Interactive Threshold Mercurial Signatures and Applications

NTT KYOTO UNIVERSITY Masayuki Abe,
KYOTO UNIVERSITY Masaya Nanri,
NTT Octavio Perez Kempner
NTT KYOTO UNIVERSITY Mehdi Tibouchi

Can Digital Signatures become more flexible?

Digital Signatures

→ Assuring information and issuers' integrity with mathematical technique

Additional Demands

- For easier development of high level security application
- For hiding privacy of credentials

Structure-Preserving Signatures (SPS) and successive tools were developed !

What is SPS and its Extension (SPS-EQ) ?

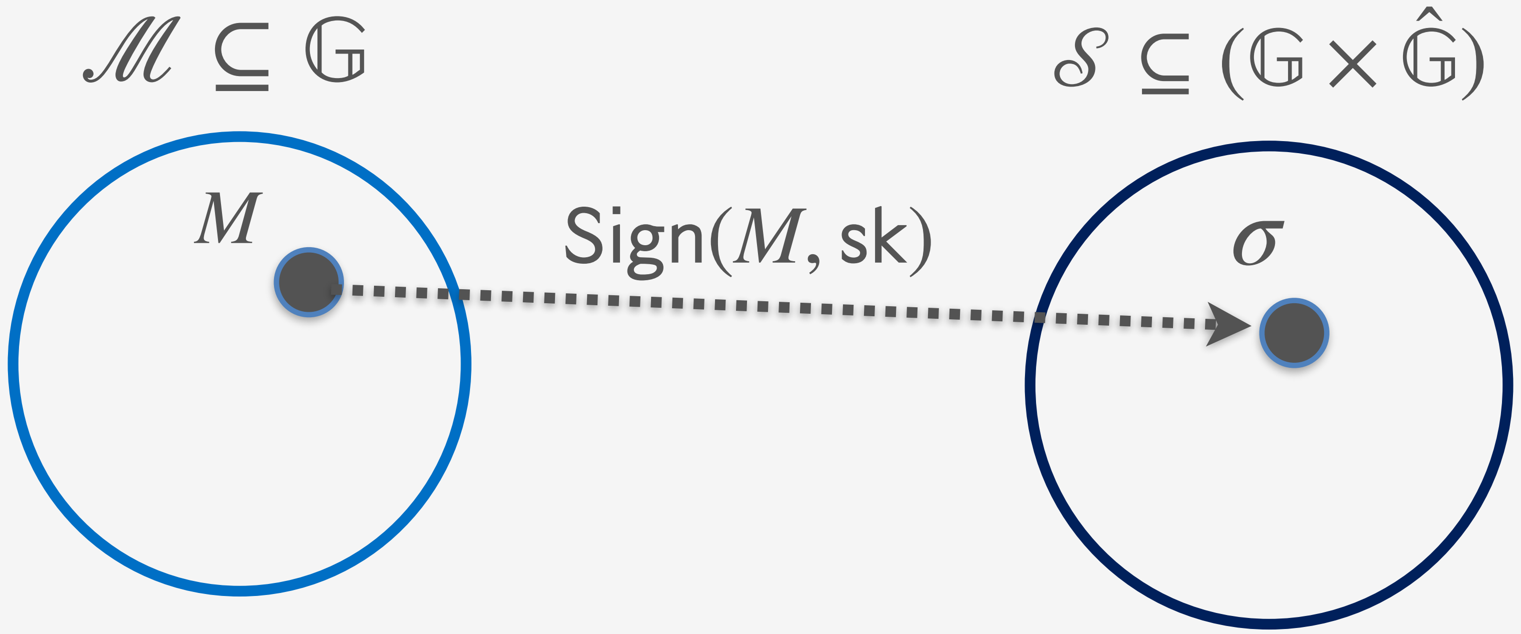
Structure Preserving Signatures [AFG10]

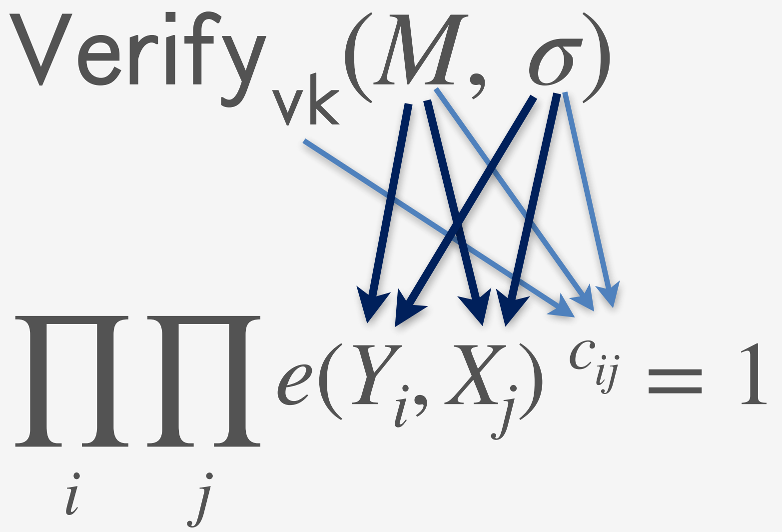
Messages, signatures and verification keys are included in the same pairing groups

Verification uses pairing operation

$\mathbb{G}$ and $\hat{\mathbb{G}}$: pairing group ordered by p, G and $\hat{G}$: generators of each group, e : map from $(\mathbb{G}, \hat{\mathbb{G}})$ to $\mathbb{G}_T$

$M \in \mathscr{M} \subseteq \mathbb{G}$: messages , $\sigma \in \mathscr{S} \subseteq (\mathbb{G} \times \hat{\mathbb{G}})$: signatures , $(\mathsf{sk}, \mathsf{vk}) \in (\mathbb{Z}_p^* \times \hat{\mathbb{G}})$: signing keys and verification keys





What is SPS and its Extension (SPS-EQ) ?

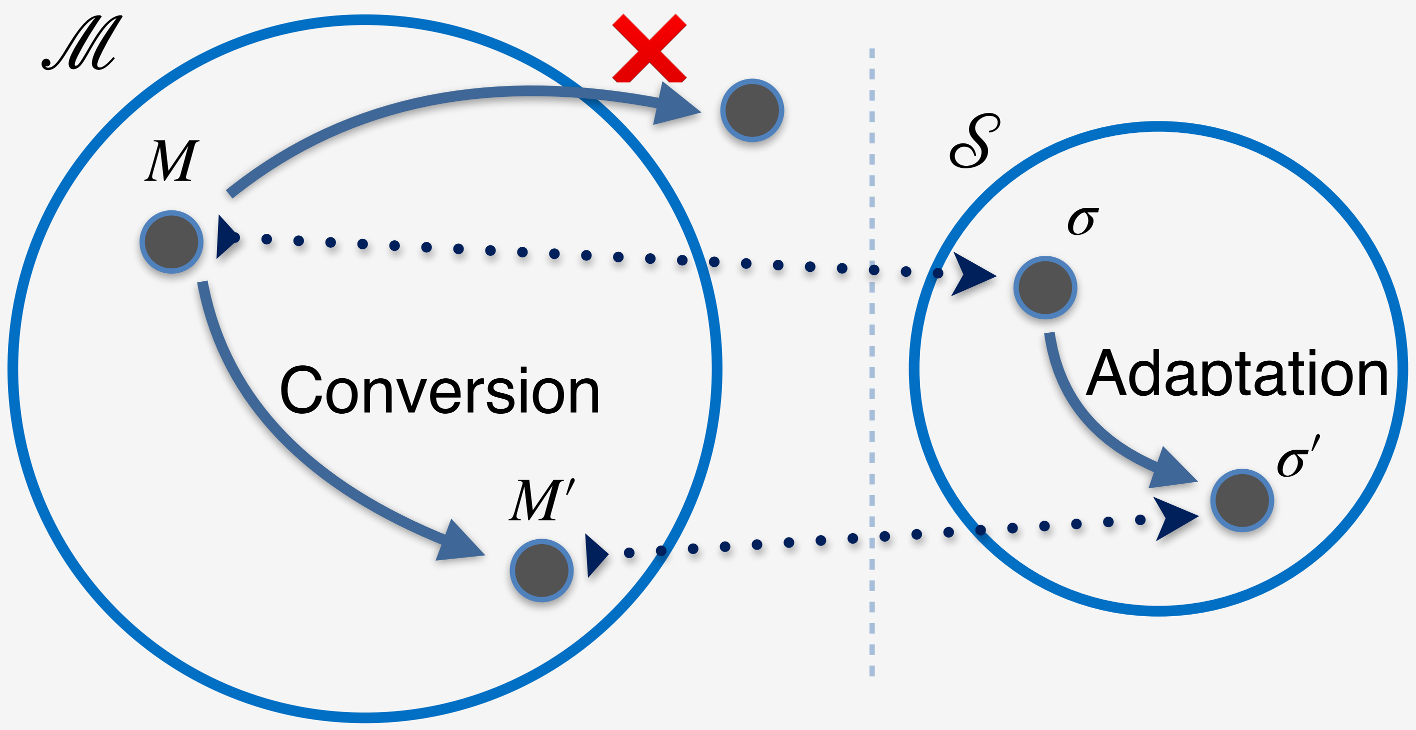
Structure Preserving Signatures [AFG10]

Messages, signatures and verification keys are included in the same pairing groups

Verification uses pairing operation

Structure Preserving Signatures on Equivalence Class [FHS14]

Signatures can be issued for a certain equivalence class defined over the message space



Anonymizing signatures leads to

Privacy enhanced credentials

Mercurial Signatures [CL19]

Methods in [FHS14] + Key Conversion

Sign(M, sk) $\rightarrow \sigma$

$$\sigma = (Z, Y, \hat{Y}) = \left(\left(\prod_{i=1}^{\ell} M_i^{x_i}\right)^y, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}}\right)$$

Verify(M, σ, vk) $\rightarrow$ 0 or 1

$$\prod_{i=1}^{\ell} e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \ \wedge \ e(Y, \hat{G}) = e(G, \hat{Y})$$

ChangeRep(vk, M, σ, μ) $\rightarrow$ (M', σ')

ConvertSig(vk, M, σ ; ρ) $\rightarrow \tilde{\sigma}$

ConvertVK(vk ; ρ) $\rightarrow \tilde{\text{vk}}$

$$\tilde{\text{vk}} = \text{vk}^{\rho} = (\hat{G}^{x_1\rho}, \ldots, \hat{G}^{x_\ell\rho})$$

ConvertSK(sk ; ρ) $\rightarrow \tilde{\text{sk}}$

$$\tilde{\text{sk}} = \text{sk}^{\rho} = (x_1^{\rho}, \ldots, x_\ell^{\rho})$$

Mercurial Signatures [CL19]

Methods in [FHS14] + Key Conversion

$\text{Sign}(M, \text{sk}) \rightarrow \sigma$

Signing Key
sk $= (x_1, \ldots, x_\ell)$

Ephemeral
Random Value

$$\sigma = (Z, Y, \hat{Y}) = \left(\left(\prod_{i=1}^{\ell} M_i^{x_i}\right)^{y}, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}}\right)$$

Mercurial Signatures [CL19]

Methods in [FHS14] + Key Conversion

$\mathrm{Sign}(M, \mathrm{sk}) \rightarrow \sigma$

Signing Key
$\mathrm{sk} = (x_1, \ldots, x_\ell)$

Ephemeral Random Value

$$\sigma = (Z, Y, \hat{Y}) = \left(\left(\prod_{i=1}^{\ell} M_i^{x_i} \right)^{y}, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}} \right)$$

$\mathrm{Verify}(M, \sigma, \mathrm{vk}) \rightarrow 0 \text{ or } 1$

$$\prod_{i=1}^{\ell} e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \ \wedge \ e(Y, \hat{G}) = e(G, \hat{Y})$$

Verification Key
$\mathrm{vk} = (\hat{X}_1, \ldots, \hat{X}_\ell) = (\hat{G}^{x_1}, \ldots, \hat{G}^{x_\ell})$

Mercurial Signatures [CL19]

Methods in [FHS14] + Key Conversion

$\underline{\text{Sign}(M, \text{sk}) \to \sigma}$

Signing Key
$\text{sk} = (x_1, \ldots, x_\ell)$

Ephemeral Random Value

$$\sigma = (Z, Y, \hat{Y}) = \left(\left(\prod_{i=1}^{\ell} M_i^{x_i} \right)^{y}, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}} \right)$$

$\underline{\text{Verify}(M, \sigma, \text{vk}) \to 0 \text{ or } 1}$

$$\prod_{i=1}^{\ell} e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \ \wedge \ e(Y, \hat{G}) = e(G, \hat{Y})$$

Verification Key
$\text{vk} = (\hat{X}_1, \ldots, \hat{X}_\ell) = (\hat{G}^{x_1}, \ldots, \hat{G}^{x_\ell})$

Sampled Random Value

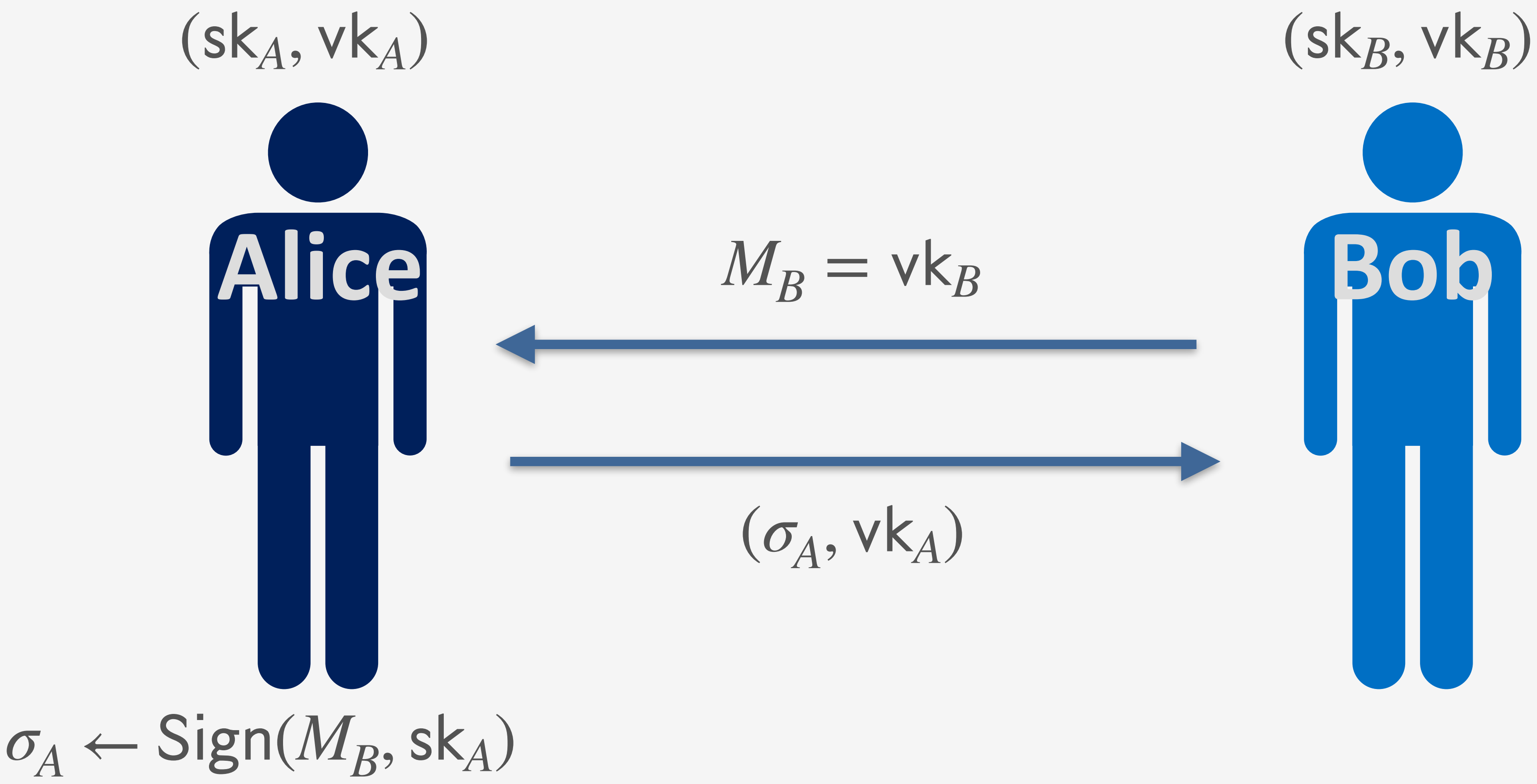
$\underline{\text{ConvertSig}(\text{vk}, M, \sigma \ ; \rho) \to \tilde{\sigma}}$

$\underline{\text{ConvertVK}(\text{vk} \ ; \rho) \to \tilde{\text{vk}}}$

$$\tilde{\text{vk}} = (\hat{X}_1^{\rho}, \ldots, \hat{X}_\ell^{\rho})$$

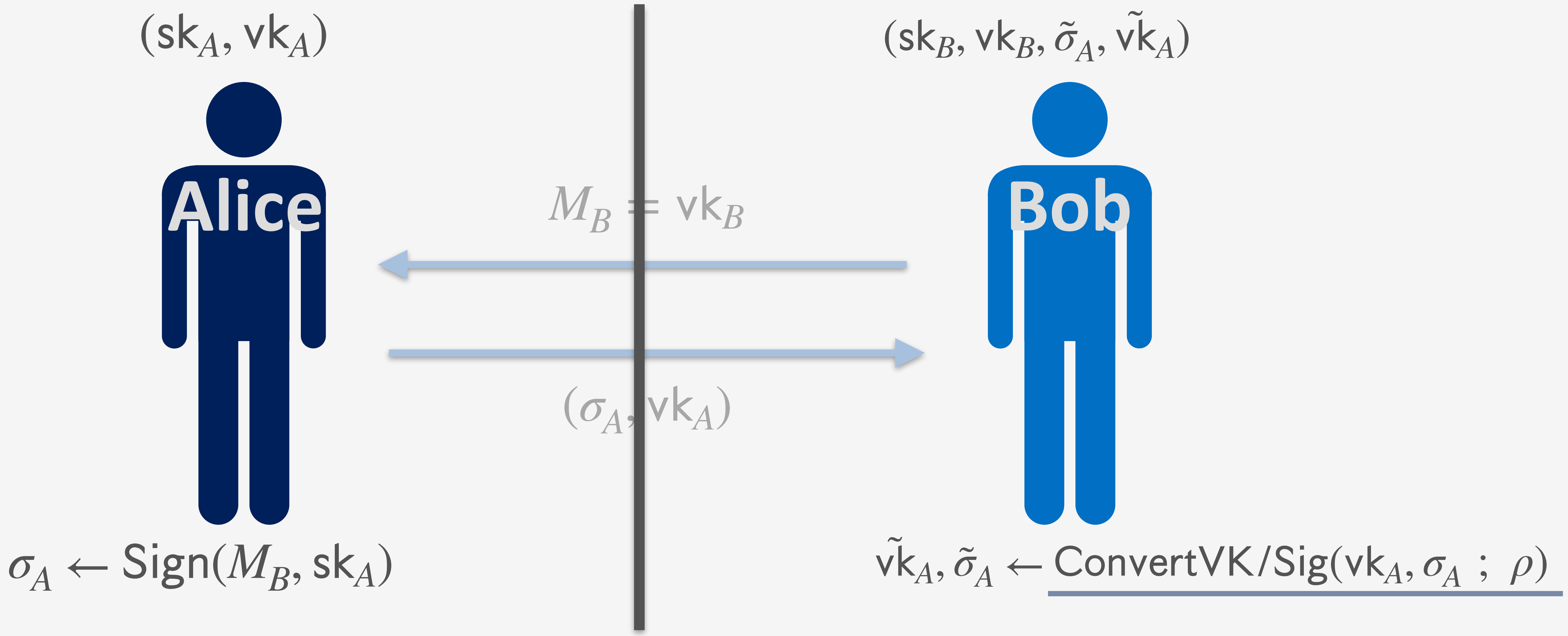
Anonymize Credentials for Privacy

Example case : Alice gives a credential to Bob like PKI



Anonymize Credentials for Privacy

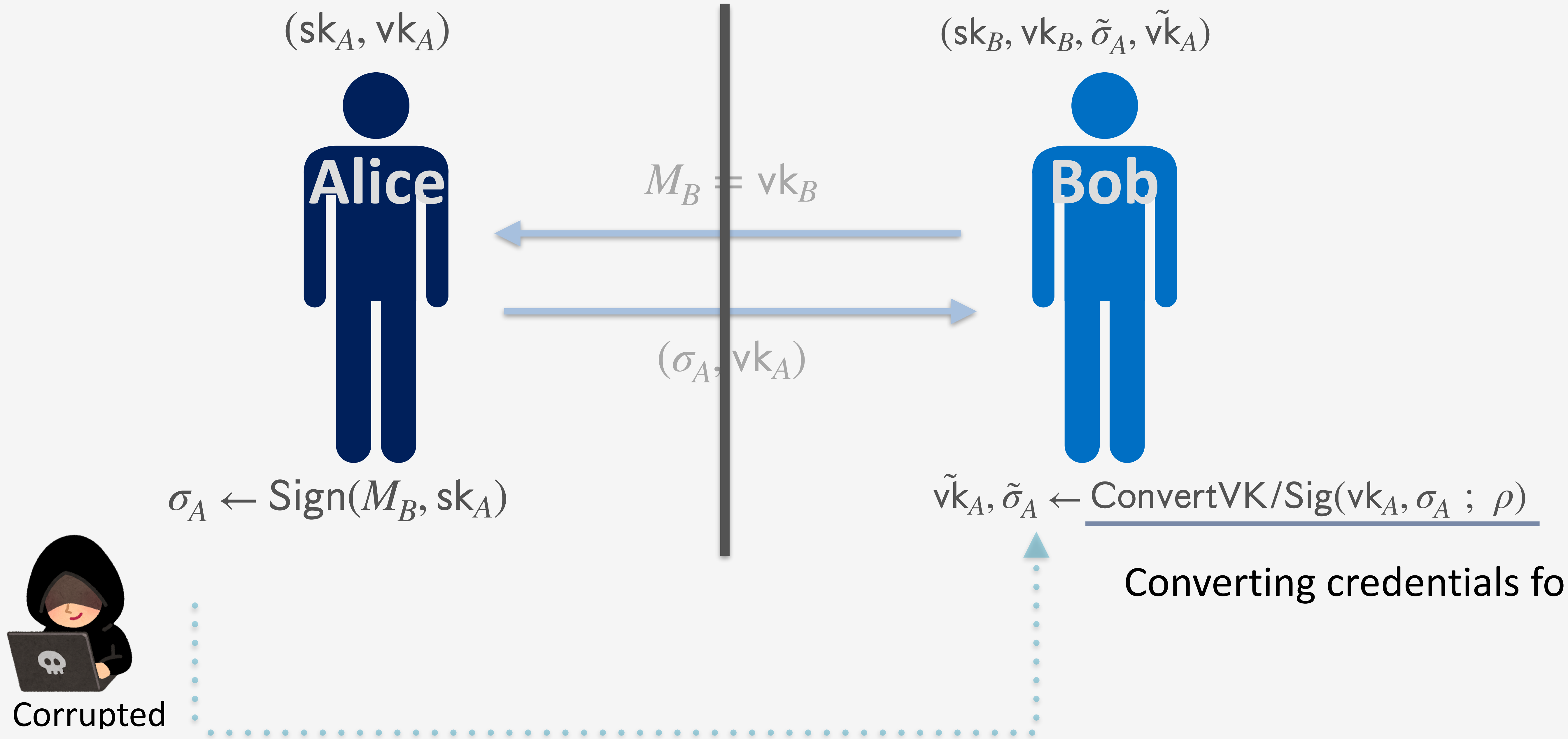
Example case : Alice gives a credential to Bob like PKI



Converting credentials for privacy as a receiver

Anonymize Credentials for Privacy

Example case : Alice gives a credential to Bob like PKI



Converting credentials for privacy as a receiver

Adversary wants to find the relation to the credentials

… Who is the issuer of the anonymized signatures ?

Problem : Weak unlinkability in Mercurial Signatures

Unlinkability doesn't hold for the corrupted signer

—> The Single malicious signer has chance to trace converted key and signatures

KeyGen(pp, $\ell(\kappa)$) $\rightarrow$ (vk, sk)

sk $= (x_1, \ldots, x_\ell)$

vk $= (\hat{X}_1, \ldots, \hat{X}_\ell) = (\hat{G}^{x_1}, \ldots, \hat{G}^{x_\ell})$

ConvertVK(vk, ρ) $\rightarrow \tilde{\mathsf{vk}}$

$\tilde{\mathsf{vk}} = \left(\hat{X}_1^\rho, \ldots \hat{X}_\ell^\rho\right)$

Obviously, this public values are the trigger

Problem : Weak unlinkability in Mercurial Signatures

Unlinkability doesn't hold for the corrupted signer

—> The Single malicious signer can trace converted keys and signatures

KeyGen(pp, $\ell(\kappa)$) → (vk, sk)

sk $= (x_1, \ldots, x_\ell)$

vk $= (\hat{X}_1, \ldots, \hat{X}_\ell) = (\hat{G}^{x_1}, \ldots, \hat{G}^{x_\ell})$

ConvertVK(vk, ρ) → $\tilde{\text{vk}}$

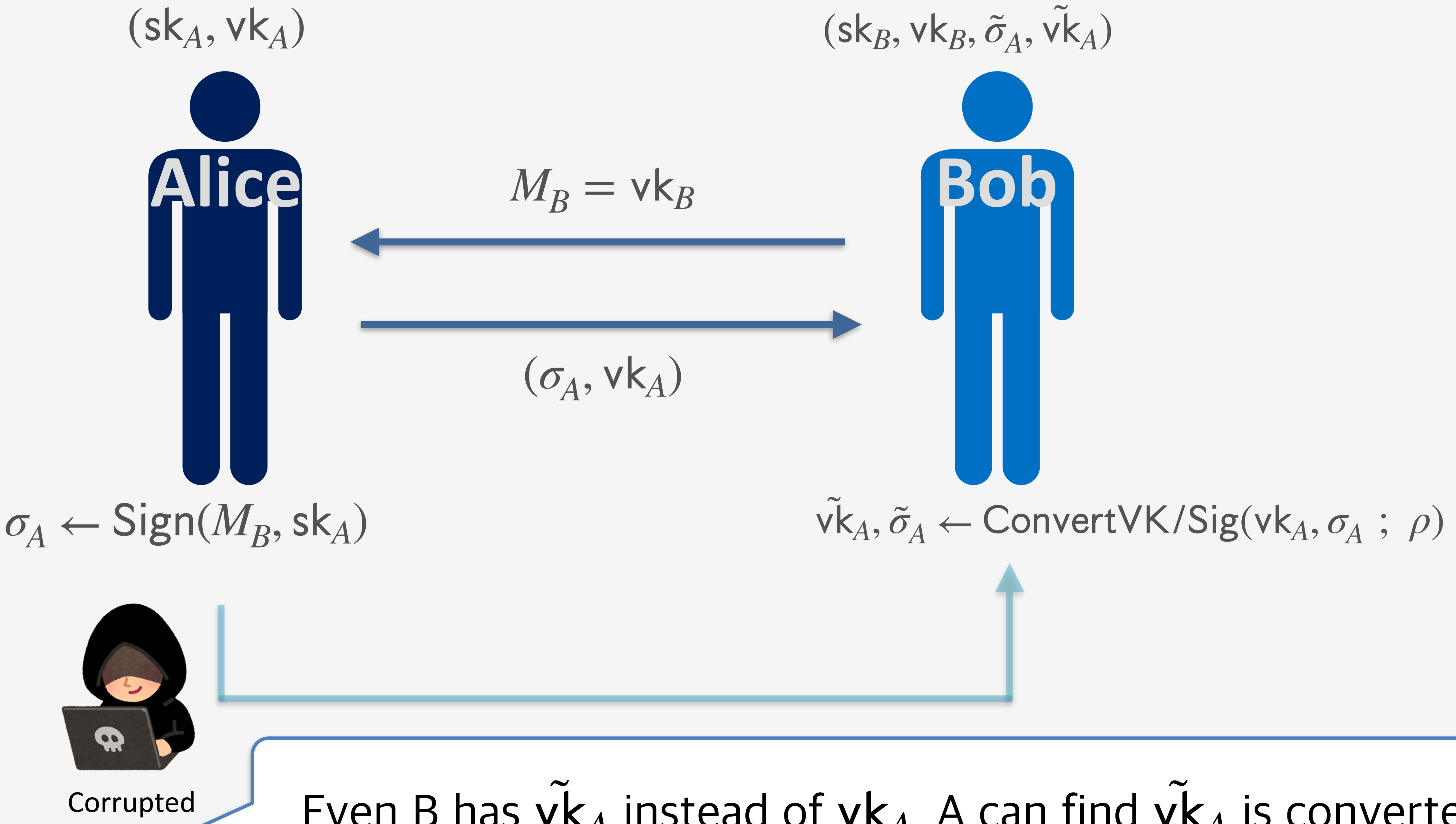
$\tilde{\text{vk}} = \left(\hat{X}_1^{\rho}, \ldots \hat{X}_\ell^{\rho}\right)$

$\tilde{\text{vk}} = (\hat{X}_1^{\rho}, \ldots, \hat{X}_\ell^{\rho})$ is in the same class as vk $= (\hat{G}^{x_1}, \ldots, \hat{G}^{x_\ell})$

If and only if $\hat{X}_1^{\rho \cdot \frac{1}{x_1}} = \cdots = \hat{X}_\ell^{\rho \cdot \frac{1}{x_\ell}}$, only the single signer has $\text{sk}^{-1} = \left(\frac{1}{x_1}, \ldots, \frac{1}{x_\ell}\right)$

Issue : Weak unlinkability makes threat for privacy…

Example case : A gives a credential to B like PKI



Even B has $\tilde{\mathsf{vk}}_A$ instead of vk_A, A can find $\tilde{\mathsf{vk}}_A$ is converted from vk_A.

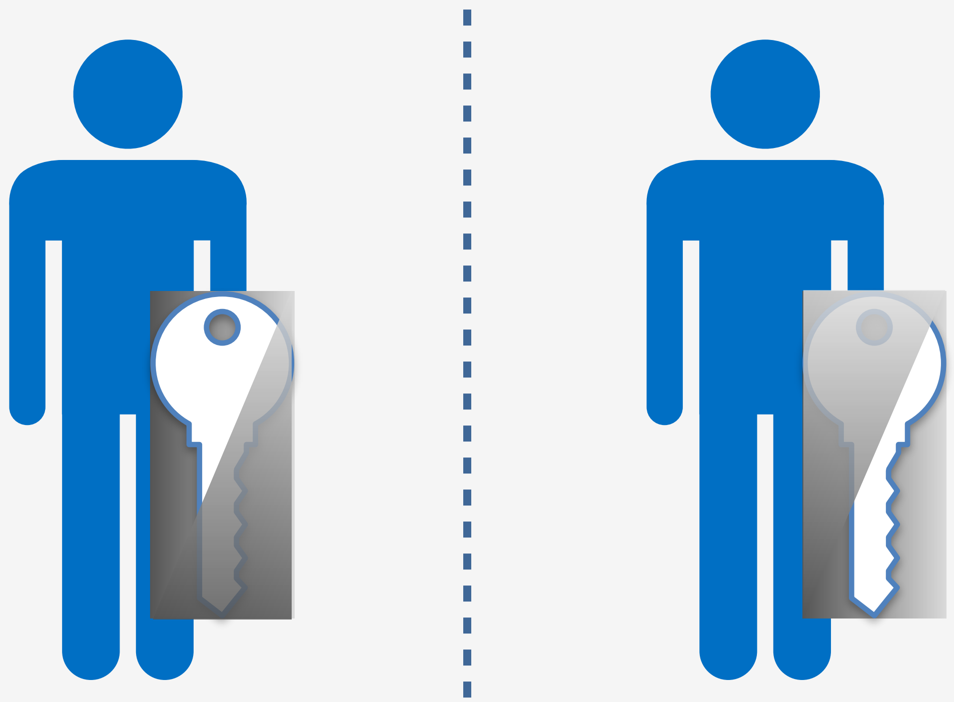
If and only if $\tilde{\sigma}_A$ is verified by $\tilde{\mathsf{vk}}_A$, the credential is issued by A.

Our approach : Splitting the Signer

No one can have the full signing key to trace the conversion

[CL19]
Single Trusted Signer

Our Approach :
Distributed Signers

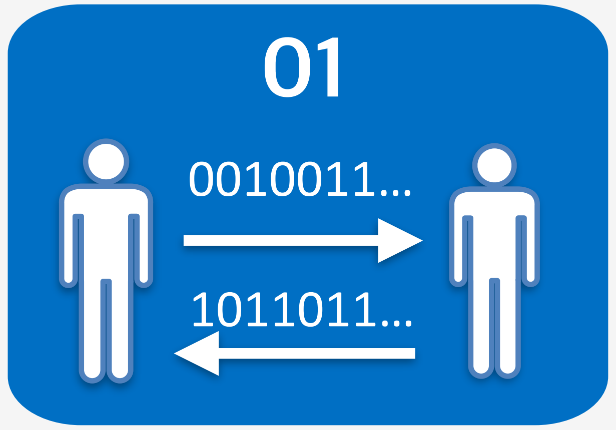


Each party doesn’t trust the opponent.

Communication among Signers

01 | Sequential communication model between Two parties

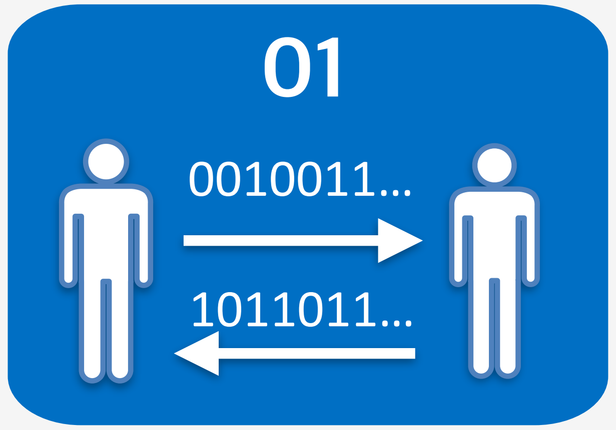
- It allows 1 party corruption



Communication among Signers

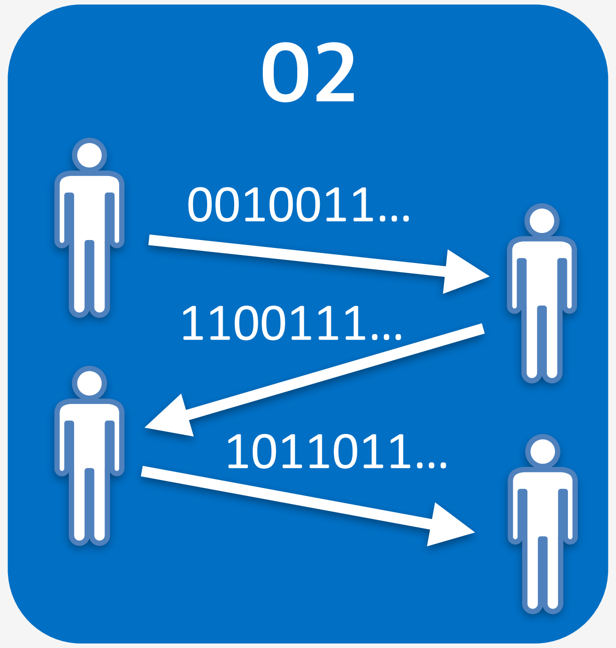
01 | Sequential communication model between Two parties

- It allows 1 party corruption



02 | Sequential communication model among t out of n parties

- It allows corruption up to t-1 party
- Pre-processing for secret sharing is required



Communication among Signers

01 | Sequential communication model between Two parties

- It allows 1 party corruption

02 | Sequential communication model among t out of n parties

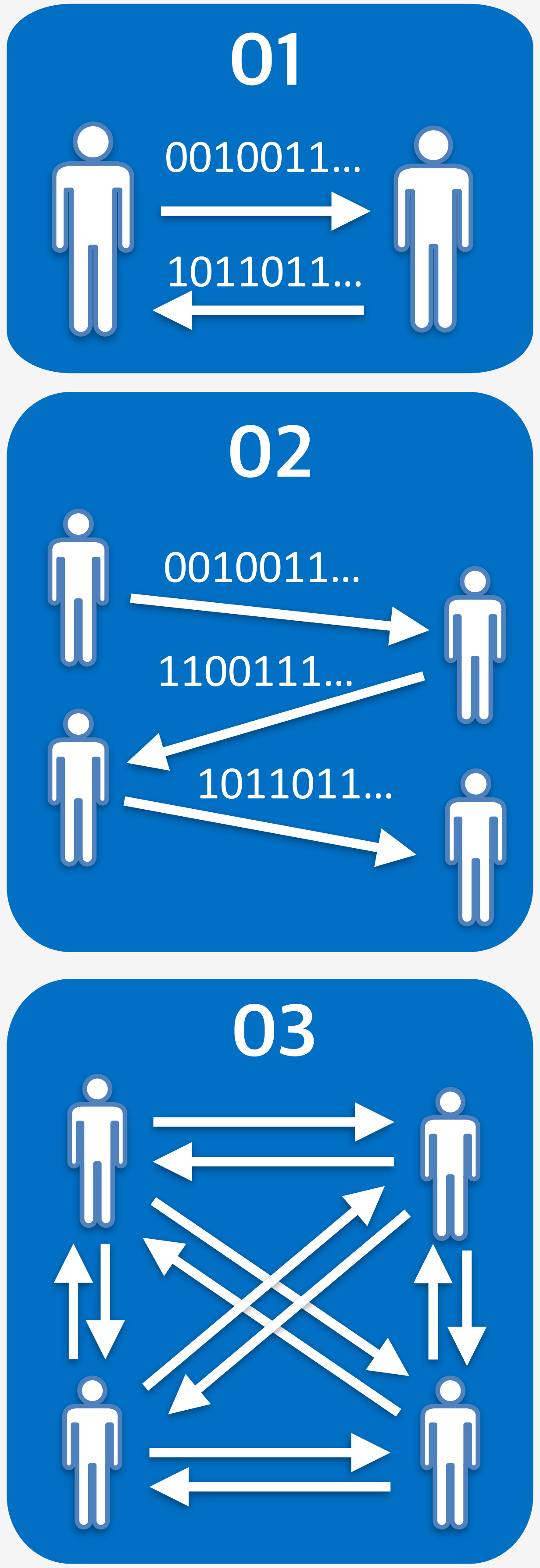
- It allows corruption up to t-1 party

- Pre-processing for secret sharing is required

03 | Synchronized communication model among t out of n parties

- It allows corruption up to t-1 party

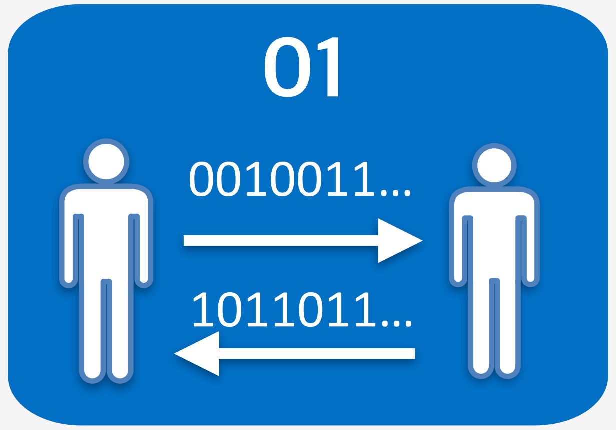
- Broadcast messages with traditional MPC



Communication among Signers

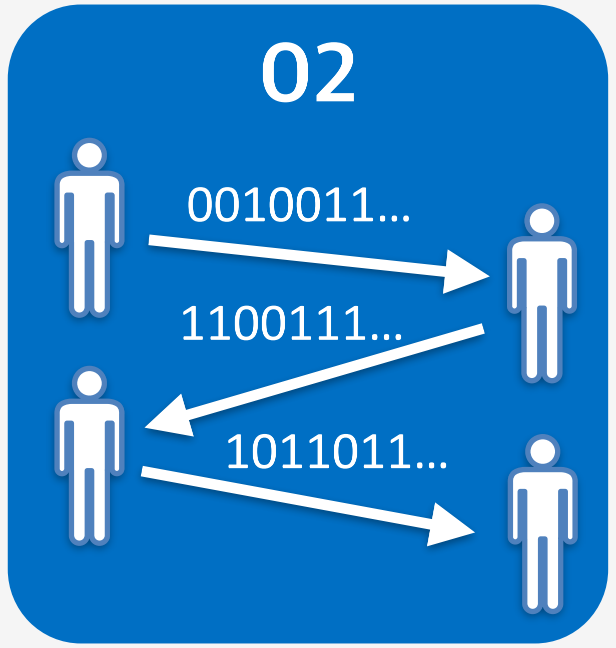
01 | Sequential communication model between Two parties

- It allows 1 party corruption



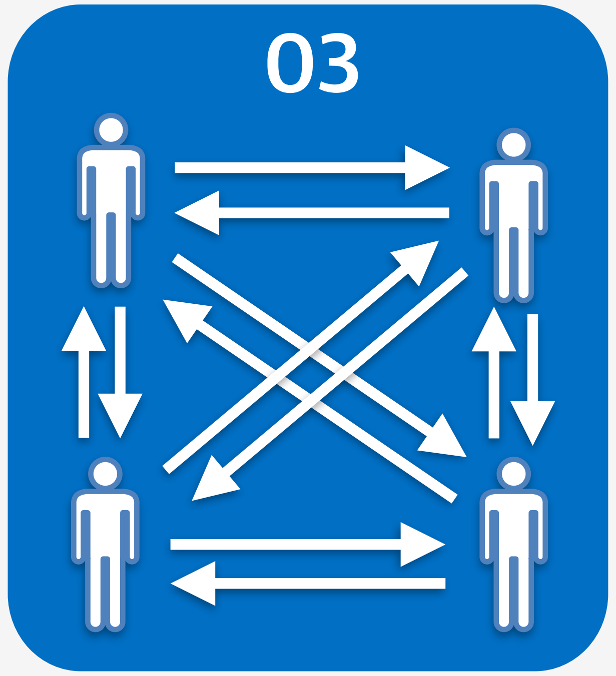
02 | Sequential communication model among t out of n parties

- It allows up to t-1 party corruption
- Pre-processing for secret sharing is required



03 | Synchronized communication model among t out of n parties

- It allows up to t-1 party corruption
- Broadcast messages with traditional MPC



Our Goal

Threshold Interactive Mercurial Signatures

$\underline{\mathsf{TSign}(M, \mathsf{sk}) \rightarrow \sigma}$

$$\sigma = (Z, Y, \hat{Y}) = \left(\left(\prod_{i=1}^{\ell} M_i^{x_i} \right)^y, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}} \right)$$

$\mathsf{ConvertSig}(\mathsf{vk}, M, \sigma \ ; \ \rho) \rightarrow \tilde{\sigma}$

$\underline{\mathsf{ConvertVK}(\mathsf{vk} \ ; \ \rho) \rightarrow \tilde{\mathsf{vk}}}$

$$\tilde{\mathsf{vk}} = \mathsf{vk}^\rho = (\hat{G}^{x_1\rho}, \ldots, \hat{G}^{x_\ell\rho})$$

Change Sign to TSign with 2 Party Interactive Protocol

Verification and Conversion method in the original are adapted directly

... To keep the flexibility for applications using Mercurial Signatures

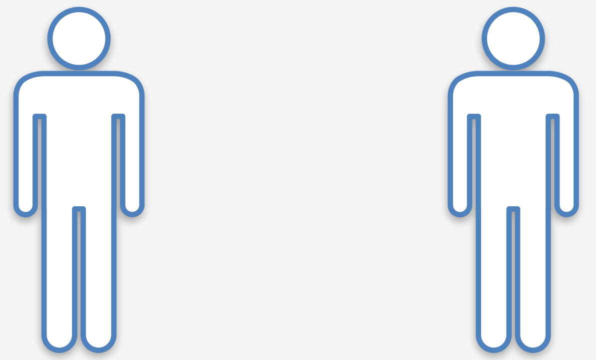
Key observations 1/2

Key is shared additively / Ephemeral Randomness is shared multiplicatively

$$\sigma = \left(\left(\prod_{i=1}^{l} M_i^{x_i} \right)^{y}, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}} \right)$$

$$x_i \rightarrow x_i^0 + x_i^1$$

$$y \rightarrow y_0 \cdot y_1$$



(x_i^0, y_0) (x_i^1, y_1)

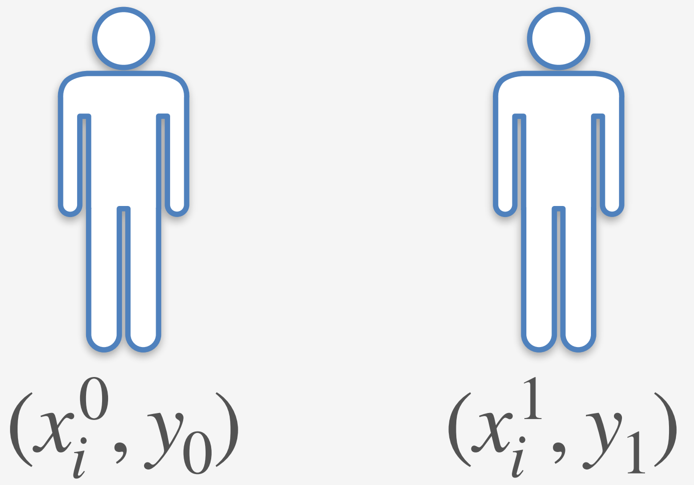
This multiplicative sharing makes easier to add randomness one by one in the Sequential stream

Key observations 2/2

Key is shared additively / Ephemeral Randomness is shared multiplicatively

$$\sigma = \left(\left(\prod_{i=1}^{l} M_i^{x_i} \right)^{y}, G^{\frac{1}{y}}, \hat{G}^{\frac{1}{y}} \right)$$

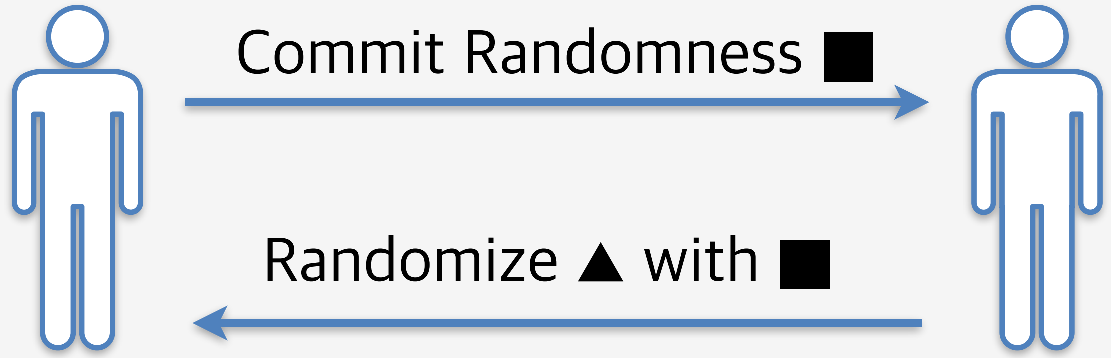
$$x_i \rightarrow x_i^0 + x_i^1$$
$$y \rightarrow y_0 \cdot y_1$$



This multiplicative sharing makes easier to add randomness one by one in the Sequential stream

Blinded Computation

⋯ Blinding local computation using other party's random factor



Naive protocol

$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$

$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$

$$Z_0 = \prod_{i=1}^{\ell} M_i^{x_i^0}$$

Z_0 →

$$y_1 \xleftarrow{\$} \mathbb{Z}_p^*, Y_1 = G^{\frac{1}{y_1}},\ \hat{Y}_1 = \hat{G}^{\frac{1}{y_1}}$$

$$Z_1 = \left(Z_0 \cdot \prod_{i=1}^{l} M_i^{x_i^1}\right)^{y_1}$$

← $Z_1, Y_1, \hat{Y}_1$

$$y_0 \xleftarrow{\$} \mathbb{Z}_p^*, Y = Y_1^{\frac{1}{y_0}},\ \hat{Y} = \hat{Y}_1^{\frac{1}{y_0}}$$

$$Z = Z_1^{y_0}$$

Return $\sigma = (Z, Y, \hat{Y})$

Problem : Naive protocol

$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$

$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$

$$Z_0 = \prod_{i=1}^{\ell} M_i^{x_i^0}$$

Z_0 is computed with only deterministic values

$P_0 \to P_1$: Z_0

$$y_1 \xleftarrow{\$} \mathbb{Z}_p^*, Y_1 = G^{\frac{1}{y_1}}, \hat{Y}_1 = \hat{G}^{\frac{1}{y_1}}$$

$$Z_1 = \left(Z_0 \cdot \prod_{i=1}^{l} M_i^{x_i^1}\right)^{y_1}$$

$P_1 \to P_0$: $Z_1, Y_1, \hat{Y}_1$

It makes a difficulty for setting secure simulator

It is required to blind Z_0 without harming protocol

$$y_0 \xleftarrow{\$} \mathbb{Z}_p^*, Y = Y_1^{\frac{1}{y_0}}, \hat{Y} = \hat{Y}_1^{\frac{1}{y_0}}$$

$$Z = Z_1^{y_0}$$

Return $\sigma = (Z, Y, \hat{Y})$

Final protocol 1/4

$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$	$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$
$y_0 \xleftarrow{\$} \mathbb{Z}_p^*,\ Y_0 = G^{\frac{1}{y_0}},\ \hat{Y}_0 = \hat{G}^{\frac{1}{y_0}}$	

$Y_0, \hat{Y}_0, \pi_0^{(1)}$ →

Commitment of y_0 (using ZK)

Return $\sigma = (Z, Y, \hat{Y})$

Return $\sigma = (Z, Y, \hat{Y})$

Final protocol 2/4

$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$		$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$
$y_0 \xleftarrow{\$} \mathbb{Z}_p^*,\ Y_0 = G^{\frac{1}{y_0}},\ \hat{Y}_0 = \hat{G}^{\frac{1}{y_0}}$	$Y_0, \hat{Y}_0, \pi_0^{(1)}$ →	
		$r \xleftarrow{\$} \mathbb{Z}_p^*,\ y_1 \xleftarrow{\$} \mathbb{Z}_p^*$
	← $K_1, \pi_1^{(1)}$	$K_1 = \boxed{Y_0^r} \cdot \prod_{i=1}^{\ell} M_i^{x_i^1},\ Y = Y_0^{\frac{1}{y_1}},\ \hat{Y} = \hat{Y}_0^{\frac{1}{y_1}}$
Return $\sigma = (Z, Y, \hat{Y})$		Return $\sigma = (Z, Y, \hat{Y})$

Commitment of y_1 (using ZK)

+

Partial signature is blinded with $Y_0^r = G^{\frac{r}{y_0}}$

Final protocol 3/4

$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$

$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$

$y_0 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*,\ Y_0 = G^{\frac{1}{y_0}},\ \hat{Y}_0 = \hat{G}^{\frac{1}{y_0}}$

$Y_0, \hat{Y}_0, \pi_0^{(1)} \rightarrow$

$r \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*,\ y_1 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$

$K_1 = Y_0^r \cdot \prod_{i=1}^{\ell} M_i^{x_i^1},\ Y = Y_0^{\frac{1}{y_1}},\ \hat{Y} = \hat{Y}_0^{\frac{1}{y_1}}$

$\leftarrow K_1, \pi_1^{(1)}$

$Z_0 = \left(K_1 \cdot \prod_{i=i}^{\ell} M_i^{x_i^0}\right)^{y_0}$

$Z_0, \pi_0^{(2)} \rightarrow$

Partial signing and randomizing

Expansion... $Z_0 = G^{\frac{r}{y_0}\cdot y_0} \cdot \left(\prod_{i=1}^{\ell} M_i^{x_i^0 + x_i^1}\right)^{y_0}$

Return $\sigma = (Z, Y, \hat{Y})$

Return $\sigma = (Z, Y, \hat{Y})$

Final protocol 4/4

$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$		$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$
$y_0 \xleftarrow{\$} \mathbb{Z}_p^*,\ Y_0 = G^{\frac{1}{y_0}},\ \hat{Y}_0 = \hat{G}^{\frac{1}{y_0}}$	$Y_0, \hat{Y}_0, \pi_0^{(1)}$ →	
	← $K_1, \pi_1^{(1)}$	$r \xleftarrow{\$} \mathbb{Z}_p^*,\ y_1 \xleftarrow{\$} \mathbb{Z}_p^*$ $K_1 = Y_0^r \cdot \prod_{i=1}^{\ell} M_i^{x_i^1},\ Y = Y_0^{\frac{1}{y_1}},\ \hat{Y} = \hat{Y}_0^{\frac{1}{y_1}}$
$Z_0 = \left(K_1 \cdot \prod_{i=i}^{\ell} M_i^{x_i^0}\right)^{y_0}$	$Z_0, \pi_0^{(2)}$ →	
Return $\sigma = (Z, Y, \hat{Y})$		$Z = \left(Z_0 \cdot G^{-r}\right)^{y_1}$ **Return** $\sigma = (Z, Y, \hat{Y})$

Offsetting blinded part and randomizing

Expansion... $Z = \left(\boxed{G^{r-r}} \cdot \prod_{i=i}^{\ell} M_i^{x_i^0 + x_i^1}\right)^{y_0 y_1}$

Signing Oracle Simulation : Corrupted P_0

Corr.		Sim. with $\mathsf{Sign}(\mathsf{sk}, \cdot)$
$\mathsf{P}_0 : M = \{M_i\}_{i \in [\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i \in [\ell]}$		$\mathsf{P}_1 : M = \{M_i\}_{i \in [\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i \in [\ell]}$
$(Y_0, \hat{Y}_0, \pi_0^{(1)}) \leftarrow \mathscr{A}(\mathsf{st})$	$Y_0, \hat{Y}_0, \pi_0^{(1)}$ →	$(Z', Y', \hat{Y}') \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$
		$K_1 \xleftarrow{\$} \mathbb{G},\ Y \leftarrow Y',\ \hat{Y} \leftarrow \hat{Y}'_0$
	← $K_1, \pi_1^{(1)}$	$\pi_1^{(1)} \leftarrow \mathsf{ZKPoK.Sim}(Z_1, Y_0, M)$
$(Z_0, \pi_0^{(2)}) \leftarrow \mathscr{A}(\mathsf{st}, K_1, \pi_1^{(1)})$	$Z_0, \pi_0^{(2)}$ →	
		$Z \leftarrow Z' \quad \sigma = (Z, Y, \hat{Y})$
	← $\sigma, \pi_1^{(1)}$	$\pi_1^{(2)} \leftarrow \mathsf{ZKPoK.Sim}(Z, Z_0, Y, Y_0)$
Return $\sigma = (Z, Y, \hat{Y})$		**Return** $\sigma = (Z, Y, \hat{Y})$

Signing Oracle Simulation : Corrupted P_1

Sim. with $\mathsf{Sign}(\mathsf{sk}, \cdot)$		Corr.
$\mathsf{P}_0 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_0 = \{x_i^0\}_{i\in[\ell]}$		$\mathsf{P}_1 : M = \{M_i\}_{i\in[\ell]} \in (\mathbb{G})^\ell \quad \mathsf{sk}_1 = \{x_i^1\}_{i\in[\ell]}$
$(Z', Y', \hat{Y}') \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ $Y_0 \leftarrow Y',\ \hat{Y}_0 \leftarrow \hat{Y}'$ $\pi_0^{(1)} \leftarrow \mathsf{ZKPoK.Sim}(Y_0, \hat{Y}_0)$	$Y_0, \hat{Y}_0, \pi_0^{(1)}$ →	
	← $K_1, \pi_1^{(1)}$	$(K_1, \pi_0^{(1)}) \leftarrow \mathscr{A}(\mathsf{st}, Y_0, \hat{Y}_0, \pi_1^{(1)})$
$r \leftarrow \mathsf{ZKPoK.Ext}(\pi_1^{(1)});\ Z_0 \leftarrow Z'G^r$ $\pi_0^{(2)} \leftarrow \mathsf{ZKPoK.Sim}(Z_0, K_1, M, Y_0)$	$Z_0, \pi_0^{(2)}$ →	
	← $\sigma, \pi_1^{(1)}$	$(\sigma, \pi_0^{(1)}) \leftarrow \mathscr{A}(\mathsf{st}, Z_0, \pi_0^{(2)})$
Return $\sigma = (Z, Y, \hat{Y})$		**Return** $\sigma = (Z, Y, \hat{Y})$

Comment for Extension

01 | Sequential communication model between Two parties

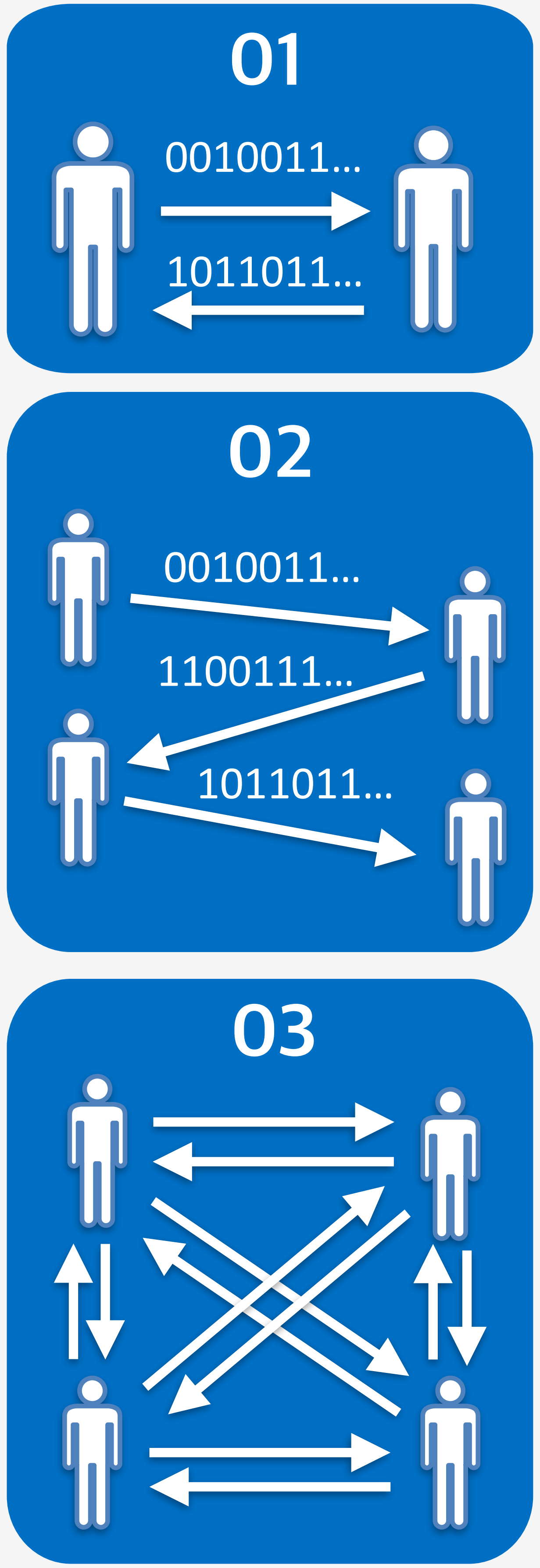
- It allows 1 party corruption

02 | Sequential communication model among t out of n parties

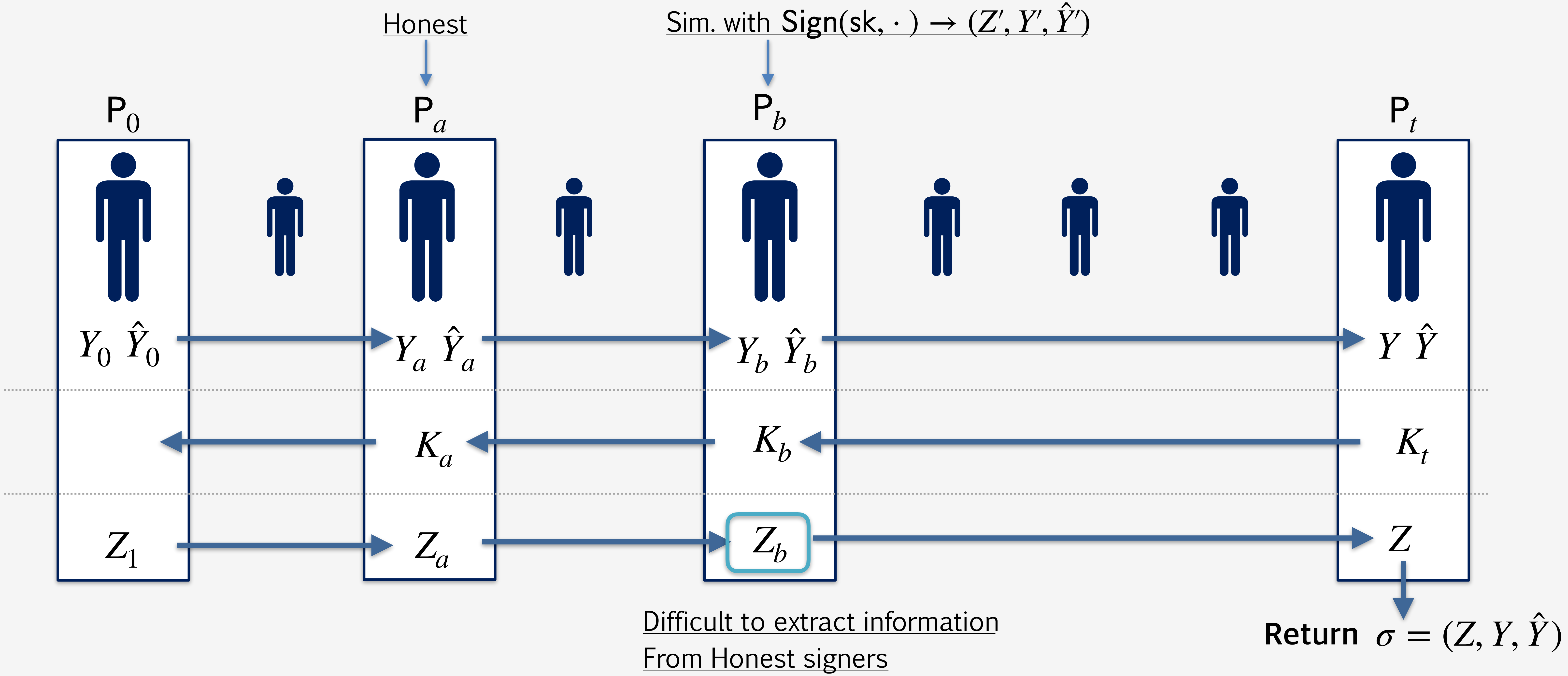
- It allows up to t-1 party corruption
- Pre-processing for secret sharing is required

03 | Synchronized communication model among t out of n parties

- It allows up to t-1 party corruption
- Broadcast messages with traditional MPC

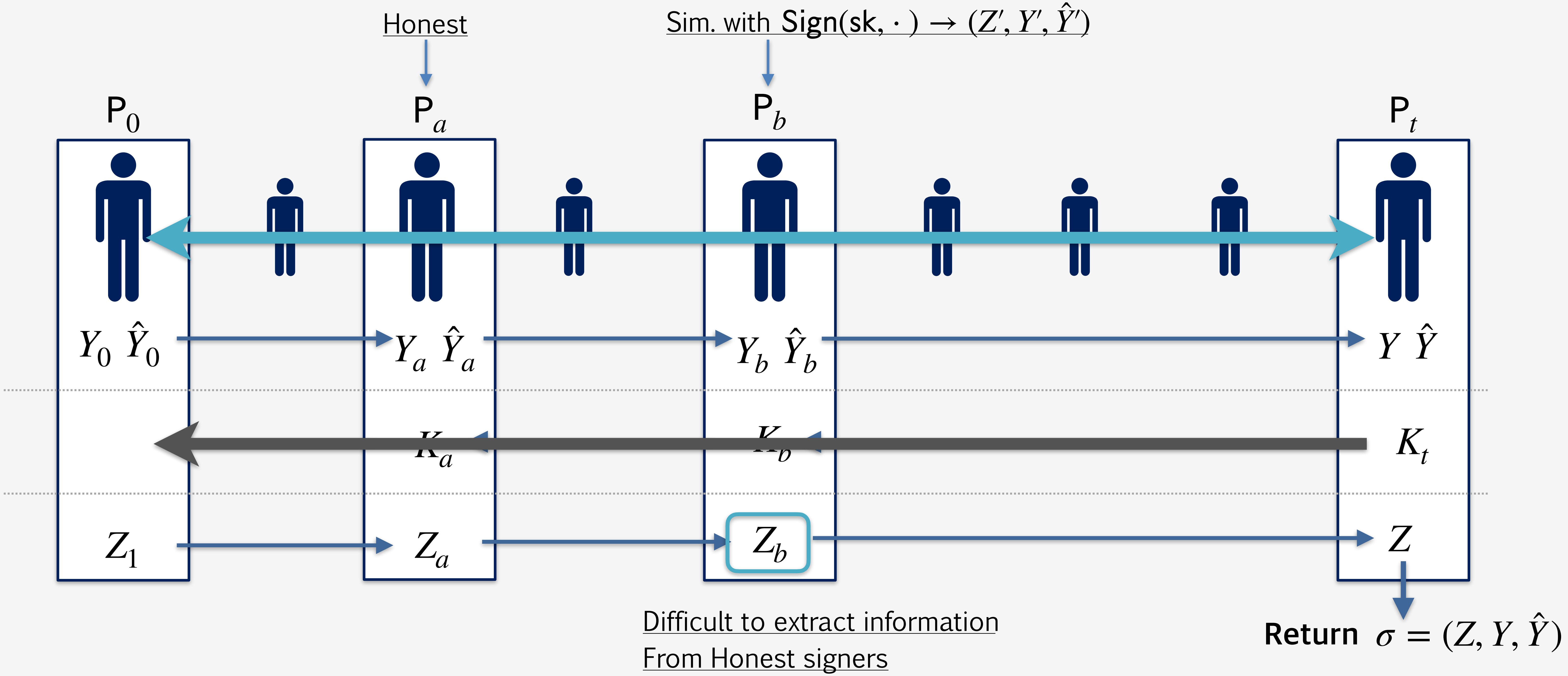


Challenge for Extension



To construct the simulator for intermediators, another blinding trick is required

Challenge for Extension



To construct the simulator for intermediators, another blinding trick is required

⋯ Zero-Sharing over Public Channel (including pre-processing phase)

Performance of Measurement

The cost is proportional to the size of the message and the number of parties

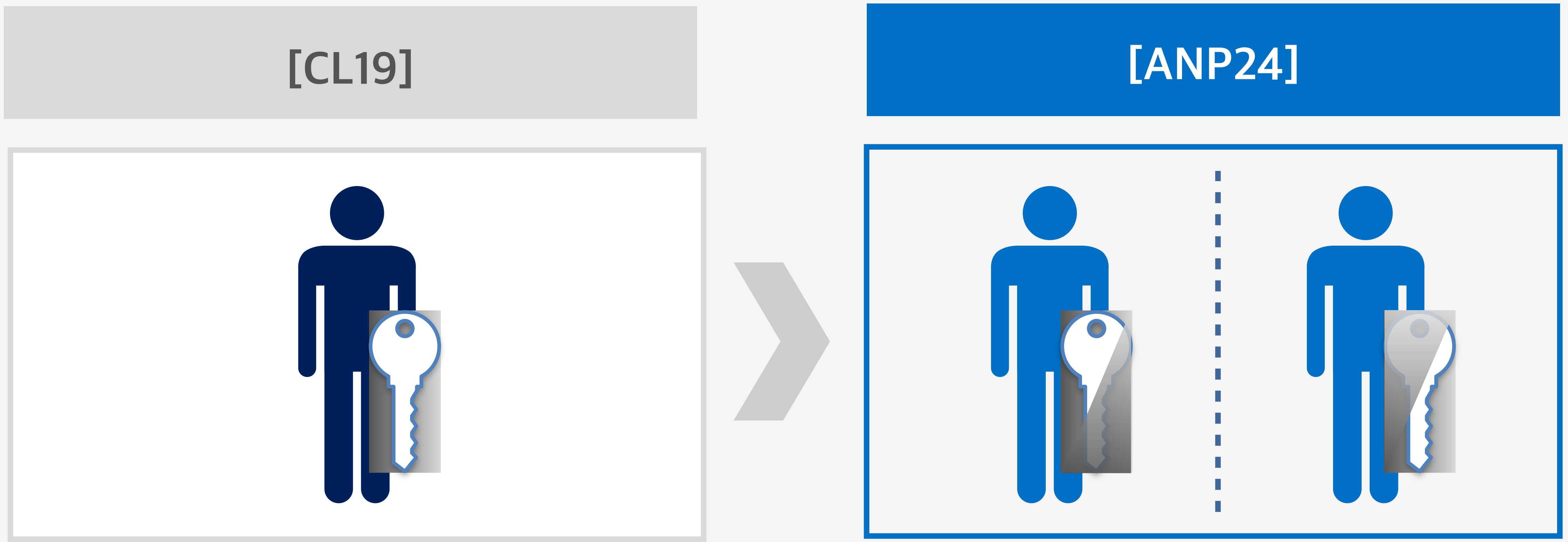
Scheme	# of Parties	Message Size 2	Message Size 5	Message Size 10
Mercurial Signatures [FHS19]	1	0.3	0.4	0.5
Sequential Communication in Two Parties	2	3.9	6.2	10.1
Sequential Communication in t out of n	5	13.3	19.3	29.6
Sequential Communication in t out of n	10	28.0	40.8	60.5

(Unit : millisecond)

Contribution of our work

> Extension for Mercurial Signatures for Distributed Parties (with threshold)

1. Provides distributed trust of the root authority for delegatable credential system
2. Improves privacy for standard anonymous credentials



> Implementation of our scheme to show its reasonable cost

Future Direction

> More Applications

… e.g. Delegatable Anonymous Credentials System

> Stronger security

… e.g. Asynchronous and non-erasable Communication Model,

Security for Adaptive Corruption

Thank you for listening

The latest version of our paper (https://eprint.iacr.org/2024/625)

Artifact of Implementation is accepted by IACR