Two-Round Threshold Signature from Algebraic One-More Learning with Errors

Thomas Espitau
(PQShield)

Shuichi Katsumata
(PQShield/AIST)

Kaoru Takemure
(PQShield/AIST)

Our Lattice-based Threshold Signature Scheme

- Signing Protocol

  👉 **2-Round with Offline-Online Efficiency**

- Security

  👉 **New Assumption** : **Algebraic One-More MLWE**

- Efficiency

  Signature Size ≈ 11 KB,

  Online Communication Cost ≈ 14 KB

Background

T-out-of-N Threshold Signatures (Key Generation)

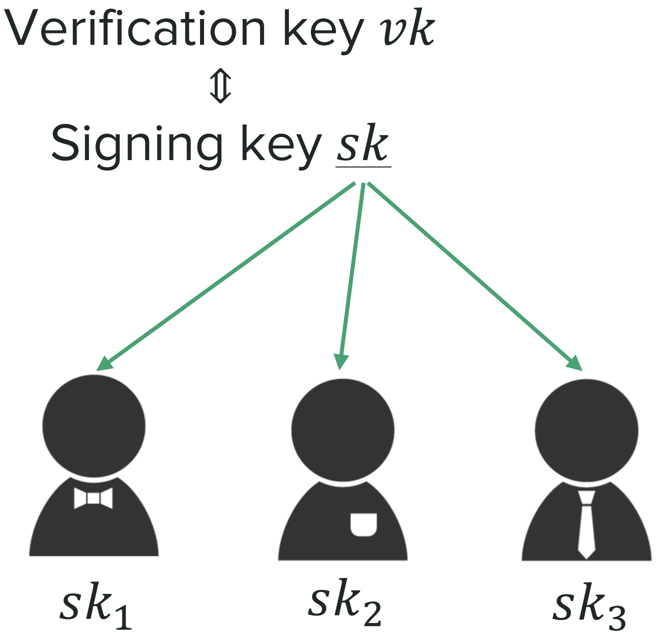
Verification key vk

$\Updownarrow$

Signing key sk

※2-out-of-3

T-out-of-N Threshold Signatures (Key Generation)



Verification key vk

$\Updownarrow$

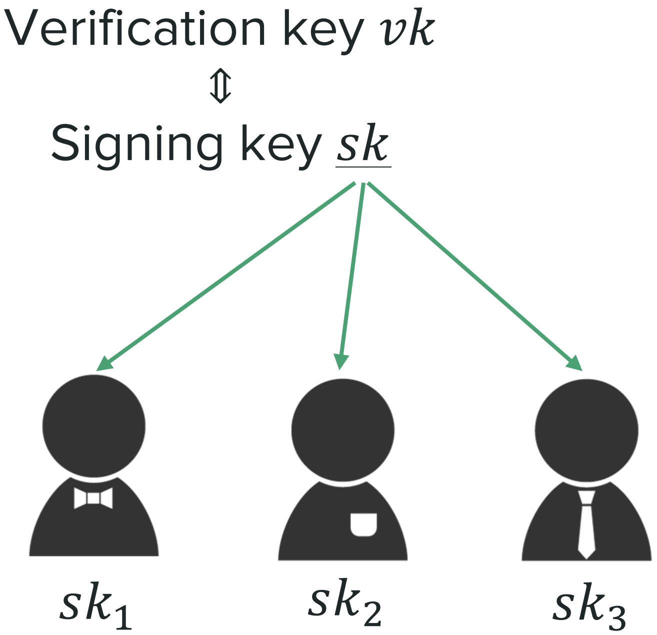
Signing key $\underline{sk}$

sk_1 sk_2 sk_3

- T or more key shares reconstruct sk

※2-out-of-3

T-out-of-N Threshold Signatures (Key Generation)



※2-out-of-3

- T or more key shares reconstruct sk

- No user knows sk
- Less than T key shares leak no information about sk

※We assume that a trusted party executes distributed key generation as well as [BCK+22,dPKM+24] etc.

T-out-of-N Threshold Signatures (Signing)

※2-out-of-3

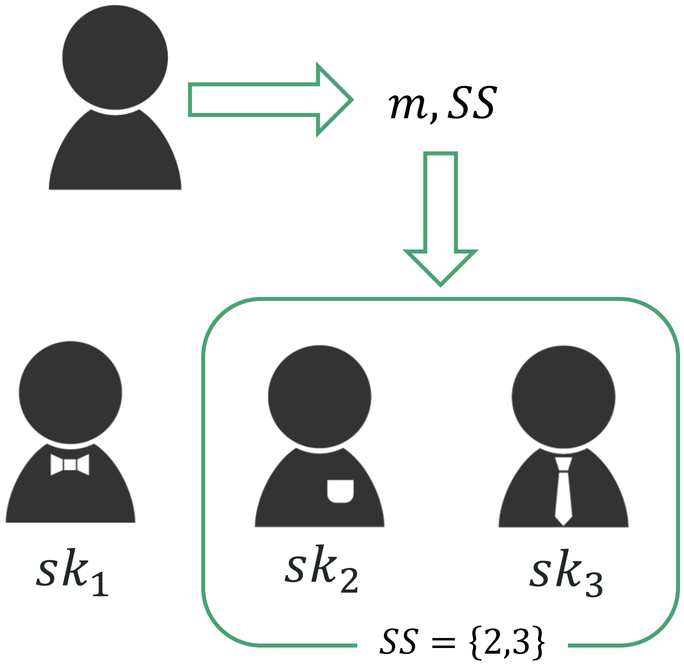
T-out-of-N Threshold Signatures (Signing)

“**Multi-Round**” Signing Protocol

General Procedure:

※2-out-of-3

T-out-of-N Threshold Signatures (Signing)



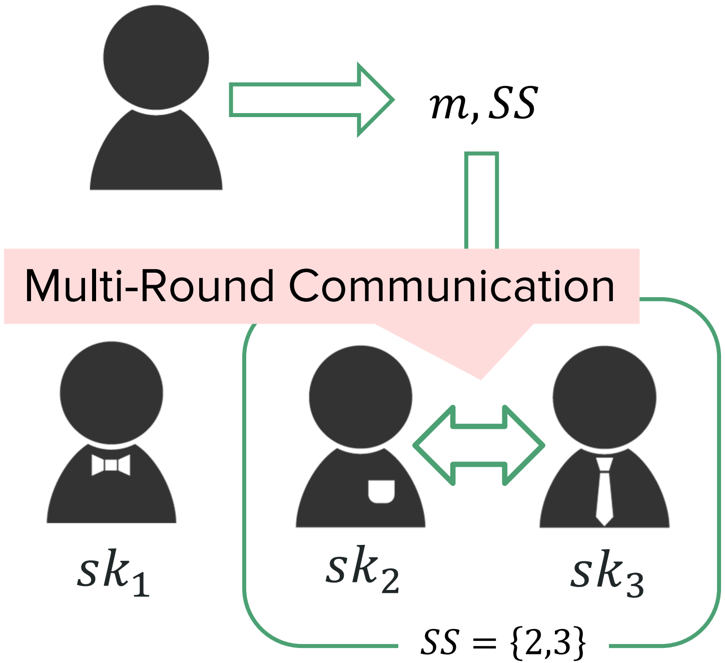
※2-out-of-3

“**Multi-Round**” Signing Protocol

General Procedure:

1. One decides message m and signer set SS

T-out-of-N Threshold Signatures (Signing)



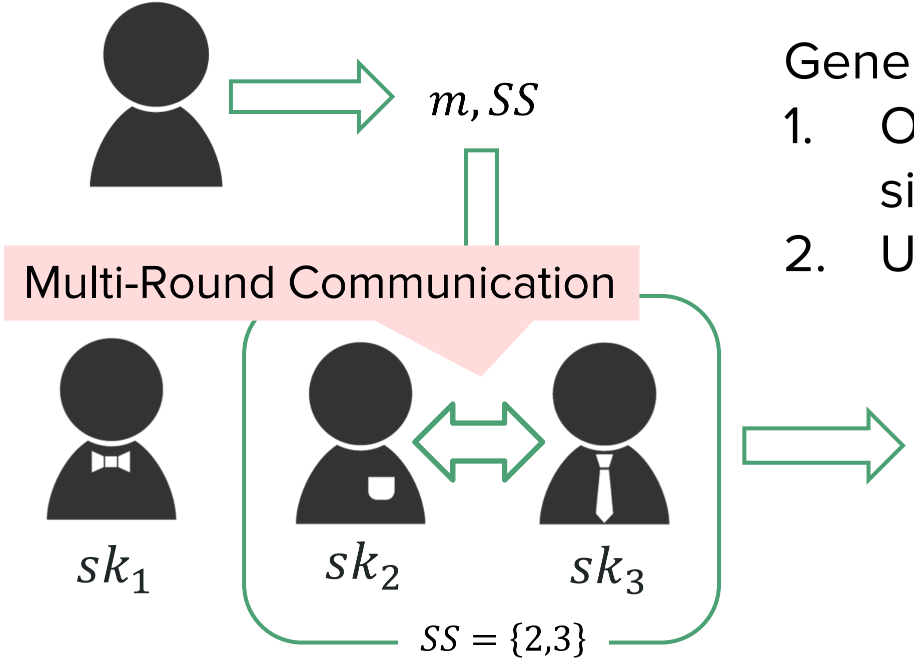
※2-out-of-3

“**Multi-Round**” Signing Protocol

General Procedure:

1. One decides message m and signer set SS
2. Users in SS execute signing protocol

T-out-of-N Threshold Signatures (Signing)



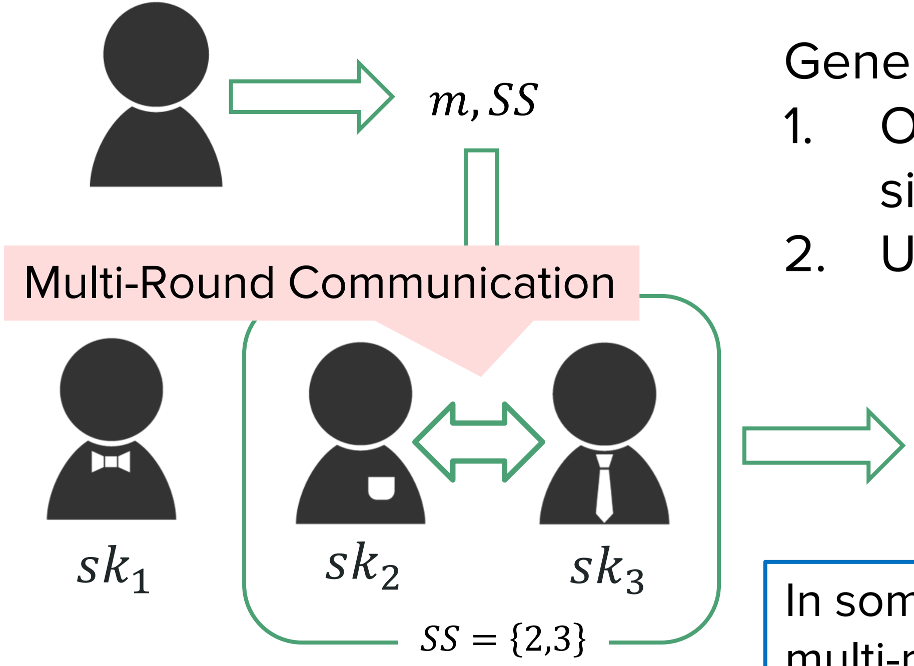
※2-out-of-3

“**Multi-Round**” Signing Protocol

General Procedure:

1. One decides message m and signer set SS
2. Users in SS execute signing protocol

T-out-of-N Threshold Signatures (Signing)



※2-out-of-3

"Multi-Round" Signing Protocol

General Procedure:

1. One decides message m and signer set SS
2. Users in SS execute signing protocol

In some restricted environments, multi-round is performance bottleneck 😞

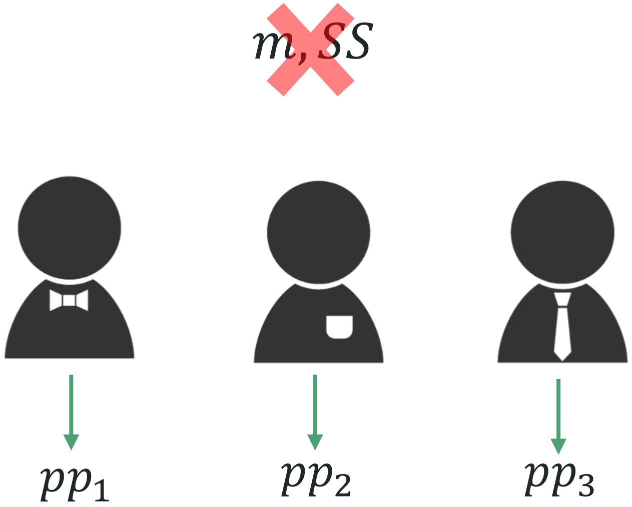
Nice Property: Offline-Online Efficiency

First Round: Pre-processing Phase

Second Round: Signing Phase

Nice Property: Offline-Online Efficiency

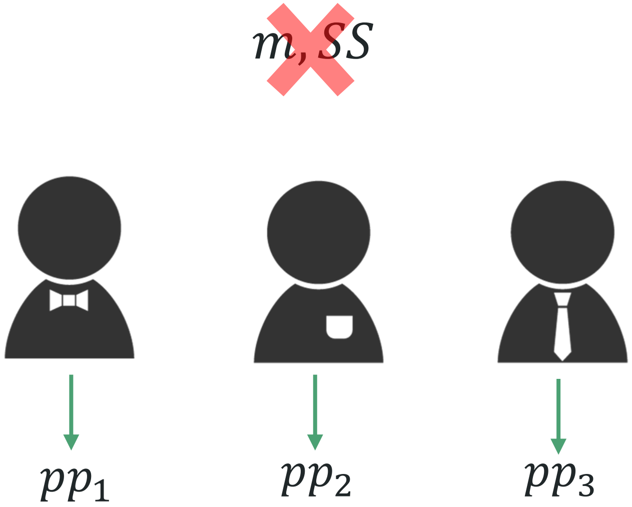
First Round: Pre-processing Phase



Second Round: Signing Phase

Nice Property: Offline-Online Efficiency

First Round: Pre-processing Phase

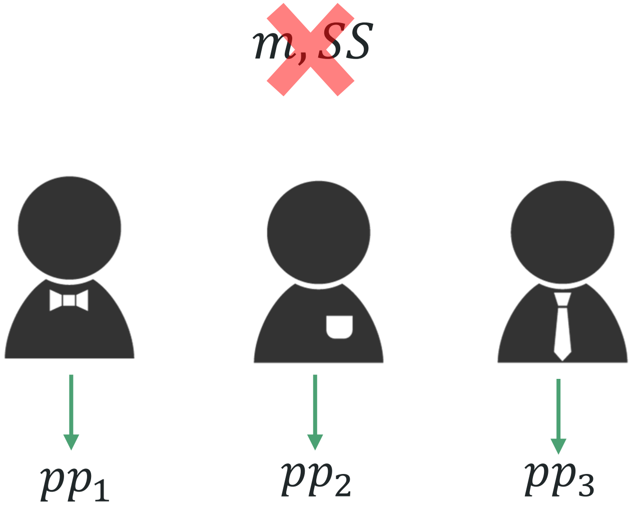


Can be executed in advance.

Second Round: Signing Phase

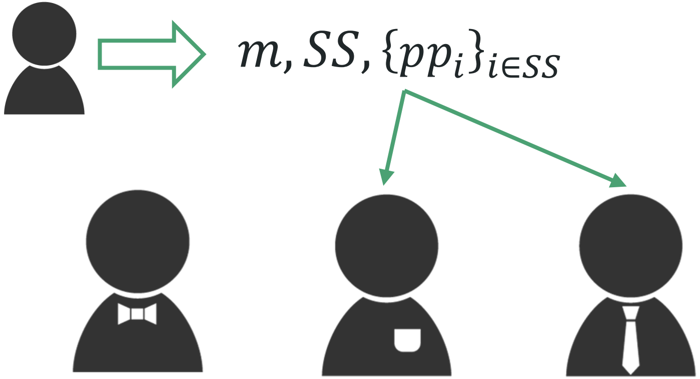
Nice Property: Offline-Online Efficiency

First Round: Pre-processing Phase



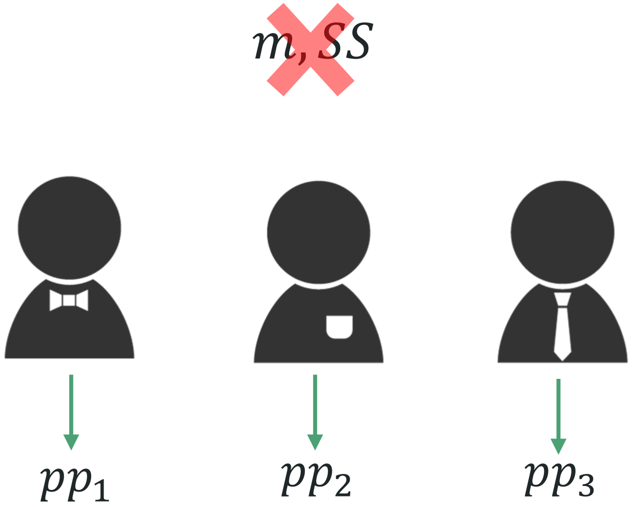
Can be executed in advance.

Second Round: Signing Phase



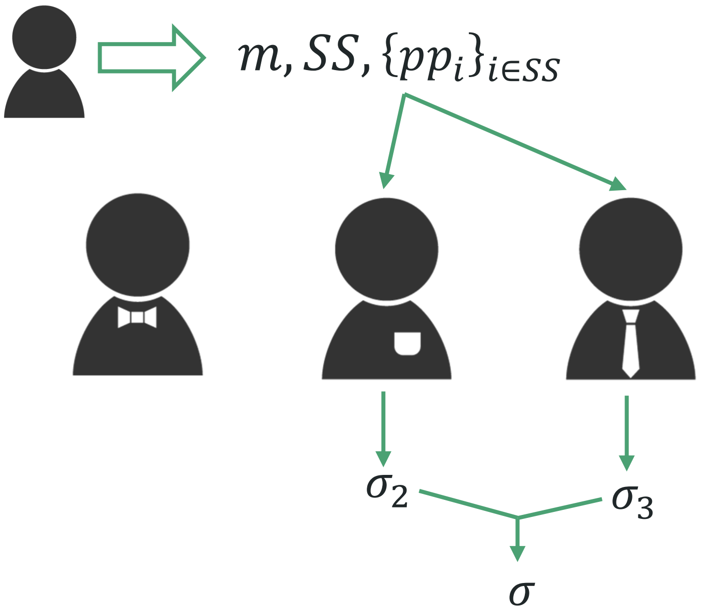
Nice Property: Offline-Online Efficiency

First Round: Pre-processing Phase



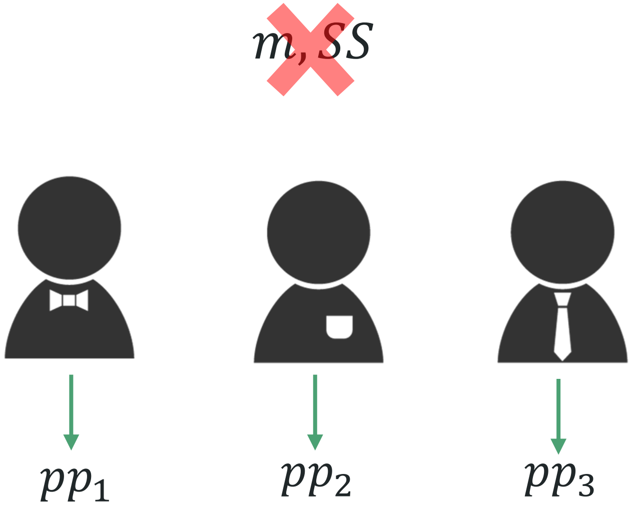
Can be executed in advance.

Second Round: Signing Phase



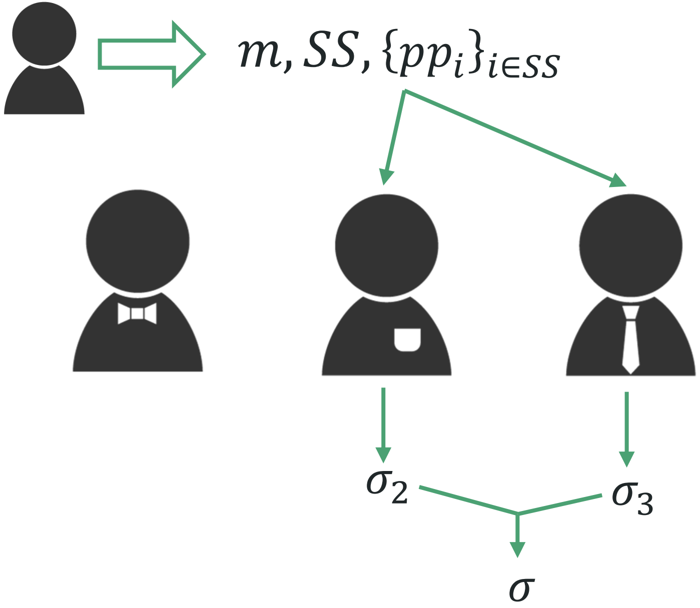
Nice Property: Offline-Online Efficiency

First Round: Pre-processing Phase



Can be executed in advance.

Second Round: Signing Phase



Non-interactive!

Recent Breakthrough

Previous: [BKP13], [BGG+18], [ASY22], [GKS23].

They rely on heavy tools like FHE and HTDC.

Very Recent:

Practical Lattice-based TS : **Threshold Raccoon (TRaccoon)** [EC:dPKM+24]

- w/o heavy tools
- |Sig| ≈ 13 KB, Comm. Cost ≈ 40 KB

TRaccoon [EC:dPKM+24]

Lattice-based TS based on **3-Round** DL-based TS Sparkle [CKM23]

TRaccoon [EC:dPKM+24]

Lattice-based TS based on **3-Round** DL-based TS Sparkle [CKM23]

Classical Setting

Sparkle [CKM23]:

- ➢ Schnorr signature
  ⇒ Discrete Log (DL)
  ⇒ Built from Fiat-Shamir Transform
- ➢ 3-round signing protocol

TRaccoon [EC:dPKM+24]

Lattice-based TS based on **3-Round** DL-based TS Sparkle [CKM23]

Classical Setting

Sparkle [CKM23]:

- Schnorr signature
  ⇒ Discrete Log (DL)
  ⇒ Built from Fiat-Shamir Transform
- 3-round signing protocol

Lattice Setting

TRaccoon:

- Raccoon signature (Dilithium-like)
  ⇒ MLWE and MSIS
  ⇒ Built from Fiat-Shamir Transform

TRaccoon [EC:dPKM+24]

Lattice-based TS based on **3-Round** DL-based TS Sparkle [CKM23]

Classical Setting

Sparkle [CKM23]:

- Schnorr signature
  ⇒ Discrete Log (DL)
  ⇒ Built from Fiat-Shamir Transform
- 3-round signing protocol

Lattice Setting

TRaccoon:

- Raccoon signature (Dilithium-like)
  ⇒ MLWE and MSIS
  ⇒ Built from Fiat-Shamir Transform
- 3-round signing protocol
  ⇒ **Masking Technique**

Lattice specific technique

Open Problem

3-round

Classical Setting

Sparkle [CKM23]

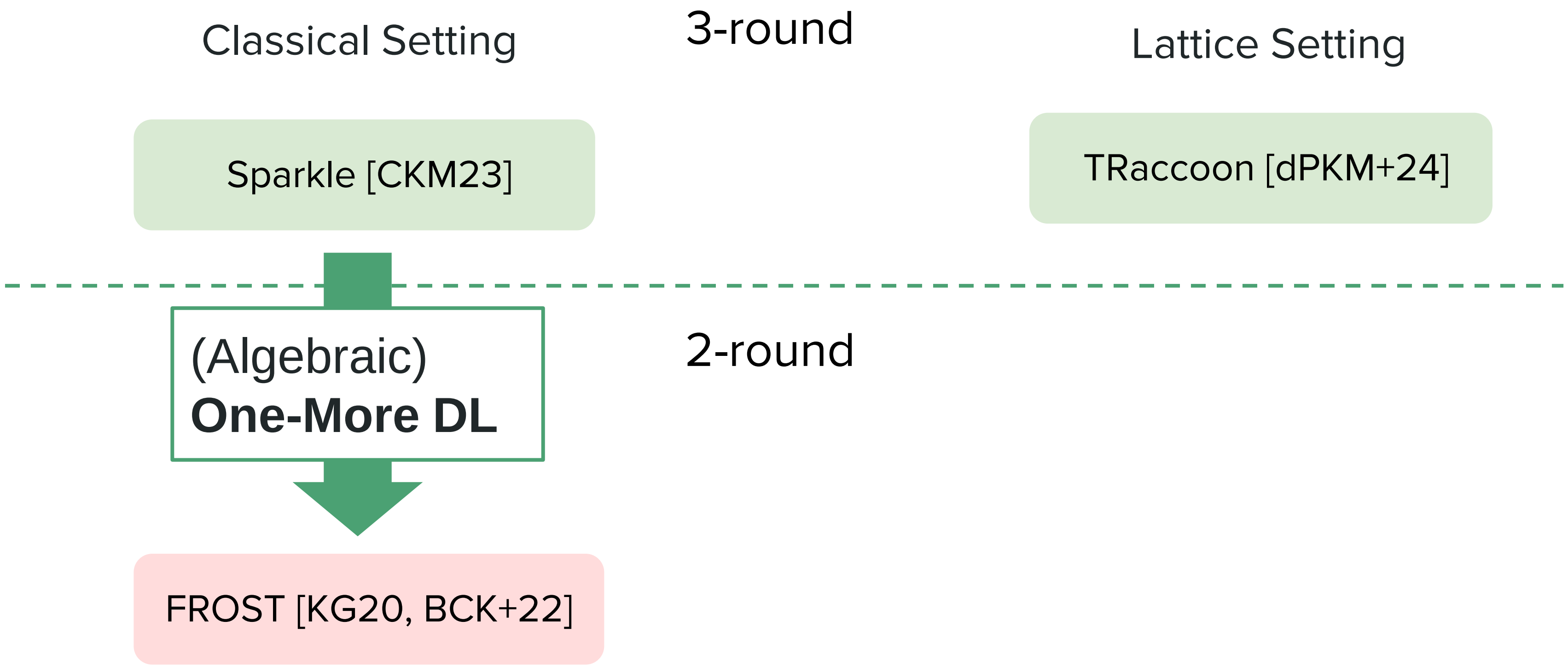
Lattice Setting

TRaccoon [dPKM+24]

Open Problem

	Classical Setting	Lattice Setting
3-round	Sparkle [CKM23]	TRaccoon [dPKM+24]
2-round	FROST [KG20, BCK+22]	

Open Problem



Classical Setting

3-round

Lattice Setting

Sparkle [CKM23]

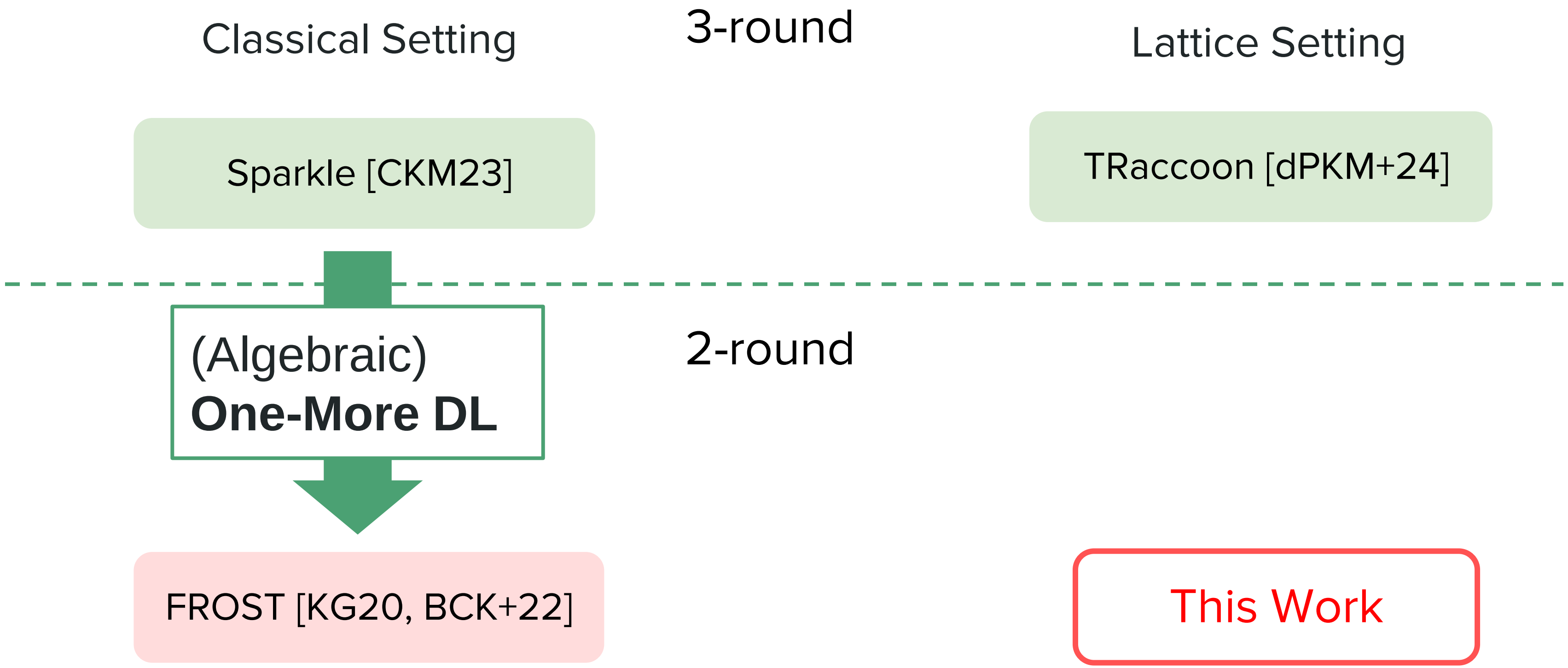
TRaccoon [dPKM+24]

(Algebraic)
One-More DL

2-round

FROST [KG20, BCK+22]

Open Problem



Classical Setting

3-round

Lattice Setting

Sparkle [CKM23]

TRaccoon [dPKM+24]

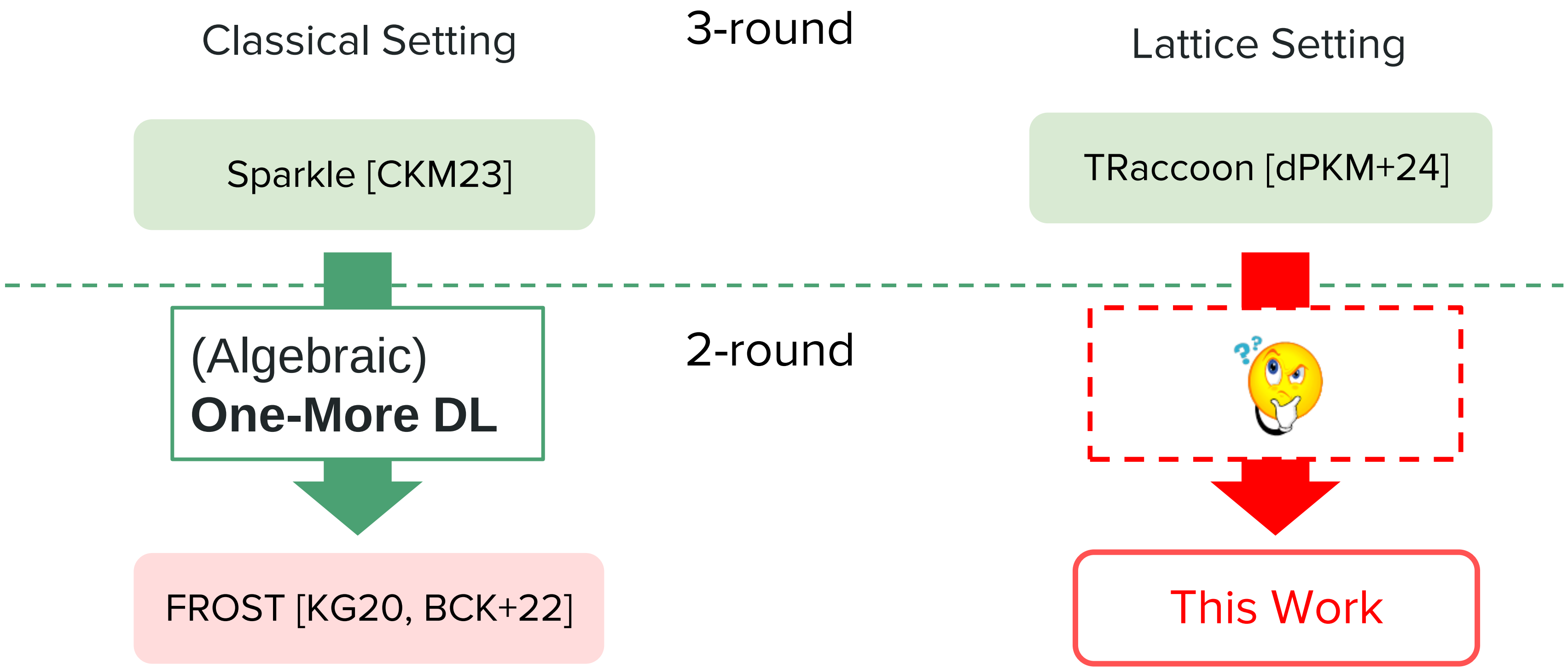
(Algebraic)
One-More DL

2-round

FROST [KG20, BCK+22]

This Work

Open Problem



Classical Setting

3-round

Lattice Setting

Sparkle [CKM23]

TRaccoon [dPKM+24]

(Algebraic)
One-More DL

2-round

FROST [KG20, BCK+22]

This Work

Introducing Algebraic One-More MLWE

One-More DL [BNPS02, BNPS03, BMV08]

Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

$g, X_0, X_1, \dots, X_Q$

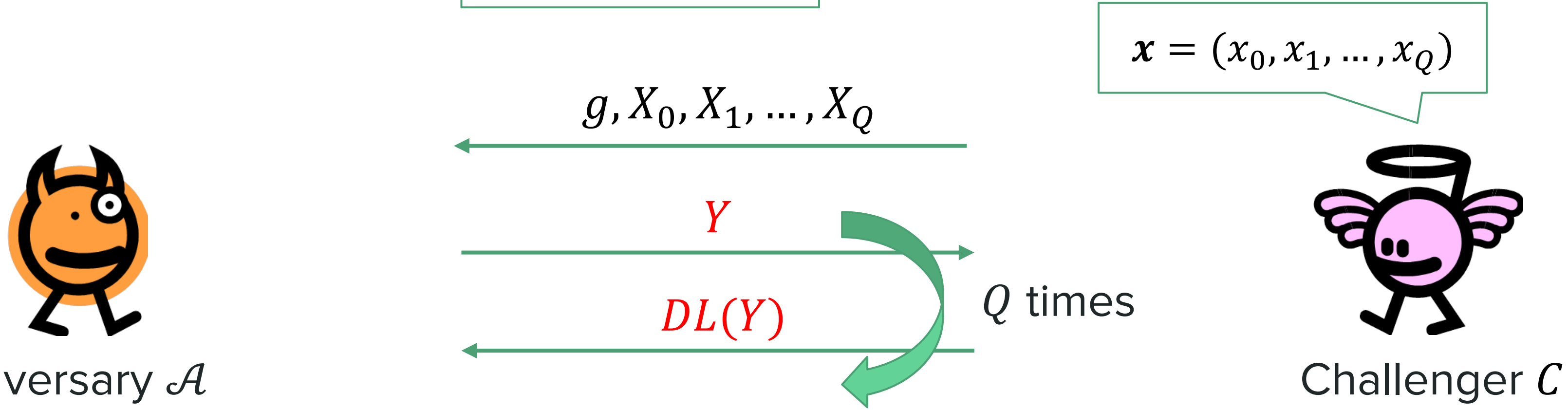
Adversary $\mathcal{A}$

Challenger C

One-More DL [BNPS02, BNPS03, BMV08]

Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$



$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

$g, X_0, X_1, \dots, X_Q$

Y

$DL(Y)$

Q times

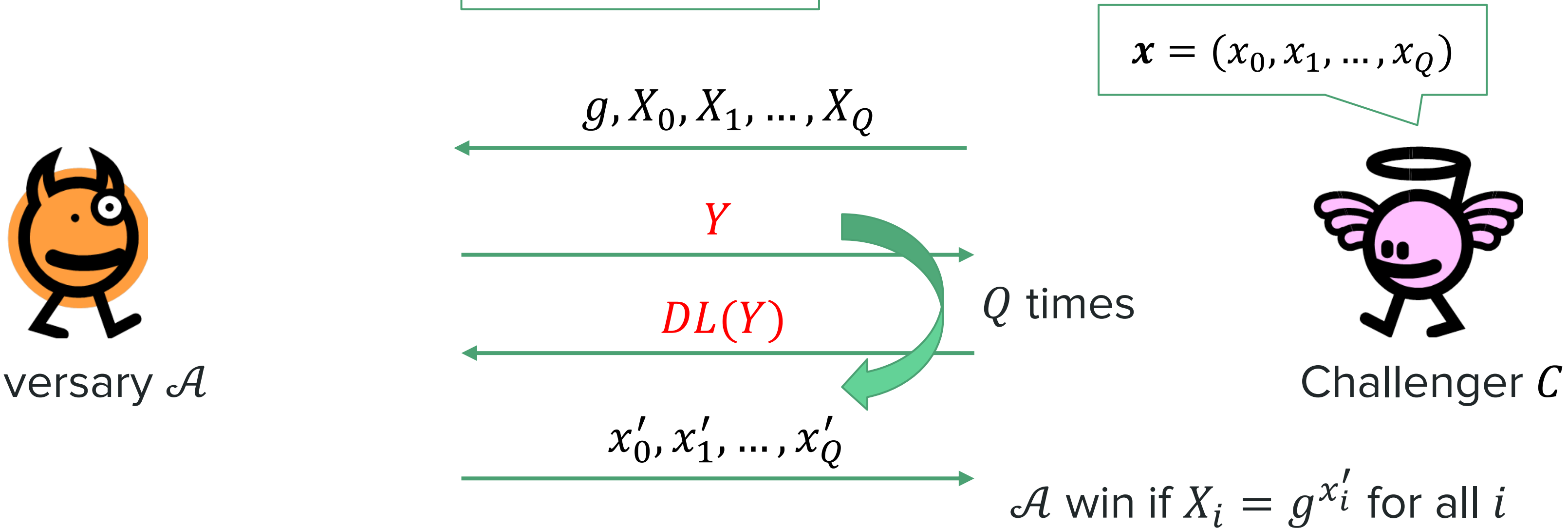
Adversary $\mathcal{A}$

Challenger C

One-More DL [BNPS02, BNPS03, BMV08]

Instance: $g, X_0, X_1, \ldots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$



$\boldsymbol{x} = (x_0, x_1, \ldots, x_Q)$

$g, X_0, X_1, \ldots, X_Q$

Y

$DL(Y)$

Q times

$x'_0, x'_1, \ldots, x'_Q$

Adversary $\mathcal{A}$

Challenger C

$\mathcal{A}$ win if $X_i = g^{x'_i}$ for all i

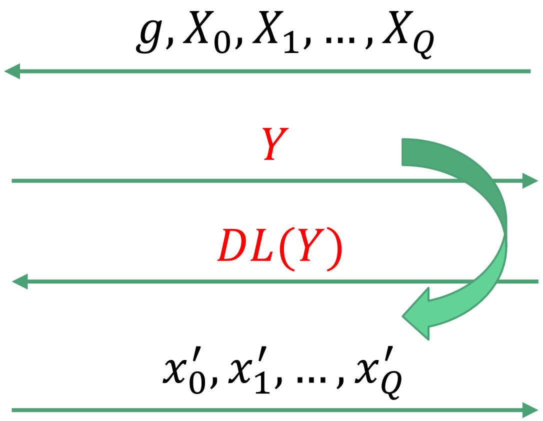
One-More DL [BNPS02, BNPS03, BMV08]

Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i}, \ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

Adversary $\mathcal{A}$



$g, X_0, X_1, \dots, X_Q$

Y

$DL(Y)$

Q times

$x'_0, x'_1, \dots, x'_Q$

Challenger C

$\mathcal{A}$ win if $X_i = g^{x'_i}$ for all i

C ***has to solve*** $DL(Y)$ to answer queries ⇒ Unfalsifiable

Algebraic One-More DL [NRS21]

$\Rightarrow$ $\mathcal{A}$ is allowed to make only **algebraic queries.**

Algebraic One-More DL [NRS21]

⇒ $\mathcal{A}$ is allowed to make only **algebraic queries.**

Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

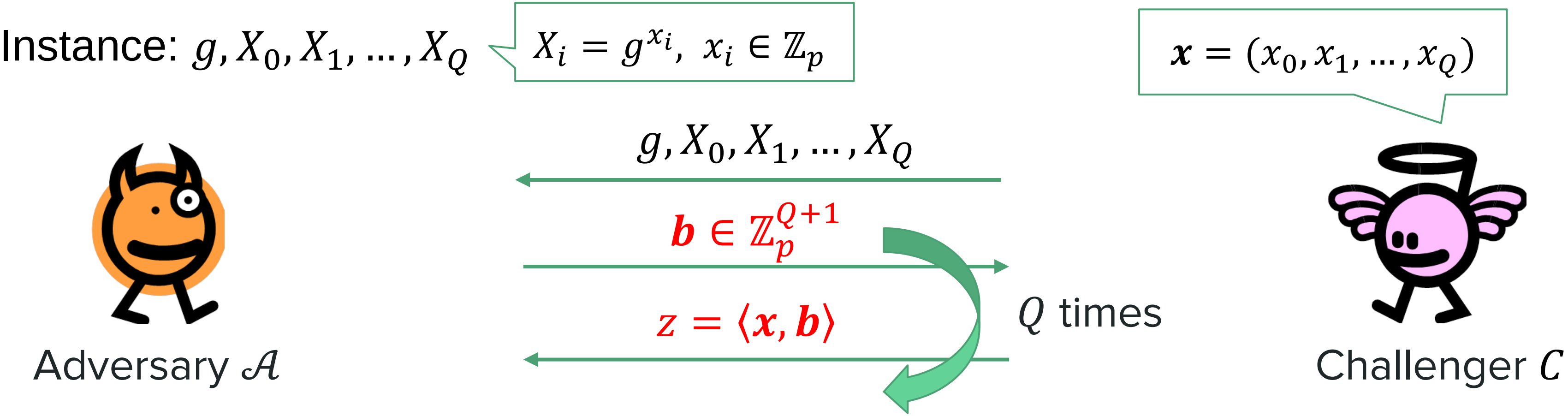
$$\xleftarrow{\quad g, X_0, X_1, \dots, X_Q \quad}$$

Adversary $\mathcal{A}$

Challenger $\mathcal{C}$

Algebraic One-More DL [NRS21]

⇒ $\mathcal{A}$ is allowed to make only **algebraic queries.**



Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

$g, X_0, X_1, \dots, X_Q$

$\boldsymbol{b} \in \mathbb{Z}_p^{Q+1}$

$z = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$

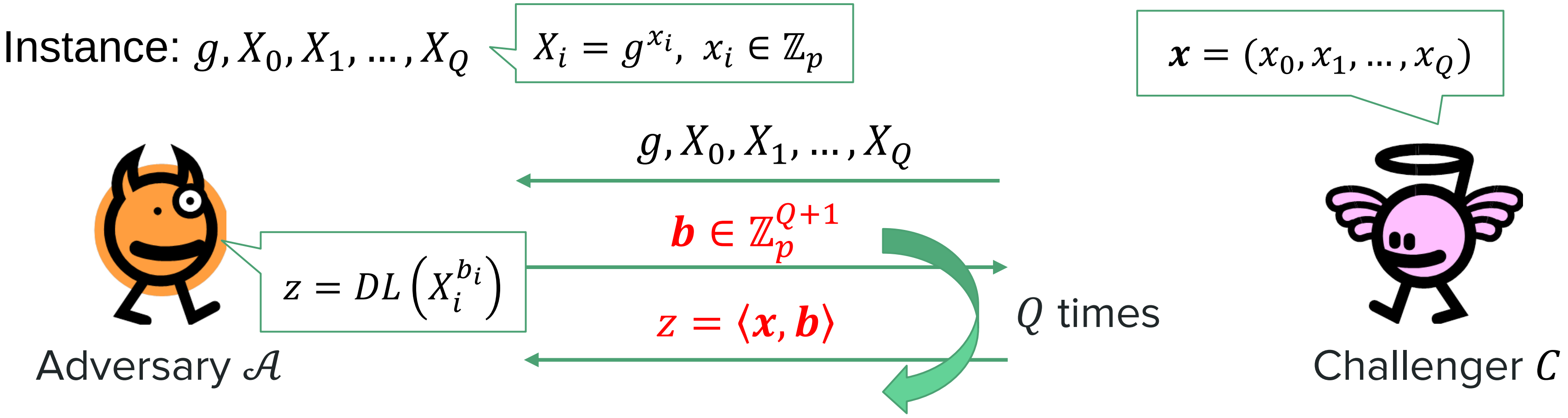
Q times

Adversary $\mathcal{A}$

Challenger C

Algebraic One-More DL [NRS21]

⇒ $\mathcal{A}$ is allowed to make only **algebraic queries.**



Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

$g, X_0, X_1, \dots, X_Q$

$\boldsymbol{b} \in \mathbb{Z}_p^{Q+1}$

$z = DL\left(X_i^{b_i}\right)$

$z = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$

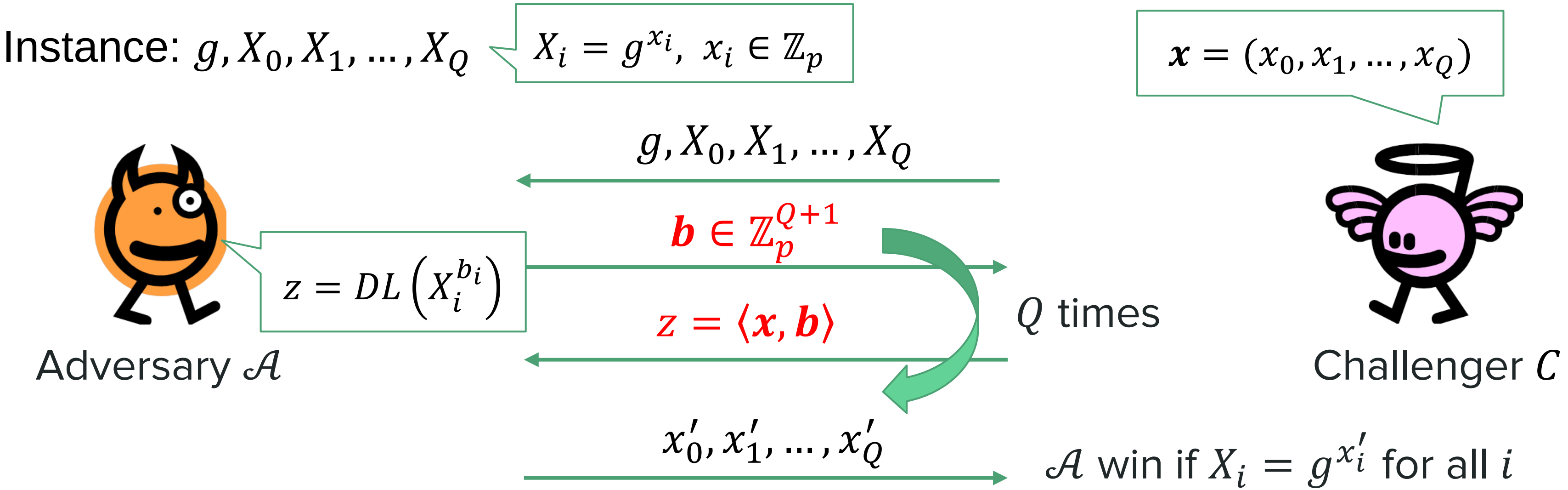
Q times

Adversary $\mathcal{A}$

Challenger C

Algebraic One-More DL [NRS21]

⇒ $\mathcal{A}$ is allowed to make only **algebraic queries.**



Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

$g, X_0, X_1, \dots, X_Q$

$\boldsymbol{b} \in \mathbb{Z}_p^{Q+1}$

$z = DL\left(X_i^{b_i}\right)$

$z = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$

Q times

Adversary $\mathcal{A}$

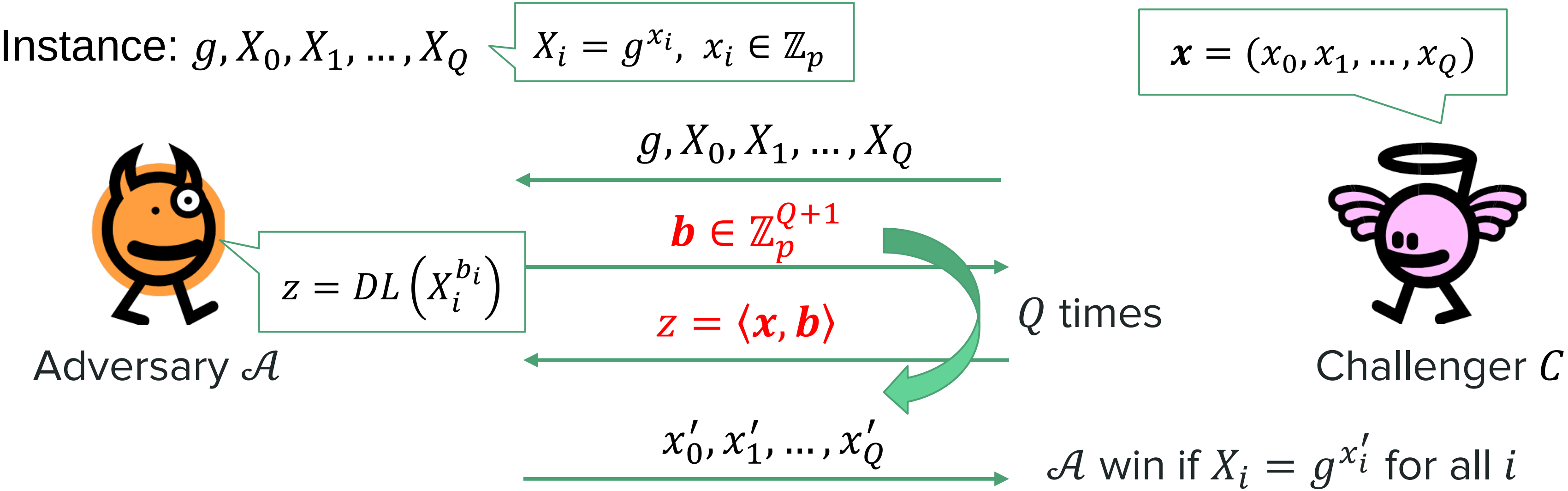
Challenger C

$x'_0, x'_1, \dots, x'_Q$

$\mathcal{A}$ win if $X_i = g^{x'_i}$ for all i

Algebraic One-More DL [NRS21]

⇒ $\mathcal{A}$ is allowed to make only **algebraic queries.**



Instance: $g, X_0, X_1, \dots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \dots, x_Q)$

$g, X_0, X_1, \dots, X_Q$
$\boldsymbol{b} \in \mathbb{Z}_p^{Q+1}$
$z = DL\left(X_i^{b_i}\right)$
$z = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$
Q times
Adversary $\mathcal{A}$
Challenger C
$x_0', x_1', \dots, x_Q'$

$\mathcal{A}$ win if $X_i = g^{x_i'}$ for all i

C ***can compute*** $z = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$ ***efficiently*** to answer queries ⇒ Falsifiable

Algebraic One-More DL [NRS21]

⇒ $\mathcal{A}$ is allowed to make only **algebraic queries.**

Instance: $g, X_0, X_1, \ldots, X_Q$

$X_i = g^{x_i},\ x_i \in \mathbb{Z}_p$

$\boldsymbol{x} = (x_0, x_1, \ldots, x_Q)$

$g, X_0, X_1, \ldots, X_Q$

AOM-DL ⇒ 2-Round TS FROST

Lattice-based variant?

C

$\mathcal{A}$ win if $X_i = g^{x_i}$ for all i

C ***can compute*** $z = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$ ***efficiently*** to answer queries ⇒ Falsifiable

Naive Attempt

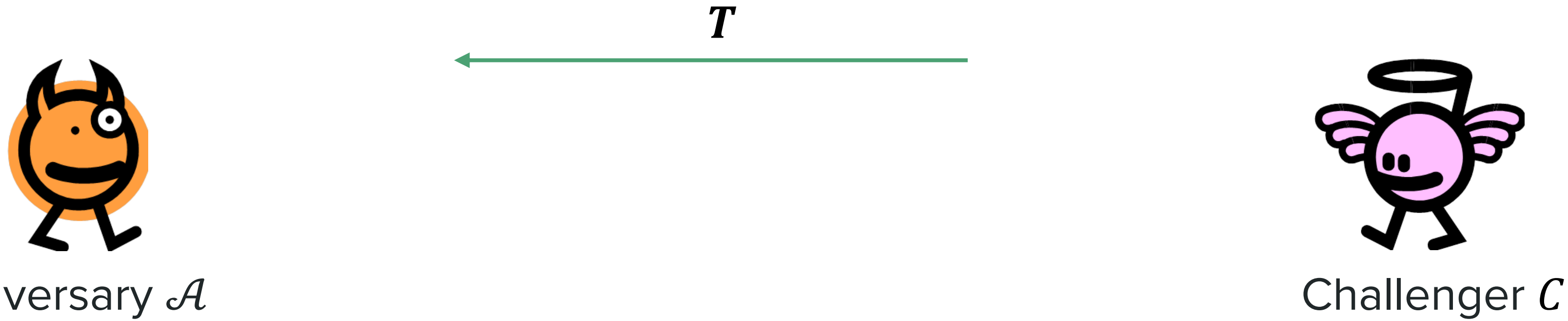
Instance: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{T} = [\boldsymbol{t}_0\boldsymbol{t}_1 \cdots \boldsymbol{t}_Q] \in \mathcal{R}_q^{k\times(Q+1)}$

$$\boxed{\boldsymbol{t}_i} = \boxed{\boldsymbol{A}' \mid \boldsymbol{I}}\,\boxed{\boldsymbol{s}_i}$$

$\boldsymbol{s}_i$ is short

$\boldsymbol{S} = [\boldsymbol{s}_0\boldsymbol{s}_1 \cdots \boldsymbol{s}_Q]$

$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}$



Naive Attempt

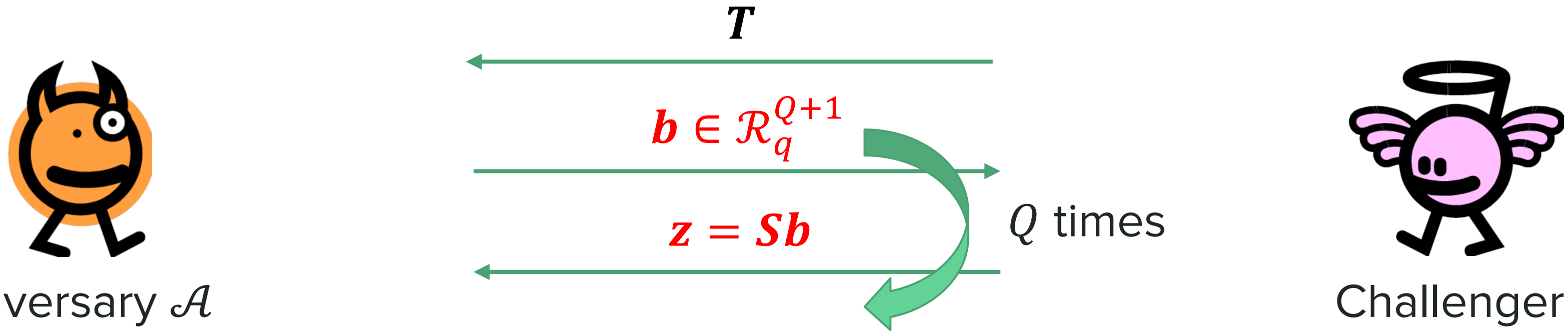
Instance: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1 \cdots \boldsymbol{t}_Q] \in \mathcal{R}_q^{k\times(Q+1)}$

$\boldsymbol{t}_i = \begin{bmatrix} \boldsymbol{A}' & \boldsymbol{I} \end{bmatrix} \boldsymbol{s}_i$

$\boldsymbol{s}_i$ is short

$\boldsymbol{S} = [\boldsymbol{s}_0 \boldsymbol{s}_1 \cdots \boldsymbol{s}_Q]$

$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}$



$\boldsymbol{T}$
$\boldsymbol{b} \in \mathcal{R}_q^{Q+1}$
$\boldsymbol{z} = \boldsymbol{S}\boldsymbol{b}$
Q times

Adversary $\mathcal{A}$

Challenger C

Naive Attempt

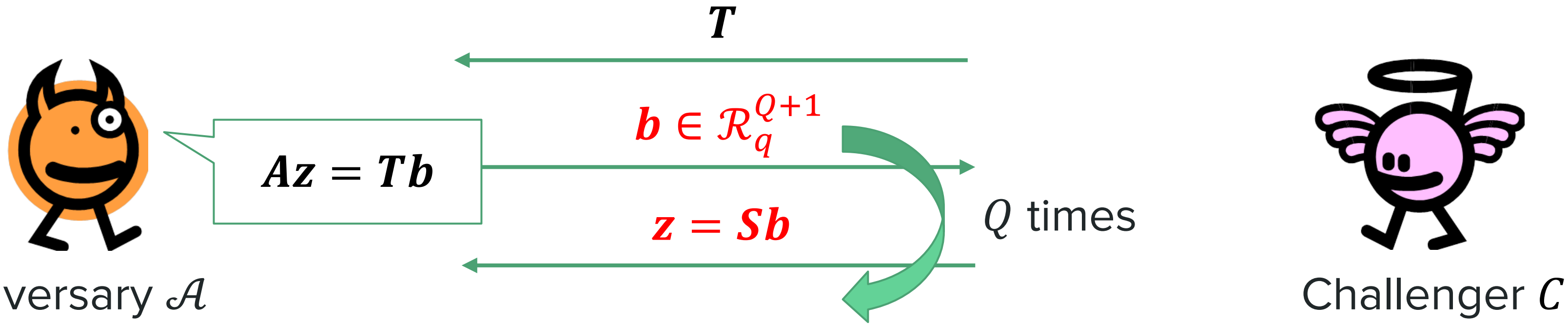
Instance: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1 \cdots \boldsymbol{t}_Q] \in \mathcal{R}_q^{k\times(Q+1)}$

$$\boldsymbol{t}_i = \begin{bmatrix} \boldsymbol{A}' & \boldsymbol{I} \end{bmatrix} \boldsymbol{s}_i$$

$\boldsymbol{s}_i$ is short

$\boldsymbol{S} = [\boldsymbol{s}_0 \boldsymbol{s}_1 \cdots \boldsymbol{s}_Q]$

$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}$



$\boldsymbol{T}$

$\boldsymbol{b} \in \mathcal{R}_q^{Q+1}$

$\boldsymbol{z} = \boldsymbol{S}\boldsymbol{b}$

Q times

$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{T}\boldsymbol{b}$

Adversary $\mathcal{A}$

Challenger C

Naive Attempt

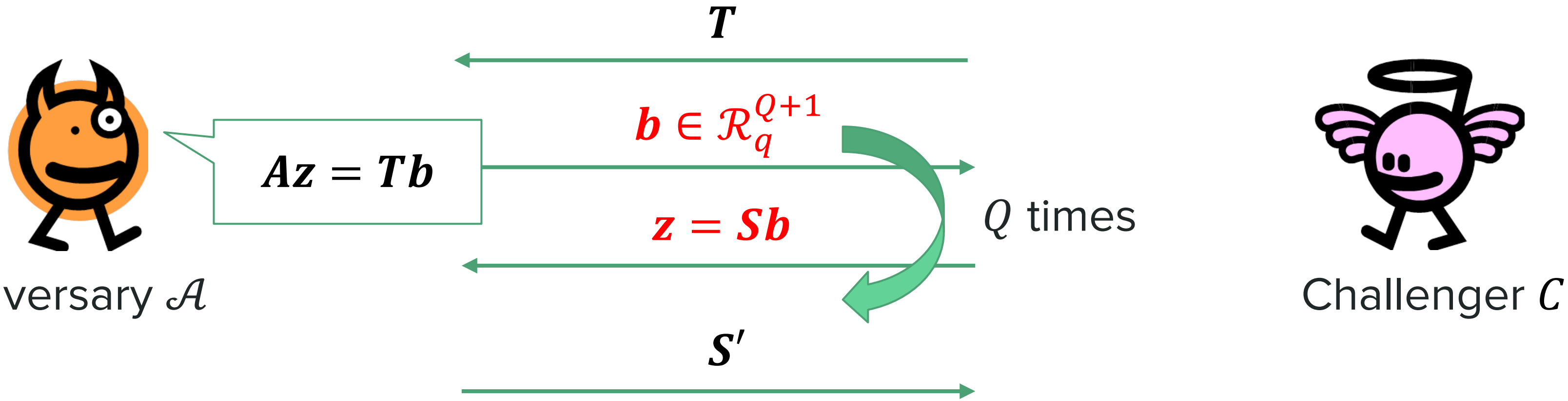
Instance: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{T} = [\boldsymbol{t}_0\boldsymbol{t}_1\cdots\boldsymbol{t}_Q] \in \mathcal{R}_q^{k\times(Q+1)}$

$\boldsymbol{t}_i = \begin{bmatrix} \boldsymbol{A}' & \boldsymbol{I} \end{bmatrix} \boldsymbol{s}_i$

$\boldsymbol{s}_i$ is short

$\boldsymbol{S} = [\boldsymbol{s}_0\boldsymbol{s}_1\cdots\boldsymbol{s}_Q]$

$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}$


$\boldsymbol{T}$
$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{T}\boldsymbol{b}$
$\boldsymbol{b} \in \mathcal{R}_q^{Q+1}$
$\boldsymbol{z} = \boldsymbol{S}\boldsymbol{b}$
Q times
$\boldsymbol{S}'$
Adversary $\mathcal{A}$
Challenger C

$\mathcal{A}$ wins if $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}'$ and $\boldsymbol{S}'$ is short

Naive Attempt

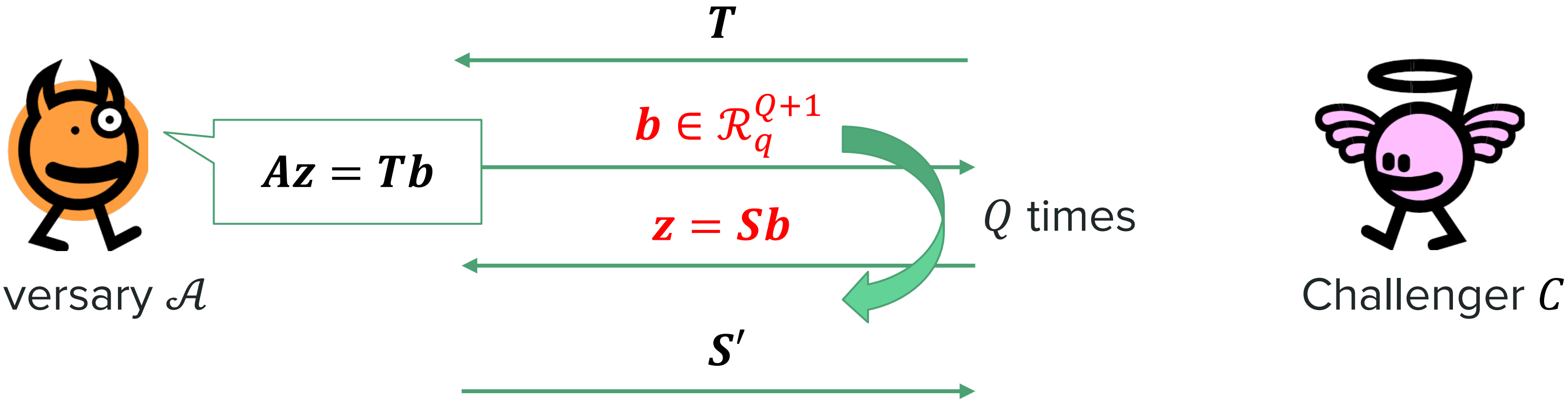
Instance: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1 \cdots \boldsymbol{t}_Q] \in \mathcal{R}_q^{k\times(Q+1)}$

$$\boldsymbol{t}_i = \begin{bmatrix} \boldsymbol{A}' & \boldsymbol{I} \end{bmatrix} \boldsymbol{s}_i$$

$\boldsymbol{s}_i$ is short

$\boldsymbol{S} = [\boldsymbol{s}_0 \boldsymbol{s}_1 \cdots \boldsymbol{s}_Q]$

$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}$



$\boldsymbol{T}$

$\boldsymbol{b} \in \mathcal{R}_q^{Q+1}$

$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{T}\boldsymbol{b}$

$\boldsymbol{z} = \boldsymbol{S}\boldsymbol{b}$

Q times

$\boldsymbol{S}'$

Adversary $\mathcal{A}$

Challenger C

$\mathcal{A}$ wins if $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}'$ and $\boldsymbol{S}'$ is short

Is this problem hard?

Naive Attempt

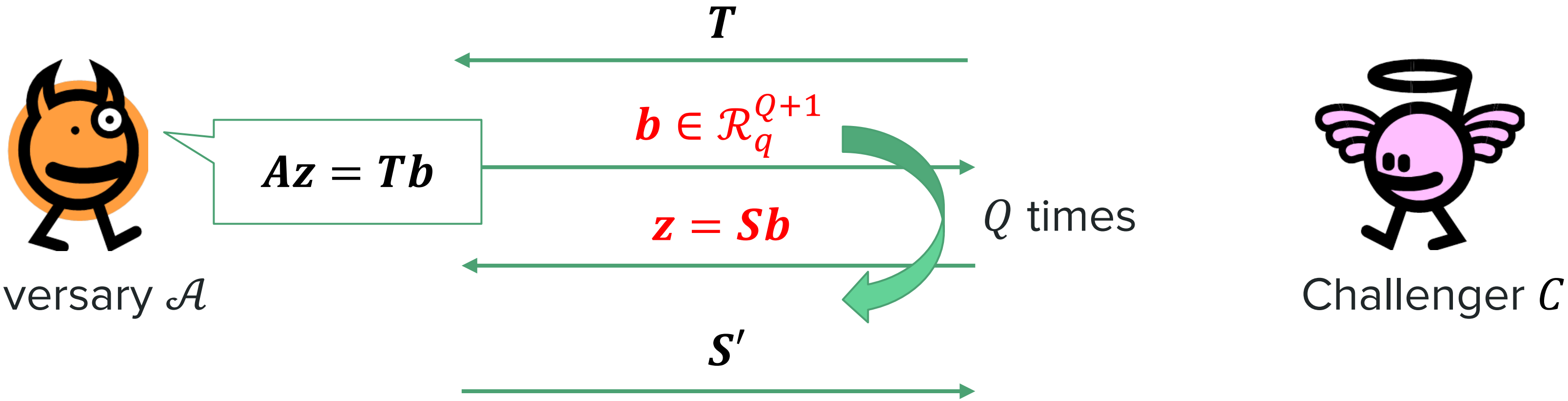
Instance: $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}, \boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1 \cdots \boldsymbol{t}_Q] \in \mathcal{R}_q^{k\times(Q+1)}$

$$\boxed{\boldsymbol{t}_i} = \boxed{\boldsymbol{A}' \mid \boldsymbol{I}}\,\boxed{\boldsymbol{s}_i}$$

$\boldsymbol{s}_i$ is short

$\boldsymbol{S} = [\boldsymbol{s}_0 \boldsymbol{s}_1 \cdots \boldsymbol{s}_Q]$

$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}$



$\boldsymbol{T}$

$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{T}\boldsymbol{b}$

$\boldsymbol{b} \in \mathcal{R}_q^{Q+1}$

$\boldsymbol{z} = \boldsymbol{S}\boldsymbol{b}$

Q times

$\boldsymbol{S}'$

Adversary $\mathcal{A}$

Challenger C

$\mathcal{A}$ wins if $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}'$ and $\boldsymbol{S}'$ is short

Is this problem hard?

No! We have attacks against this problem.

Insecure Example: Large Algebraic Query

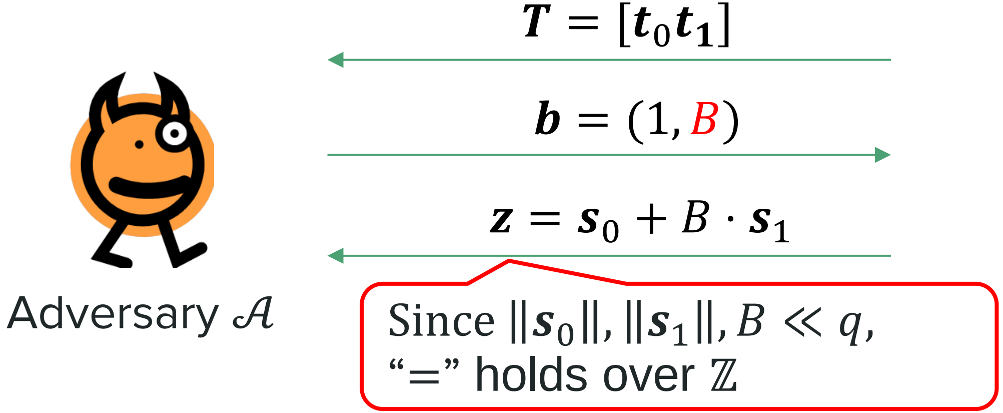
Assume
- $Q = 1$
- $B \ll$ modulus q
- $\|\boldsymbol{s}_i\|_\infty < B$

$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$
Adversary $\mathcal{A}$

Insecure Example: Large Algebraic Query

Assume

- $Q = 1$
- $B \ll$ modulus q
- $\|\boldsymbol{s}_i\|_\infty < B$



$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$

$\boldsymbol{b} = (1, B)$

$\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1$

Adversary $\mathcal{A}$

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$,
"$=$" holds over $\mathbb{Z}$

Insecure Example: Large Algebraic Query

Assume

- $Q = 1$
- $B \ll$ modulus q
- $\|\boldsymbol{s}_i\|_\infty < B$

1. $\boldsymbol{z} \bmod B = \boldsymbol{s}_0$

Adversary $\mathcal{A}$

$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$ ←

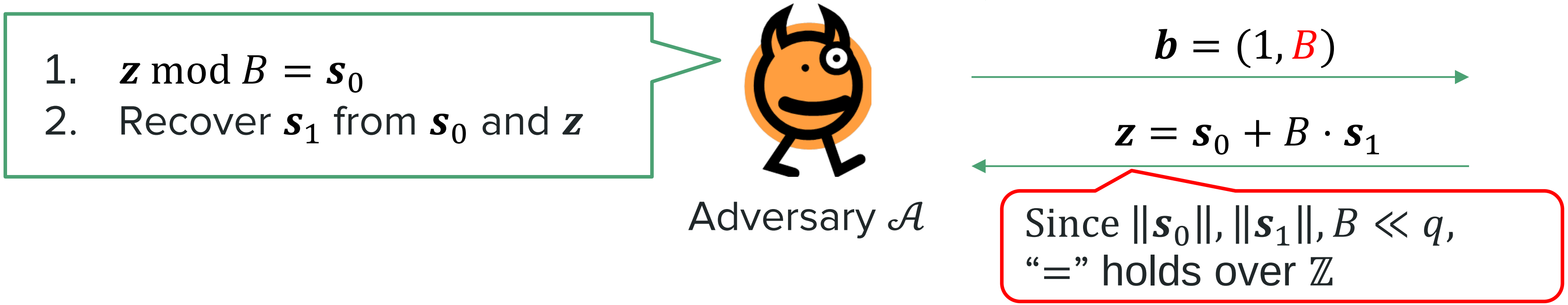
$\boldsymbol{b} = (1, B)$ →

$\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1$ ←

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$,
“=” holds over $\mathbb{Z}$

Insecure Example: Large Algebraic Query

Assume
- $Q = 1$
- $B \ll$ modulus q
- $\|\boldsymbol{s}_i\|_\infty < B$



1. $\boldsymbol{z} \bmod B = \boldsymbol{s}_0$
2. Recover $\boldsymbol{s}_1$ from $\boldsymbol{s}_0$ and $\boldsymbol{z}$

Adversary $\mathcal{A}$

$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$

$\boldsymbol{b} = (1, B)$

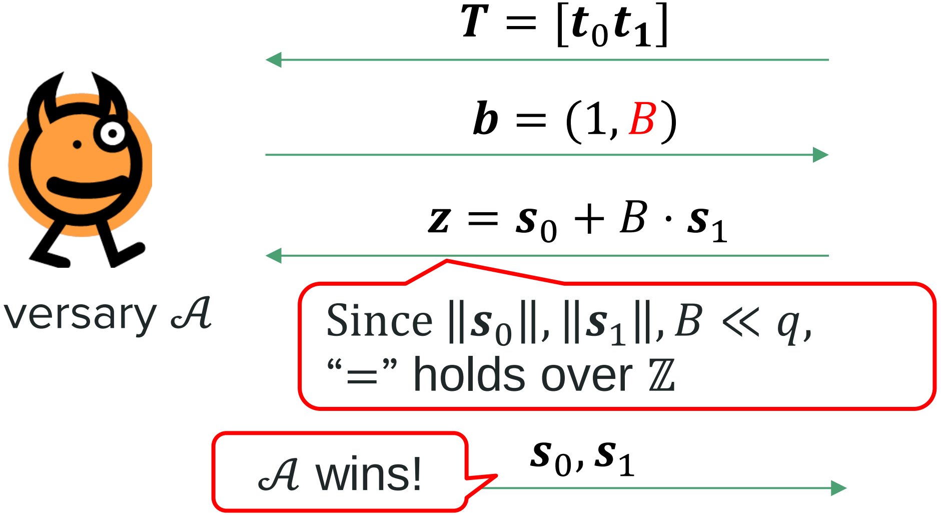
$\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1$

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$,
"=" holds over $\mathbb{Z}$

Insecure Example: Large Algebraic Query

Assume
- $Q = 1$
- $B \ll$ modulus q
- $\|\boldsymbol{s}_i\|_\infty < B$

1. $\boldsymbol{z} \bmod B = \boldsymbol{s}_0$
2. Recover $\boldsymbol{s}_1$ from $\boldsymbol{s}_0$ and $\boldsymbol{z}$



Adversary $\mathcal{A}$

$\xleftarrow{\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]}$

$\xrightarrow{\boldsymbol{b} = (1, B)}$

$\xleftarrow{\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1}$

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$,
"$=$" holds over $\mathbb{Z}$

$\mathcal{A}$ wins!

$\xrightarrow{\boldsymbol{s}_0, \boldsymbol{s}_1}$

Insecure Example: Large Algebraic Query

Assume
- $Q = 1$
- $B \ll$ modulus q
- $\boxed{\|\boldsymbol{s}_i\|_\infty < B}$

1. $\boldsymbol{z} \bmod B = \boldsymbol{s}_0$
2. Recover $\boldsymbol{s}_1$ from $\boldsymbol{s}_0$ and $\boldsymbol{z}$

Adversary $\mathcal{A}$

Lattice-specific attack exploiting "small" secret

$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$

$\boldsymbol{b} = (1, B)$

$\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1$

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$, "=" holds over $\mathbb{Z}$

$\mathcal{A}$ wins!

$\boldsymbol{s}_0, \boldsymbol{s}_1$

Insecure Example: Large Algebraic Query

Assume

- $Q = 1$
- $B \ll$ modulus q
- $\boxed{\|\boldsymbol{s}_i\|_\infty < B}$

1. $\boldsymbol{z} \bmod B = \boldsymbol{s}_0$
2. Recover $\boldsymbol{s}_1$ from $\boldsymbol{s}_0$ and $\boldsymbol{z}$

Adversary $\mathcal{A}$

$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$ ←

→ $\boldsymbol{b} = (1, B)$

$\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1$ ←

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$, "$=$" holds over $\mathbb{Z}$

Lattice-specific attack exploiting "small" secret

$\mathcal{A}$ wins!

→ $\boldsymbol{s}_0, \boldsymbol{s}_1$

Since $s_i \leftarrow \mathbb{Z}_q$ in classical setting, Attack does not work well.

Insecure Example: Large Algebraic Query

Assume

- $Q = 1$
- $B \ll$ modulus q
- $\boxed{\|\boldsymbol{s}_i\|_\infty < B}$

1. $\boldsymbol{z} \bmod B = \boldsymbol{s}_0$
2. Recover $\boldsymbol{s}_1$ from $\boldsymbol{s}_0$ and $\boldsymbol{z}$

Adversary $\mathcal{A}$

$\boldsymbol{T} = [\boldsymbol{t}_0 \boldsymbol{t}_1]$ ←

$\boldsymbol{b} = (1, B)$ →

$\boldsymbol{z} = \boldsymbol{s}_0 + B \cdot \boldsymbol{s}_1$ ←

Since $\|\boldsymbol{s}_0\|, \|\boldsymbol{s}_1\|, B \ll q$, "$=$" holds over $\mathbb{Z}$

Lattice-specific attack exploiting "small" secret

$\mathcal{A}$ wins!

$\boldsymbol{s}_0, \boldsymbol{s}_1$ →

Since $s_i \leftarrow \mathbb{Z}_q$ in classical setting,
Attack does not work well.

We have to ***restrict the shape of LWE queries*** to ensure hardness!

Accepted Linear Combination

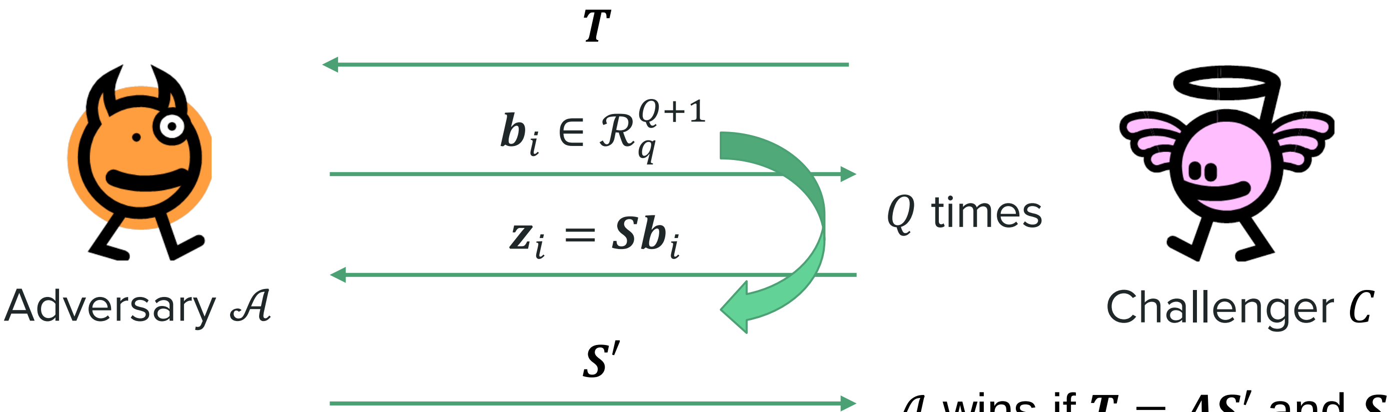
We ***have to restrict the shape of queries*** to ensure hardness.

➡ "***Accepted Linear Combination***" (ALC): $\mathcal{L} \subseteq \mathcal{R}_q^{(Q+1)\times Q}$

Accepted Linear Combination

We ***have to restrict the shape of queries*** to ensure hardness.

"***Accepted Linear Combination***" (ALC): $\mathcal{L} \subseteq \mathcal{R}_q^{(Q+1)\times Q}$



$\boldsymbol{T}$

$\boldsymbol{b}_i \in \mathcal{R}_q^{Q+1}$

$\boldsymbol{z}_i = \boldsymbol{S}\boldsymbol{b}_i$

Q times

Adversary $\mathcal{A}$

Challenger C

$\boldsymbol{S}'$

$\mathcal{A}$ wins if $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{S}'$ and $\boldsymbol{S}'$ is short

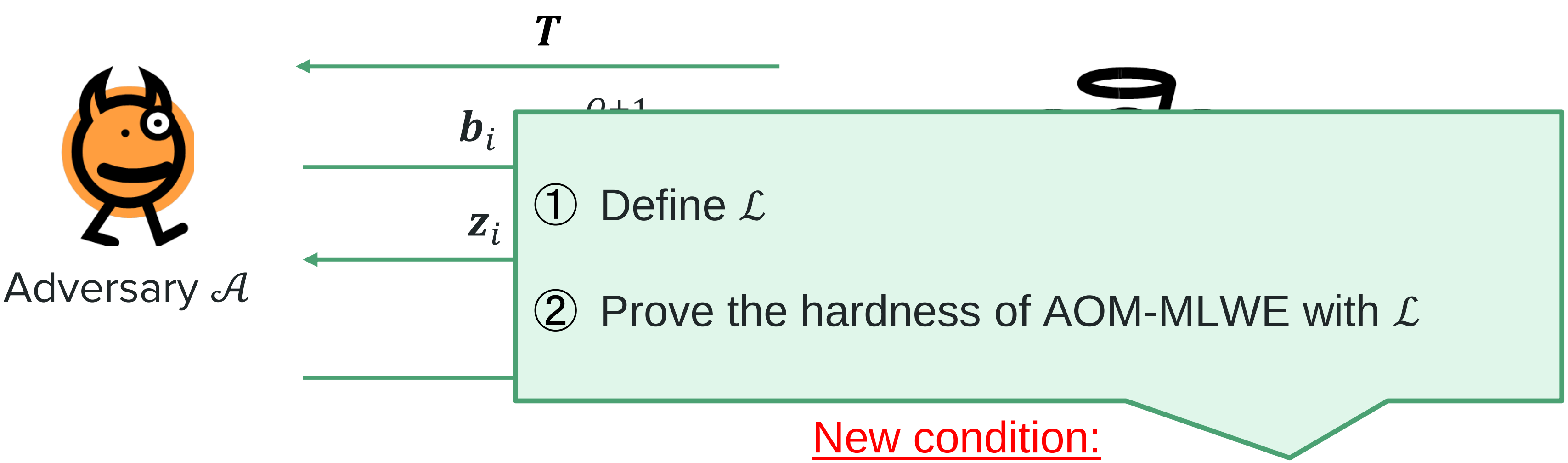
New condition:

$$\boldsymbol{B} = \left[\boldsymbol{b}_1 \cdots \boldsymbol{b}_Q\right] \in \mathcal{L}$$

Accepted Linear Combination

We ***have to restrict the shape of queries*** to ensure hardness.

“***Accepted Linear Combination***” (ALC): $\mathcal{L} \subseteq \mathcal{R}_q^{(Q+1)\times Q}$



① Define $\mathcal{L}$

② Prove the hardness of AOM-MLWE with $\mathcal{L}$

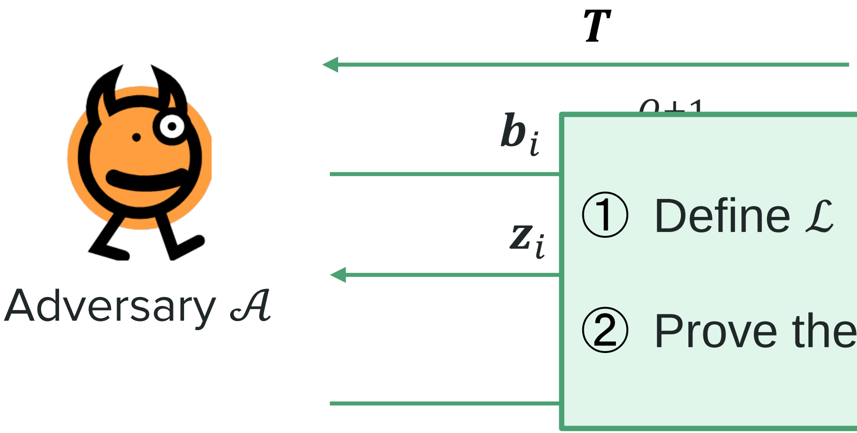
New condition:

$$\boldsymbol{B} = \left[\boldsymbol{b}_1 \cdots \boldsymbol{b}_Q\right] \in \mathcal{L}$$

Accepted Linear Combination

We ***have to restrict the shape of queries*** to ensure hardness.

“***Accepted Linear Combination***” (ALC): $\mathcal{L} \subseteq \mathcal{R}_q^{(Q+1)\times Q}$



① Define $\mathcal{L}$ ⇒ *protocol-specific*

② Prove the hardness of AOM-MLWE with $\mathcal{L}$

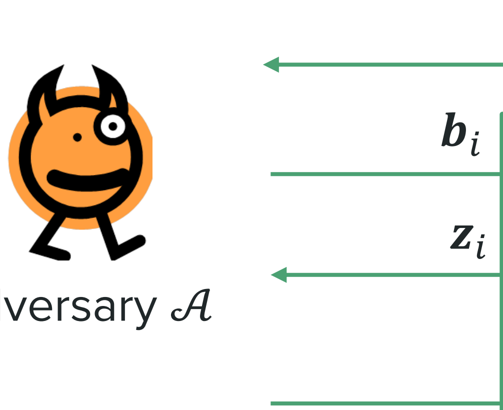
New condition:

$$\boldsymbol{B} = \left[\boldsymbol{b}_1 \cdots \boldsymbol{b}_Q\right] \in \mathcal{L}$$

Accepted Linear Combination

We ***have to restrict the shape of queries*** to ensure hardness.

“***Accepted Linear Combination***” (ALC): $\mathcal{L} \subseteq \mathcal{R}_q^{(Q+1)\times Q}$



Adversary $\mathcal{A}$

① Define $\mathcal{L}$ ⇒ *protocol-specific*

② Prove the hardness of AOM-MLWE with $\mathcal{L}$

New condition:

$\boldsymbol{B} = \left[\boldsymbol{b}_1 \cdots \boldsymbol{b}_Q\right] \in \mathcal{L}$

How do we establish the hardness under specific $\mathcal{L}$?

Classical Setting:

⇒ **Use Generic Group Model (GGM)**

⇒ (A)OM-DL is as hard as DLP under the GGM [AC:BFP21].

How do we establish the hardness under specific $\mathcal{L}$?

Classical Setting:

⇒ **Use Generic Group Model (GGM)**

⇒ (A)OM-DL is as hard as DLP under the GGM [AC:BFP21].

Lattice Setting:

⇒ **No model like GGM!!**

How do we establish the hardness?

How do we establish the hardness under specific $\mathcal{L}$?

Classical Setting:

⇒ **Use Generic Group Model (GGM)**

⇒ (A)OM-DL is as hard as DLP under the GGM [AC:BFP21].

Lattice Setting:

⇒ **No model like GGM!!**

How do we establish the hardness?

We *heuristically* establish it in ***two steps***:

1. "**Selective**" AOM-MLWE with specific $\mathcal{L}$ is hard under standard assumptions.
2. Practical cryptanalysis against **adaptive** adversary.

Step 1: Hardness of Selective AOM-MLWE with $\mathcal{L}$

What is sel-AOMMLWE?

$\mathcal{A}$ has to output a query matrix $\mathcal{B} \in \mathcal{L}$ at the beginning of the game.

Why selective?

Previous insecure example induces **a statistical attack**.

⇒ reveals obvious “weak” parameters

It does not exploit adaptive query.

Step 1: Hardness of Selective AOM-MLWE with $\mathcal{L}$

What is sel-AOMMLWE?

$\mathcal{A}$ has to output a query matrix $\mathcal{B} \in \mathcal{L}$ at the beginning of the game.

Why selective?

Previous insecure example induces **a statistical attack**.

⇒ reveals obvious “weak” parameters

It does not exploit adaptive query.

Goal of this step: To show sel-AOM-MLWE with specific $\mathcal{L}$ is hard

i.e., needs to exploit adaptive query to break

Step 1: Hardness of Selective AOM-MLWE with $\mathcal{L}$

What is sel-AOMMLWE?

$\mathcal{A}$ has to output a query matrix $\mathcal{B} \in \mathcal{L}$ at the beginning of the game.

Why selective?

Previous insecure example induces **a statistical attack**.

⇒ reveals obvious "weak" parameters

It does not exploit adaptive query.

Goal of this step: To show sel-AOM-MLWE with specific $\mathcal{L}$ is hard

i.e., needs to exploit adaptive query to break

We showed that

The **sel**-AOM-LWE with certain $\mathcal{L}$ is **hard under MLWE + MSIS.**

Step 2: Practical Cryptanalysis against Adaptive $\mathcal{A}$

Consider a generic attack for certain $\mathcal{L}$.

$\mathcal{A}$'s strategy in simple example:

Step 2: Practical Cryptanalysis against Adaptive $\mathcal{A}$

Consider a generic attack for certain $\mathcal{L}$.

$\mathcal{A}$'s strategy in simple example:

$$\text{From LWE queries, obtain } \{\boldsymbol{s}_0 + \boldsymbol{s}_i\}_{i\in[Q]} \xrightarrow{\text{Sum}} Q \cdot \boldsymbol{s}_0 + \sum_{i=1}^{Q} \boldsymbol{s}_i$$

Step 2: Practical Cryptanalysis against Adaptive $\mathcal{A}$

Consider a generic attack for certain $\mathcal{L}$.

$\mathcal{A}$'s strategy in simple example:

From LWE queries, obtain $\{\boldsymbol{s}_0 + \boldsymbol{s}_i\}_{i\in[Q]}$ $\xrightarrow{\text{Sum}}$ $Q \cdot \boldsymbol{s}_0 + \boxed{\sum_{i=1}^{Q} \boldsymbol{s}_i}$

By Gaussian convolution,
this is $\sqrt{Q}$ times smaller than $Q \cdot \boldsymbol{s}_0$

Step 2: Practical Cryptanalysis against Adaptive $\mathcal{A}$

Consider a generic attack for certain $\mathcal{L}$.

$\mathcal{A}$'s strategy in simple example:

From LWE queries, obtain $\{\boldsymbol{s}_0 + \boldsymbol{s}_i\}_{i \in [Q]}$ $\xrightarrow{\text{Sum}}$ $Q \cdot \boldsymbol{s}_0 + \boxed{\sum_{i=1}^{Q} \boldsymbol{s}_i}$

By Gaussian convolution, this is $\sqrt{Q}$ times smaller than $Q \cdot \boldsymbol{s}_0$

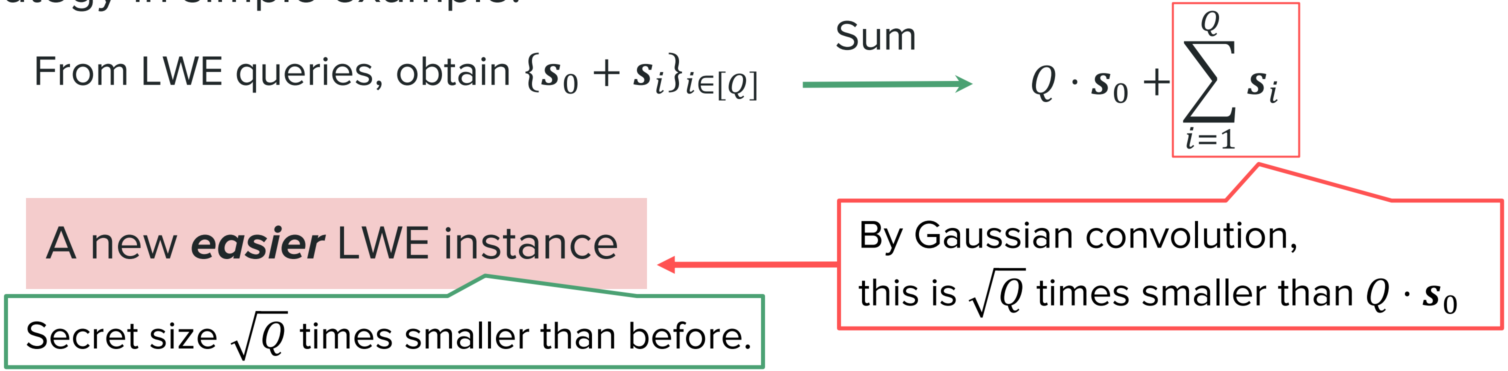
A new ***easier*** LWE instance

Secret size $\sqrt{Q}$ times smaller than before.

Step 2: Practical Cryptanalysis against Adaptive $\mathcal{A}$

Consider a generic attack for certain $\mathcal{L}$.

$\mathcal{A}$'s strategy in simple example:



From LWE queries, obtain $\{\boldsymbol{s}_0 + \boldsymbol{s}_i\}_{i\in[Q]}$ $\xrightarrow{\text{Sum}}$ $Q \cdot \boldsymbol{s}_0 + \boxed{\sum_{i=1}^{Q} \boldsymbol{s}_i}$

By Gaussian convolution, this is $\sqrt{Q}$ times smaller than $Q \cdot \boldsymbol{s}_0$

A new ***easier*** LWE instance

Secret size $\sqrt{Q}$ times smaller than before.

Generalize this strategy to all accepted queries in $\mathcal{L}$.

Heuristically show that for $\mathcal{A}$ following this strategy

An adaptive $\mathcal{A}$ is *no stronger* than a selective $\mathcal{A}$.

Two-Round Threshold Raccoon

Construction

Construct by combining FROST + TRaccoon

Construction

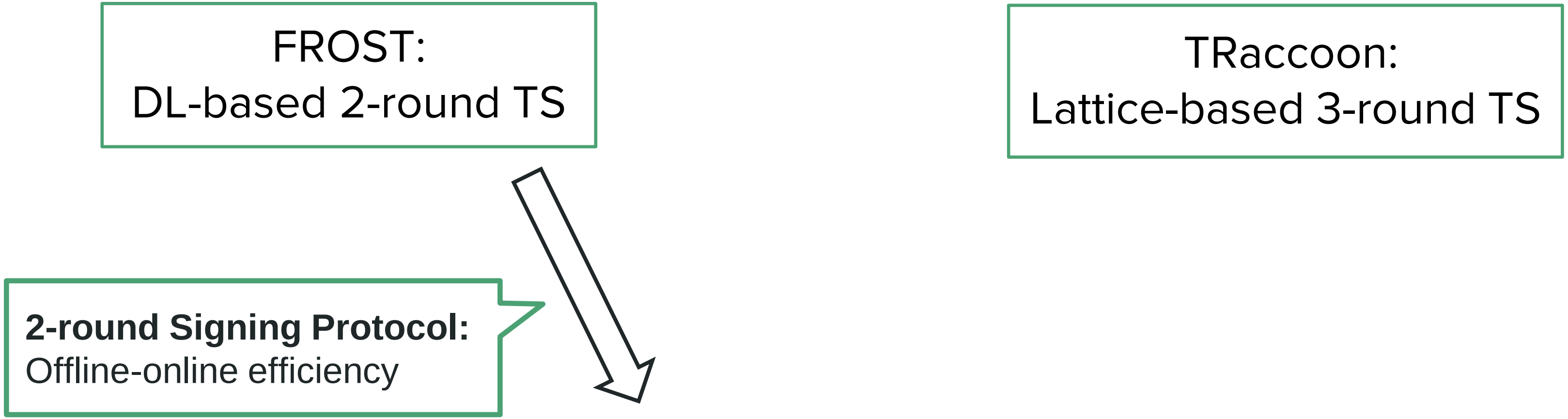
Construct by combining FROST + TRaccoon

FROST:
DL-based 2-round TS

TRaccoon:
Lattice-based 3-round TS

Construction

Construct by combining FROST + TRaccoon



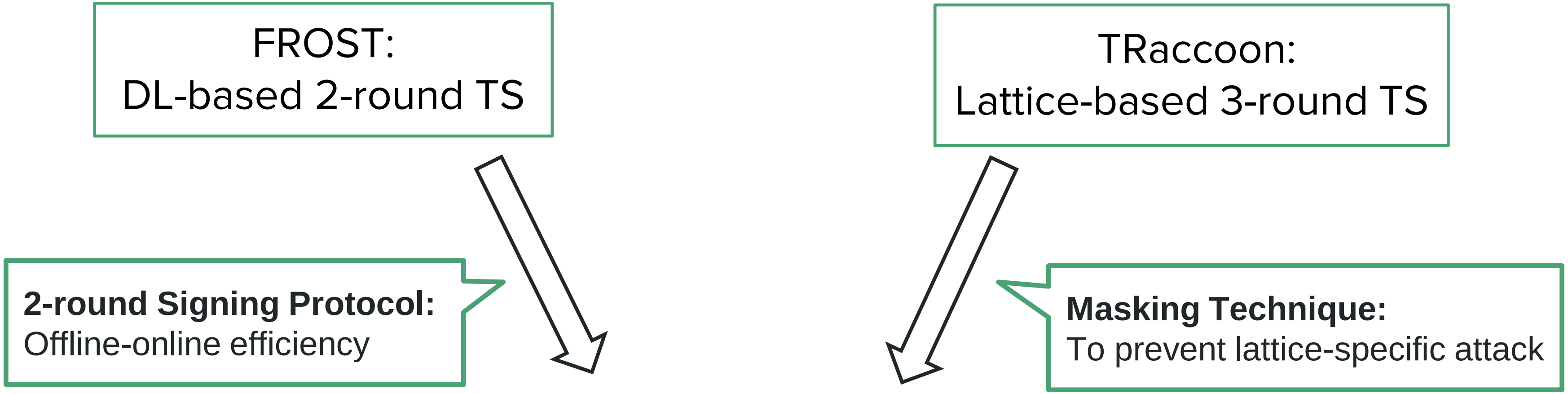
FROST:
DL-based 2-round TS

TRaccoon:
Lattice-based 3-round TS

2-round Signing Protocol:
Offline-online efficiency

Construction

Construct by combining FROST + TRaccoon



FROST:
DL-based 2-round TS

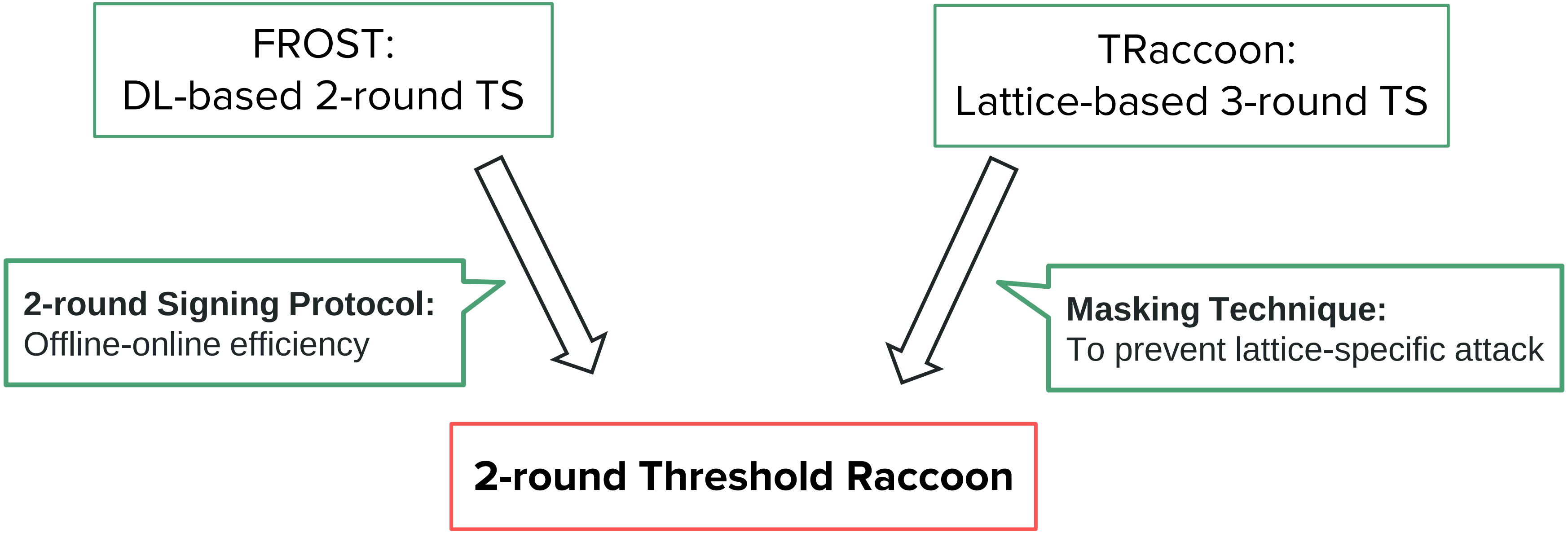
TRaccoon:
Lattice-based 3-round TS

2-round Signing Protocol:
Offline-online efficiency

Masking Technique:
To prevent lattice-specific attack

Construction

Construct by combining FROST + TRaccoon



FROST:
DL-based 2-round TS

TRaccoon:
Lattice-based 3-round TS

2-round Signing Protocol:
Offline-online efficiency

Masking Technique:
To prevent lattice-specific attack

2-round Threshold Raccoon

Security

We proved the unforgeability ***under AOM-MLWE.***

Proof strategy is almost the same as FROST's proof, but...

Check **if query matrix made by reduction is contained in our ALC!!**

Security

We proved the unforgeability ***under AOM-MLWE.***

Proof strategy is almost the same as FROST’s proof, but…

Check **if query matrix made by reduction is contained in our ALC!!**

Our ALC :

Row vector

$$\mathcal{L}_{\mathsf{TS}} = \left\{ \begin{bmatrix} 1 & \\ & \mathbf{P}_{\text{row}} \end{bmatrix} \begin{bmatrix} \mathbf{c}_1^\top & \mathbf{c}_2^\top & \cdots & \mathbf{c}_{Q'}^\top \\ \mathbf{B}_1 & & & \\ & \mathbf{B}_2 & & \\ & & \ddots & \\ & & & \mathbf{B}_{Q'} \end{bmatrix} \cdot \mathbf{P}_{\text{column}} \subset \mathcal{R}_q^{Q\times(Q-1)} \;\middle|\; \begin{array}{l} \forall i \in [Q'], (\mathbf{c}_i, \mathbf{B}_i) \in \mathcal{C}_{\mathsf{TS}} \times \mathcal{B}_{\mathsf{TS}}, \\ (\mathbf{P}_{\text{row}}, \mathbf{P}_{\text{column}}) \in \mathcal{P}_{Q-1}^2 \end{array} \right\}$$

Block diagonal

Performances

Under $T \leq 1024$ setting, for 128-bit security,

Scheme	|vk|	|Sig|	Online Comm./User	Offline Comm./User
3-round	3.9 KB	12.7 KB	40.8 KB	-
2-round	5.5 KB	10.8 KB	14.1 KB	262 KB

Almost the same

Efficient!

Overhead

Thank You!

Important Future Work:

- ➢ To prove **the hardness of adaptive AOM-MLWE.**

Concurrent Works:

- ➢ "Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding" [C: KTR24] (Next Talk!!)
- ➢ "Flood and submerse: Verifiable short secret sharing and application to robust threshold signatures on lattices" [C: EPN24] (Talk was this morning!!)
- ➢ "Partially Non-Interactive Two-Round Lattice-Based Threshold Signatures" [Eprint:CATZ24]

Recent Related Works:

- ➢ "Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors" [Eprint:BKL+24]
  (※ partially offline-online efficient)