@NTU

Exploring Links Between Lattice-based NIZKs and Various Signature Schemes (in the Standard Model)

Shuichi Katsumata (AIST, PQShield)

Overview of This Talk

1. Non-Interactive Zero-Knowledge (NIZK)

2. Links Between NIZKs and Signatures

3. Result: Group Signatures w/o CRS-NIZK

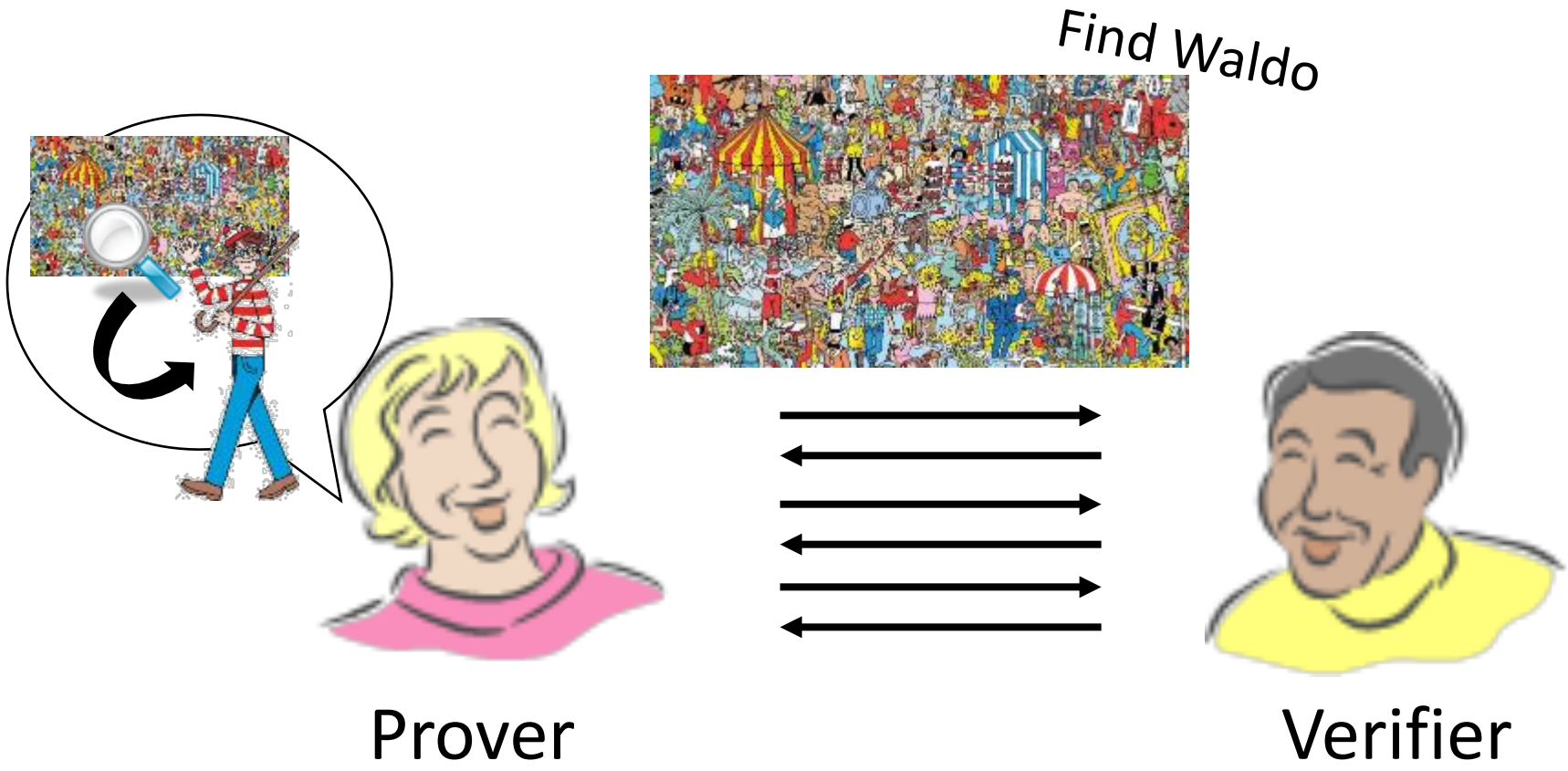
[KY19@EC]

- New Notion:
  Multi-User Designated-Prover NIZKs

1. Introduction:

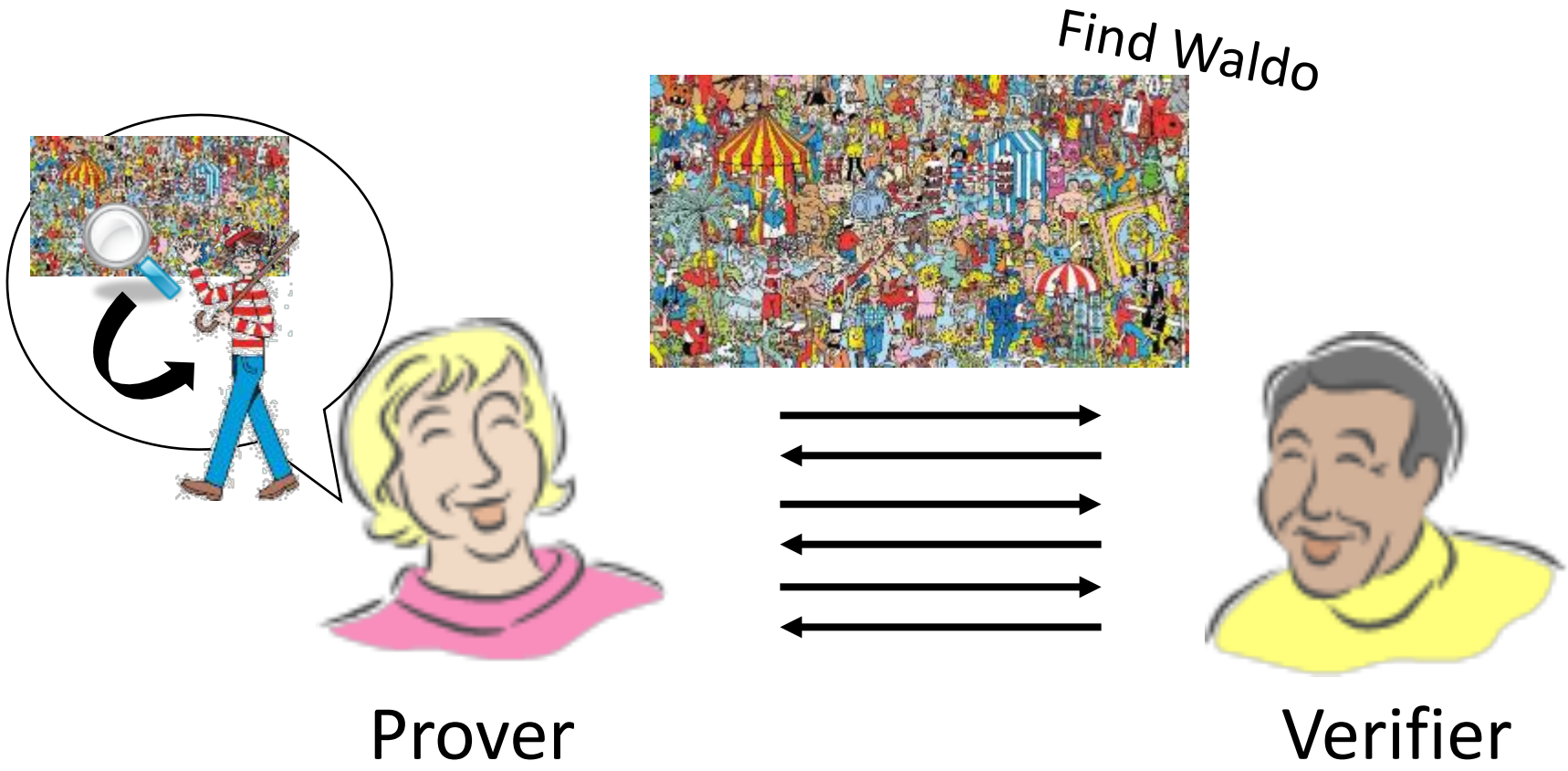
Non-Interactive Zero-Knowledge

Zero-Knowledge Proof Systems



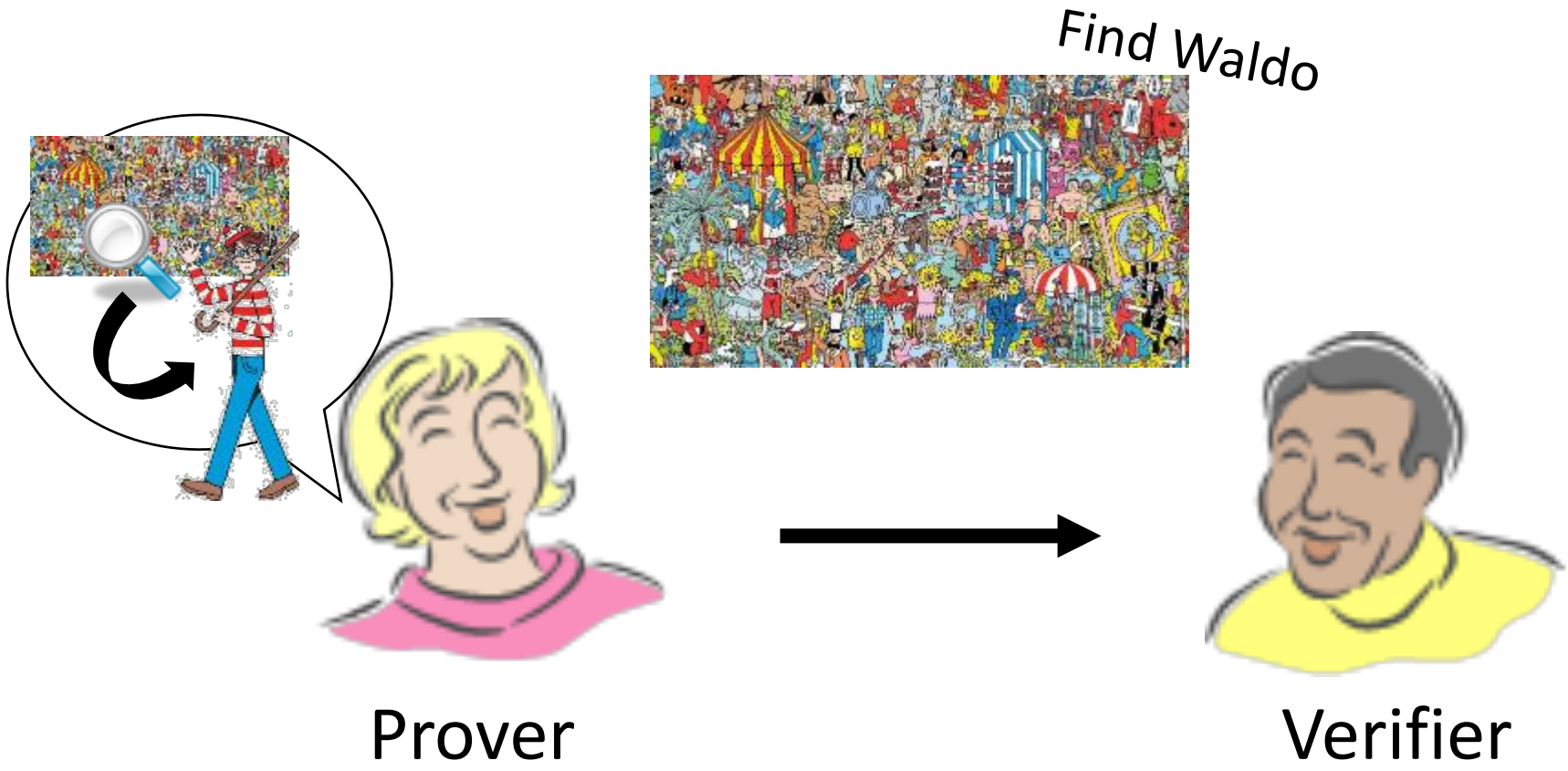
- Verifier is convinced that Prover knows where Waldo is.

Zero-Knowledge Proof Systems



- Verifier is convinced that Prover knows where Waldo is.
- …BUT, Verifier doesn't learn where Waldo is!

Non-Interactive ZK (NIZK)



- Prover **<u>sends only one message</u>** to Verifier.

More Formally: NIZKs

$x \in L$

Prover (x, w) Verifier x

$\xrightarrow{\pi}$

✓ **Completeness**: If $(x, w) \in R_L$, then Verifier is convinced.

More Formally: NIZKs

Cheating Prover x

$x \notin L$

Verifier x

π

➡ Reject π.

- ✓ **Completeness**: If $(x, w) \in R_L$, then Verifier is convinced.
- ✓ **Soundness**: If $x \notin L$, cheating Prover cannot convince Verifier.

More Formally: NIZKs

$x \in L$

Prover (x, w)

π

Verifier x

- ✓ **Completeness**: If $(x, w) \in R_L$, then Verifier is convinced.
- ✓ **Soundness**: If $x \notin L$, cheating Prover cannot convince Verifier.
- ✓ **Zero-Knowledge**: If $x \in L$, Verifier only learns that $x \in L$.

Motivation for NIZK

Many Applications of NIZKs

- OWF + NIZK ⇒ signature scheme [BG89@CRYPTO]
- CPA-PKE + NIZK ⇒ CCA-PKE [NY90@STOC]
- Semi-honest secure MPC + NIZK ⇒ Malicious secure MPC [GMW86@CRYPTO]
- ….

Theoretical Interest

- Connections with complexity theory

Building NIZKs (for all of NP)

Do not exist w/o trusted setup! ☹ [GO94]

Building NIZKs (for all of NP)

Do not exist w/o trusted setup! ☹ [GO94]

- Random Oracle Model [FS87]
  - Practically appealing solution.
- With Trusted Setup [FLS90]
  - Provable security.
  - Theoretically appealing solution.

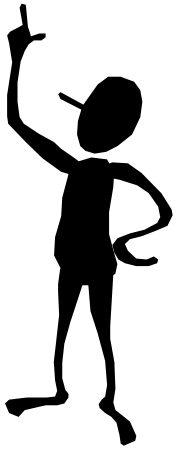
Building NIZKs (for all of NP)

Do not exist w/o trusted setup! ☹ [GO94]

- Random Oracle Model [FS87]
  - Practically appealing solution.

- With Trusted Setup [FLS90]
  - Provable security.
  - Theoretically appealing solution.

This Talk



Various Types of Trusted Setup

→ **CRS**: (public) common reference string

Various Types of Trusted Setup

→ **CRS**: (public) common reference string

CRS, $x \in L$

Prover (x, w)

$\xrightarrow{\pi}$

Verifier x

CRS-NIZK

(Most standard NIZK)

Various Types of Trusted Setup

CRS: (public) common reference string

$\mathbf{k_v}$: (private) verification key

Various Types of Trusted Setup

CRS: (public) common reference string

$\mathbf{k_v}$: (private) verification key

Prover (x, w)

CRS, $x \in L$

Verifier x

π

Designated Verifier-NIZK (DV-NIZK)

$\Rightarrow$Require private $\mathrm{k_v}$ to verify proof π!

Various Types of Trusted Setup

CRS: (public) common reference string

$\mathbf{k_p}$: (private) proving key

Prover (x, w)

CRS, $x \in L$

Verifier x

$\xrightarrow{\pi}$

Designated Prover-NIZK (DP-NIZK)

$\Rightarrow$Require private $\mathrm{k_p}$ to generate proof π!

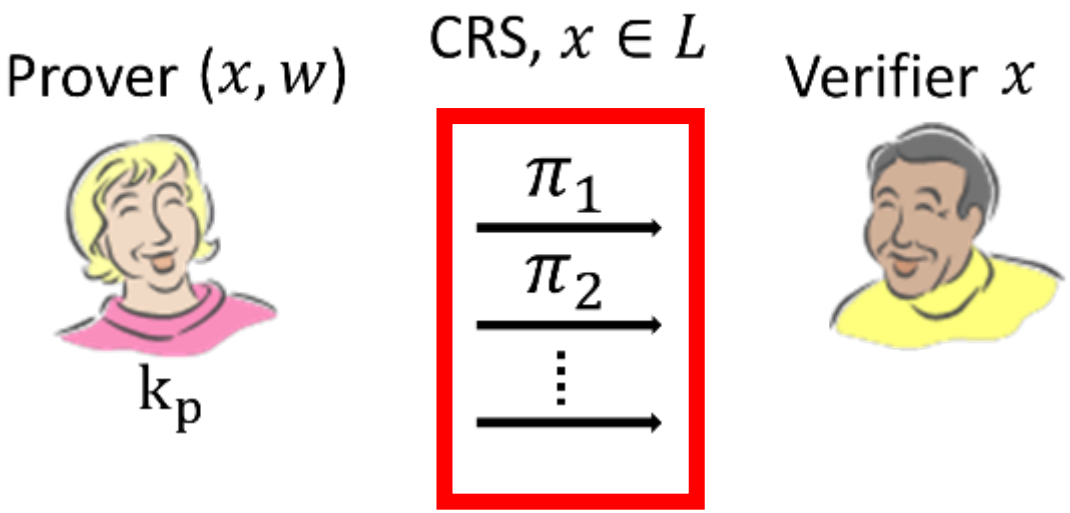
*Subtleties in (DV, DP)-NIZKs

Since verifier/prover maintains secret information, definition requires more care than CRS-NIZK.

*Subtleties in (DV, DP)-NIZKs

Since verifier/prover maintains secret information, definition requires more care than CRS-NIZK.

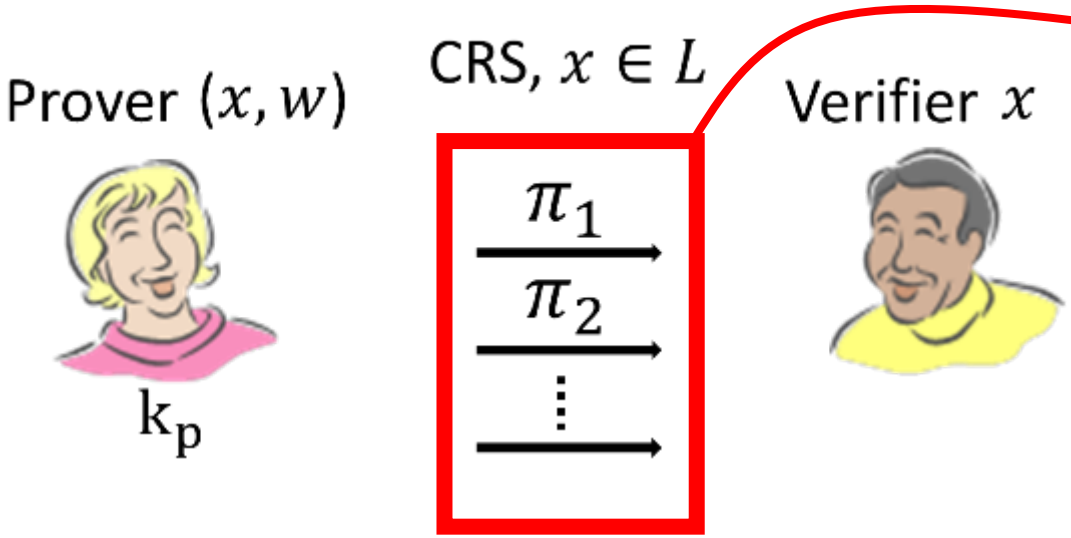
Ex) DP-NIZK: **Unbounded** Zero-Knowledge



*Subtleties in (DV, DP)-NIZKs

Since verifier/prover maintains secret information, definition requires more care than CRS-NIZK.

Ex) DP-NIZK: **Unbounded** Zero-Knowledge



Each proof may leak some information of k_p!

***Consequence**
Parallel repetition may fail.

State-of-Affairs

*Non Exhaustive

1990

CRS-NIZK

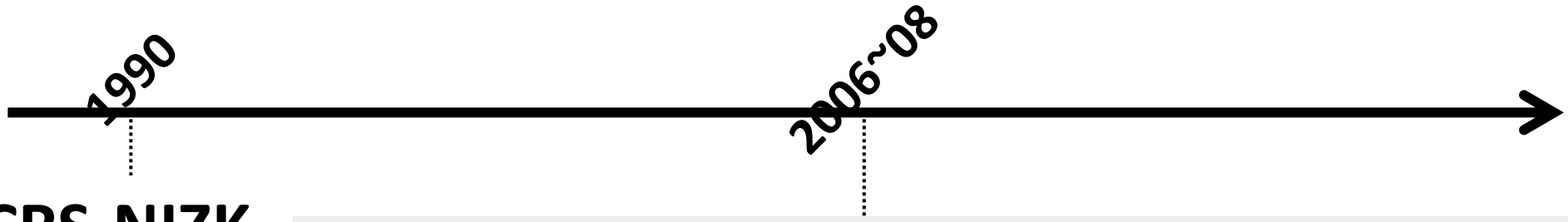
Trapdoor Permutations [FLS90, BY96, G04]

DV-NIZK

DP-NIZK

State-of-Affairs

*Non Exhaustive



CRS-NIZK

Pairings [GOS06, GS08,…]

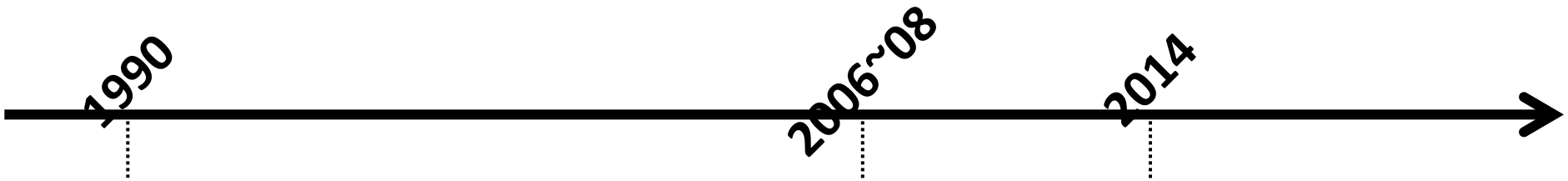
Trapdoor Permutations [FLS90, BY96, G04]

DV-NIZK

DP-NIZK

State-of-Affairs

*Non Exhaustive



CRS-NIZK

Pairings [GOS06, GS08,…]

Trapdoor Permutations [FLS90, BY96, G04]

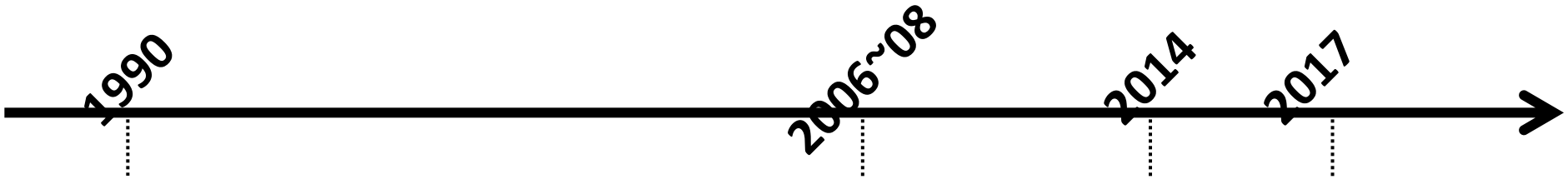
iO [SW14,…]

DV-NIZK

DP-NIZK

State-of-Affairs

*Non Exhaustive



CRS-NIZK

Pairings [GOS06, GS08,…]

Trapdoor Permutations [FLS90, BY96, G04]

iO [SW14,…]

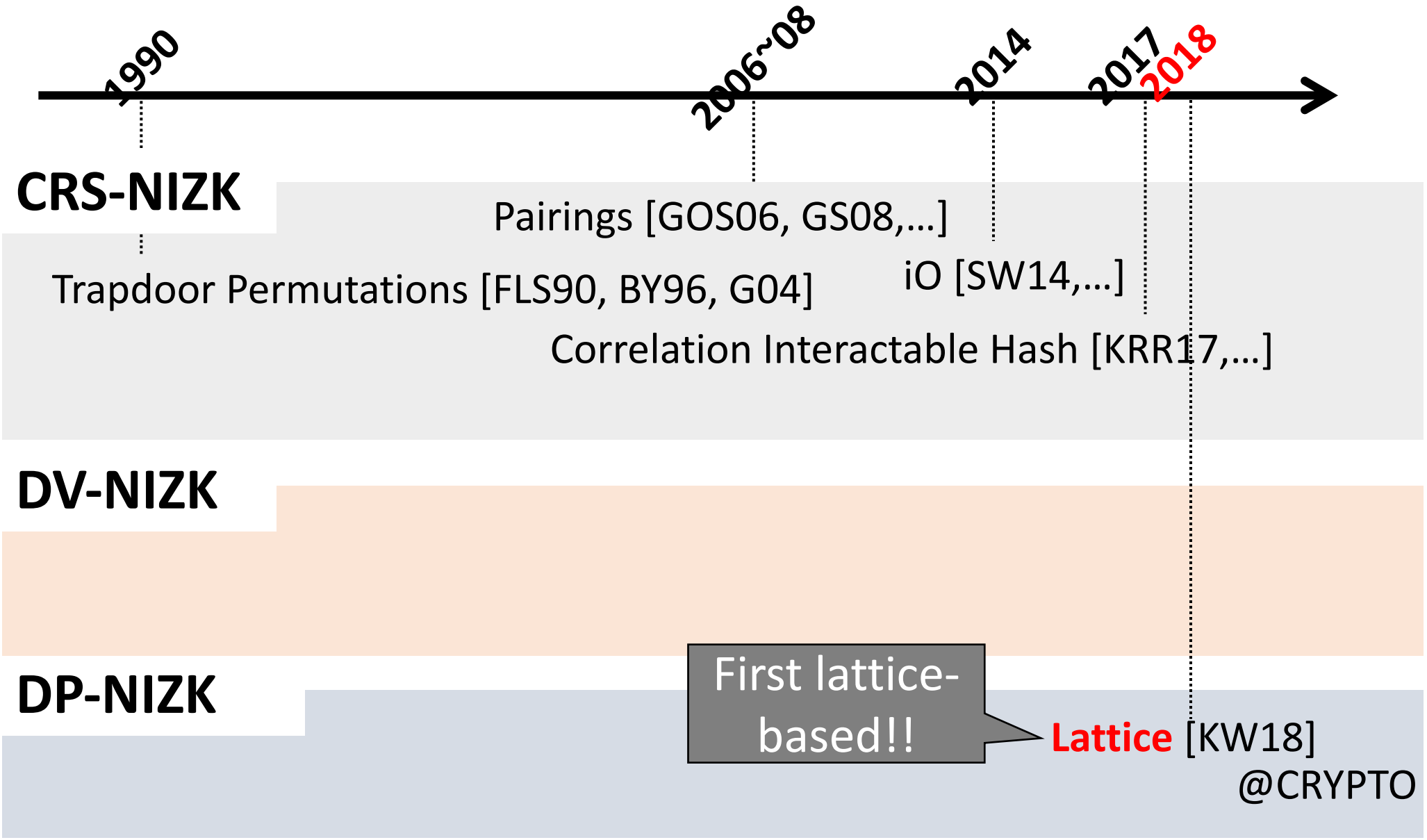
Correlation Interactable Hash [KRR17,…]

DV-NIZK

DP-NIZK

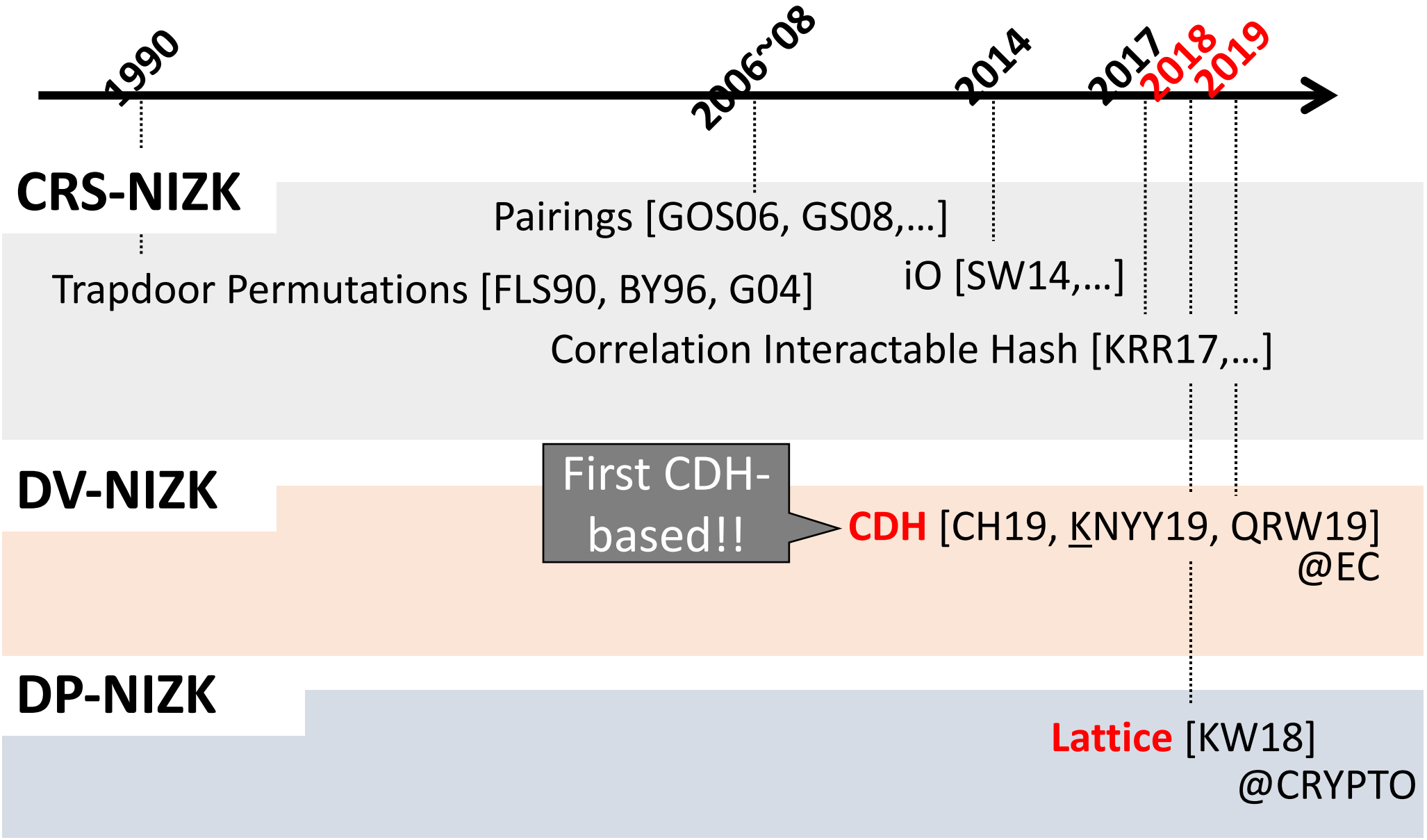
State-of-Affairs

*Non Exhaustive



State-of-Affairs

*Non Exhaustive



State-of-Affairs

*Non Exhaustive

1990 | 2006~08 | 2014 | 2017 | 2018 | 2019

CRS-NIZK

Pairings [GOS06, GS08,…]

Trapdoor Permutations [FLS90, BY96, G04]

iO [SW14,…]

Correlation Interactable Hash [KRR17,…]

Lattice [PS19] @CRYPTO

Finally!!

DV-NIZK

CDH [CH19, KNYY19, QRW19] @EC

Lattice [LQRWW19] @CRYPTO

DP-NIZK

Lattice [KW18] @CRYPTO

State-of-Affairs

*Non Exhaustive

1990 | 2006~08 | 2014 | 2017 | 2018 | 2019

CRS-NIZK

Pairings [GOS06, GS08,…]

Trapdoor Permutations [FLS90, BY96, G04]

iO [SW14,…]

Correlation Interactable Hash [KRR17,…]

Lattice [PS19] @CRYPTO

DV-NIZK

CDH [CH19, KNYY19, QRW19] @EC

Lattice [LQRWW19] @CRYPTO

DP-NIZK

Lattice [KW18] @CRYPTO

Is [KW18] and [LQRWW19] simply a special case of [PS19]??

Closer Look at Lattice-based NIZKs

- CRS-NIZK [PS19]
  - Follows the correlation interactable hash paradigm.
  - Based on **polynomial LWE** **(Learning with Errors)**.

Closer Look at Lattice-based NIZKs

- CRS-NIZK [PS19]
  - Follows the correlation interactable hash paradigm.
  - Based on **polynomial LWE** **(Learning with Errors)**.
- DV-NIZK [LQRWW19]
  - Use new tool: "Function-hiding" Attribute-based Encryption.
  - Based on **super-polynomial LWE**. (Can use LPN and CDH too!)

Closer Look at Lattice-based NIZKs

- CRS-NIZK [PS19]
  - Follows the correlation interactable hash paradigm.
  - Based on **polynomial LWE** **(Learning with Errors)**.
- DV-NIZK [LQRWW19]
  - Use new tool: "Function-hiding" Attribute-based Encryption.
  - Based on **super-polynomial LWE**. (Can use LPN and CDH too!)
- DP-NIZK [KW18]
  - Generic construction from Fully-Homomorphic Signatures.
  - Based on **polynomial SIS** **(Short Integer Solution)**.

Closer Look at Lattice-based NIZKs

- CRS-NIZK [PS19]
  - Follows the correlation interactable hash paradigm.
  - Based on **polynomial LWE** **(Learning with Errors)**.
- DV-NIZK [LQRWW19]
  - Use new tool: "Function-hiding" Attribute-based Encryption.
  - Based on **super-polynomial LWE**. (Can use LPN and CDH too!)
- DP-NIZK [KW18]
  - Generic construction from Fully-Homomorphic Signatures.
  - Based on **polynomial SIS** **(Short Integer Solution)**.

➡ From a theoretical stand point,
DP-NIZK requires the **weakest assumption**!

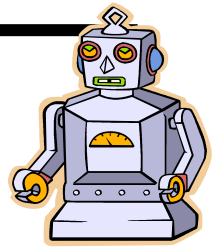
*The state-of-the-art can change anytime! ☺

SIS vs LWE

SIS

- One way function
- Collision resistant hash
- Digital signature scheme
- Attribute-based Signature
- Fully homomorphic signature
- …

LWE



- Public key encryption
- Oblivious transfer
- Attribute-based Encryption
- Fully homomorphic encryption
- …

+

Whatever SIS can.

*In a world of PT quantum algorithms, LWE and SIS are equivalent.

2. Exploration:

Links Between NIZKs and Signature Schemes

Warm Up: Standard Signatures

Well Known Fact…

OWF + **CRS**-NIZK ⇒ Signature Schemes [BG89]

Warm Up: Standard Signatures

Well Known Fact…

OWF + **CRS**-NIZK ⇒ Signature Schemes [BG89]

However,

OWF + **DP**-NIZK ⇒ Signature Schemes

Warm Up: Standard Signatures

Well Known Fact…

OWF + CRS-NIZK ⇒ Signature Schemes [BG89]

However,

OWF + DP-NIZK ⇒ Signature Schemes

Why? At a high level…

In a signature scheme, the **keys are generated honestly** and **secret key is never revealed** to an adversary.

Warm Up: Standard Signatures

Well Known Fact…

OWF + CRS-NIZK ⇒ Signature Schemes [BG89]

→ **LWE**

However,

OWF + DP-NIZK ⇒ Signature Schemes

→ **SIS**

Why? At a high level…

In a signat

honestly a

an adversa

Aligns with prior knowledge that SIS implies signature schemes.

In a Bit More Detail…

*Use fact that OWF implies MAC, COM

Signer

$$\mathrm{vk} = (\mathrm{crs}_{\mathrm{DP}}, \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}})$$

$$\mathrm{sk} = (\mathrm{k}_{\mathrm{P}}, \mathrm{sk}_{\mathrm{MAC}})$$

Verifier

In a Bit More Detail…

*Use fact that OWF implies MAC, COM

Signer

$$\mathrm{vk} = (\mathrm{crs}_{\mathrm{DP}}, \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}})$$

Verifier

$$\mathrm{sk} = (\mathrm{k}_{\mathrm{P}}, \mathrm{sk}_{\mathrm{MAC}})$$

$\mathbf{Sign}(\mathrm{sk}, \mathrm{M})$:

1. $\sigma_{\mathrm{M}} \leftarrow \mathrm{Sign}_{\mathrm{MAC}}(\mathrm{sk}_{\mathrm{MAC}}, \mathrm{M})$

In a Bit More Detail…

*Use fact that OWF implies MAC, COM

Signer

$$\mathrm{vk} = (\mathrm{crs}_{\mathrm{DP}}, \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}})$$

$$\mathrm{sk} = (\mathrm{k}_{\mathrm{P}}, \mathrm{sk}_{\mathrm{MAC}})$$

Verifier

$\mathbf{Sign}(\mathrm{sk}, \mathrm{M})$:

1. $\sigma_{\mathrm{M}} \leftarrow \mathrm{Sign}_{\mathrm{MAC}}(\mathrm{sk}_{\mathrm{MAC}}, \mathrm{M})$
2. $\pi \leftarrow \mathrm{Prove}(\mathrm{x} = (\mathrm{M}, \sigma_{\mathrm{M}}, \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}}), \mathrm{w} = \mathrm{sk}_{\mathrm{MAC}})$

$$\boxed{\begin{aligned}(\mathrm{x}, \mathrm{w}) \in R \Leftrightarrow (&\mathrm{Verify}_{\mathrm{MAC}}(\mathrm{sk}_{\mathrm{MAC}}, \mathrm{M}, \sigma_{\mathrm{M}}) = \top \\ &\wedge \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}} = \mathrm{COM}(\mathrm{sk}_{\mathrm{MAC}}))\end{aligned}}$$

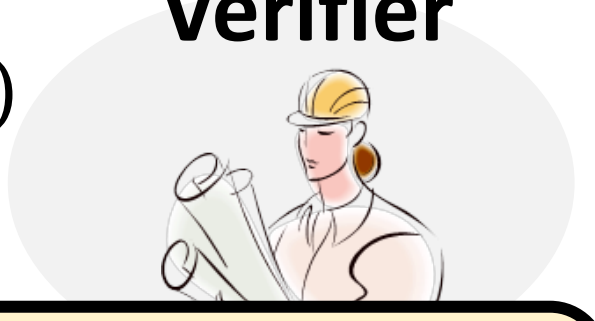
In a Bit More Detail…

*Use fact that OWF implies MAC, COM

Signer

$\mathrm{vk} = (\mathrm{crs}_{\mathrm{DP}}, \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}})$

Verifier



$\mathrm{sk} = (\mathrm{k}_{\mathrm{P}}, \mathrm{sk}_{\mathrm{MAC}})$

Prove that

✓Signature is valid

✓Signed using the committed $\mathrm{sk}_{\mathrm{MAC}}$

Sign$(\mathrm{sk}, \mathrm{M})$:

1. $\sigma_{\mathrm{M}} \leftarrow \mathrm{Sign}_{\mathrm{MAC}}(\mathrm{sk}_{\mathrm{MAC}}, \mathrm{M})$
2. $\pi \leftarrow \mathrm{Prove}(\mathrm{x} = (\mathrm{M}, \sigma_{\mathrm{M}}, \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}}), \mathrm{w} = \mathrm{sk}_{\mathrm{MAC}})$

$$\boxed{(\mathrm{x}, \mathrm{w}) \in R \Leftrightarrow (\mathrm{Verify}_{\mathrm{MAC}}(\mathrm{sk}_{\mathrm{MAC}}, \mathrm{M}, \sigma_{\mathrm{M}}) = \top \wedge \mathrm{com}_{\mathrm{sk}_{\mathrm{MAC}}} = \mathrm{COM}(\mathrm{sk}_{\mathrm{MAC}}))}$$

In a Bit More Detail…

*Use fact that OWF implies MAC, COM

Signer

$\mathrm{sk} = (\mathrm{k_P}, \mathrm{sk_{MAC}})$

$\mathrm{vk} = (\mathrm{crs_{DP}}, \mathrm{com_{sk_{MAC}}})$

$= (\pi, \sigma_\mathrm{M})$

Verifier

Sign$(\mathrm{sk}, \mathrm{M})$:

1. $\sigma_\mathrm{M} \leftarrow \mathrm{Sign_{MAC}}(\mathrm{sk_{MAC}}, \mathrm{M})$
2. $\pi \leftarrow \mathrm{Prove}(\mathrm{x} = (\mathrm{M}, \sigma_\mathrm{M}, \mathrm{com_{sk_{MAC}}}), \mathrm{w} = \mathrm{sk_{MAC}})$

$$\boxed{(\mathrm{x}, \mathrm{w}) \in R \Leftrightarrow (\mathrm{Verify_{MAC}}(\mathrm{sk_{MAC}}, \mathrm{M}, \sigma_\mathrm{M}) = \top \wedge \mathrm{com_{sk_{MAC}}} = \mathrm{COM}(\mathrm{sk_{MAC}}))}$$

Verify$(\mathrm{vk}, \mathrm{M},$ $)$:

Check validity of proof

In a Bit More Detail…

*Use fact that OWF implies MAC, COM

Signer

$\mathrm{sk} = (\mathrm{k_P}, \mathrm{sk_{MAC}})$

$\mathrm{vk} = (\mathrm{crs_{DP}}, \mathrm{com_{sk_{MAC}}})$

$= (\pi, \sigma_{\mathrm{M}})$

Verifier

Sign(sk, M

1. $\sigma_{\mathrm{M}} \leftarrow$ Si
2. $\pi \leftarrow$ Prov

$(\mathrm{x}, \mathrm{w}) \in$

$\wedge\ \mathrm{com_{sk_{MAC}}} = \mathrm{COM}(\mathrm{sk_{MAC}}))$

):

f proof

Take Away

DP-NIZK suffices since the "**signer**" is the "**designated prover**".

Lattice NIZKs and Signatures (Feasibility)

Fully-Hom. Signature

↓ [KW18]

DP-NIZK **SIS**

Correlation Interactable Hash + Fiat-Shamir

↓ [PS 19]

CRS-NIZK **LWE**

*Ignore DV-NIZK since it doesn't seem useful for signatures.

Lattice NIZKs and Signatures (Feasibility)

Fully-Hom. Signature

↓ [KW18]

DP-NIZK **SIS**

↓

Digital Signature

Correlation Interactable Hash + Fiat-Shamir

↓ [PS 19]

CRS-NIZK **LWE**

↓ [BG89]

Digital Signature

*All arrow assumes "+OWF"

Lattice NIZKs and Signatures (Feasibility)

Fully-Hom. Signature

↓ [KW18]

DP-NIZK

↓

Digital Signature

? →

Attribute-based Signature

SIS

Correlation Interactable Hash + Fiat-Shamir

↓ [PS 19]

CRS-NIZK

↓ [BG89]

Digital Signature

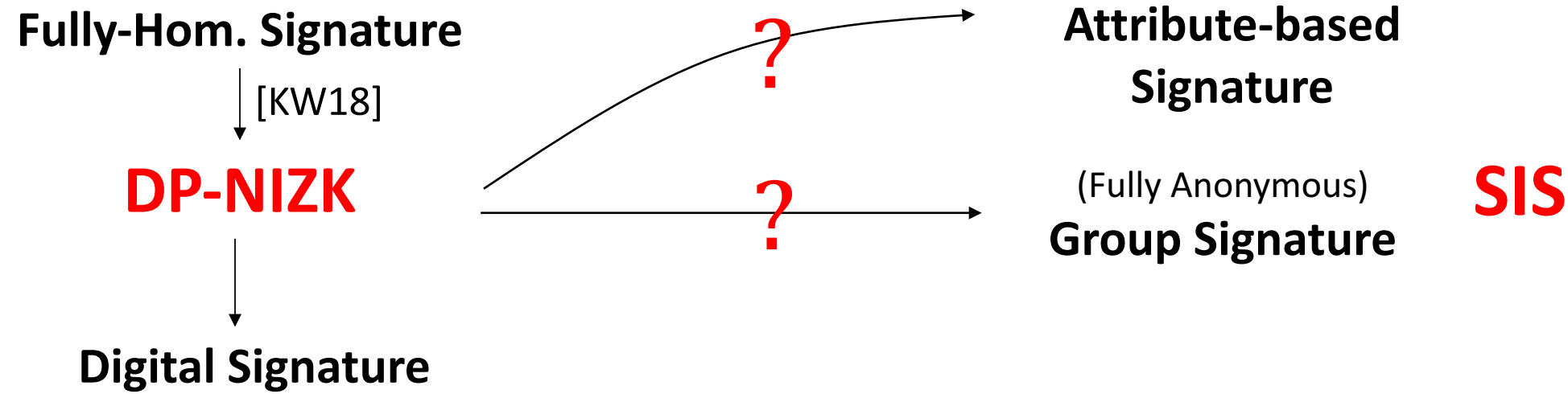
[MPR11] →

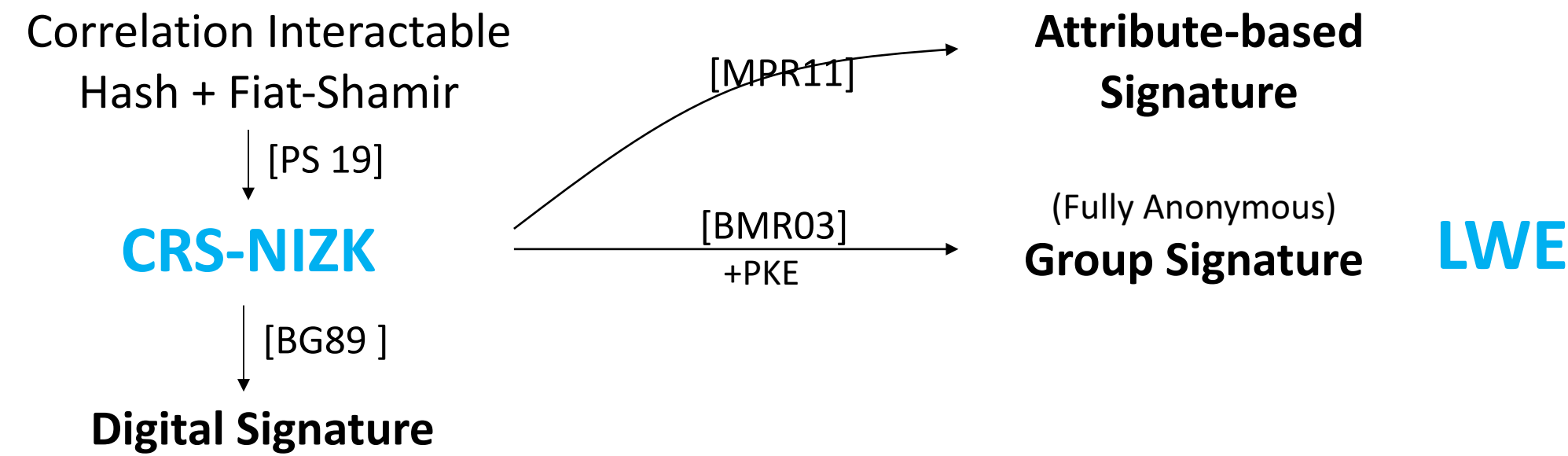
Attribute-based Signature

LWE

*All arrow assumes "+OWF"

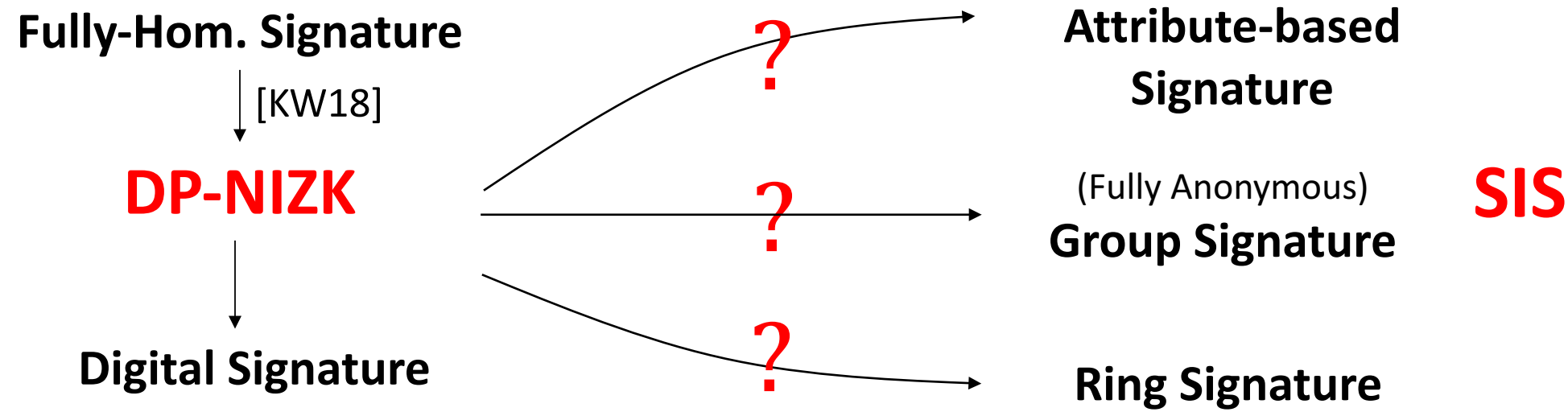
Lattice NIZKs and Signatures (Feasibility)

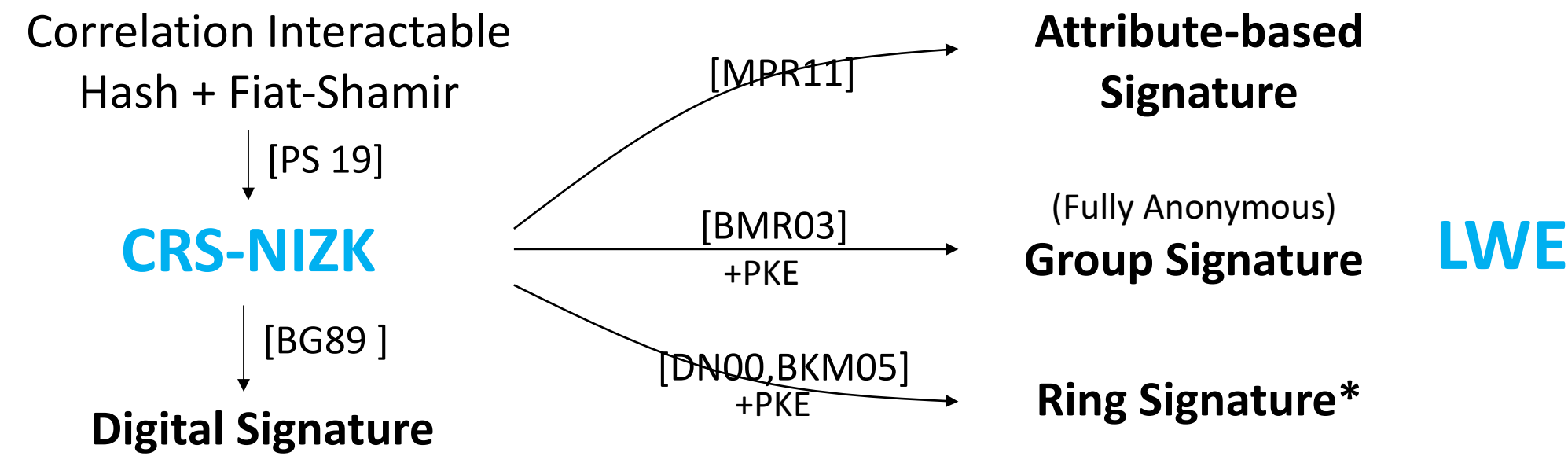




*All arrow assumes "+OWF"

Lattice NIZKs and Signatures (Feasibility)





*Unfortunately, this implication does not hold for lattice-based CRS-NIZKs ☹

Lattice NIZKs and Signatures (Feasibility)

Fully-Hom. Signature

[KW18]

DP-NIZK → [KY19] *generalization* → **Multi-User DP-NIZK**

Digital Signature

Attribute-based Signature

(Fully Anonymous) **Group Signature**

Ring Signature

SIS

Correlation Interactable Hash + Fiat-Shamir

[PS 19]

CRS-NIZK

[BG89]

Digital Signature

[MPR11] → **Attribute-based Signature**

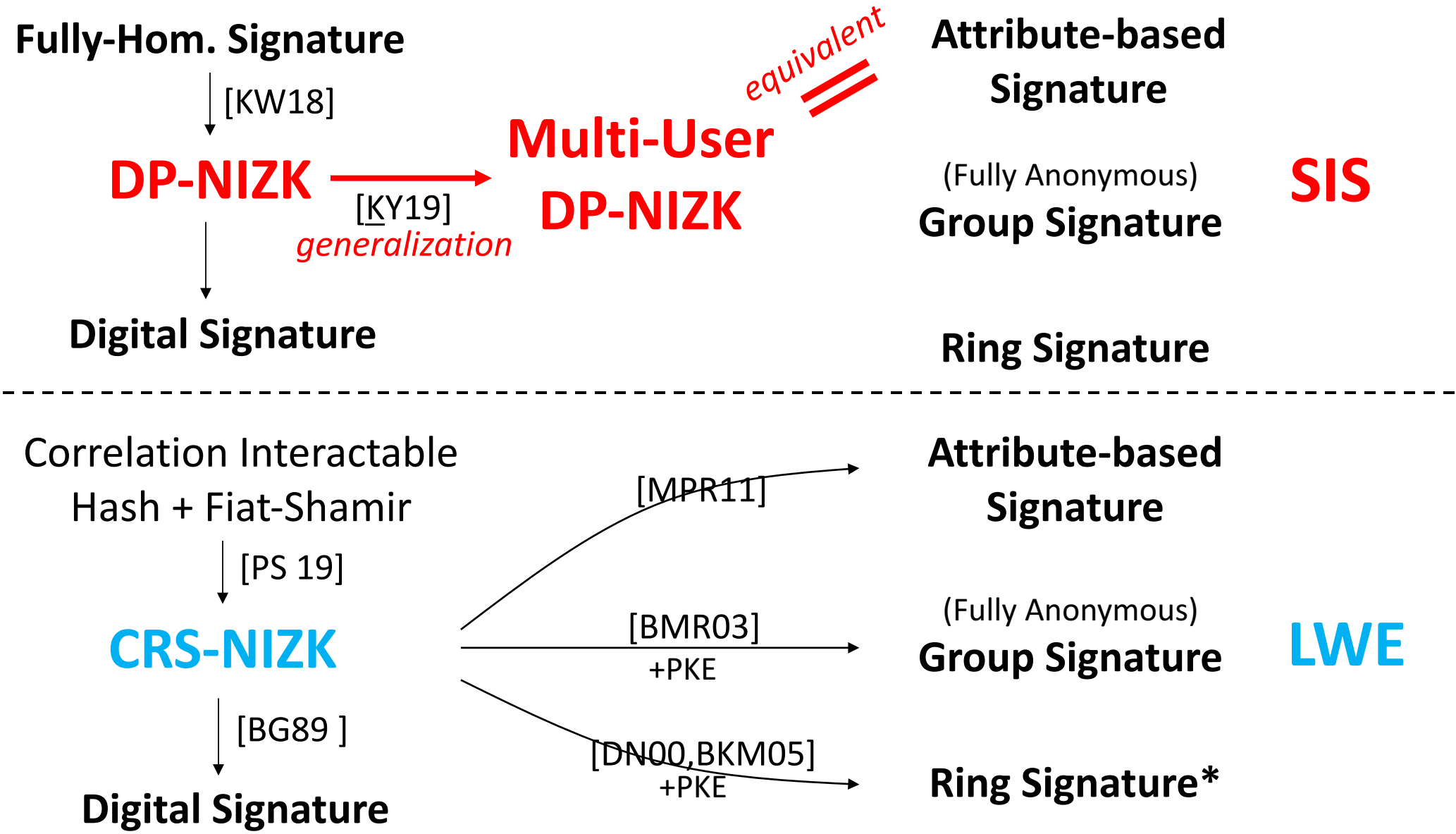
[BMR03] +PKE → (Fully Anonymous) **Group Signature**

[DN00,BKM05] +PKE → **Ring Signature***

LWE

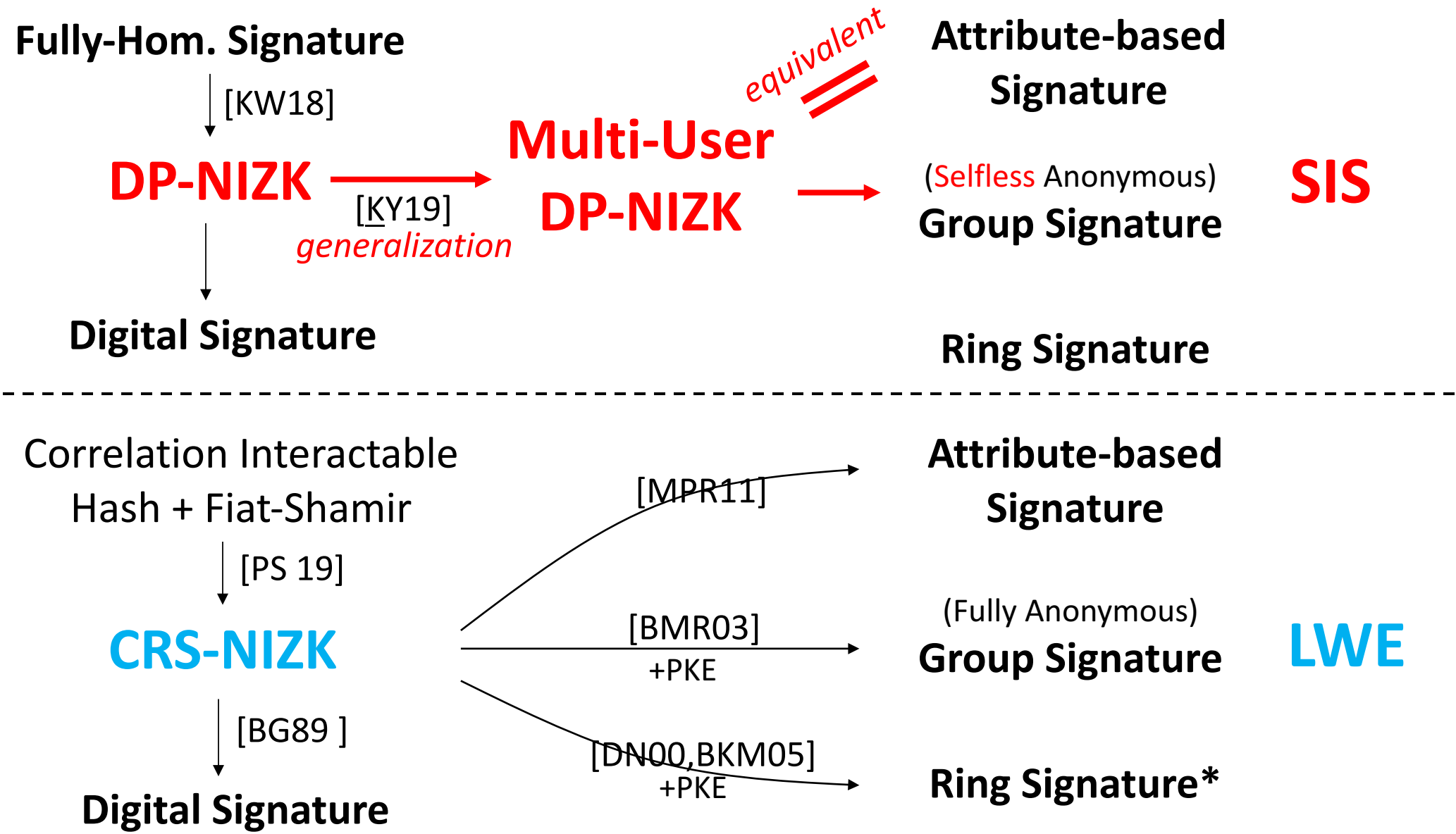
*All arrow assumes "+OWF"

Lattice NIZKs and Signatures (Feasibility)



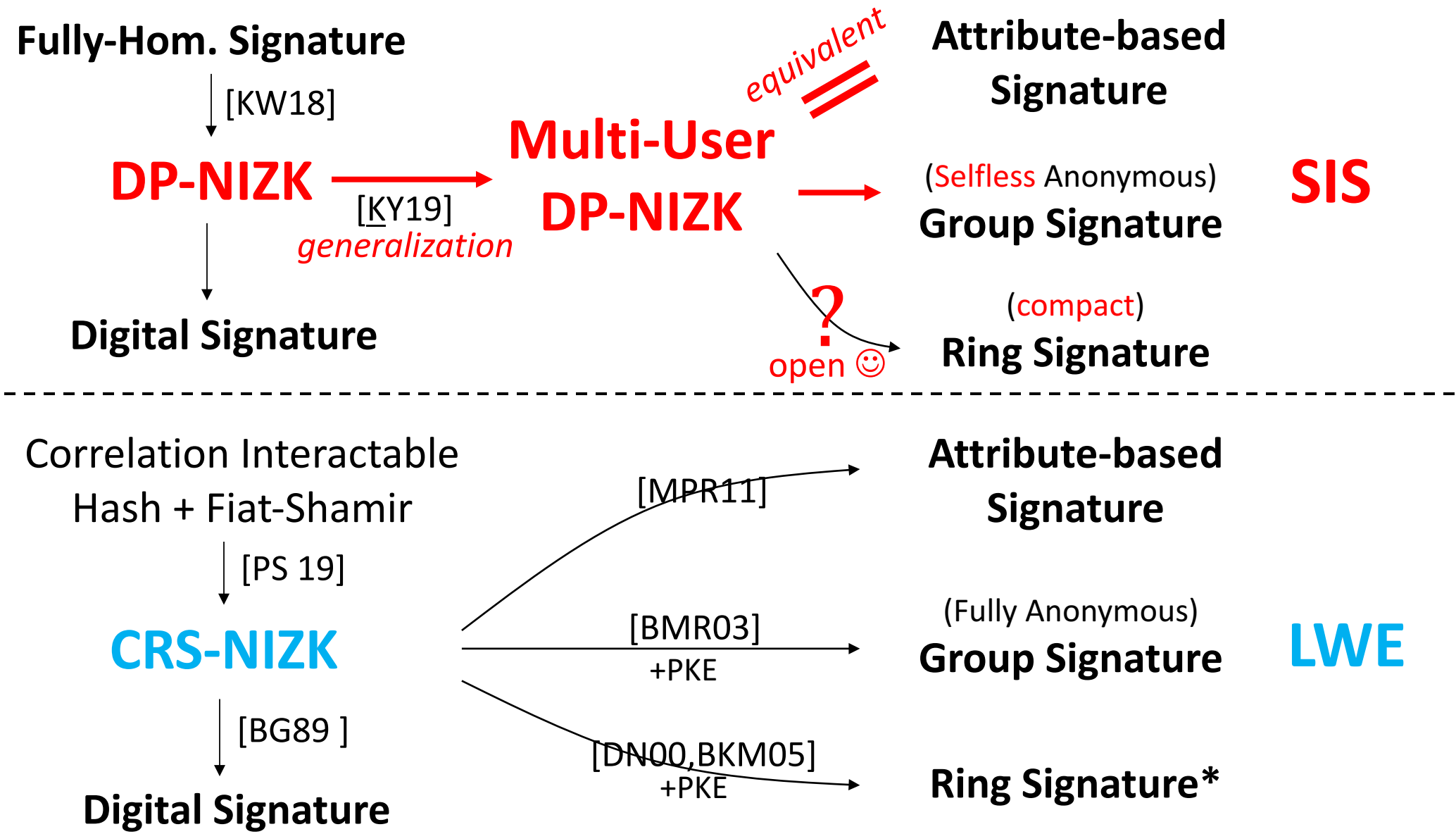
*All arrow assumes "+OWF"

Lattice NIZKs and Signatures (Feasibility)



*All arrow assumes "+OWF"

Lattice NIZKs and Signatures (Feasibility)



*All arrow assumes "+OWF"

3. Result:

Lattice-based Group Signature

[KatYam@EC'19]

via "*Multi-User DP-NIZKs*"

*Disclaimers

After our paper was accepted@EC, **CRS**-NIZK from LWE was finally resolved [PS19@CRYPTO].

Accordingly, the following presentation@EC is made under the “old” fact that CRS-NIZK from lattices do not exist yet.

Our Result in Short

① Construct the **first group signatures from lattices in the standard model**.

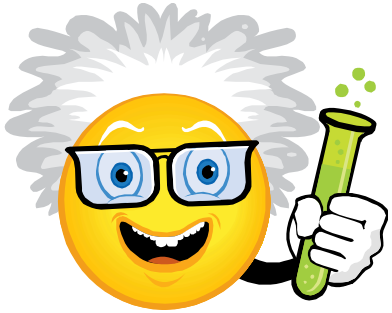
② Achieves full traceability [BMW03] and selfless anonymity [CG04].

③ Constructions from various assumptions.

- ✓ SIS w/ subexp-modulus.
- ✓ LWE w/ poly-modulus.
- ✓ SIS w/ poly-modulus + LPN w/ const. noise rate

Our Techniques in Short

Avoid using CRS-NIZK, a necessary component in existing frameworks [BMW03, CG04,…], but not known from lattices.



A) Extend DP-NIZK to **Multi-User (MU) DP-NIZK**.
B) Show MU-DP-NIZK ⇒ GS.
C) Provide construction of MU-DP-NIZK from SIS.

Outline of “3. Result”

Definition of Group Signatures

Previous Techniques

Our Work

Outline of “3. Result”

Definition of Group Signatures

Previous Techniques

Our Work

Syntax of Group Signature (GS)

$$\left(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}\right) \leftarrow \mathrm{GS.KeyGen}(1^k, 1^N)$$

Syntax of Group Signature (GS)

$$(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}) \leftarrow \mathrm{GS.KeyGen}(1^k, 1^N)$$

$$\Sigma \leftarrow \mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, \mathrm{M})$$

Syntax of Group Signature (GS)

$$(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}) \leftarrow \mathrm{GS.KeyGen}(1^k, 1^N)$$

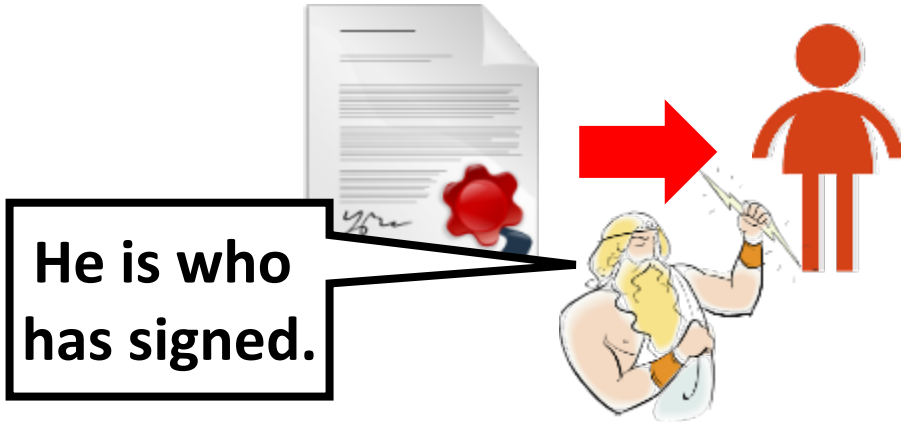
$$\Sigma \leftarrow \mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, \mathrm{M})$$

$$\top/\bot \leftarrow \mathrm{GS.Vrfy}(\mathrm{gpk}, \mathrm{M}, \Sigma)$$

*Signature signed by **SOMEBODY** in the group.

Syntax of Group Signature (GS)

$$(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}) \leftarrow \mathrm{GS.KeyGen}(1^k, 1^N)$$

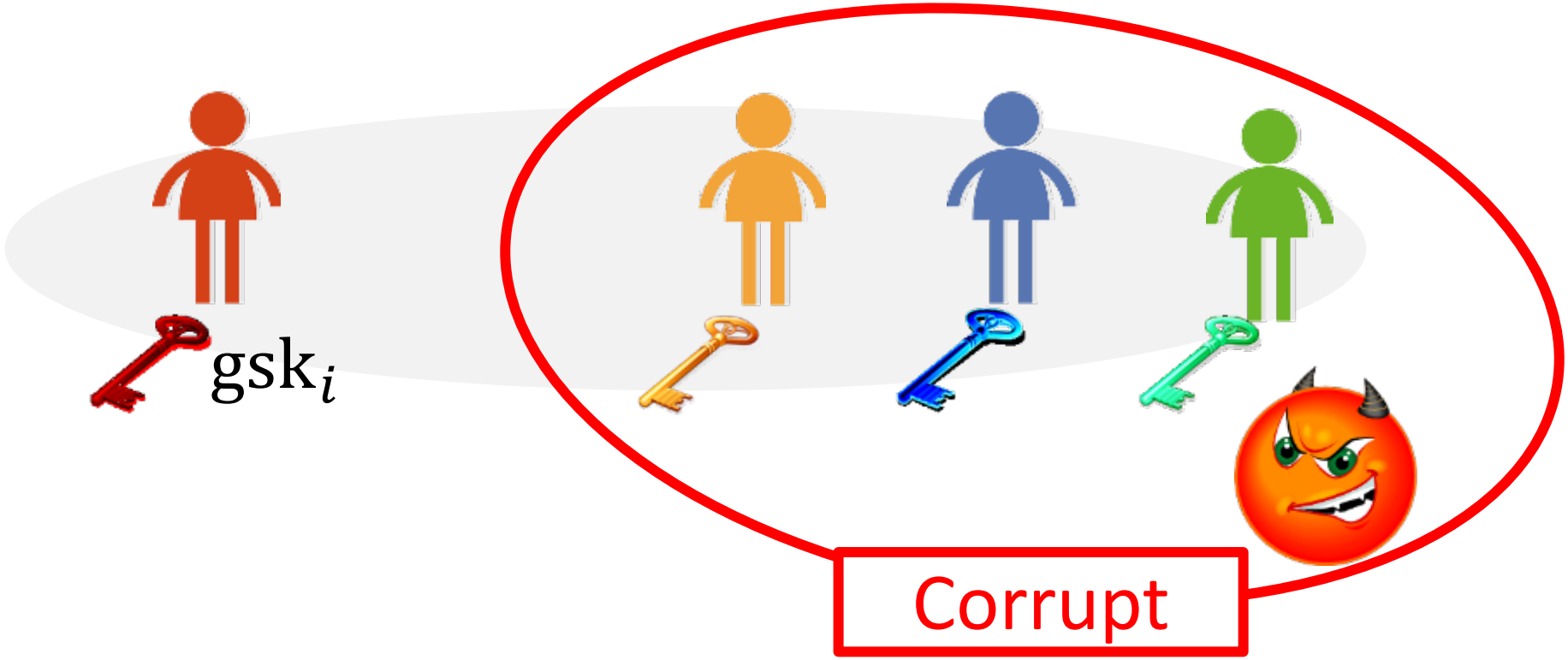


$$\Sigma \leftarrow \mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, \mathrm{M})$$

$$\top/\bot \leftarrow \mathrm{GS.Vrfy}(\mathrm{gpk}, \mathrm{M}, \Sigma)$$

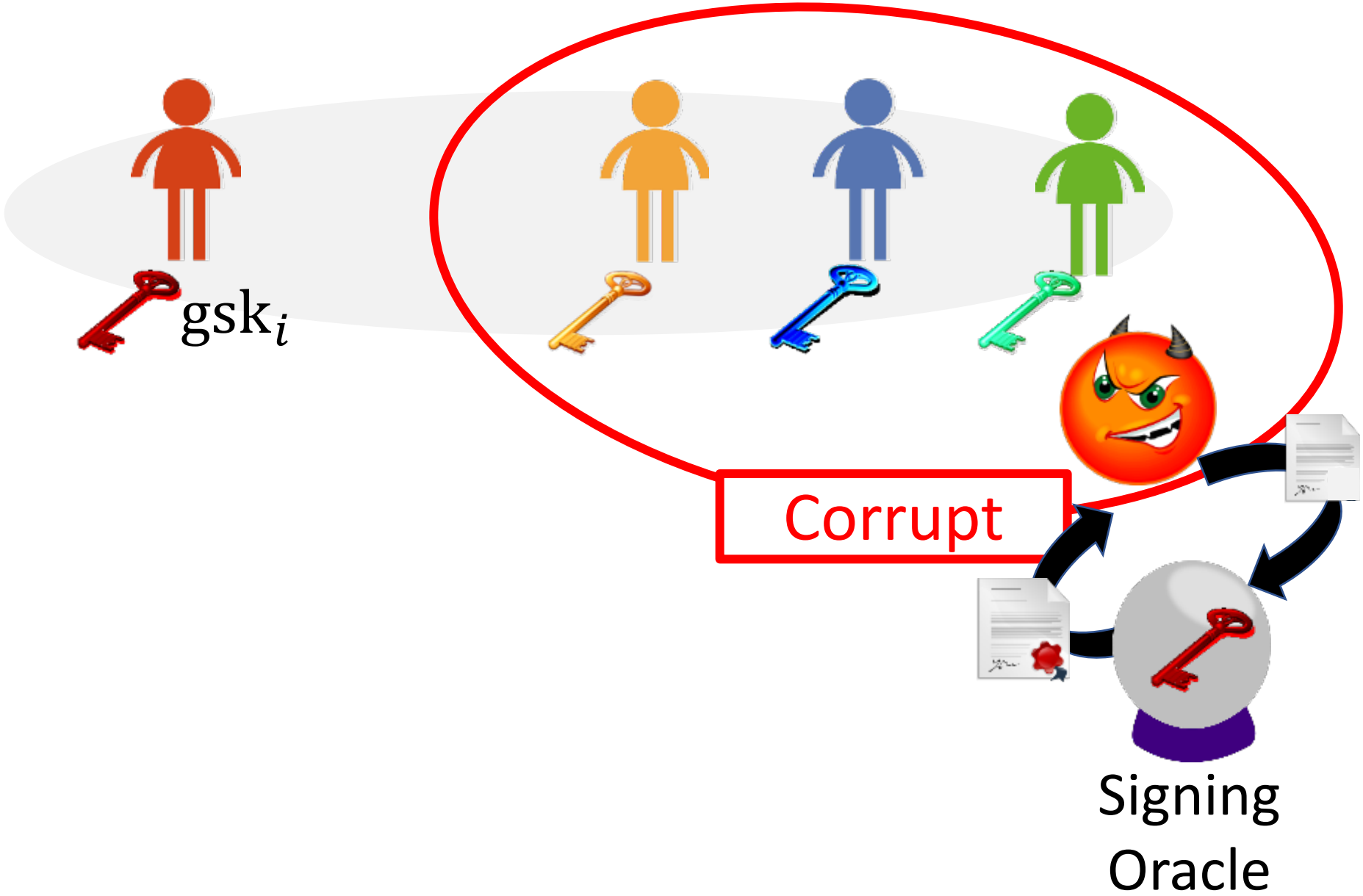
$$i/\bot \leftarrow \mathrm{GS.Open}(\mathrm{gok}, \mathrm{M}, \Sigma)$$

Security: Full Traceability [BMW03]

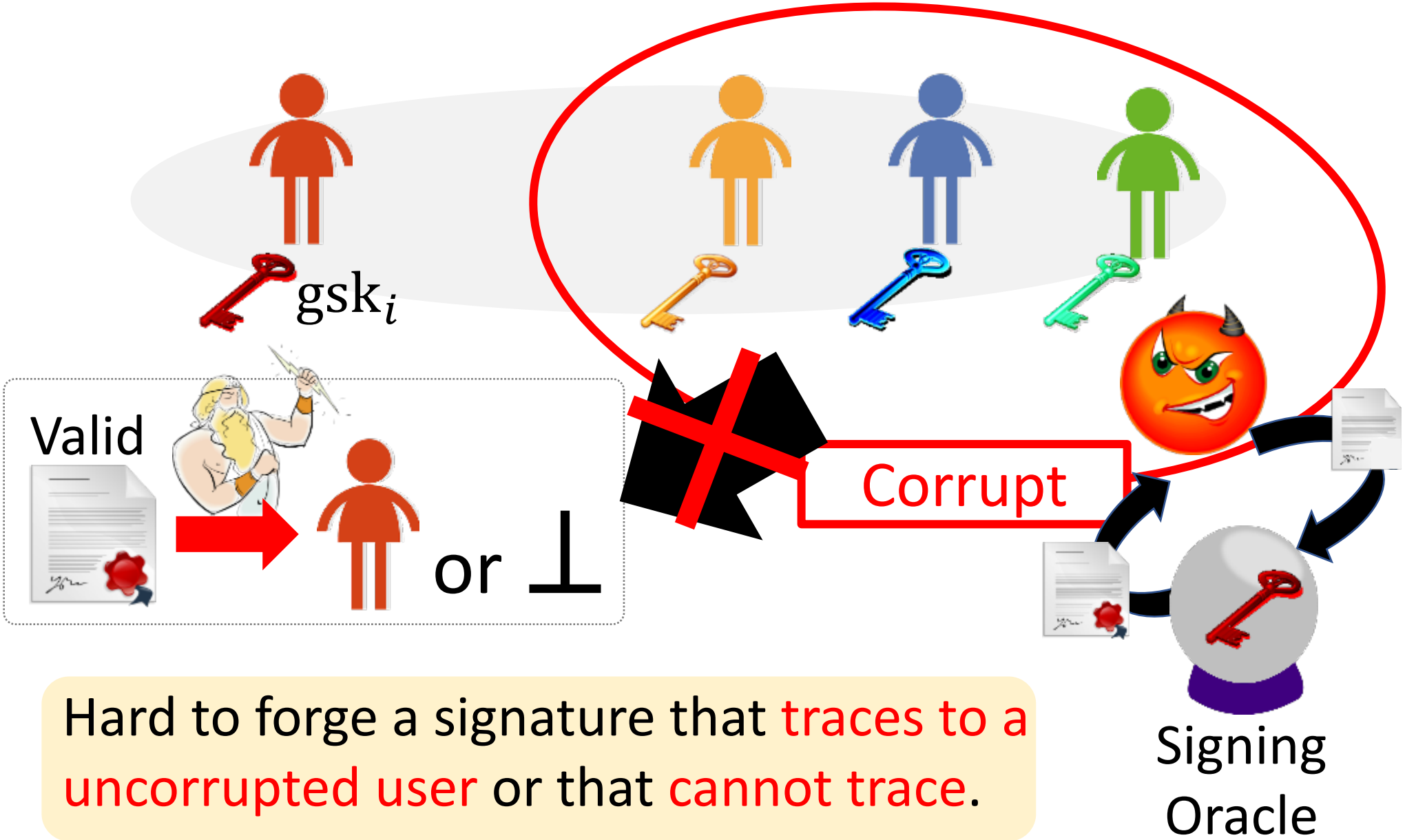


*All but 1 user

Security: Full Traceability [BMW03]

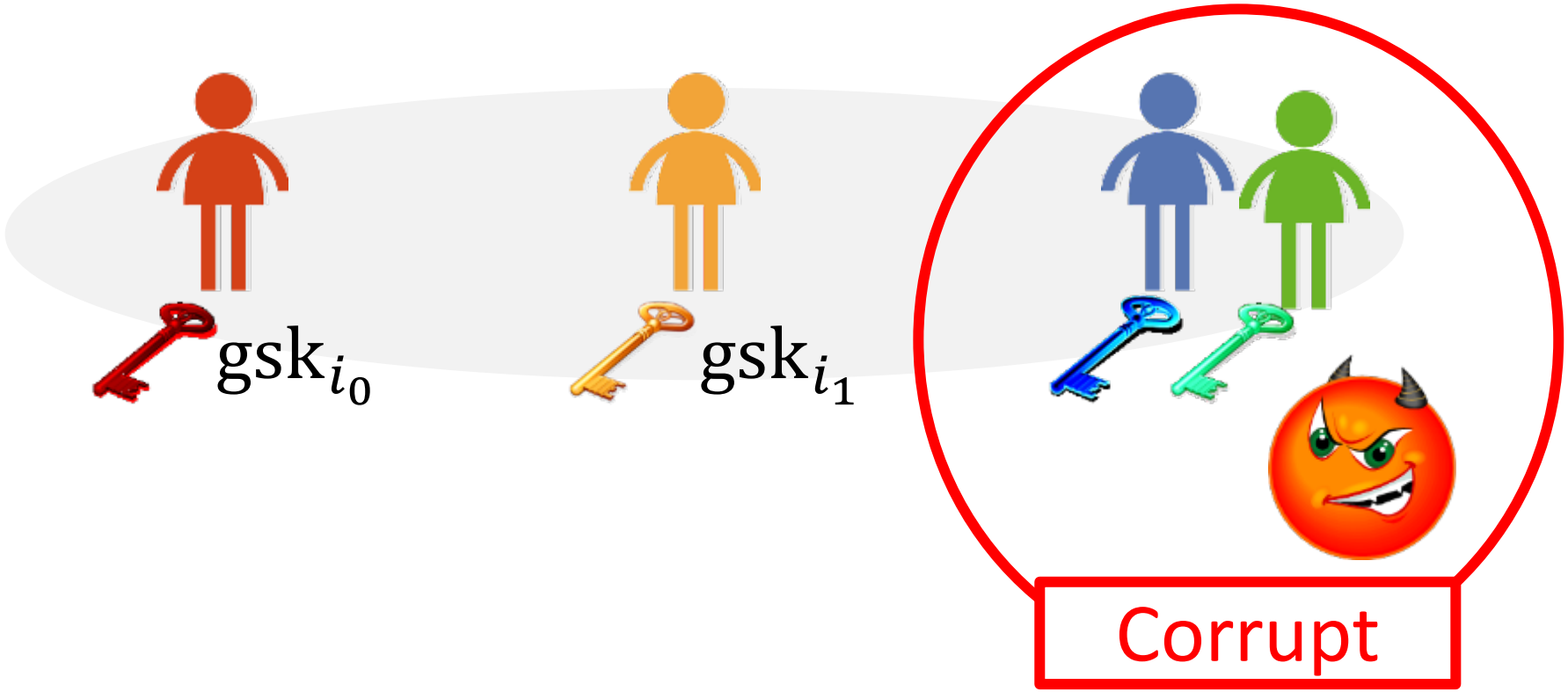


Security: Full Traceability [BMW03]



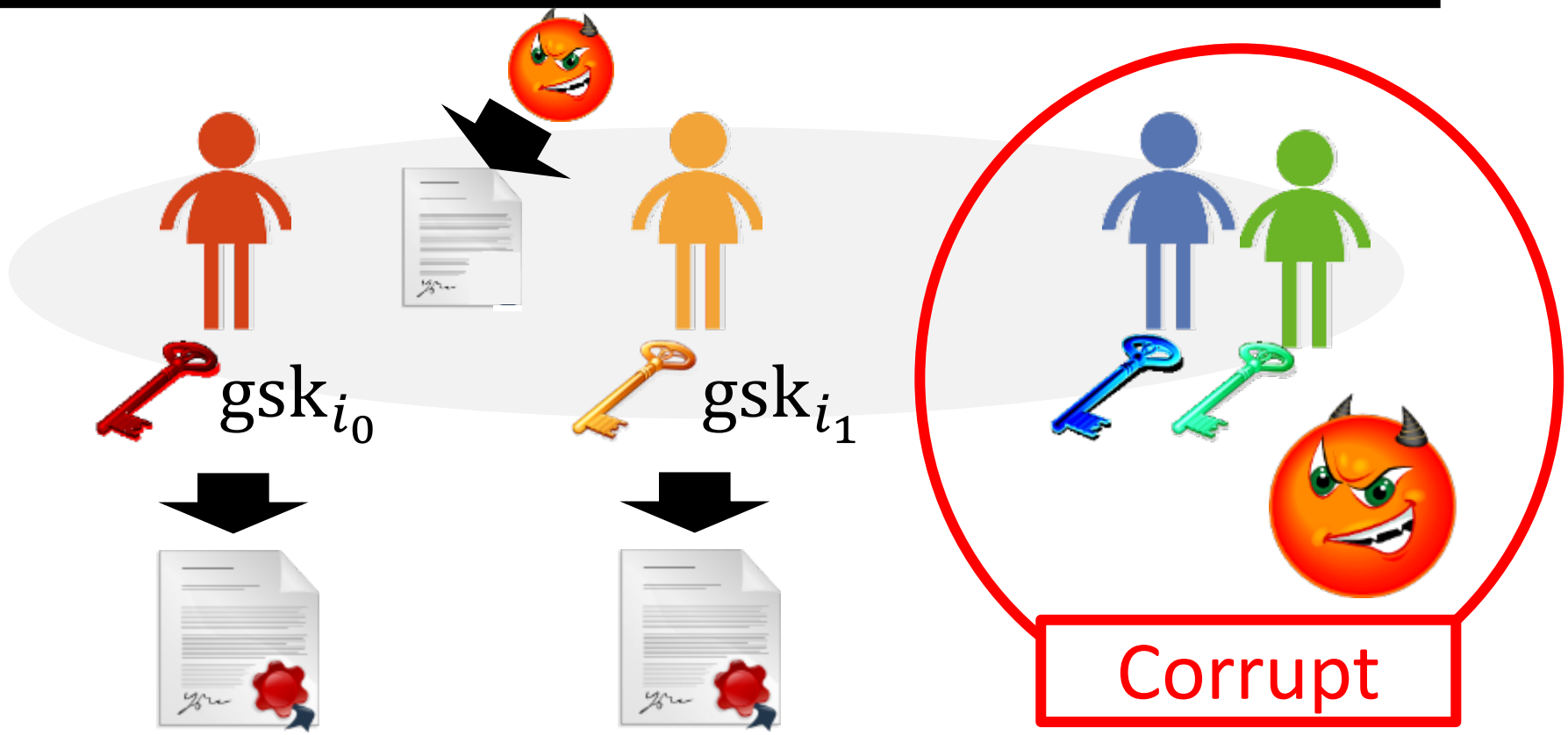
Hard to forge a signature that traces to a uncorrupted user or that cannot trace.

Security: Selfless Anonymity [CG04]

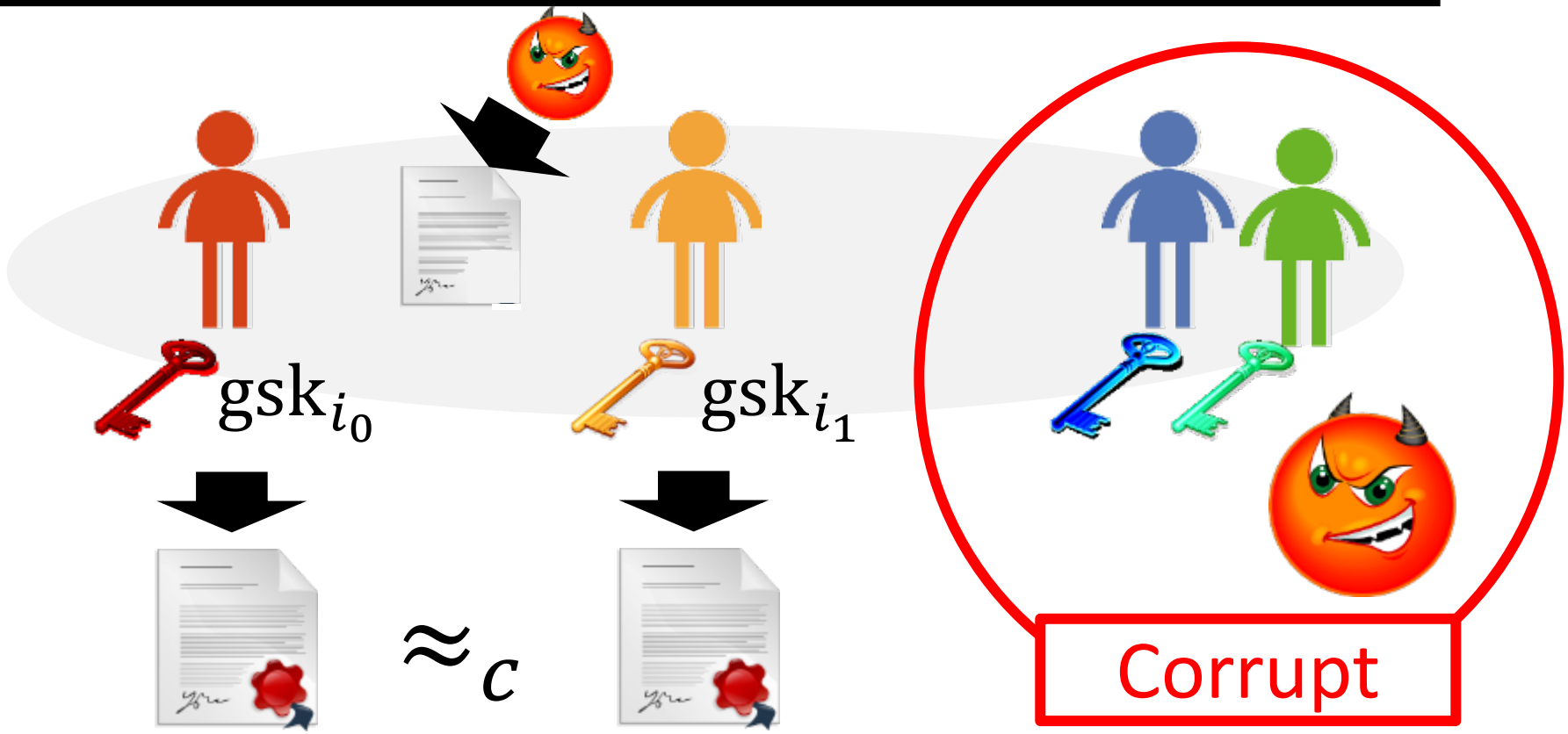


*All but 2 users

Security: Selfless Anonymity [CG04]



Security: Selfless Anonymity [CG04]

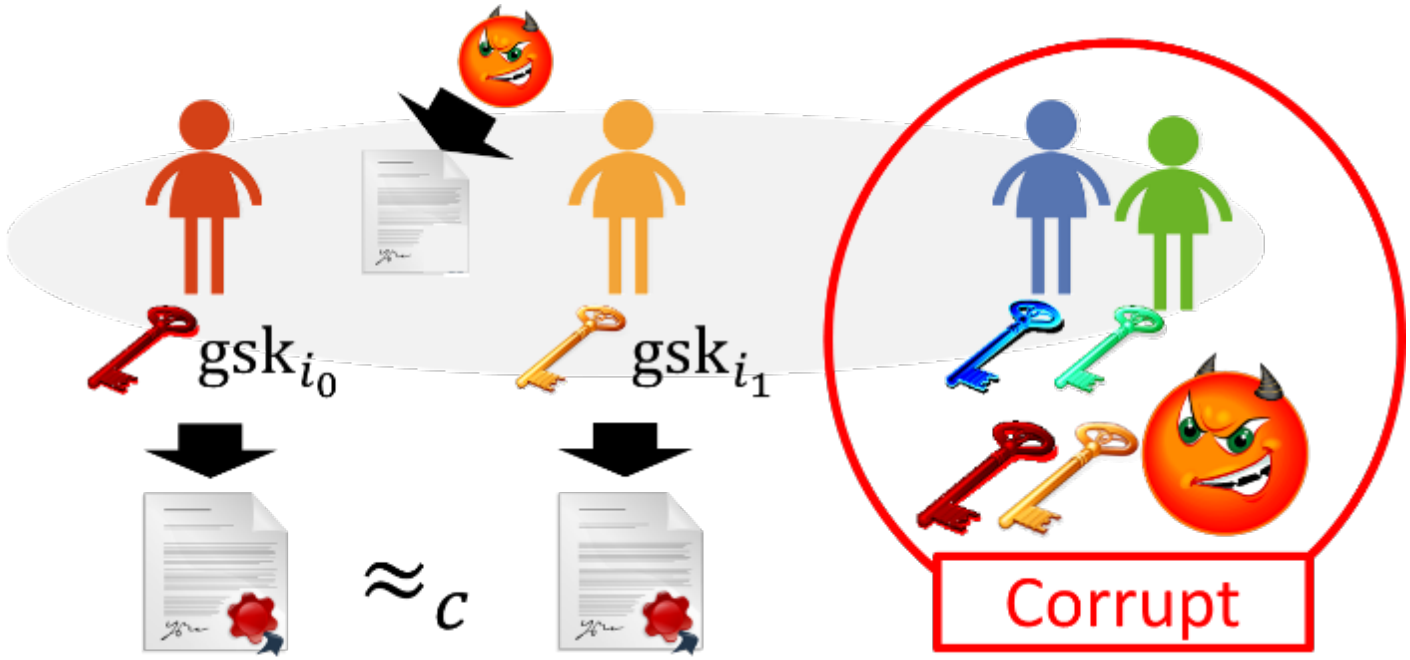


Signatures of two users i_0 and i_1 are ind,
even given gpk, $\{\mathrm{gsk}_j\}_{j \neq i_0, i_1}$ (and open oracle)

*Full Anonymity [BMW03]

A **stronger** notion than "selfless" anonymity.

The adversary is also given gsk_{i_0} and gsk_{i_1}.



Known to imply public-key encryption. [CG04,AW04]

(Hence, probably non-obtainable from SIS.)

*Full Anonymity [BMW03]

A **stronger** notion than "selfless" anonymity.

The adversary is also given gsk_{i_0} and gsk_{i_1}.

This Talk

We will only focus on "selfless" anonymity.

Known to imply pu W04]

(Hence, probably non-obtainable from SIS.)

Outline of “3. Result”

Definition of Group Signatures

Previous Techniques

Our Work

“Sign then Enc and Prove” Framework

*Modified version of original [BMW03]

Ingredients: Signature + PKE + CRS-NIZK

$$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow \left(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i \in [N]}\right)$$

“Sign then Enc and Prove” Framework

*Modified version of original [BMW03]

Ingredients: Signature + PKE + CRS-NIZK

$$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow \left(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}\right)$$

- $\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs})$

“Sign then Enc and Prove” Framework

*Modified version of original [BMW03]

Ingredients: Signature + PKE + CRS-NIZK

$$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow \left(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}\right)$$

- $\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs})$
- $\mathrm{gsk}_i = \mathrm{sk}_i$: Signing key of signature scheme

“Sign then Enc and Prove” Framework

*Modified version of original [BMW03]

Ingredients: Signature + PKE + CRS-NIZK

$$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow \left(\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i\in[N]}\right)$$

- $\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs})$
- $\mathrm{gsk}_i = \mathrm{sk}_i$: Signing key of signature scheme
- $\mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$: Secret key of PKE

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$

User $\boldsymbol{i}$

$\mathrm{gsk}_i = \mathrm{sk}_i$

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$

User $\boldsymbol{i}$

$\mathrm{gsk}_i = \mathrm{sk}_i$

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. $\sigma \leftarrow \mathrm{Sign}(\mathrm{sk}_i, \mathrm{M})$

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$

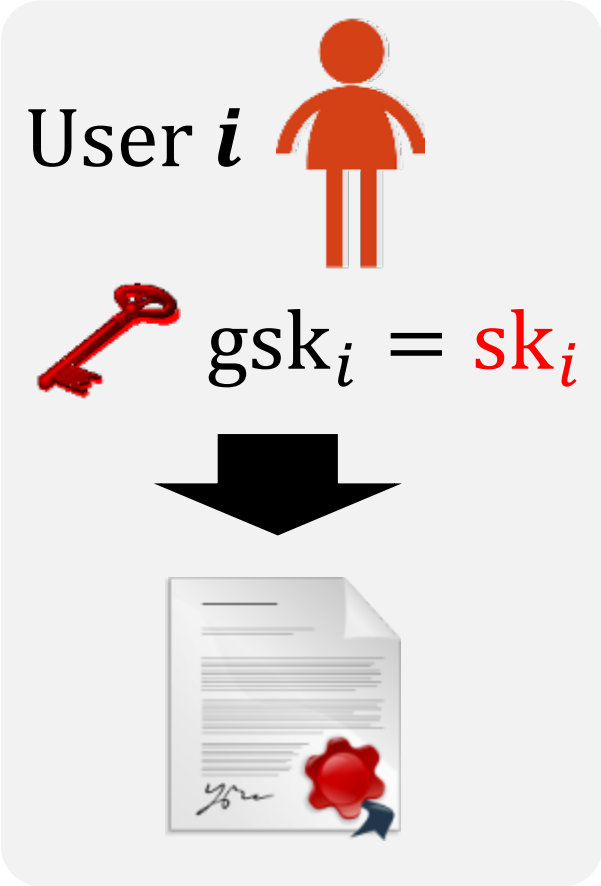
User $\boldsymbol{i}$

$\mathrm{gsk}_i = \mathrm{sk}_i$

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. $\sigma \leftarrow \mathrm{Sign}(\mathrm{sk}_i, \mathrm{M})$
  2. $\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$



- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$

  1. $\sigma \leftarrow \mathrm{Sign}(\mathrm{sk}_i, \mathrm{M})$
  2. $\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$

     and obtain proof π.

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$

User $\boldsymbol{i}$

$\mathrm{gsk}_i = \mathrm{sk}_i$

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$

1. $\sigma \leftarrow \mathrm{Sign}(\mathrm{sk}_i, \mathrm{M})$
2. $\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$
3. Prove using CRS-NIZK that $\exists R\ \exists\sigma\ \exists i$ s.t.
   - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
   - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$

   and obtain proof π.

$= (\mathrm{ct}, \pi)$

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$

User $\boldsymbol{i}$

$\mathrm{gsk}_i = \mathrm{sk}_i$

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. $\sigma \leftarrow \mathrm{Sign}(\mathrm{sk}_i, \mathrm{M})$
  2. $\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$
  3. Prove using CRS-NIZK that $\exists R\ \exists\sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$

     and obtain proof π.

$= (\mathrm{ct}, \pi)$

- GS.Verify $\Rightarrow$ Verify π

“Sign then Enc and Prove” Framework

$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]},\ \mathrm{pk}_{\mathrm{PKE}},\ \mathrm{crs}),\quad \mathrm{gok} = \mathrm{sk}_{\mathrm{PKE}}$

User $\boldsymbol{i}$

$\mathrm{gsk}_i = \mathrm{sk}_i$

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. $\sigma \leftarrow \mathrm{Sign}(\mathrm{sk}_i, \mathrm{M})$
  2. $\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$
  3. Prove using CRS-NIZK that $\exists R\ \exists\sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$

     and obtain proof π.

$= (\mathrm{ct}, \pi)$

- GS.Verify $\Rightarrow$ Verify π
- GS.Open $\Rightarrow$ Dec. ct

Intuition for Security: Anonymity

Prove user i's information doesn't leak from  .

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \boxed{\exists i}$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π.

 $= \boxed{(\mathrm{ct}, \pi)}$

Intuition for Security: Anonymity

Prove user i's information doesn't leak from  .

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \boxed{\exists i}$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π.

 $= \boxed{(\mathrm{ct}, \pi)}$

- $\boldsymbol{\pi}$ reveals nothing about i due to CRS-NIZK being ZK.

Intuition for Security: Anonymity

Prove user i's information doesn't leak from  .

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \boxed{\exists i}$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π.

 $= \boxed{(\mathrm{ct}, \pi)}$

- $\boldsymbol{\pi}$ reveals nothing about i due to CRS-NIZK being ZK.
- **ct** reveals nothing about i due to security of PKE.

Intuition for Security: Anonymity

Prove user i's information doesn't leak from .

- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \boxed{\exists i}$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π.

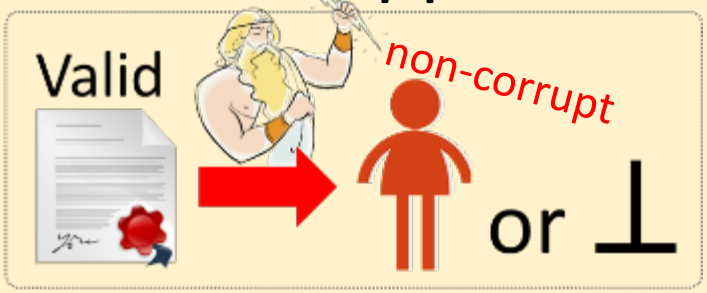
$= \boxed{(\mathrm{ct}, \pi)}$

- $\boldsymbol{\pi}$ reveals nothing about i due to CRS-NIZK being ZK.
- **ct** reveals nothing about i due to security of PKE.

The information of who signed the message is hidden!

Intuition for Security: Traceability

Prove the following doesn't happen.



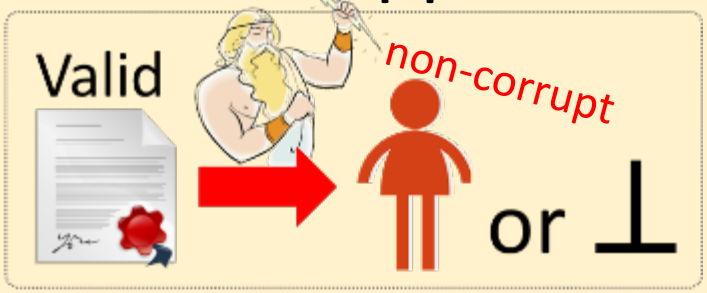
- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π.

$= (\mathrm{ct}, \pi)$

Intuition for Security: Traceability

Prove the following doesn't happen.



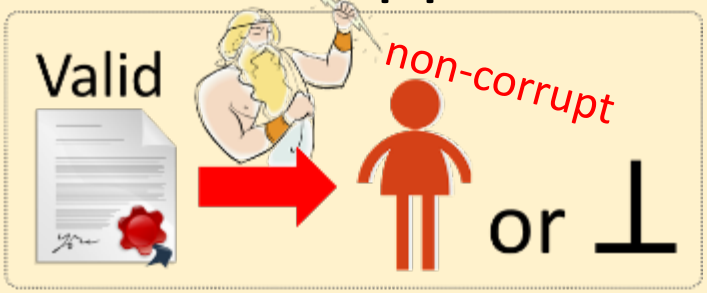
- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π. $= (\mathrm{ct}, \pi)$

- Due to soundness of CRS-NIZK, must have ct of the form $\mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$ for $\exists R\ \exists \sigma\ \exists i$.

Intuition for Security: Traceability

Prove the following doesn't happen.



- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

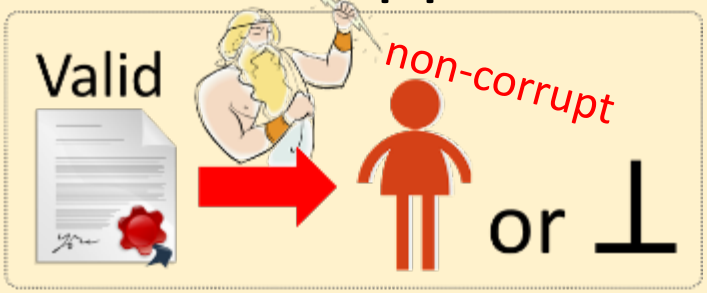
     and obtain proof π. $= (\mathrm{ct}, \pi)$

- Due to soundness of CRS-NIZK, must have ct of the form $\mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$ for $\exists R\ \exists \sigma\ \exists i$.

  $\Rightarrow$ Open algorithm does not output $\perp$.

Intuition for Security: Traceability

Prove the following doesn't happen.



- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π. $= (\mathrm{ct}, \pi)$

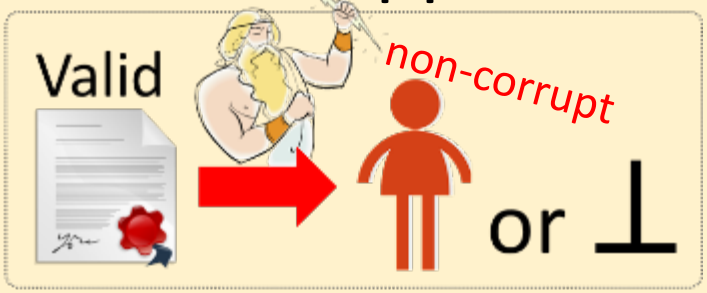
- Due to soundness of CRS-NIZK, must have ct of the form $\mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$ for $\exists R\ \exists \sigma\ \exists i$.

  $\Rightarrow$ Open algorithm does not output $\perp$.

- Due to unforgeability of signature, impossible to forge σ s.t. $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$ for **non-corrupt** user i.

Intuition for Security: Traceability

Prove the following doesn't happen.



- $\mathrm{GS.Sign}(\mathrm{gpk}, \mathrm{gsk}_i, M)$
  1. 2. (Skip)
  3. Prove using CRS-NIZK that $\exists R\ \exists \sigma\ \exists i$ s.t.
     - $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$
     - $\mathrm{ct} = \mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}}\ \ i||\sigma;\ R)$

     and obtain proof π. $= (\mathrm{ct}, \pi)$

- Due to soundness of CRS-NIZK, must have ct of the form $\mathrm{Enc}(\mathrm{pk}_{\mathrm{PKE}},\ i||\sigma;\ R)$ for $\exists R\ \exists \sigma\ \exists i$.

  $\Rightarrow$ Open algorithm does not output $\perp$.

- Due to unforgeability of signature, impossible to forge σ s.t. $\mathrm{Verify}(\mathrm{vk}_i, \sigma, \mathrm{M}) = \top$ for **non-corrupt** user i.

  $\Rightarrow$ Open algorithm does not output non-corrupt user.

Outline of “3. Result”

Definition of Group Signatures

Previous Techniques

Our Work

Motivation of this Work

How to construct lattice-based GS w/o CRS-NIZK??

Ingredients: Signature + PKE + CRS-NIZK

$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow (\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i \in [N]})$

Motivation of this Work

How to construct lattice-based GS w/o CRS-NIZK??

Ingredients: Signature + PKE + $\boxed{\text{CRS-NIZK}}$

$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow (\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i \in [N]})$

- ✓ If we can replace CRS-NIZK with DP-NIZK, then we can use SIS-based construction of [KW18].
- ✓ Getting away with SKE instead of PKE is easy.

Motivation of this Work

How to construct lattice-based GS w/o CRS-NIZK??

Ingredients: Signature + PKE + $\boxed{\text{CRS-NIZK}}$

$\mathrm{GS.KeyGen}(1^k, 1^N) \rightarrow (\mathrm{gpk}, \mathrm{gok}, \{\mathrm{gsk}_i\}_{i \in [N]})$

- ✓ If we can replace CRS-NIZK with DP-NIZK, then we can use SIS-based construction of [KW18].
- ✓ Getting away with SKE instead of PKE is easy.

▶ Will result in the first lattice-based GS! Moreover, from the SIS assumption ☺

Overview of Our Approach

① Unfortunately, simply plugging in DP-NIZK in place of CRS-NIZK does not work ☹

Overview of Our Approach

① Unfortunately, simply plugging in DP-NIZK in place of CRS-NIZK does not work ☹

② To avoid problem, we introduce a **Multi-User DP-NIZK** and use it as replacement.

Overview of Our Approach

① Unfortunately, simply plugging in DP-NIZK in place of CRS-NIZK does not work ☹

② To avoid problem, we introduce a **Multi-User DP-NIZK** and use it as replacement.

③ We observe that MU-DP-NIZK is implied by attribute-based signatures (**ABS**) and construct a new lattice-based ABS.

Recap: DP-NIZK

CRS: (public) common reference string

$\mathbf{k_p}$: (private) proving key

Prover (x, w)

CRS, $x \in L$

Verifier x

$\mathbf{k_p}$

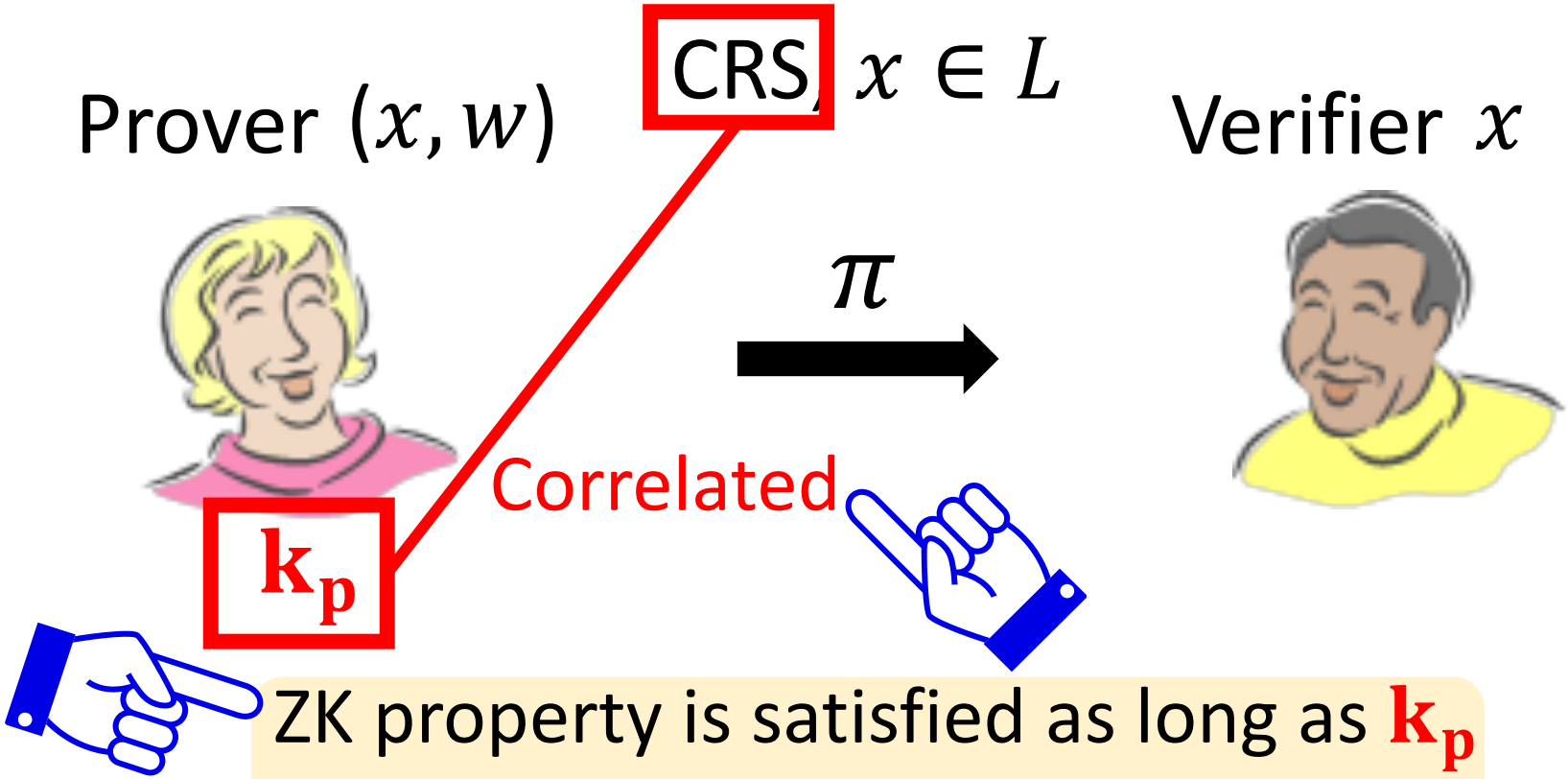
π

ZK property is satisfied as long as $\mathbf{k_p}$ is kept secret from the Verifier.

Recap: DP-NIZK

CRS: (public) common reference string

$\mathbf{k_p}$: (private) proving key



ZK property is satisfied as long as $\mathbf{k_p}$ is kept secret from the Verifier.

First Attempt

Plug in DP-NIZK in [BMW03] framework.

<u>Group public key:</u>

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk}_{\mathrm{PKE}}, \quad \mathrm{crs})$$

First Attempt

Plug in DP-NIZK in [BMW03] framework.

<u>Group public key:</u>

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk}_{\mathrm{PKE}}, \quad \mathrm{crs})$$

<u>Group signing key for user i:</u>

$$\mathrm{gsk}_i = (\mathrm{sk}_i, \mathrm{k_p})$$

Proving key for DP-NIZK
(Same for all users)

First Attempt

Plug in DP-NIZK in [BMW03] framework.

Group public key:

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk}_{\mathrm{PKE}}, \quad \mathrm{crs})$$

Group signing key for user i:

$$\mathrm{gsk}_i = (\mathrm{sk}_i, \mathrm{k_p})$$

Proving key for DP-NIZK
(Same for all users)

Signature made by user i:

$$\Sigma = (\mathrm{ct}, \pi)$$

DP-NIZK proof

First Attempt

Plug in DP-NIZK in [BMW03] framework.

Group public key:

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk}_{\mathrm{PKE}}, \quad \mathrm{crs})$$

Group signing key for user i:

$$\mathrm{gsk}_i = (\mathrm{sk}_i, \mathrm{k_p})$$

Proving key for DP-NIZK
(Same for all users)

Signature made by user i:

$$\Sigma = (\mathrm{ct}, \pi)$$

DP-NIZK proof

Corruption of single user reveals $\mathrm{k_P}$

$\Rightarrow$ Ruins ZK property of DP-NIZK

$\Rightarrow$ Breaks Anonymity of the resulting GS

Second Attempt

Use different $\mathrm{k_p}$ for different users.

<u>Group public key:</u>

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk_{PKE}}, \quad \mathrm{crs}^{(1)}, \ldots, \mathrm{crs}^{(N)})$$

Second Attempt

Use different $\mathrm{k_p}$ for different users.

<u>Group public key:</u>

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk_{PKE}}, \quad \mathrm{crs}^{(1)}, \ldots, \mathrm{crs}^{(N)})$$

<u>Group signing key for user i:</u>

$$\mathrm{gsk}_i = (\mathrm{sk}_i, \mathrm{k_p}^{(i)})$$

Proving key for the ***i*-th instance** of DP-NIZK

Second Attempt

Use different $\mathrm{k_p}$ for different users.

Group public key:

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk_{PKE}}, \quad \mathrm{crs}^{(1)}, \ldots, \mathrm{crs}^{(N)})$$

Group signing key for user i:

$$\mathrm{gsk}_i = (\mathrm{sk}_i, \mathrm{k_p}^{(i)})$$

Proving key for the ***i*-th instance** of DP-NIZK

Signature made by user i:

$$\Sigma = (\mathrm{ct}, \pi)$$

W.R.T $\mathrm{crs}^{(i)}$

Second Attempt

Use different $\mathrm{k_p}$ for different users.

<u>Group public key:</u>

$$\mathrm{gpk} = (\{\mathrm{vk}_i\}_{i\in[N]}, \quad \mathrm{pk_{PKE}}, \quad \mathrm{crs}^{(1)}, \ldots, \mathrm{crs}^{(N)})$$

<u>Group signing key for user i:</u>

$$\mathrm{gsk}_i = (\mathrm{sk}_i, \mathrm{k_p}^{(i)})$$

Proving key for the ***i*-th instance** of DP-NIZK

<u>Signature made by user i:</u>

$$\Sigma = (\mathrm{ct}, \pi)$$

W.R.T $\mathrm{crs}^{(i)}$

The DP-NIZK proof (GS signature) does not hide the instance i.

$\Rightarrow$ Breaks Anonymity of the resulting GS

Lesson Learned from Failures

- Need multiple prover variant of DP-NIZK.
- Need security against corruption of provers.
- Proof should not leak prover identity.

DP-NIZK seems to be too weak….

Lesson Learned from Failures

- Need multiple prover variant of DP-NIZK.
- Need security against corruption of provers.
- Proof should not leak prover identity.

DP-NIZK seems to be too weak….

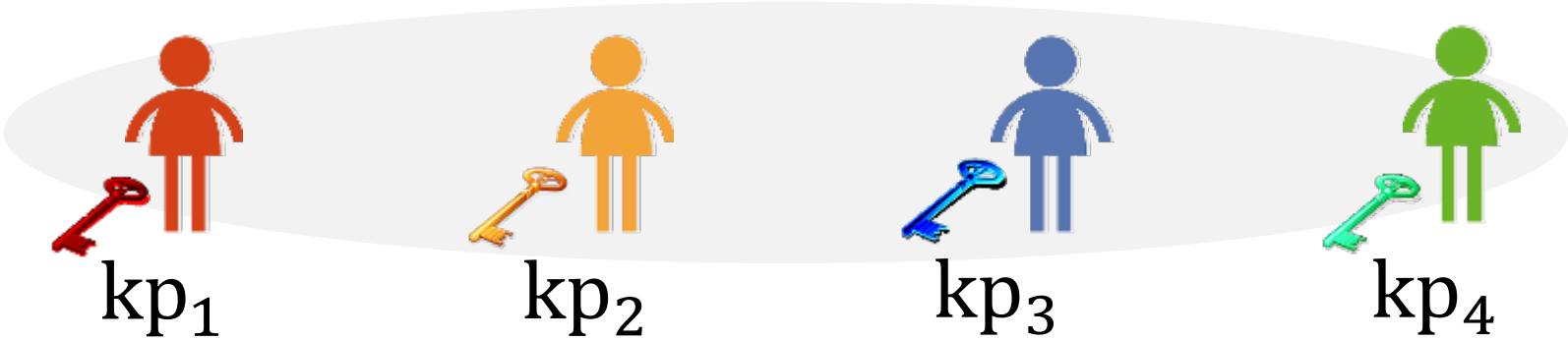
Our Solution

Construct anonymous ***Multi-User DP-NIZK***
(Attribute-based signature + [KW18] technique)

New Notion: Multi-User DP-NIZK

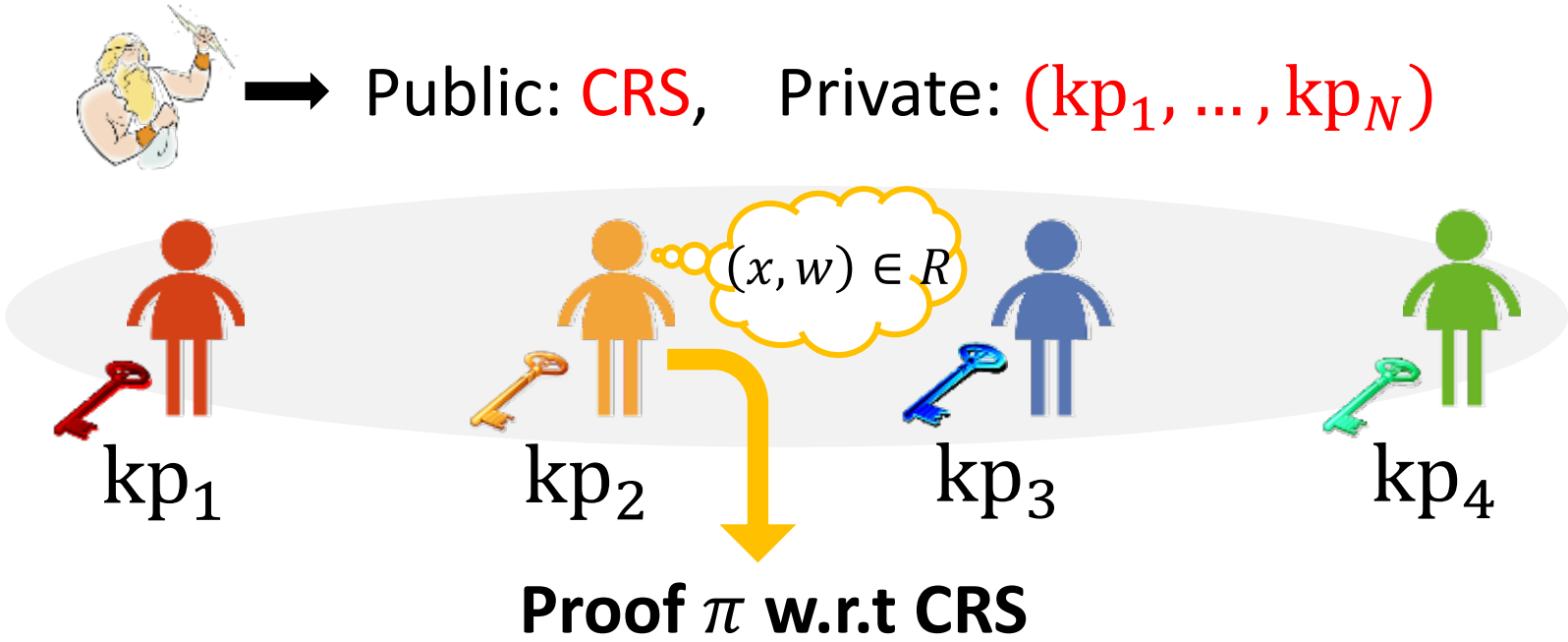
Essentially, a DP-NIZK with multiple users ☺

→ Public: CRS, Private: $(\mathrm{kp}_1, \ldots, \mathrm{kp}_N)$



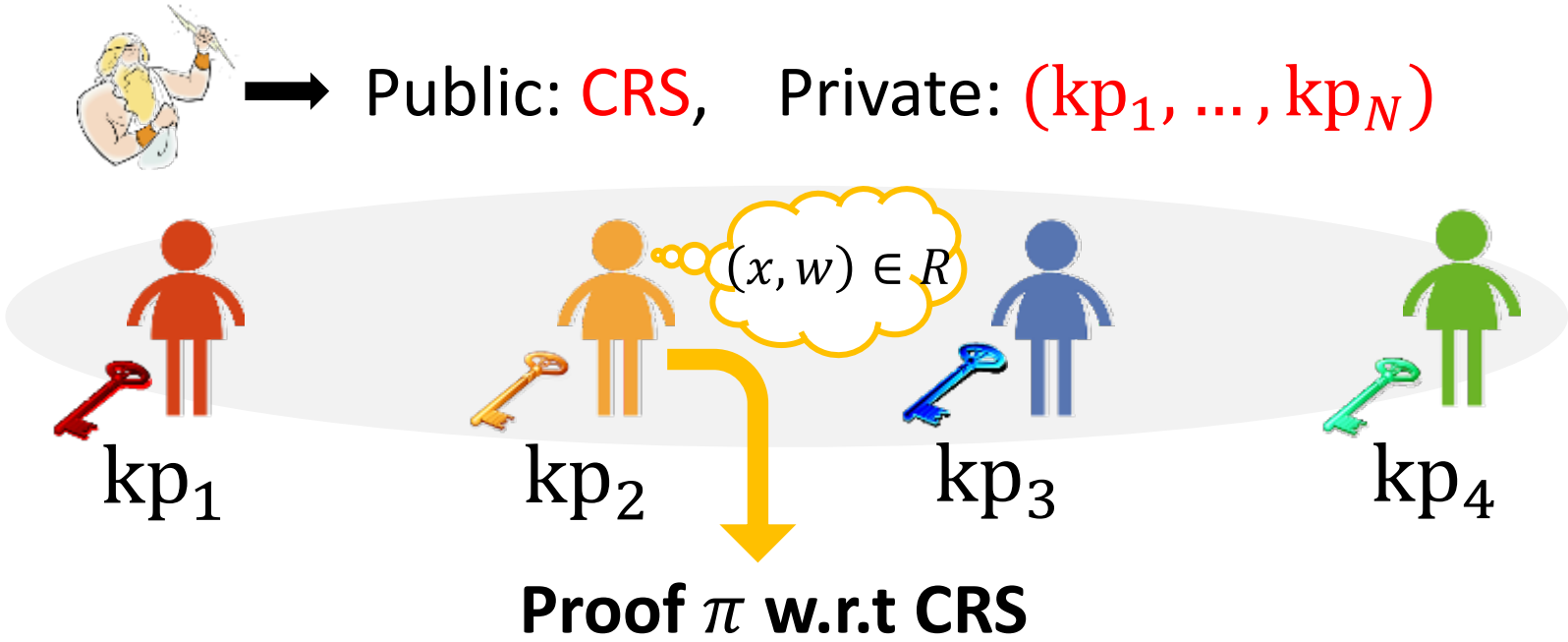
New Notion: Multi-User DP-NIZK

Essentially, a DP-NIZK with multiple users ☺



New Notion: Multi-User DP-NIZK

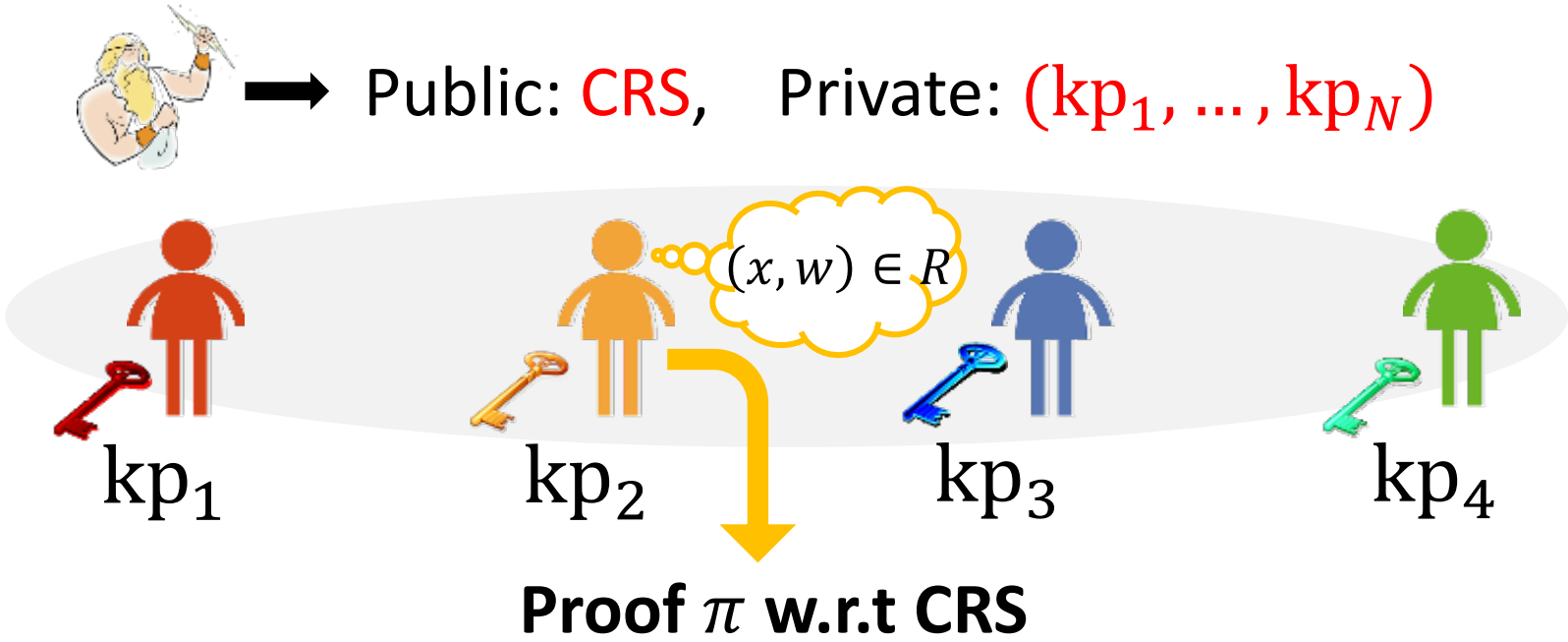
Essentially, a DP-NIZK with multiple users ☺



✓ **Zero-Knowledge**: π leaks no information even with corruption.

New Notion: Multi-User DP-NIZK

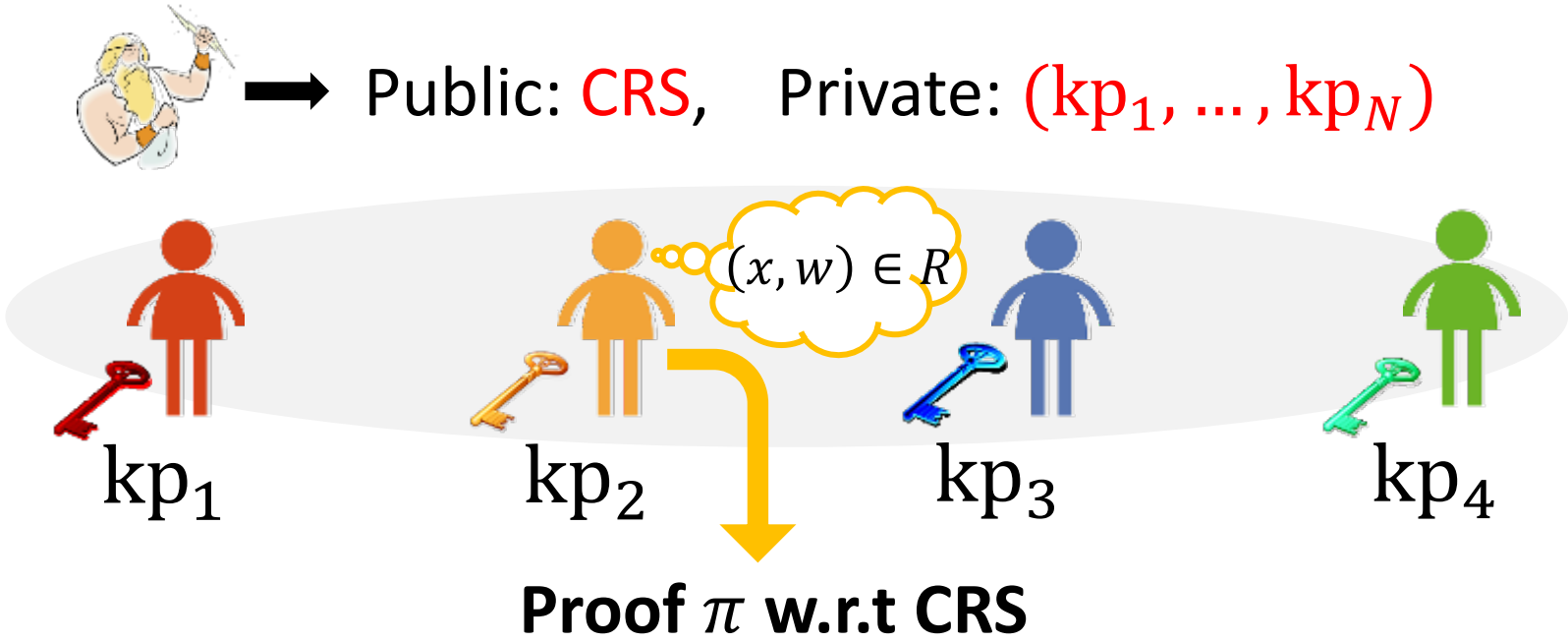
Essentially, a DP-NIZK with multiple users ☺



- ✓ **Zero-Knowledge**: π leaks no information even with corruption.
- ✓ **Soundness**: $x \notin L$ cannot be proven even with corruption.

New Notion: Multi-User DP-NIZK

Essentially, a DP-NIZK with multiple users ☺



- ✓ **Zero-Knowledge**: π leaks no information even with corruption.
- ✓ **Soundness**: $x \notin L$ cannot be proven even with corruption.
- ✓ **Anonymity**: Information of who generated π is not leaked even with corruption.

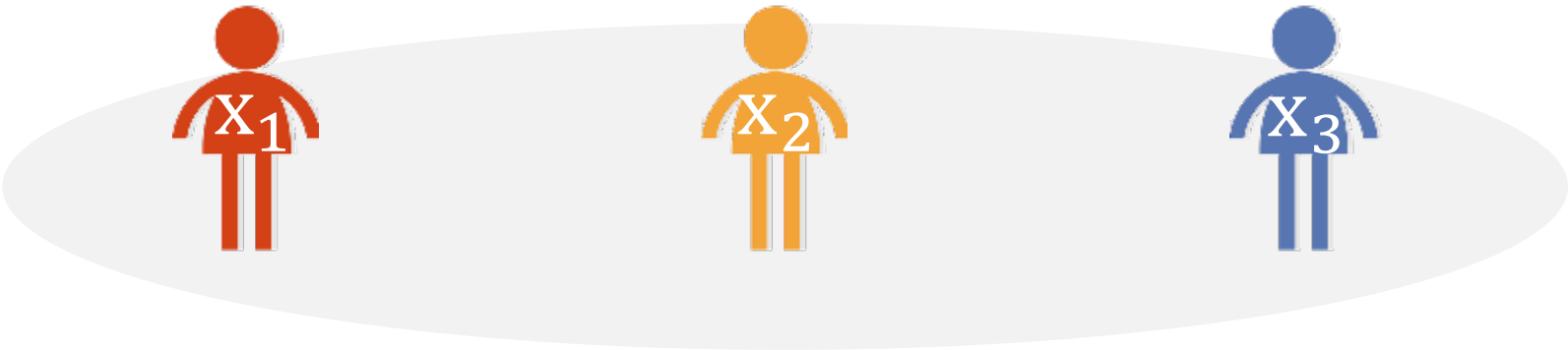
How to Construct MU-DP-NIZK??

The Plan

1. Review Attribute-based Signatures (ABS).
2. Compile ABS into MU-DP-NIZK using the technique developed in [KW18].

Attribute-based Signatures (ABS)

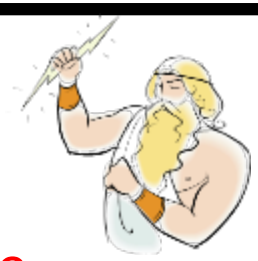
$$(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{ABS.Setup}(1^k)$$



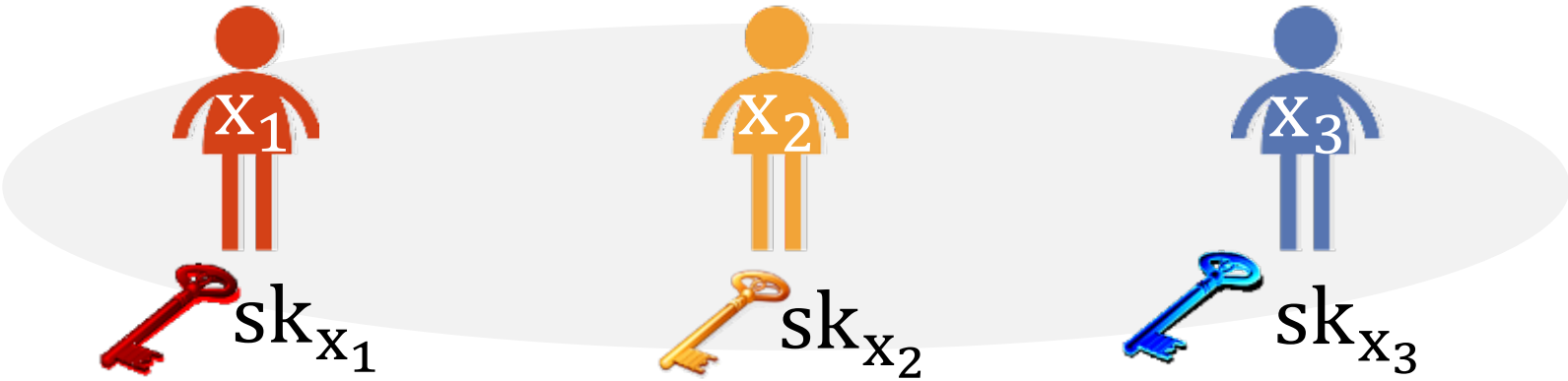
Attribute-based Signatures (ABS)

$$(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{ABS.Setup}(1^k)$$

$$\mathrm{sk}_{\mathrm{x}} \leftarrow \mathrm{ABS.KeyGen}(\mathrm{msk}, \underline{\mathrm{x}})$$



*attribute

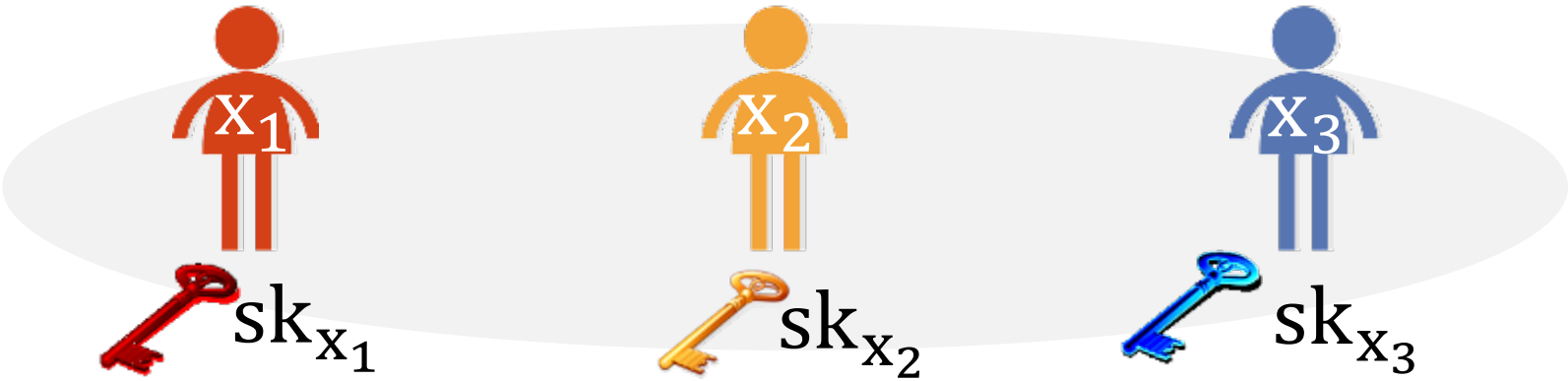


Attribute-based Signatures (ABS)

$$(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{ABS.Setup}(1^k)$$

$$\mathrm{sk}_{\mathrm{x}} \leftarrow \mathrm{ABS.KeyGen}(\mathrm{msk}, \underline{\mathrm{x}})$$

*attribute



$$\sigma \leftarrow \mathrm{ABS.Sign}(\mathrm{mpk}, \mathrm{sk}_{\mathrm{x}}, \underline{\mathrm{C}}, \mathrm{M})$$

*policy

Can sign on a policy C iff $\mathrm{C}(\mathrm{x}) = 1$.

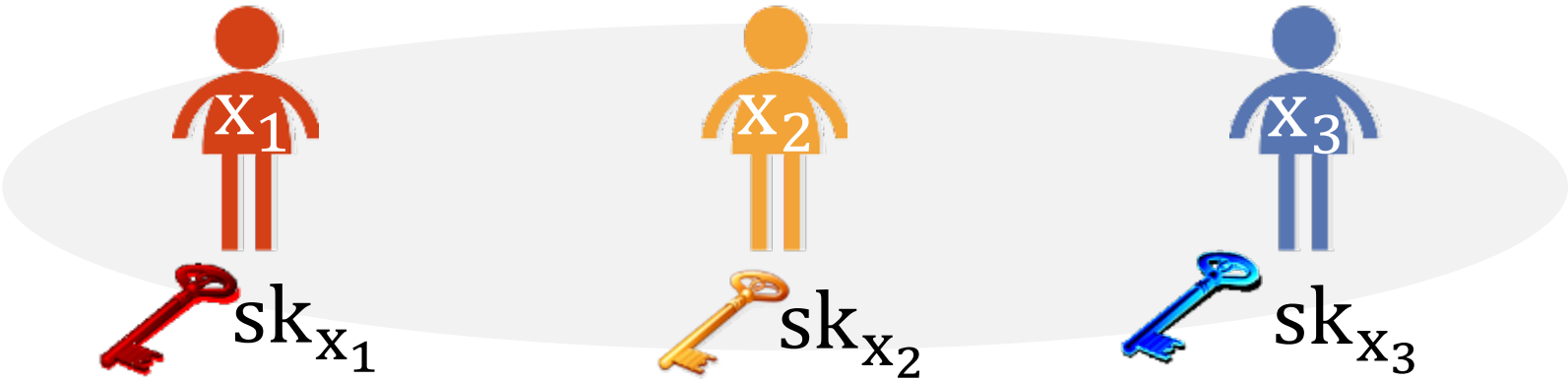
*attribute satisfies policy.

Attribute-based Signatures (ABS)

$$(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{ABS.Setup}(1^k)$$

$$\mathrm{sk}_{\mathrm{x}} \leftarrow \mathrm{ABS.KeyGen}(\mathrm{msk}, \underline{\mathrm{x}})$$

*attribute



$$\sigma \leftarrow \mathrm{ABS.Sign}(\mathrm{mpk}, \mathrm{sk}_{\mathrm{x}}, \underline{\mathrm{C}}, \mathrm{M})$$

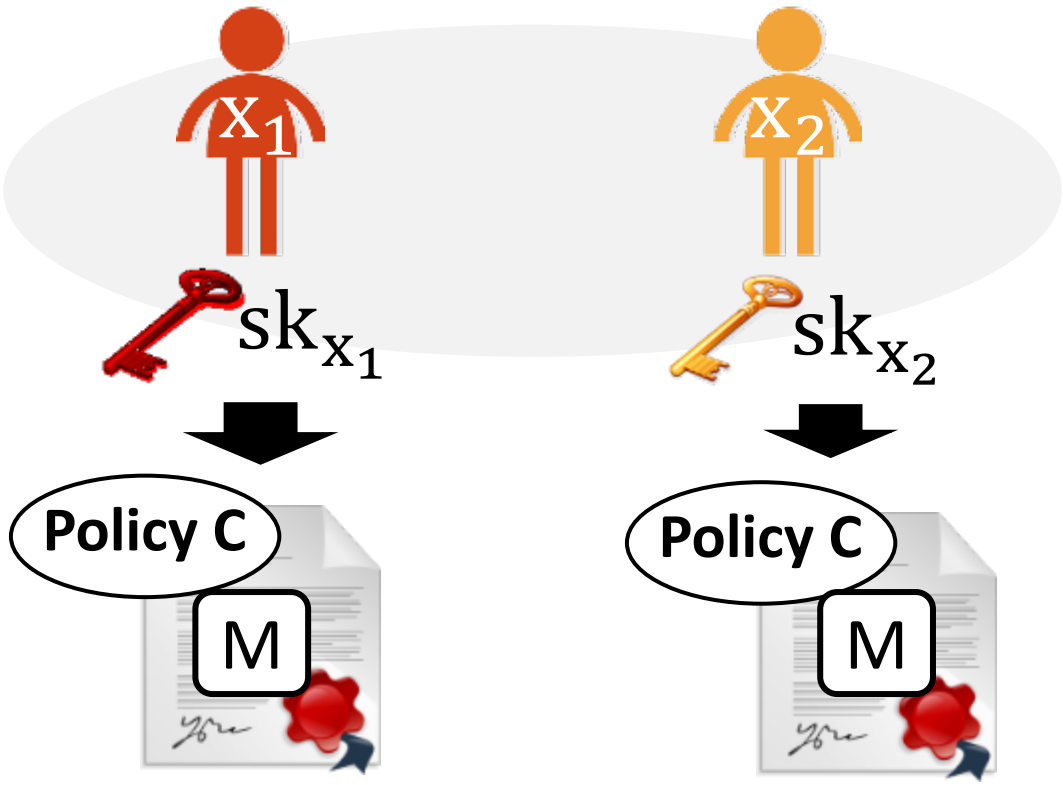
*policy

Can sign on a policy C iff $\mathrm{C}(\mathrm{x}) = 1$.

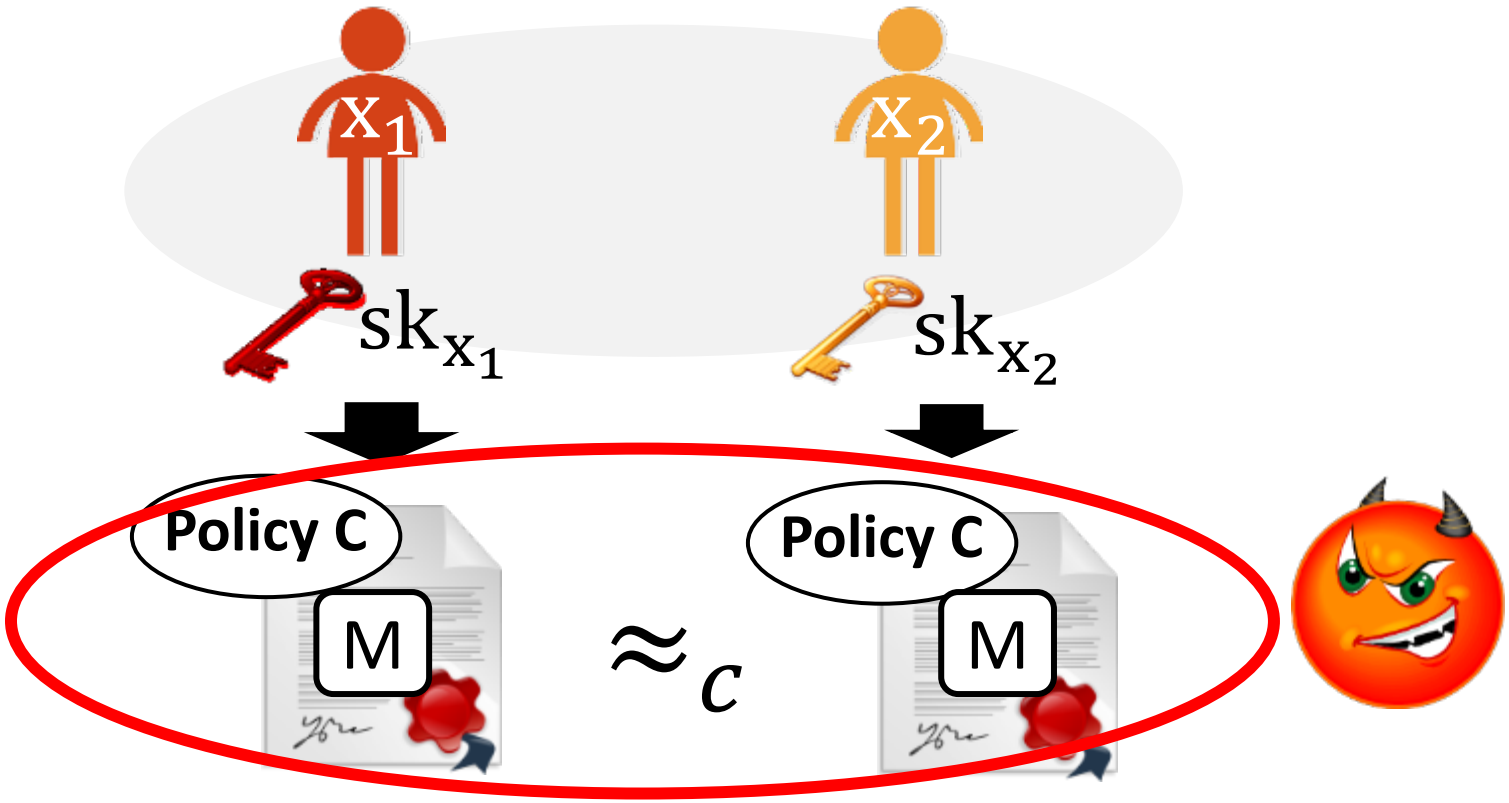
*attribute satisfies policy.

$$\top/\bot \leftarrow \mathrm{ABS.Verify}(\mathrm{mpk}, \mathrm{C}, \sigma, \mathrm{M})$$

Security of ABS: Anonymity

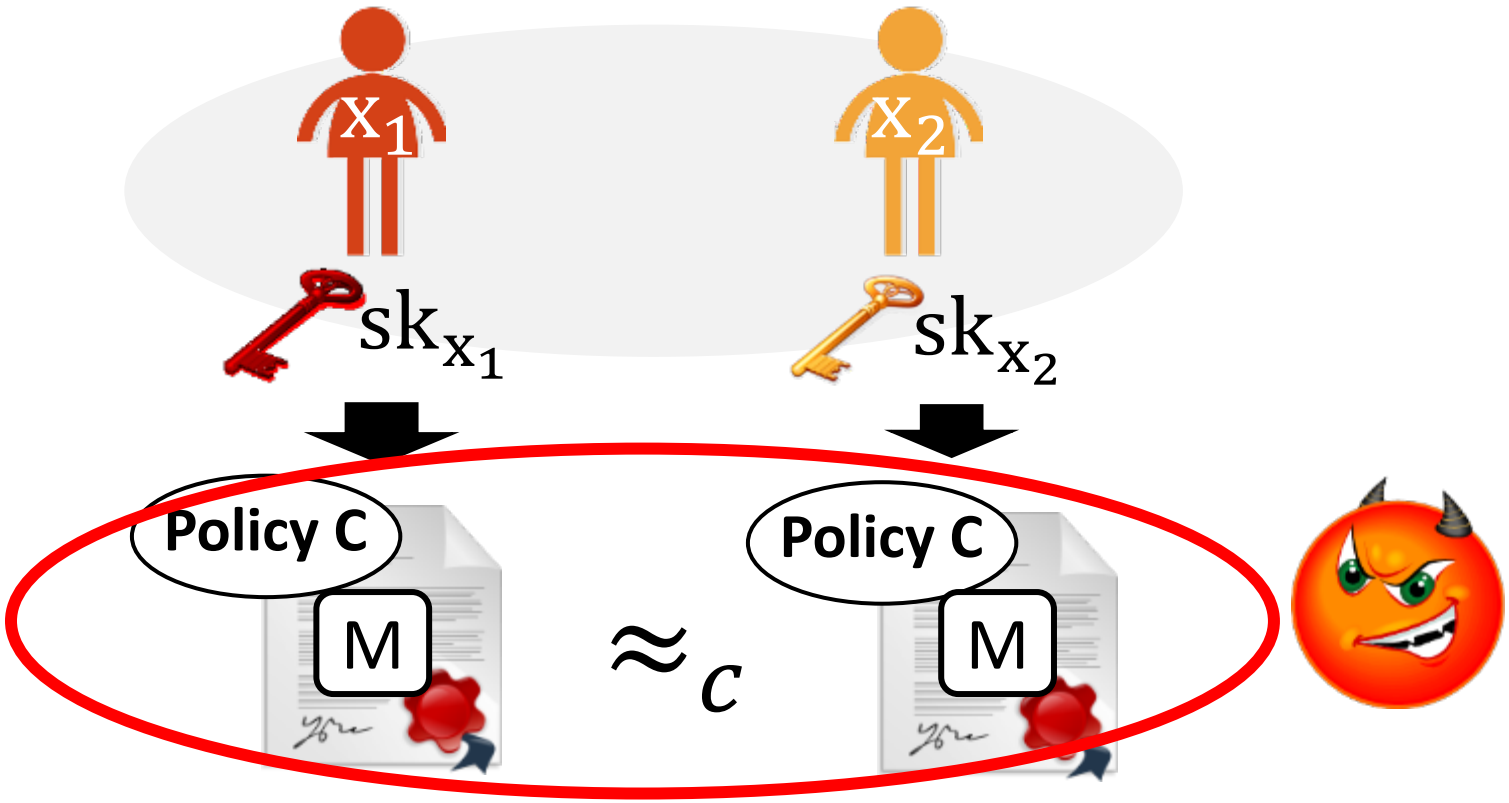


Security of ABS: Anonymity



If attribute x_1 and x_2 satisfy $C(x_1) = C(x_2) = 1$, then the signatures are indistinguishable.

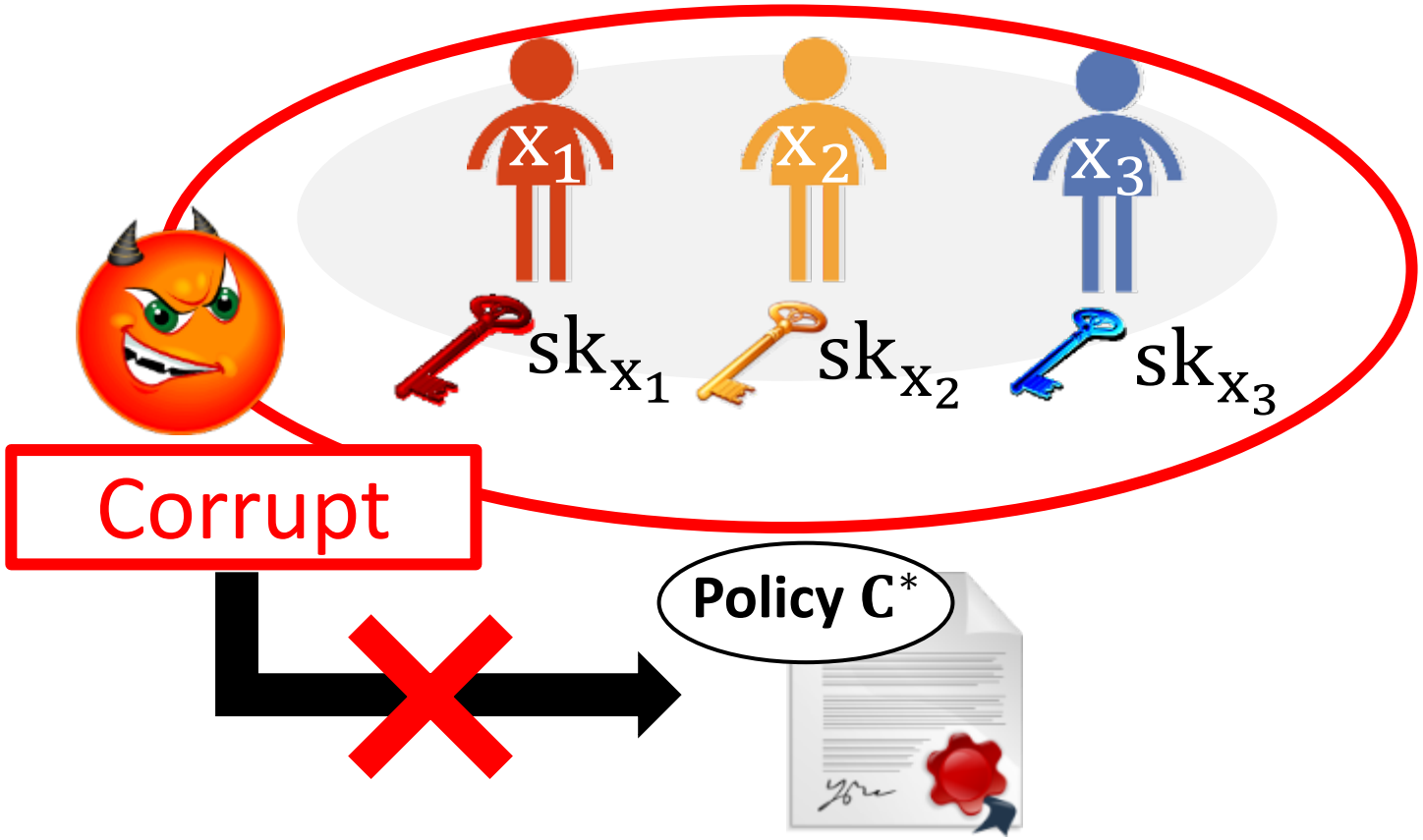
Security of ABS: Anonymity



If attribute x_1 and x_2 satisfy $C(x_1) = C(x_2) = 1$, then the signatures are indistinguishable.

*Signature only leaks that the signer had a satisfying attribute x for policy C.

Security of ABS: Unforgeability

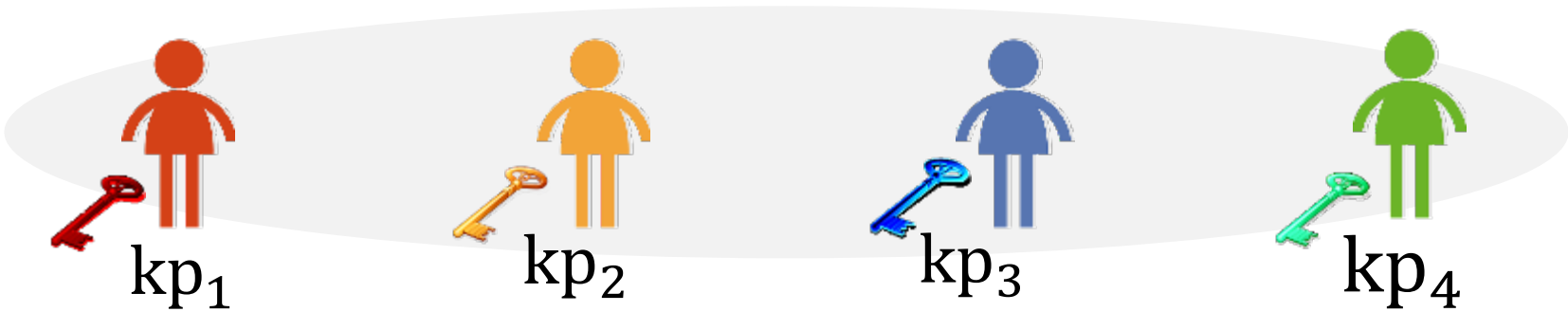


Hard to forge a signature on C^* even if given signing keys $\{sk_x\}$ that are not allowed to sign on C^* (i.e., $C^*(x) = 0$)

*Secret key sk_x can only be used with respect to C such that $C(x) = 1$.

MU-DP-NIZK from ABS (+ SKE)

Apply the idea of [KW18] to ABS instead of FHS.



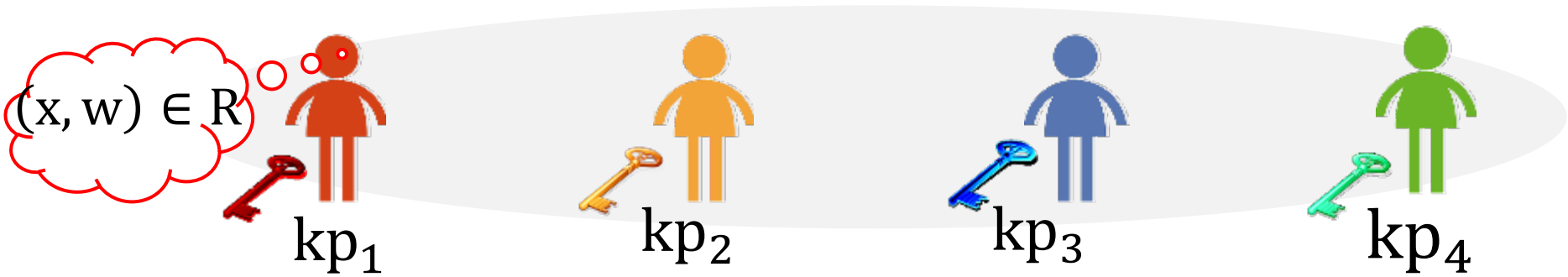
$= (\mathrm{K}_1, \mathrm{sk}^{\mathrm{ABS}}_{\mathrm{K}_1})$ ..... $= (\mathrm{K}_4, \mathrm{sk}^{\mathrm{ABS}}_{\mathrm{K}_4})$

SKE secret key

ABS signing key viewing K_1 as attribute

MU-DP-NIZK from ABS (+ SKE)

Apply the idea of [KW18] to ABS instead of FHS.



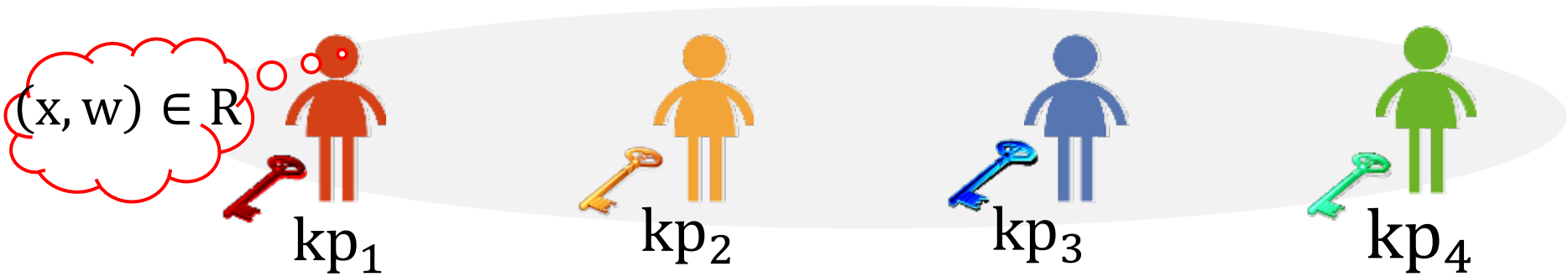
$= (K_1, sk_{K_1}^{ABS})$ ..... $= (K_4, sk_{K_4}^{ABS})$

<u>Constructing MU-DP-NIZK proof π</u>

1. $ct \leftarrow SKE.Enc(K_1, w)$

MU-DP-NIZK from ABS (+ SKE)

Apply the idea of [KW18] to ABS instead of FHS.



$= (K_1, sk_{K_1}^{ABS})$ ..... $= (K_4, sk_{K_4}^{ABS})$

<u>Constructing MU-DP-NIZK proof π</u>

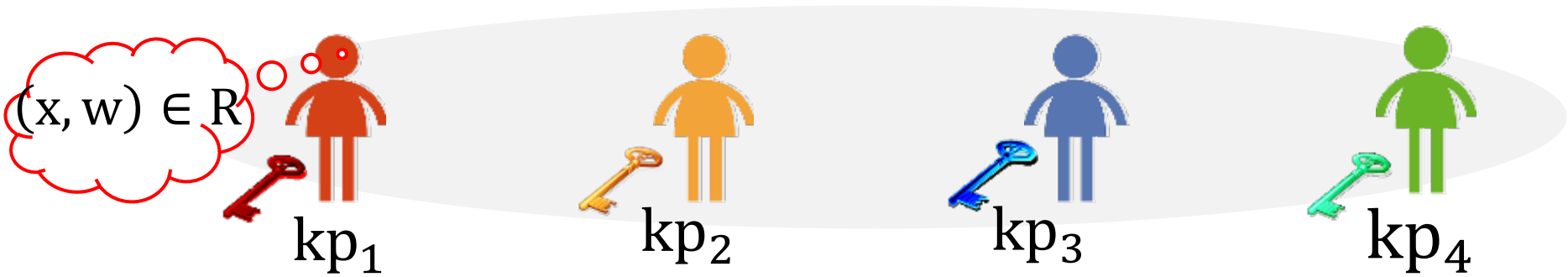
1. $ct \leftarrow SKE.Enc(K_1, w)$
2. $\sigma \leftarrow ABS.Sign(mpk, sk_{K_1}^{ABS}, C_{x,ct}, \text{"}\exists \text{fixed M"})$

   where policy $C_{x,ct}(K) := R(x, SKE.Dec(K, ct))$.

Public Function

MU-DP-NIZK from ABS (+ SKE)

Apply the idea of [KW18] to ABS instead of FHS.



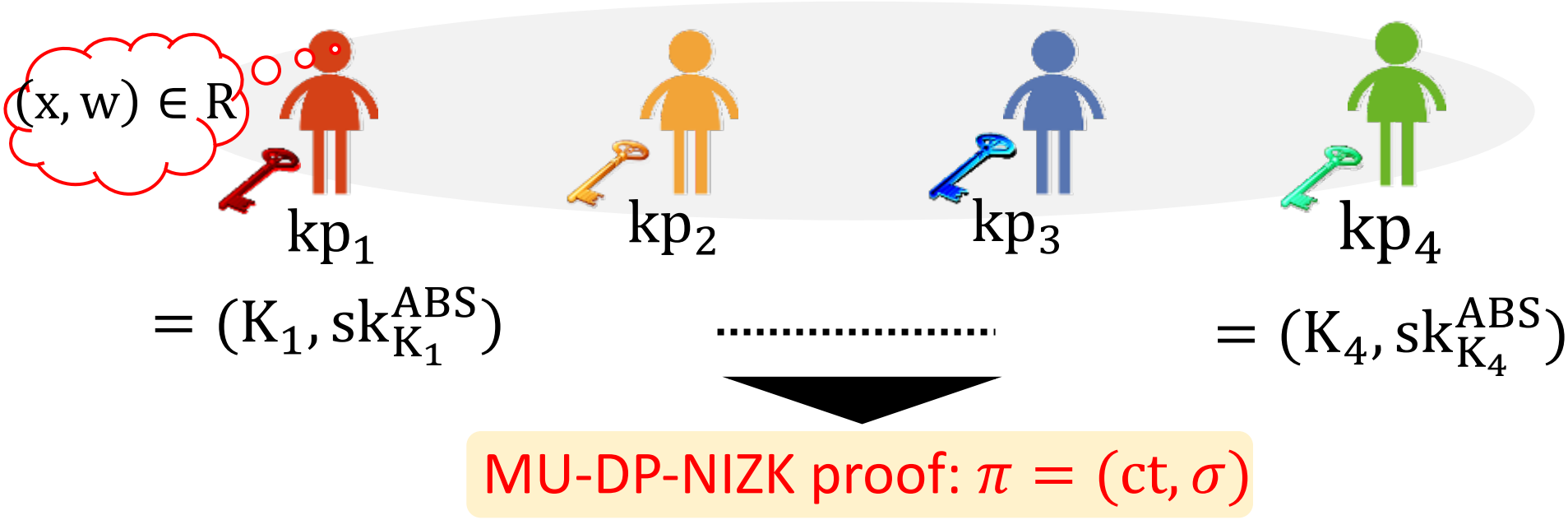
$= (K_1, sk_{K_1}^{ABS})$ ..... $= (K_4, sk_{K_4}^{ABS})$

<u>Constructing MU-DP-NIZK proof π</u>

1. $ct \leftarrow SKE.Enc(K_1, w)$
2. $\sigma \leftarrow ABS.Sign(mpk, sk_{K_1}^{ABS}, C_{x,ct}, \text{"}\exists \text{fixed M"})$
   where policy $C_{x,ct}(K) := R(x, SKE.Dec(K, ct))$.
3. $\pi := (ct, \sigma)$

Public Function

Security: Soundness



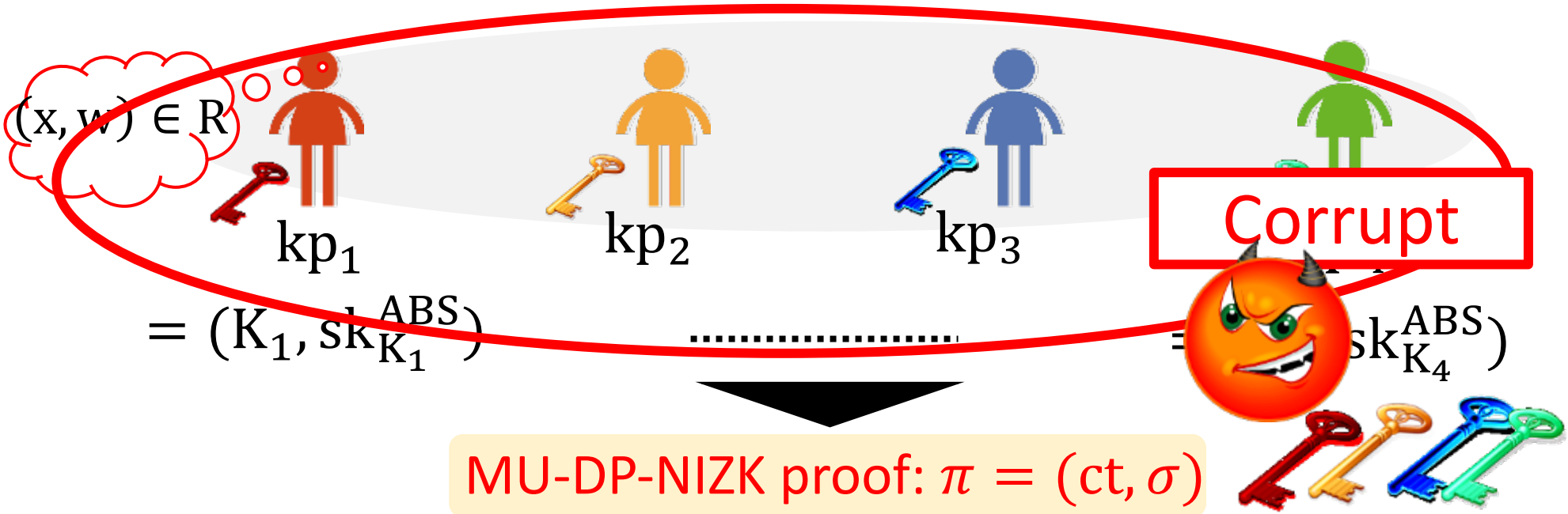
Soundness

If x is not in the language, then for all ct and K,

$$C_{x,ct}(K) := R\big(x, SKE.Dec(K, ct)\big) = 0.$$

Hence, unforgeability of ABS implies soundness ☺

Security: Soundness



Soundness

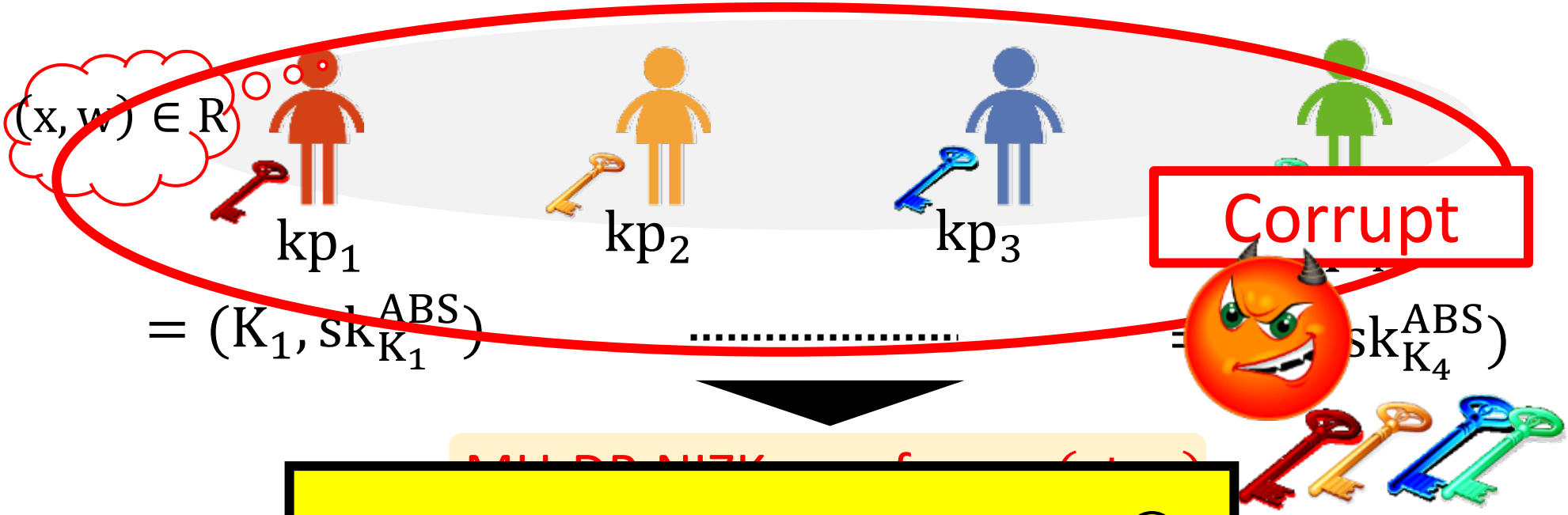
If x is not in the language, then for all ct and K,

$$C_{x,ct}(K) := R\big(x, SKE.Dec(K, ct)\big) = 0.$$

Hence, unforgeability of ABS implies soundness ☺

…w/ corruption too??

Security: Soundness



Soundne

Sound even with corruption ☺
(∵ ABS is unforgeable even with corruption)

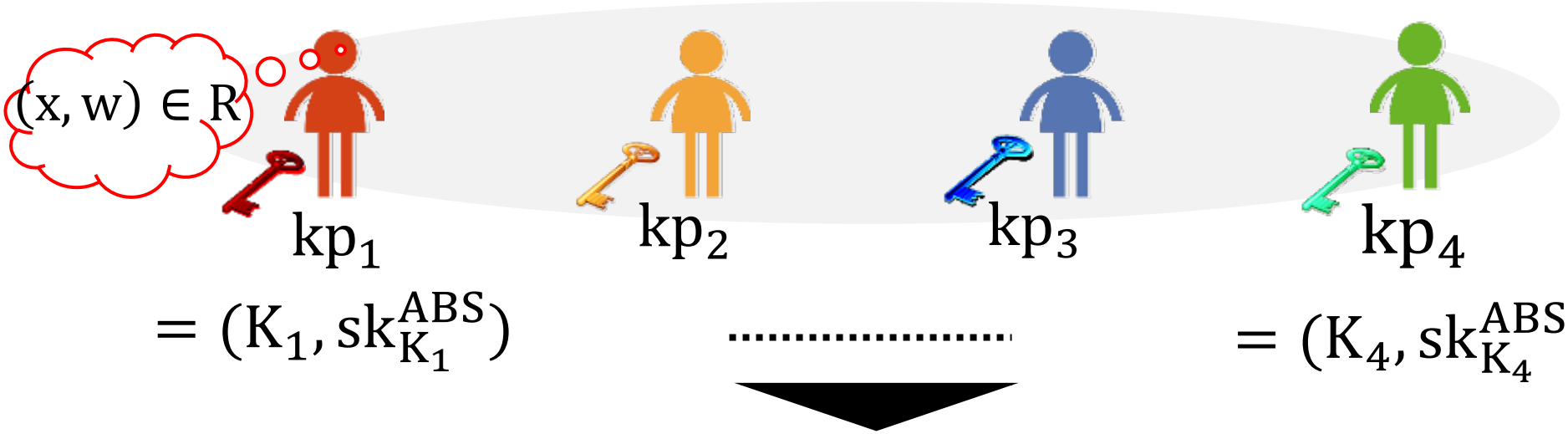
If x is

$$C_{x,ct}(K) := R(x, SKE.Dec(K, ct)) = 0.$$

Hence, unforgeability of ABS implies soundness ☺

…w/ corruption too??

Security: Zero-Knowledge/Anonymity

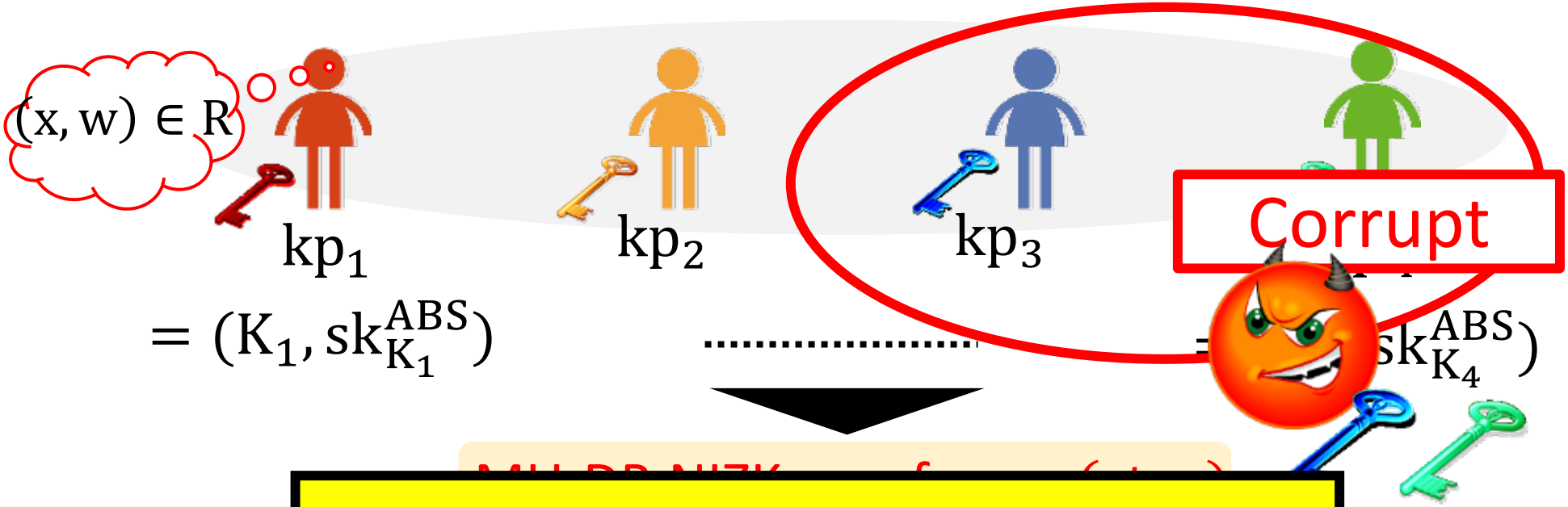


MU-DP-NIZK proof: $\pi = (ct, \sigma)$

ZK/Anonymity

- ct$= \mathrm{SKE.Enc}(K, w)$: Does not leak w due to security of SKE.
- σ is a ABS signature using sk_K^{ABS}: Does not leak (attribute) K due to anonymity of ABS.

Security: Zero-Knowledge/Anonymity



ZK/Anon

- ct= SK rity of SKE.
- σ is a A ttribute) K due to anonymity of ABS.

ZK and Anonymous even with corruption ☺
(∵SKE Keys are independent and ABS is anonymous)

…w/ corruption too??

Piecing Everything Together

Plug in MU-DP-NIZK in place of CRS-NIZK in the "Sign-then-Enc-and-Prove" paradigm [BMW03, CG04].

$$\text{SKE} + \text{Signature} + \underbrace{\text{Multi-User DP-NIZK}}_{\text{SKE} + \text{ABS} \Longrightarrow} \Longrightarrow \text{GS}$$

Piecing Everything Together

Plug in MU-DP-NIZK in place of CRS-NIZK in the "Sign-then-Enc-and-Prove" paradigm [BMW03, CG04].

$$\text{SKE} + \text{Signature} + \underbrace{\text{Multi-User DP-NIZK}} \Longrightarrow \text{GS}$$

$$\text{SKE} + \text{ABS} \Longrightarrow$$

Straightforward to instantiate SKE and Signature using existing constructions (SIS, LWE, LPN,…).

How about Attribute-based Signature??

Instantiating ABS

Instantiation 1:

[Tsabary17] gives ABS, but needs complexity leveraging for our purpose $\Rightarrow$ Need (subexp) SIS (due to mismatch of security notions).

Instantiating ABS

Instantiation 1:

[Tsabary17] gives ABS, but needs complexity leveraging for our purpose ⇒ Need (subexp) SIS (due to mismatch of security notions).

Instantiation 2:

Weaken the security requirements for ABS by the following observations

- Bounded key queries is sufficient
- Attributes for signing key can be determined before the setup of system (They are SKE keys)

⇒Directly construction from (poly) SIS.

Conclusion of [KY19]

① Construct the **first group signatures from lattices in the standard model**.

② Consider a new type of Multi-User DP-NIZK and construct it from ABS.

③ Constructions from various assumptions.

- ✓ SIS w/ subexp-modulus.
- ✓ LWE w/ poly-modulus.
- ✓ SIS w/ poly-modulus + LPN w/ const. noise rate

Some Open Questions

① Construct **group signatures** based on **poly SIS**.

*Ours require sub-exp SIS.

② Construct **ring signatures** w/o setup with **logarithmic signature size from lattices**.

*Linear size known from SIS.

③ Any other interesting notions for NIZKs??
(e.g., malicious DV-NIZK [QRW19])

④ Explore other links between various types of NIZK and signatures.