP (merely a demonstration of what is not covered by the attack).

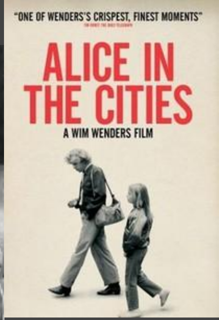
Other related works & Implications

The lattice lemmas appear in the concurrent work of [Goyal, Koppula, Waters 18] that builds traitor tracing from LWE.

[Bartusek, Guan, Ma, Zhandry] limitation of the attacks on GGH15-based iO candidates.

One of the future direction:
Build applications from multilinear maps with “slots”
=> instantiate using GGH15 with diagonal matrices,
see if there is a chance of proving from LWE

Only
25 mins?



Thanks for your time!

GGH15 Beyond Permutation Branching Programs:
Proofs, Attacks, and Candidates

https://eprint.iacr.org/2018/360