Vector Semi-Commitment:

Optimizing MPC-in-the-Head based Signatures

Seongkwang Kim[1], **Byeonghak Lee**[1], and Mincheol Son[2]

[1]Samsung SDS, Korea [2]KAIST, Korea

Public Key Crypto in GAPS?

Public Key Crypto in GAPS?

- During my PhD, I have primarily focused on symmetric provable security
  - Tweakable Block Ciphers, MAC, AEAD, …

Public Key Crypto in GAPS?

- During my PhD, I have primarily focused on symmetric provable security

- But when I attended Eurocrypt 2022, I saw following session titles:

Public Key Crypto in GAPS?

- During my PhD, I have primarily focused on symmetric provable security

- But when I attended Eurocrypt 2022, I saw following session titles:

Secure multiparty computation 1

⋮

Secure multiparty computation 5

Post-quantum cryptography 1

⋮

Post-quantum cryptography 4

Public Key Crypto in GAPS?

- During my PhD, I have primarily focused on symmetric provable security
- But when I attended Eurocrypt 2022, I saw following session titles:

Secure multiparty computation 1

⋮

Secure multiparty computation 5

Post-quantum cryptography 1

⋮

Post-quantum cryptography 4

- I began exploring research fields closely related to secret key cryptography

Public Key Crypto in GAPS?

- Good news! Some PQ Signatures are based on symmetric key assumptions
  - SPHINCS+: Pure hash-based digital signature standardized by NIST (FIPS-205, SLH-DSA)
  - PICNIC: MPC-in-the-Head + LowMC block cipher
  - AIMer: MPC-in-the-Head + dedicated one-way function
  - FAEST: VOLE-in-the-Head + AES block cipher
  - …

Public Key Crypto in GAPS?

- Good news! Some PQ Signatures are based on symmetric key assumptions
  - SPHINCS+, PICNIC, AIMer, FAEST, …

- MPC-in-the-Head (MPCitH)
  - Enables post-quantum digital signatures from one-way function
  - Some tools for symmetric key proofs (e.g. H-coefficient technique) are used

➔ It felt relatively familiar to me, and I imagine it will be same for you

Public Key Crypto in GAPS?

- Good news! Some PQ Signatures are based on symmetric key assumptions
  - SPHINCS+, PICNIC, AIMer, FAEST, …
- MPC-in-the-Head (MPCitH)
  - Enables post-quantum digital signatures from one-way function
- In this talk, I will briefly introduce
  - MPC-in-the-Head paradigm and
  - Recent optimization: Vector Semi-Commitment

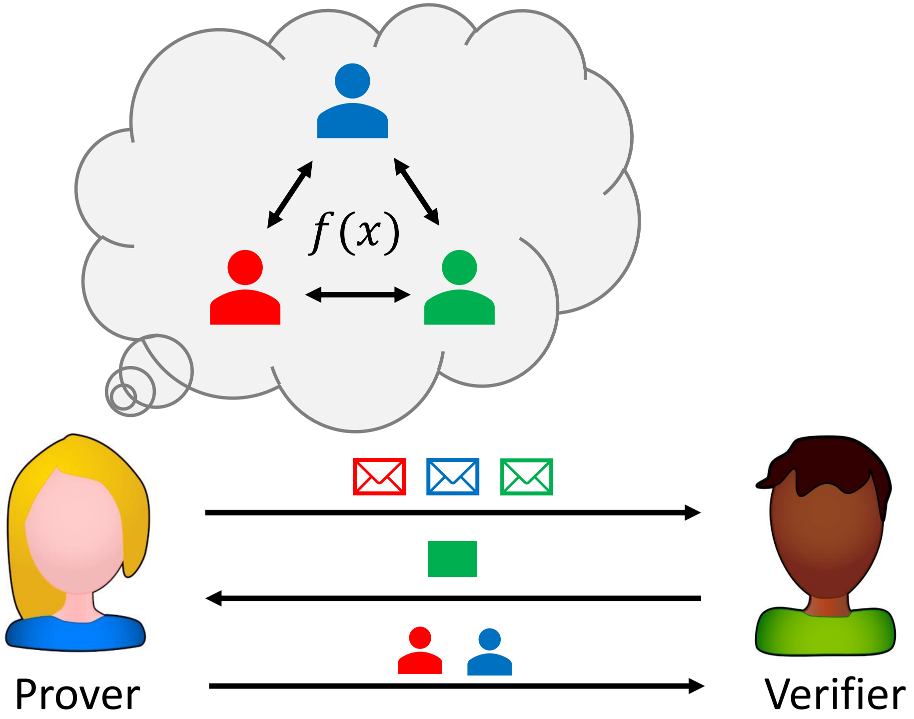
MPC-in-the-Head

MPC-in-the-Head (MPCitH)

- [IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Amit Sahai: “Zero-knowledge from secure multiparty computation” (STOC 2007)

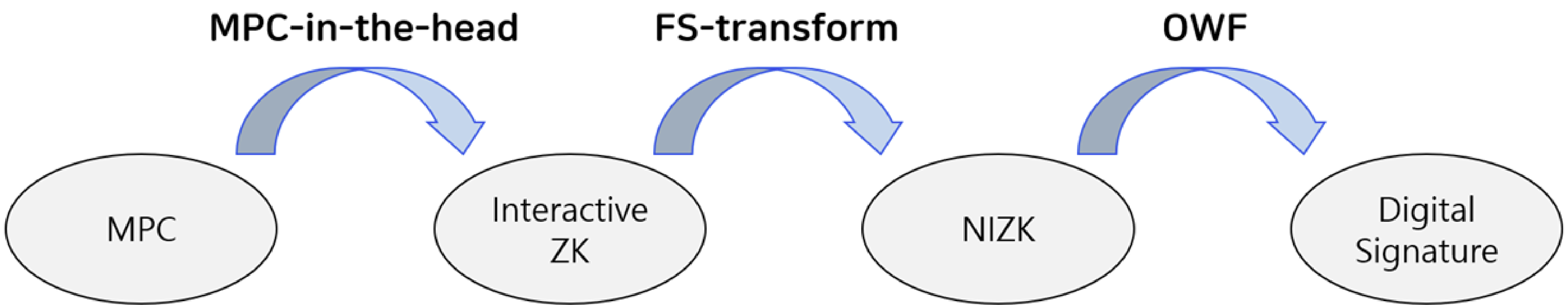
- Turns Multiparty Computation (MPC) into Zero-Knowledge-Proof-of-Knowledge (ZKPoK)

- Can be applied to any cryptographic problem
  - E.g. Knowledge of block cipher key



MPCitH-based Signatures

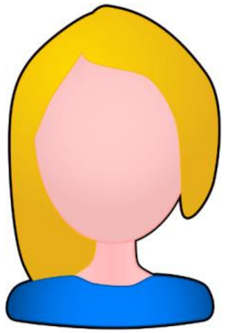
- MPCitH enables post-quantum signature schemes
  - Minimal assumption: Security of digital signature only relies on the one-wayness of OWF
  - 6 of 15 in NIST additional PQC standardization are based on MPCitH
  - MIRA, MQOM, …



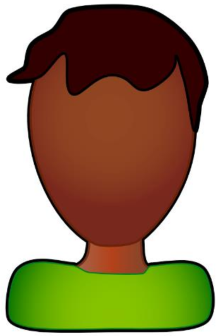
MPC-in-the-Head Transform

- Prover wants to prove the knowledge of x s.t. $F(x) = y$

Prover



Verifier



MPC-in-the-Head Transform

- Prover wants to prove the knowledge of x s.t. $F(x) = y$

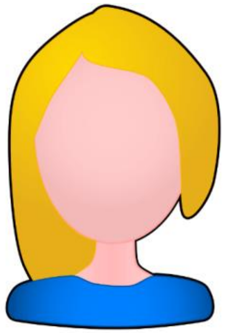
(1) Generate and commit shares

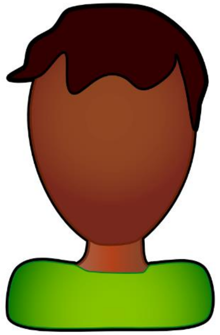
$$x = x^{(1)} + \cdots + x^{(N)}$$

Send Commits

$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$

Prover $\longrightarrow$ Verifier





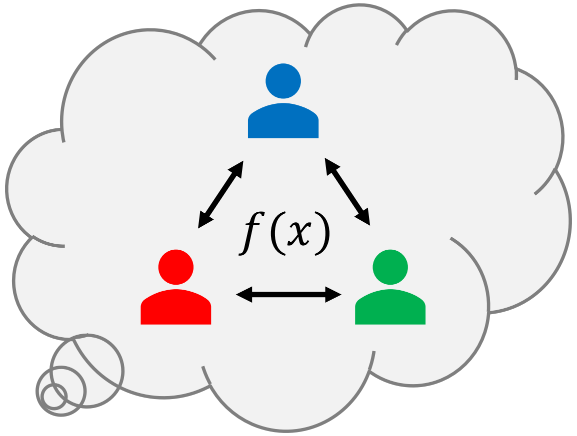
MPC-in-the-Head Transform

- Prover wants to prove the knowledge of x s.t. $F(x) = y$

(1) Generate and commit shares

$$x = x^{(1)} + \cdots + x^{(N)}$$

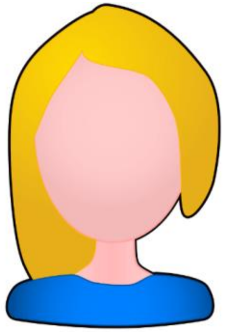
(2) Run MPC in their Head



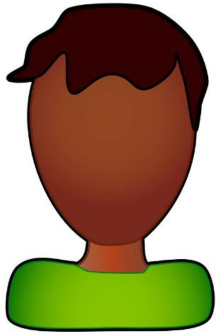
Send Commits

$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$

Prover



Verifier



Send Broadcasts

$\alpha^{(1)}, \ldots, \alpha^{(n)}$

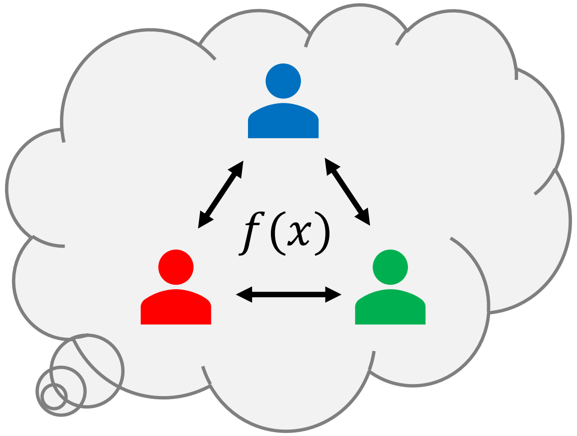
MPC-in-the-Head Transform

- Prover wants to prove the knowledge of x s.t. $F(x) = y$

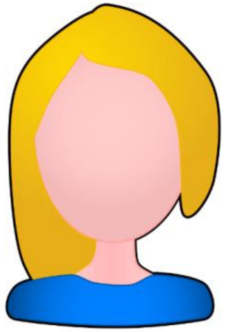
(1) Generate and commit shares

$$x = x^{(1)} + \cdots + x^{(N)}$$

(2) Run MPC in their Head



Prover



Send Commits

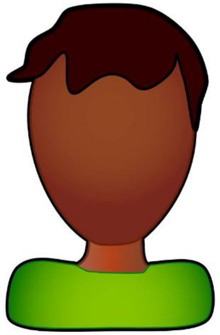
$\mathrm{Com}(x^{(1)}), \ldots, \mathrm{Com}(x^{(N)})$ $\longrightarrow$

Send Broadcasts

$\alpha^{(1)}, \ldots, \alpha^{(n)}$ $\longrightarrow$

i^* $\longleftarrow$

Verifier



(3) Choose a party

$$i^* \leftarrow_{\$} \{1, \ldots, N\}$$

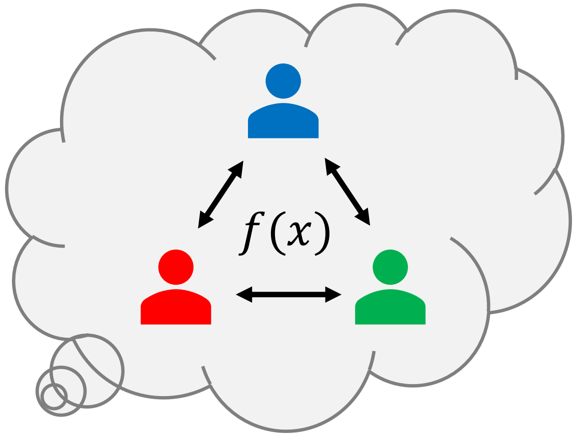
MPC-in-the-Head Transform

- Prover wants to prove the knowledge of x s.t. $F(x) = y$

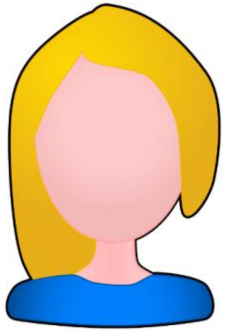
(1) Generate and commit shares

$$x = x^{(1)} + \cdots + x^{(N)}$$

(2) Run MPC in their Head



Prover



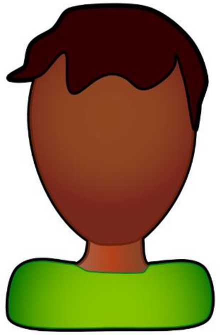
Send Commits
$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$ $\longrightarrow$

Send Broadcasts
$\alpha^{(1)}, \ldots, \alpha^{(n)}$ $\longrightarrow$

i^* $\longleftarrow$

$\left\{x^{(i)}\right\}_{i \neq i^*}$ $\longrightarrow$

Verifier



(3) Choose a party
$i^* \leftarrow_\$ \{1, \ldots, N\}$

(4) Open parties $\{1, \ldots, N\} \setminus \{i^*\}$

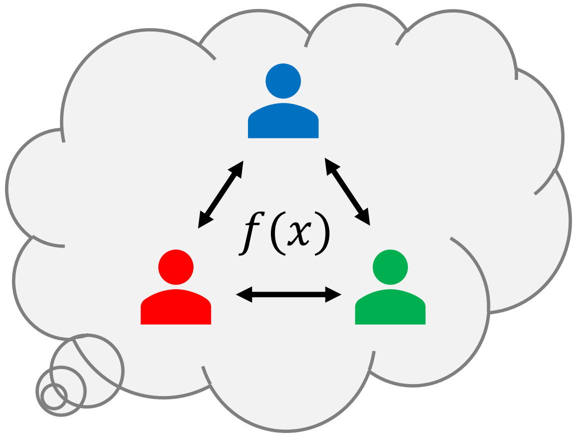
MPC-in-the-Head Transform

- Prover wants to prove the knowledge of x s.t. $F(x) = y$

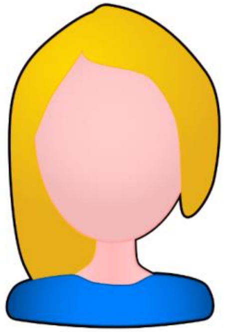
(1) Generate and commit shares

$$x = x^{(1)} + \cdots + x^{(N)}$$

(2) Run MPC in their Head



Prover



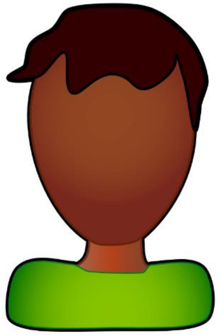
Send Commits $\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$ →

Send Broadcasts $\alpha^{(1)}, \ldots, \alpha^{(n)}$ →

← i^*

$\left\{x^{(i)}\right\}_{i \neq i^*}$ →

Verifier



(3) Choose a party

$$i^* \leftarrow_{\$} \{1, \ldots, N\}$$

(4) Open parties $\{1, \ldots, N\} \setminus \{i^*\}$

(5) Check $\forall i \neq i^*$

- Commits $\mathrm{Com}\left(x^{(i)}\right)$
- Broadcast values

$$\alpha^{(i)} = \phi\left(x^{(i)}\right)$$

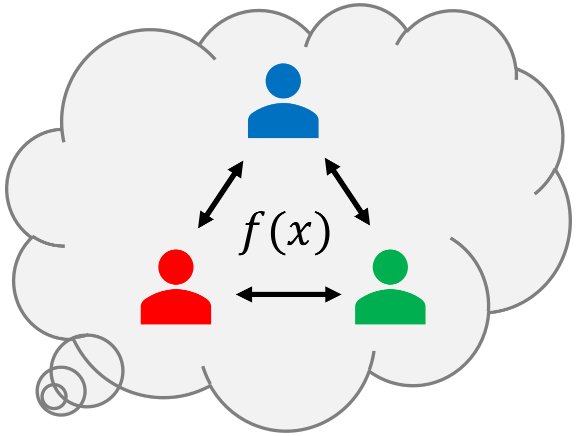
Check MPC result

$$\overline{F}(\alpha) = y$$

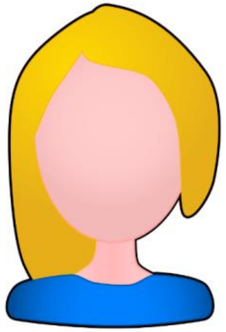
(1) Generate and commit shares

$x = x^{(1)} + \cdots + x^{(N)}$

(2) Run MPC in their Head



Prover



Send Commits

$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$ →

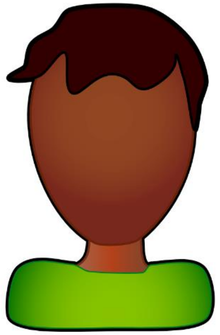
Send Broadcasts

$\alpha^{(1)}, \ldots, \alpha^{(n)}$ →

← i^*

$\left\{x^{(i)}\right\}_{i \neq i^*}$ →

Verifier



(3) Choose a party

$i^* \leftarrow_{\$} \{1, \ldots, N\}$

(5) Check $\forall i \neq i^*$

- Commits $\mathrm{Com}\left(x^{(i)}\right)$
- MPC computations

$\alpha^{(i)} = \phi\left(x^{(i)}\right)$

Check MPC result

$\overline{F}(\alpha) = y$

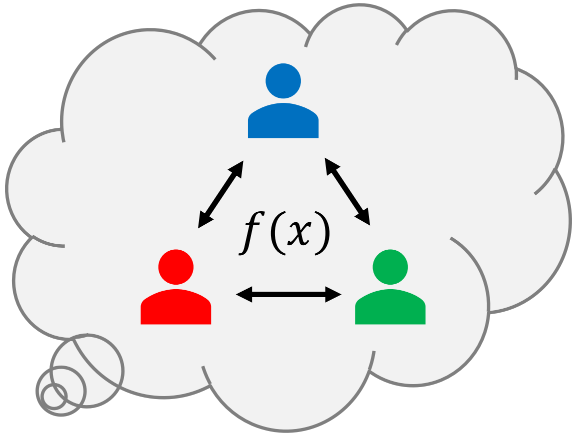
(4) Open parties $\{1, \ldots, N\} \setminus \{i^*\}$

- Zero-knowledge for verifier
  - x is still secret because $x^{(i^*)}$ is unknown to verifier
  - unopened party's secret cannot be revealed: $x^{(i^*)}$ from $\mathrm{Com}(x^{(i^*)})$
  - $\mathrm{Com}(x^{(i^*)})$ should be indistinguishable to random (hiding property of Commitment)

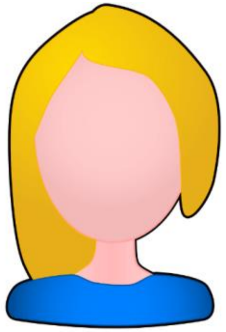
(1) Generate and commit shares

$$x = x^{(1)} + \cdots + x^{(N)}$$

(2) Run MPC in their Head



Prover



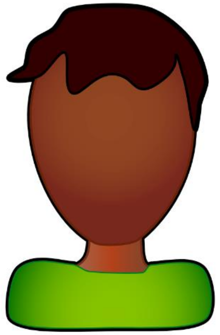
Send Commits $\mathrm{Com}(x^{(1)}), \ldots, \mathrm{Com}(x^{(N)})$ →

Send Broadcasts $\alpha^{(1)}, \ldots, \alpha^{(n)}$ →

← i^*

$\{x^{(i)}\}_{i \neq i^*}$ →

Verifier



(3) Choose a party

$$i^* \leftarrow_\$ \{1, \ldots, N\}$$

(4) Open parties $\{1, \ldots, N\} \setminus \{i^*\}$

(5) Check $\forall i \neq i^*$

- Commits $\mathrm{Com}(x^{(i)})$
- MPC computations

$$\alpha^{(i)} = \phi(x^{(i)})$$

Check MPC result

$$\overline{F}(\alpha) = y$$

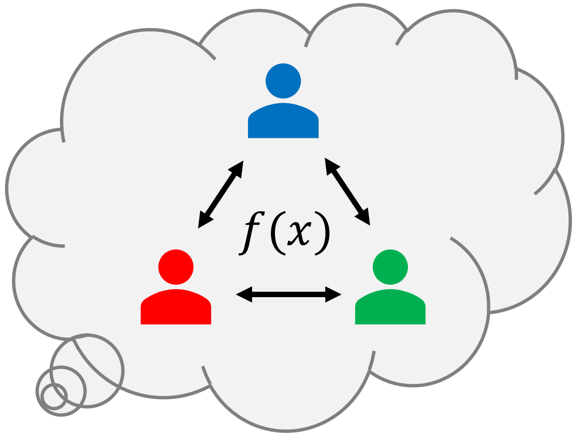
- Malicious Prover cheats successfully if:
  - unopened party was corrupted: $\mathrm{Com}(x^{(i^*)})$ and $\alpha^{(i^*)}$ are maliciously chosen without $x^{(i^*)}$

    ➔ probability: 1/N
  - Corruption of $i \neq i^*$ did not detected: Commitment check or MPC computation check failed

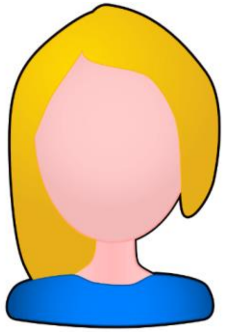
(1) Generate and commit shares

$$x = x^{(1)} + \cdots + x^{(N)}$$

(2) Run MPC in their Head



Prover



Send Commits

$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$ $\longrightarrow$

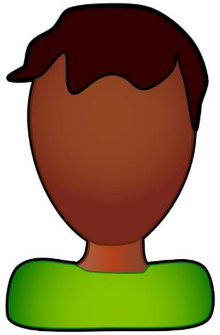
Send Broadcasts

$\alpha^{(1)}, \ldots, \alpha^{(n)}$ $\longrightarrow$

i^* $\longleftarrow$

$\left\{x^{(i)}\right\}_{i \neq i^*}$ $\longrightarrow$

Verifier



(3) Choose a party

$$i^* \leftarrow_\$ \{1, \ldots, N\}$$

(5) Check $\forall i \neq i^*$

- Commits $\mathrm{Com}\left(x^{(i)}\right)$
- MPC computations

$$\alpha^{(i)} = \phi\left(x^{(i)}\right)$$

Check MPC result

$$\overline{F}(\alpha) = y$$

(4) Open parties $\{1, \ldots, N\} \setminus \{i^*\}$

- Malicious Prover cheats successfully if:
  - unopened party was corrupted ➔ probability: 1/N
  - Corruption of $i \neq i^*$ did not detected ➔ probability: ϵ (typically, small)

Repeat τ times where

$$\left(\frac{1}{N} + \epsilon\right)^\tau \simeq 2^{-\lambda}$$

(1) Generate and commit shares

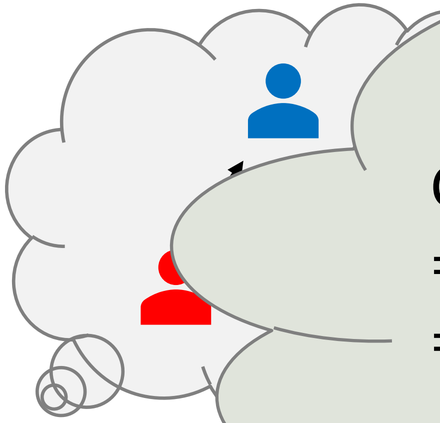
$$x = x^{(1)} + \cdots + x^{(N)}$$

Send Commits

$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$

(2) Run MPC in their Head

Prover



(3) Choose a party

$i^* \leftarrow_\$ \{1, \ldots, N\}$

$\neq i^*$

$\mathrm{Com}\left(x^{(i)}\right)$

tations

$\varphi\left(x^{(i)}\right)$

MPC result

$\overline{F}(\alpha) = y$

Commits are binding & No parties are corrupted

⇒ the input to MPC protocol is binded

⇒ can cheats only if MPC check fails for the binded input

(4) Open parties $\{1, \ldots, N\}$

- Malicious Prover cheats successfully if:
  - unopened party was corrupted ➔ probability: 1/N
  - Corruption of $i \neq i^*$ did not detected ➔ probability: ϵ (typically, small)

Repeat τ times where

$$\left(\frac{1}{N} + \epsilon\right)^{\tau} \simeq 2^{-\lambda}$$

(1) Generate and commit shares

$x = x^{(1)} + \cdots + x^{(N)}$

Send Commits

$\mathrm{Com}(x^{(1)}), \ldots, \mathrm{Com}(x^{(N)})$

(2) Run MPC in their Head

Prover

(3) Choose a party

$i^* \leftarrow_\$ \{1, \ldots, N\}$

$\neq i^*$

$\mathrm{Com}(x^{(i)})$

tations

$\varphi(x^{(i)})$

MPC result

$\overline{F}(\alpha) = y$

Commits are semi-binding & No parties are corrupted

$\Rightarrow$ some(=u) inputs to MPC protocol are binded

$\Rightarrow$ can cheats only if MPC check fails for binded(=u) inputs

$\Rightarrow \epsilon$ become $u\epsilon$

(4) Open parties $\{1, \ldots, N\}$

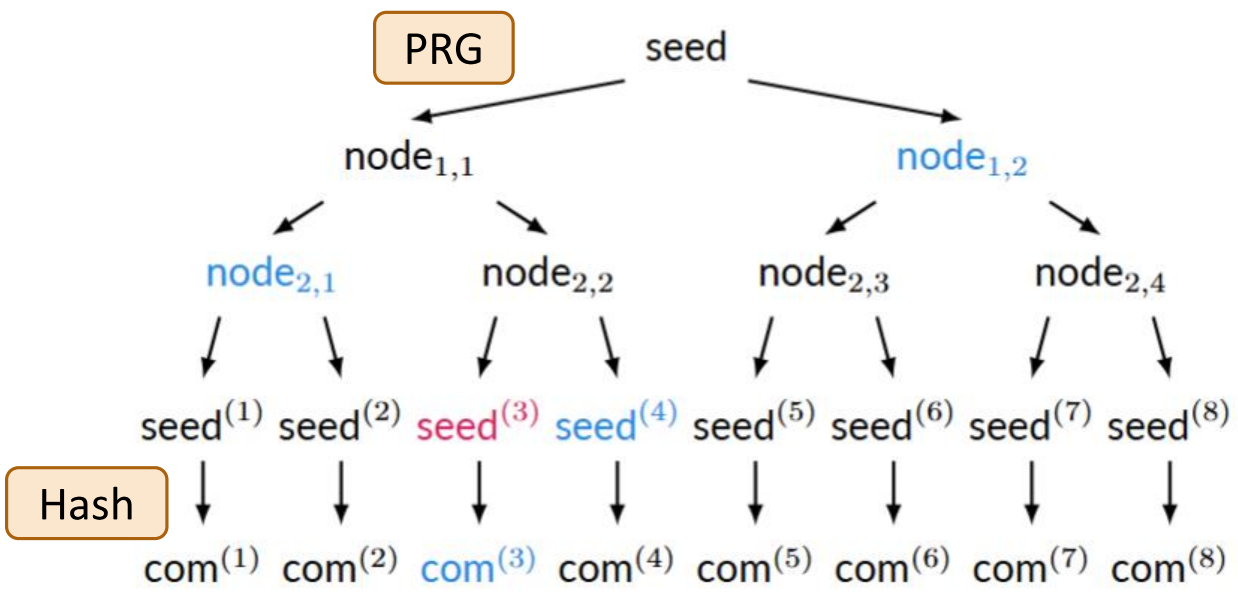
- Malicious Prover cheats successfully if:
  - unopened party was corrupted ➔ probability: 1/N
  - Corruption of $i \neq i^*$ did not detected ➔ probability: ϵ (typically, small)

Repeat τ times where

$$\left(\frac{1}{N} + \epsilon\right)^{\tau} \simeq 2^{-\lambda}$$

Vector (Semi-)Commitment

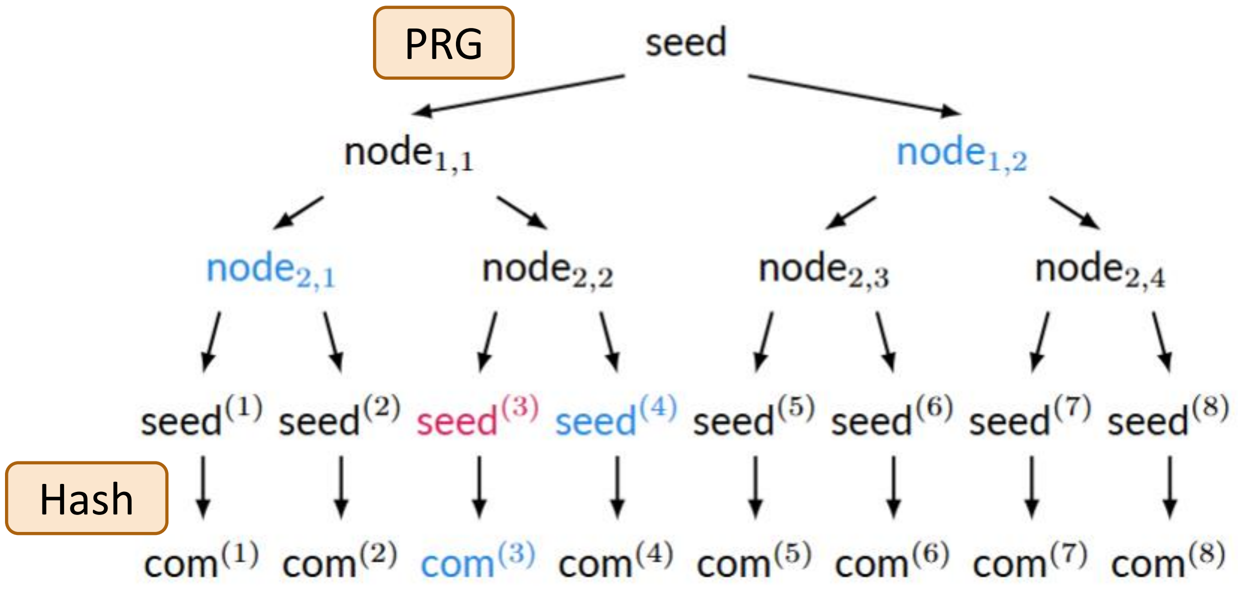
Vector Commitments (VC)



- $\mathrm{VC.Commit(seed)} = (\mathrm{decom}, \mathrm{com})$
  - $\mathrm{com} \coloneqq \left(\mathrm{com}^{(1)}, \ldots, \mathrm{com}^{(8)}\right)$
- $\mathrm{VC.Open}(\mathrm{decom}, \bar{3}) = \mathrm{pdecom}$
  - $\mathrm{pdecom} \coloneqq \left(\mathrm{node}_{1,2}, \mathrm{node}_{2,1}, \mathrm{seed}^{(4)}, \mathrm{com}^{(3)}\right)$
  - All information to evaluate $\mathrm{seed}^{(i)}$ for $i \neq \bar{3}$
- $\mathrm{VC.Verify}(\mathrm{com}, \mathrm{pdecom}, \bar{3}) = \left(\mathrm{seed}^{(i)}\right)_{i \neq 3}$ or $\perp$

$\mathrm{PRG}\left(\mathrm{seed}^{(i)}\right)$ = internal values for i-th party (including $x^{(i)}$

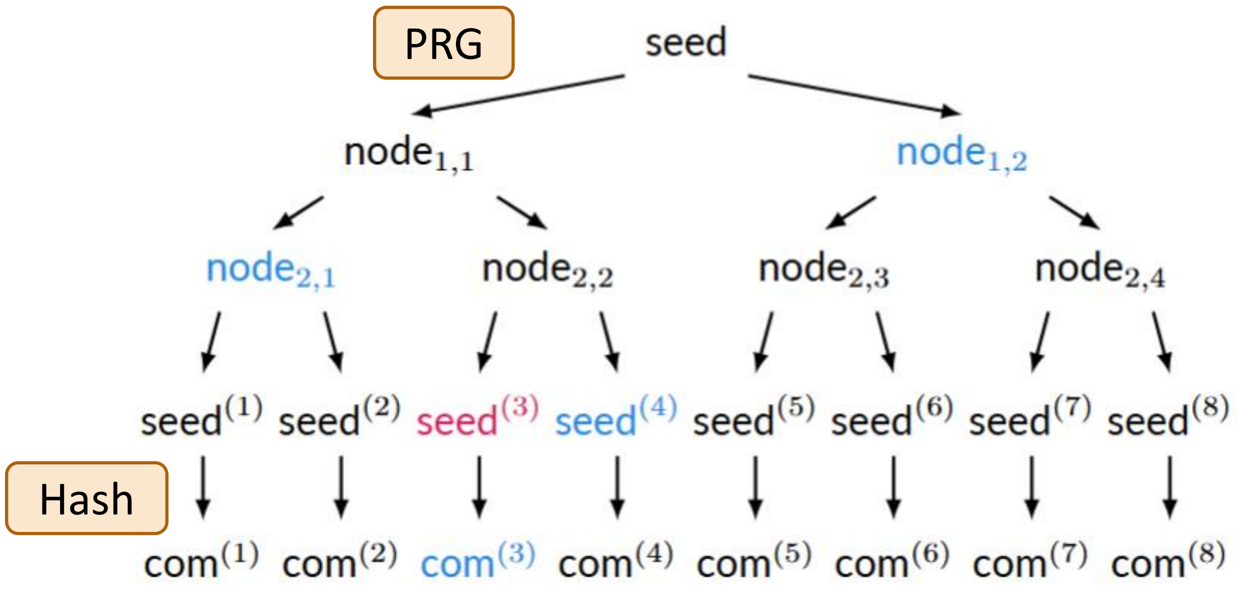
Vector Commitments (VC)



- VC is binding: $\left(\mathrm{com}^{(i)}\right)_{i\in[N]}$ binds $\left(\mathrm{seed}^{(i)}\right)_{i\in[N]}$
  - ➔ One cannot find collisions of Hash
  - ➔ requires $\left|\mathrm{com}^{(i)}\right| \geq 2\lambda$
- VC is hiding: hidden seed cannot be discovered from pdecom
  - ➔ One cannot find preimage of Hash
  - ➔ requires $\left|\mathrm{com}^{(i)}\right| \geq \lambda$

Proof of binding: Collision resistance of Hash / Simple analysis with RO

Vector Commitments (VC)

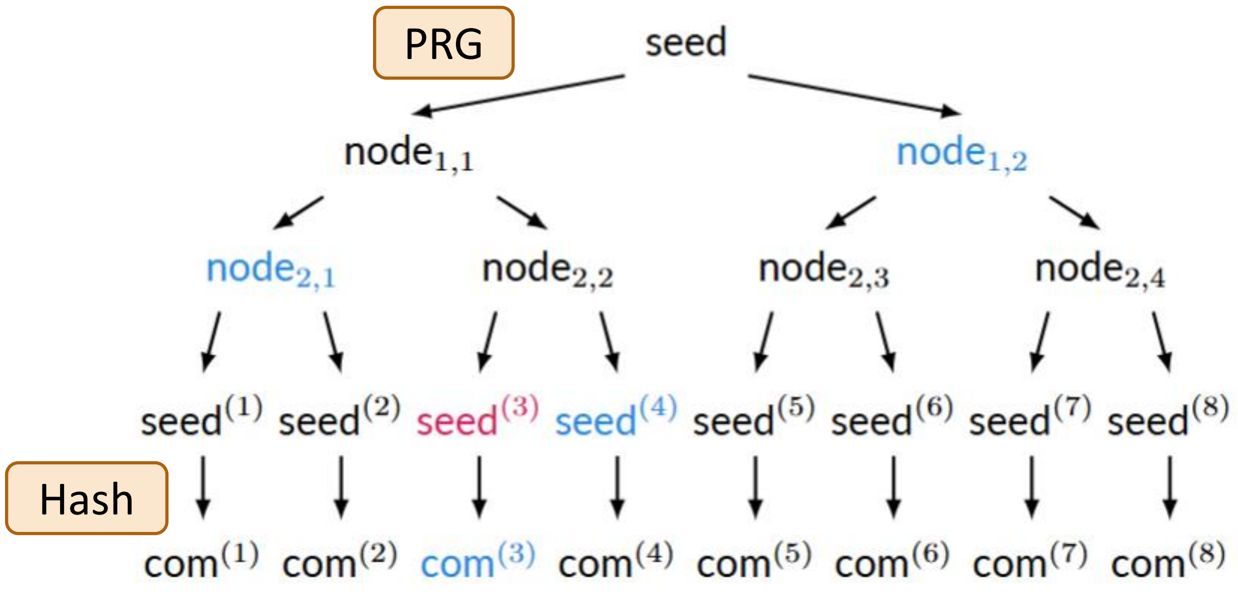


- VC is binding: $\left(\text{com}^{(i)}\right)_{i\in[N]}$ binds $\left(\text{seed}^{(i)}\right)_{i\in[N]}$
  - ➔ One cannot find collisions of Hash
  - ➔ requires $\left|\text{com}^{(i)}\right| \geq 2\lambda$
- VC is hiding: hidden seed cannot be discovered from pdecom
  - ➔ One cannot find preimage of Hash
  - ➔ requires $\left|\text{com}^{(i)}\right| \geq \lambda$

Proof of hiding: the adversary cannot distinguish $\text{seed}^{(3)}$ from random λ-bit string

➔ PRG assumption + preimage resistance / H-coefficient technique

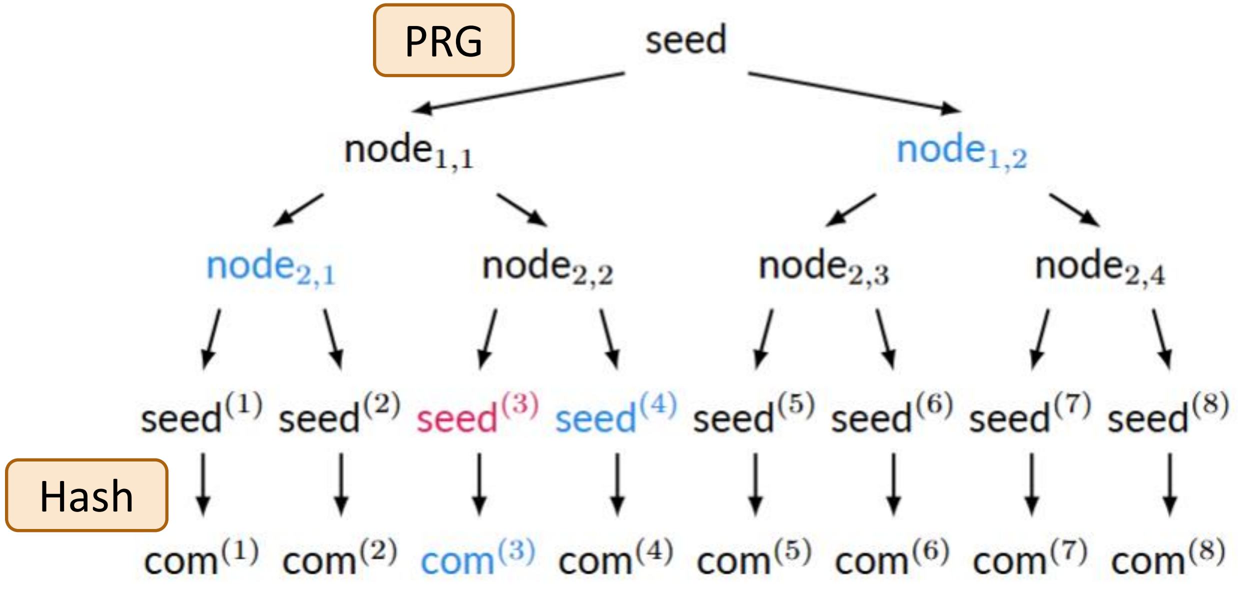
Vector Commitments (VC)



- VC is binding: $\left(\text{com}^{(i)}\right)_{i\in[N]}$ binds $\left(\text{seed}^{(i)}\right)_{i\in[N]}$
  - ➔ One cannot find collisions of Hash
  - ➔ requires $\left|\text{com}^{(i)}\right| \geq 2\lambda$
- VC is hiding: hidden seed cannot be discovered from pdecom
  - ➔ One cannot find preimage of Hash
  - ➔ requires $\left|\text{com}^{(i)}\right| \geq \lambda$

Relaxing the binding property of VC will reduce communication cost (=signature size)

Vector Semi-Commitments (VSC)

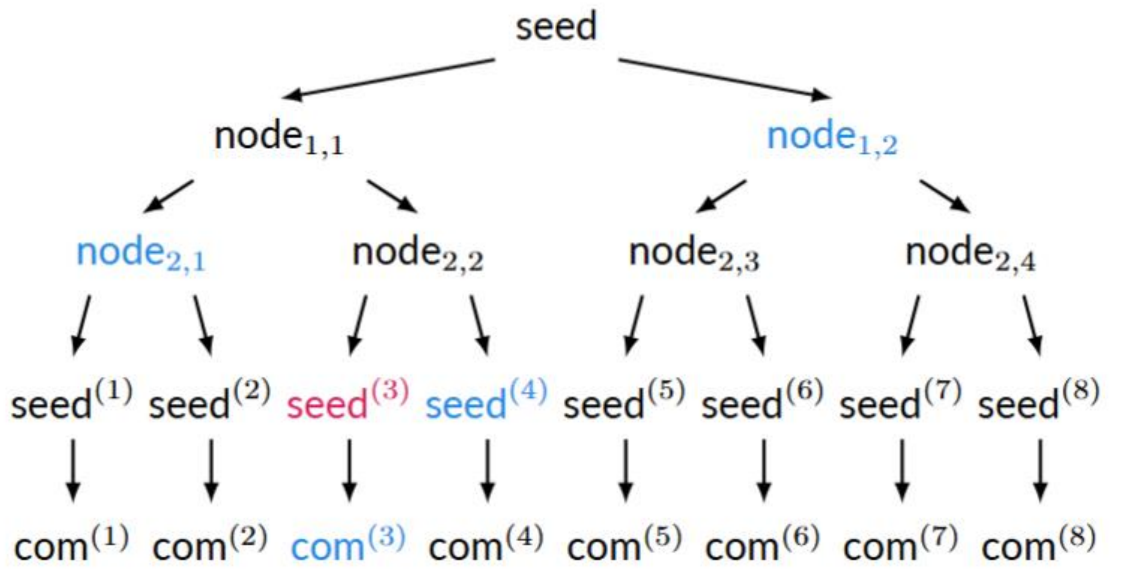


- VC is u-semi-binding
  - $\left(\mathrm{com}^{(i)}\right)_{i\in[N]}$ binds few (=u) of $\left(\mathrm{seed}^{(i)}\right)_{i\in[N]}$
  - One cannot find large multi-collisions of Hash
- Balls-into-Bins Game
  - If Q balls are randomly assigned into 2^{λ} bins

$$\Pr\left[\mathrm{max{-}load} \geq \frac{2\lambda}{\log\lambda}\right] \leq \mathrm{O}\left(\frac{Q}{2^{\lambda}}\right)$$

  - Set $\left|\mathrm{com}^{(i)}\right| = \lambda$ then $u = ??$

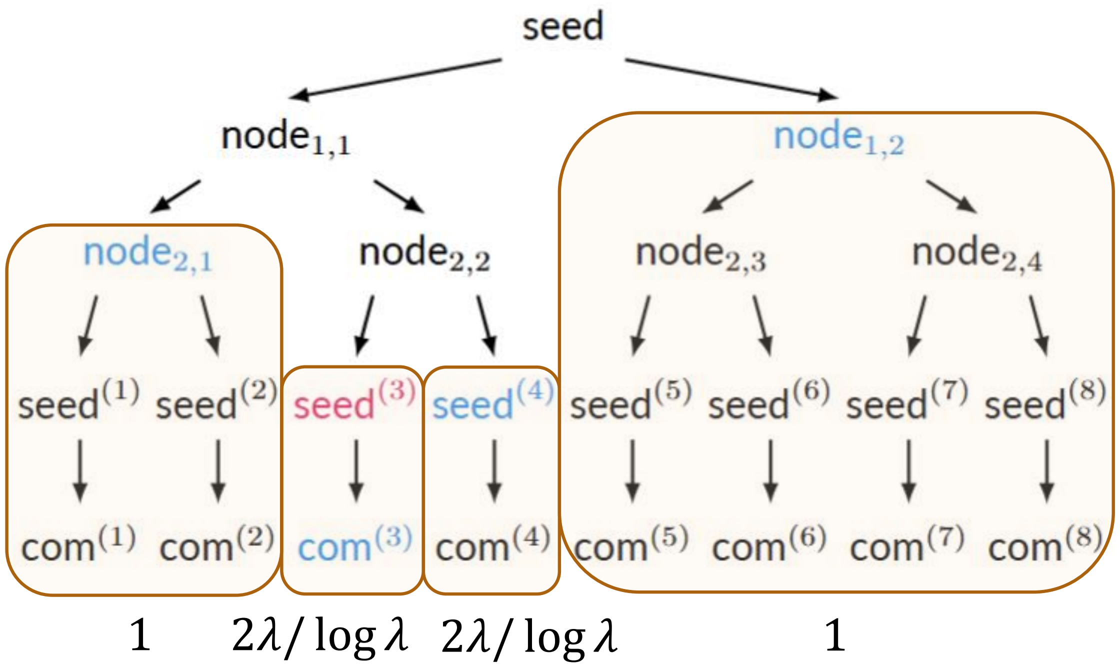
Vector Semi-Commitments (VSC)



$2\lambda/\log\lambda$ $\cdots$ $2\lambda/\log\lambda$

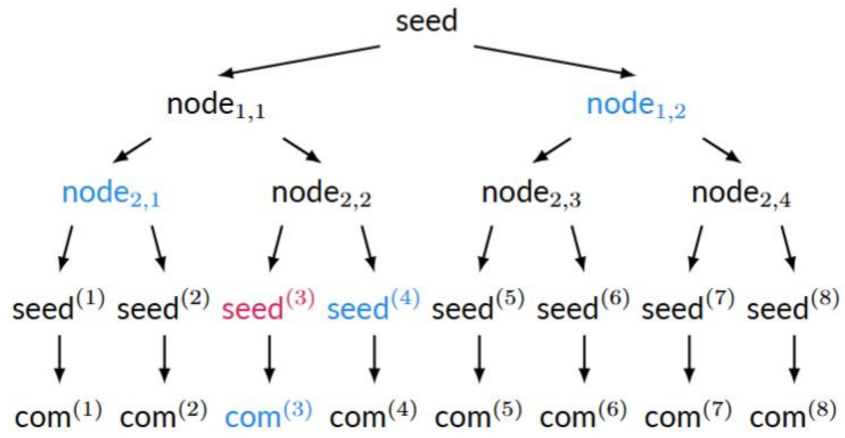
- Naive computation: $\mathrm{u} = \left(\frac{2\lambda}{\log\lambda}\right)^N$ which seems quite large
  - But malicious prover should find $\left(\mathrm{seed}^{(i)}\right)_{i\in[N]}$ with valid pdecom

Vector Semi-Commitments (VSC)



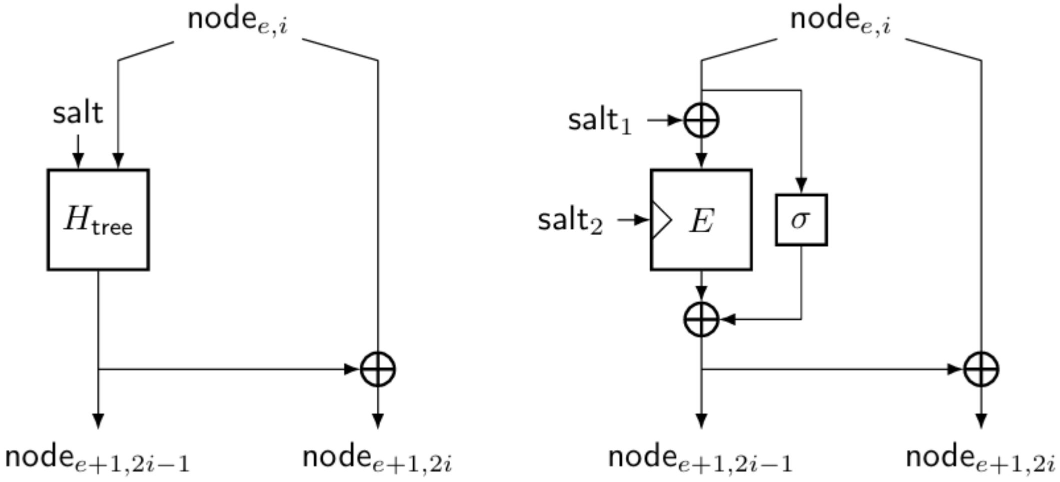
- # of $\left(\mathrm{seed}^{(i)}\right)_{i\in[N]}$ with valid pdecom: $u = \frac{N}{2}\cdot\left(\frac{2\lambda}{\log\lambda}\right)^2$ ➔ VSC is u-semi-binding

Vector Semi-Commitments (VSC)



- Halved commit size by relaxing binding property
  - Reduce $\tau \cdot \lambda$ bits of signature size

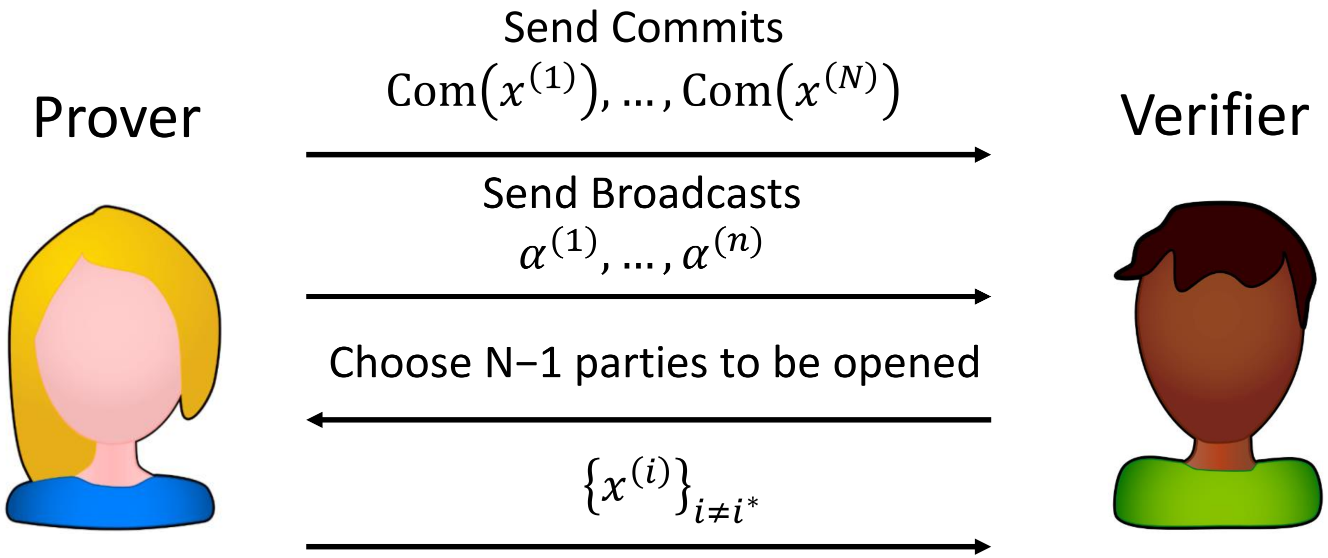
- Two instantiations: RO-VSC and IC-VSC
  - For IC-VSC, we use fixed key AES for tree expansion
    ➔ a lot faster VSC evaluation
  - We provide security proof in ROM/ICM



(a) RO-VSC (b) IC-VSC

Differences in Security Proofs

- The happy illusion in the beginning
  - VSC has u-semi-binding instead of binding(=1-semi-binding)
  - MPC check failure probability becomes u-times larger

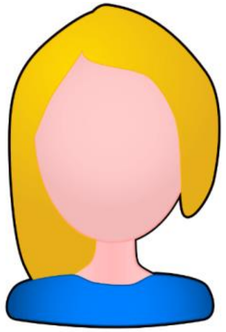


Differences in Security Proofs

- The happy illusion in the beginning
  - VSC has u-semi-binding instead of binding(=1-semi-binding)
  - MPC check failure probability becomes u-times larger

But the world was not so simple

Prover



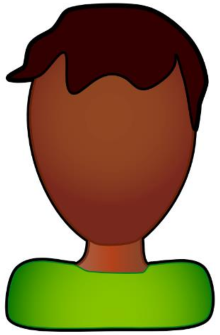
Send Commits
$\mathrm{Com}\left(x^{(1)}\right), \ldots, \mathrm{Com}\left(x^{(N)}\right)$
$\longrightarrow$

Send Broadcasts
$\alpha^{(1)}, \ldots, \alpha^{(n)}$
$\longrightarrow$

Choose N−1 parties to be opened
$\longleftarrow$

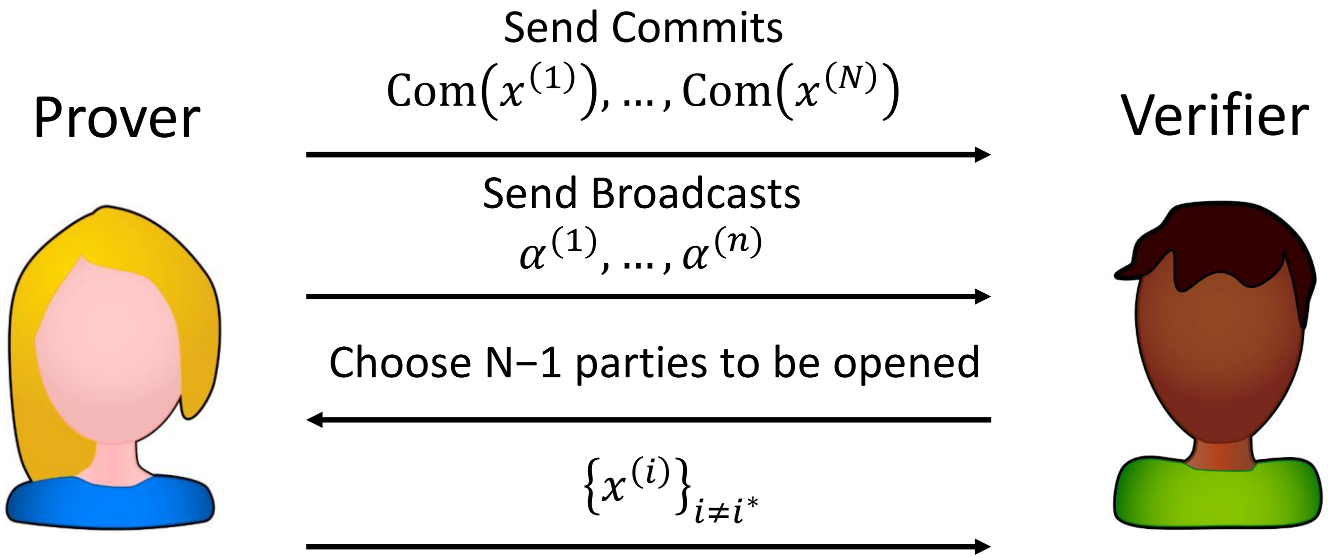
$\left\{x^{(i)}\right\}_{i \neq i^*}$
$\longrightarrow$

Verifier



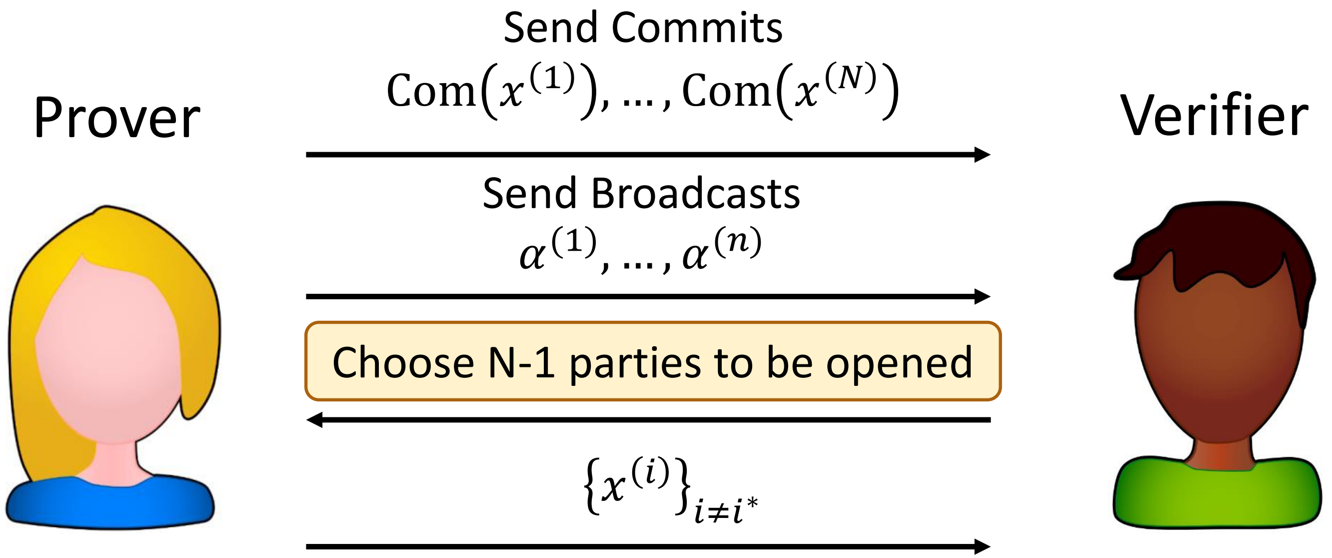
Differences in Security Proofs

- The reality is quite complicated
- MPC check failure probability becomes u-times larger and



Differences in Security Proofs

- The reality is quite complicated
- MPC check failure probability becomes u-times larger and
- Malicious prover can find new seeds those are consistent to previously generated commitments
  - Even after opening parties are known

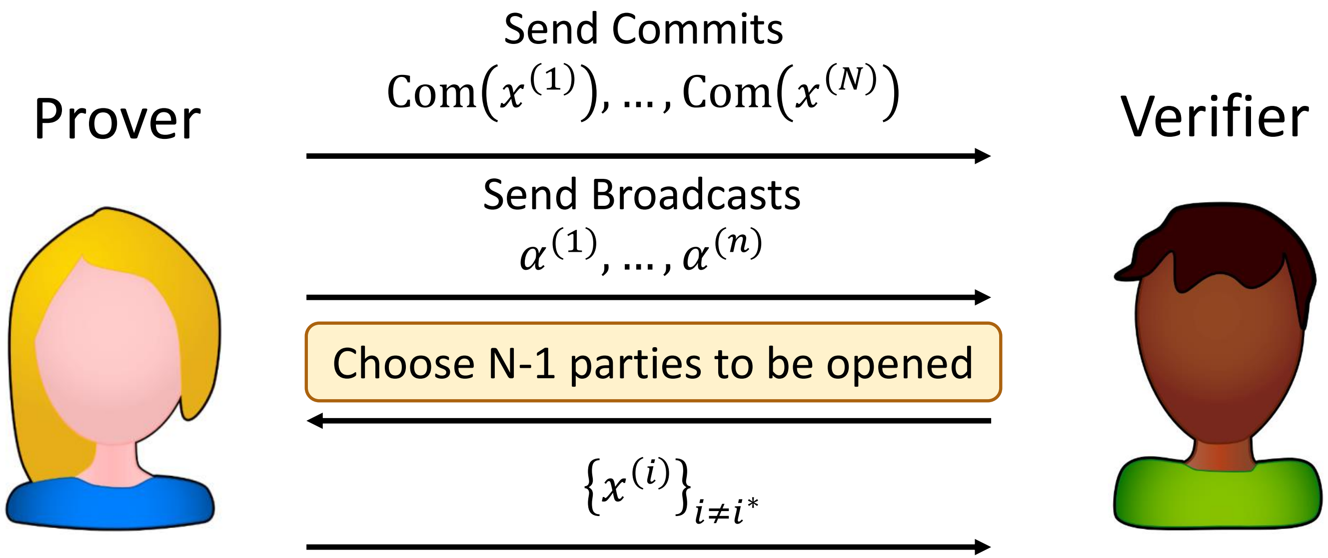


Differences in Security Proofs

So, we should prove followings

1. u-semi-binding property of VSC
2. Malicious prover cannot find a new seed which is
   - Consistent to previously generated commitments and
   - Pass the MPC check

➔ Analyzing more bad events, …



Result

Scheme	Field Size	N	τ	RO call	PRG or IC call	Sig. size (B)
BN++	2^{128}	16	33	532	$1056C + 1518$	$1056C + 3792$
	2^{128}	256	17	4356	$8704C + 13022$	$544C + 3088$
rBN++	2^{128}	16	33	5	$1056C + 1551$	$1056C + 2736$
	2^{128}	256	17	5	$8704C + 13039$	$544C + 2544$

- reduced BN++: BN++ with IC-VSC
  - Shorter commitment size ➔ Shorter signature size
  - Use fixed key AES ➔ Faster evaluation

Conclusion

- Vector semi-commitment (VSC)
  - relaxing binding property of vector commitment
  - VSC makes signatures shorter and faster

- Future Works
  - VOLE-in-the-Head with VSC? ➔ In progress
  - VSC based on standard (PRG) assumption ➔ Useful for Quantum proofs

Thank you

Q&A : byghak.lee@samsung.com