Relaxed Vector Commitment for Shorter Signatures

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

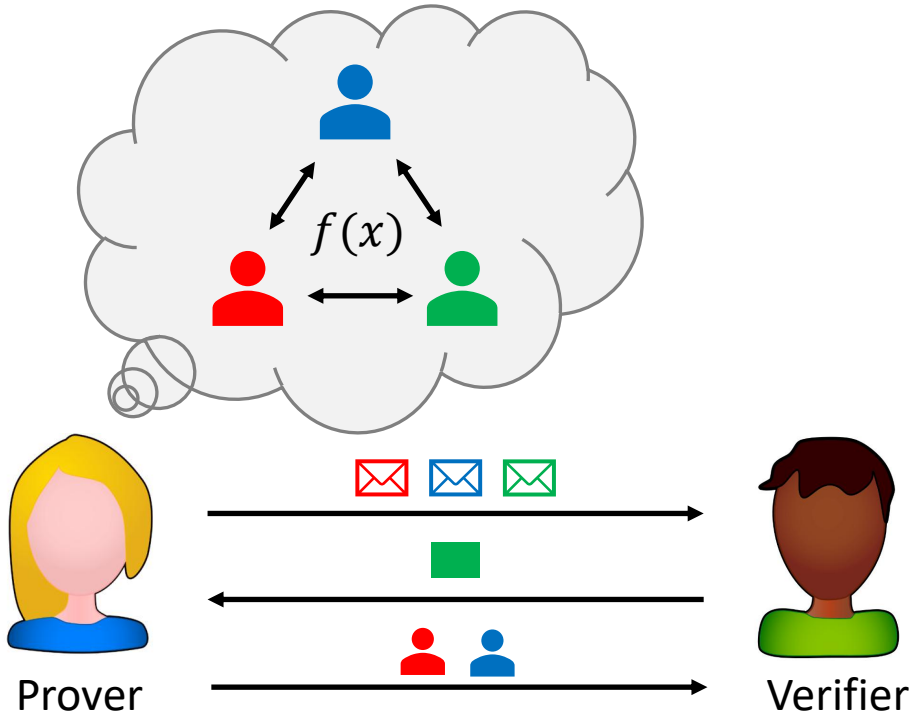
[1]Samsung SDS, Korea [2]KAIST, Korea

Brief Overview

- Relax the vector commitment scheme used in MPCitH-based signature
- Vector semi-commitment (VSC)
  - relaxing binding property of vector commitment
  - further optimized by utilizing correlated GGM tree
- Application of VSC ➔ rAIMer
  - By utilizing VSC, rAIMer has 18% shorter signatures and 112% faster signing speed

MPCitH-based Signatures

- Ishai et al. proposed a generic conversion from MPC to ZKPoK
  - MPCitH + OWF + FS = Digital Signature



MPCitH-based Signatures

- Ishai et al. proposed a generic conversion from MPC to ZKPoK
  - MPCitH + OWF + FS = Digital Signature
- 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, …
- Variants are still being researched and proposed
  - VOLE-in-the-Head, Threshold-Computation-in-the-Head, …

MPCitH-based Signatures

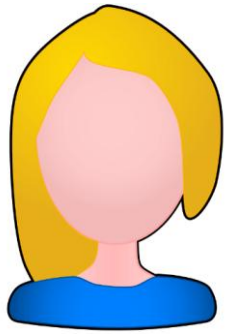
- Ishai et al. proposed a generic conversion from MPC to ZKPoK
  - MPCitH + OWF + FS = Digital Signature
- MPCitH enables post-quantum signature schemes
  - Minimal assumption: Security of digital signature only relies on the one-wayness of OWF
  - 6 of 15 in NIST additional PQC standardization are based on MPCitH: MIRA, MQOM, …
- Variants are still being researched and proposed
  - VOLE-in-the-Head, Threshold-Computation-in-the-Head, …

➔ Optimizing MPCitH is important

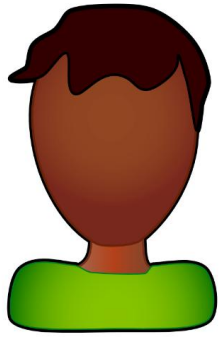
Deep Dive into Recent MPCitH (BN Protocol)

- BN: MPC-in-the-Head based NIZKPoK for arithmetic circuits

Prover



Verifier



commits to following for all N parties
1. Additive shares of beaver triples
2. Additive shares of all wires of the circuit

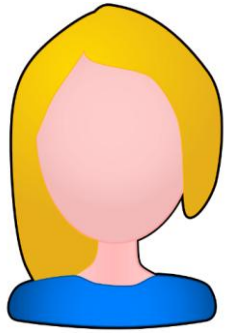
Randomness for the verification

1. Simulate multiplication check protocols
2. Commit to views of parties

Choose $N - 1$ parties to open

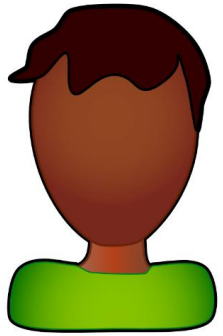
Open the views for chosen parties

Prover



commits to following for all N parties
1. Additive shares of beaver triples
2. Additive shares of all wires of the circuit

Verifier



Randomness for the verification

1. Simulate multiplication check protocols
2. Commit to views of parties

Choose $N-1$ parties to open

Open the views for chosen parties

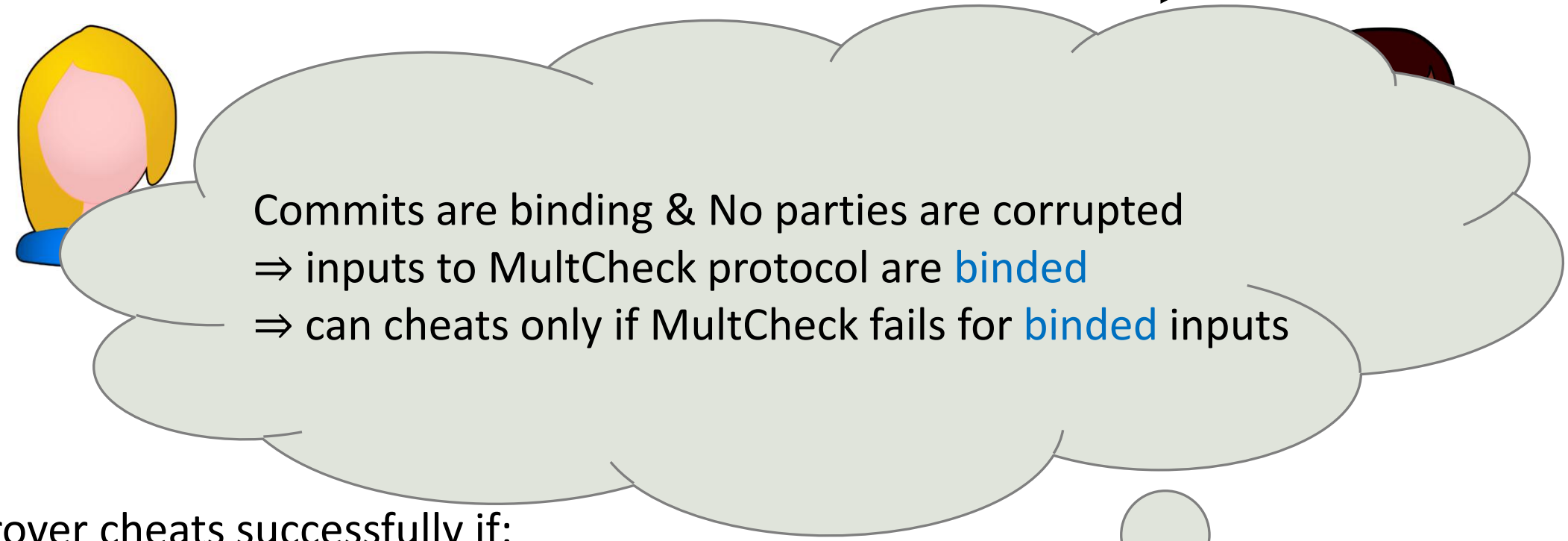
- Prover cheats successfully if:
  - Prover corrupted the unopened party ➔ $1/N$
  - multiplication check protocol fails ➔ soundness error = typically $\frac{1}{\|\mathbb{F}\|}$

} Repeat τ times where $\left(\frac{1}{N}+\frac{1}{\|\mathbb{F}\|}\right)^{\tau} \simeq 2^{-\lambda}$

Prover

commits to following for all N parties
1. Additive shares of beaver triples
2. Additive shares of all wires of the circuit

Verifier



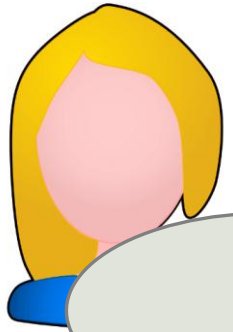
- Prover cheats successfully if:
  - Prover corrupted the unopened party ➔ $1/N$
  - multiplication check protocol fails ➔ soundness error = typically $\frac{1}{\|\mathbb{F}\|}$

Repeat τ times where $\left(\frac{1}{N}+\frac{1}{\|\mathbb{F}\|}\right)^{\tau} \simeq 2^{-\lambda}$

Prover

commits to following for all N parties
1. Additive shares of beaver triples
2. Additive shares of all wires of the circuit

Verifier



Commits are semi-binding & No parties are corrupted
⇒ some(=u) inputs to MultCheck protocol are binded
⇒ can cheats only if MultCheck fails for binded inputs

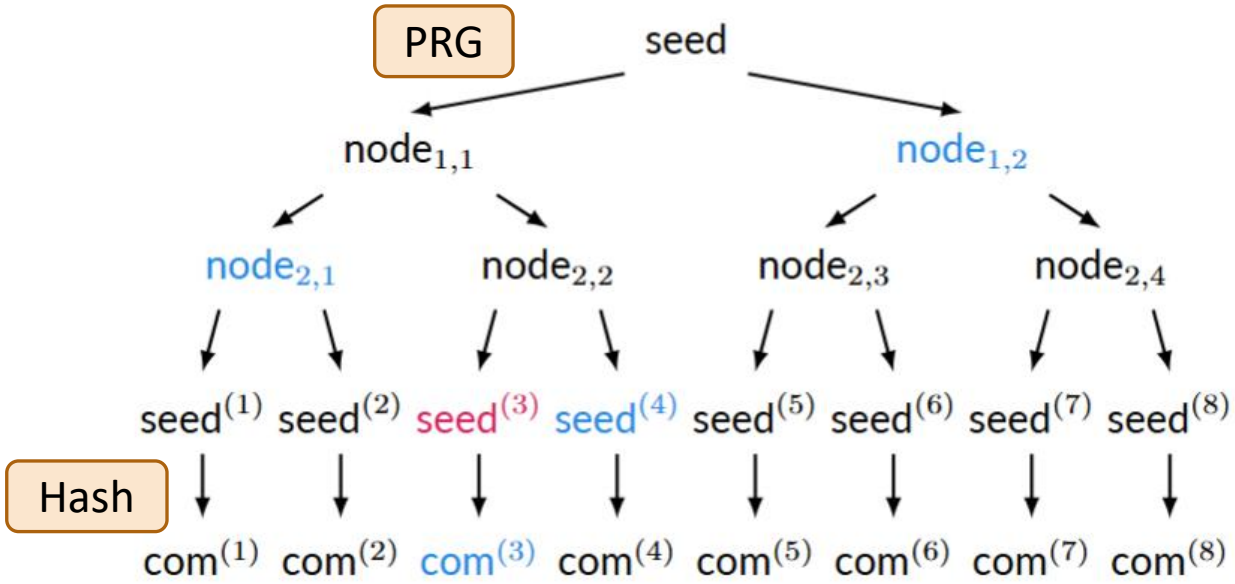
$$\frac{1}{\|\mathbb{F}\|} \rightarrow \frac{u}{\|\mathbb{F}\|} \ ?$$

- Prover cheats successfully if:
  - Prover corrupted the unopened party ➔ $1/N$
  - multiplication check protocol fails ➔ soundness error = typically $\frac{1}{\|\mathbb{F}\|}$

} Repeat τ times where $\left(\frac{1}{N}+\frac{1}{\|\mathbb{F}\|}\right)^{\tau} \simeq 2^{-\lambda}$

Our Contribution

Vector Commitments (VC)

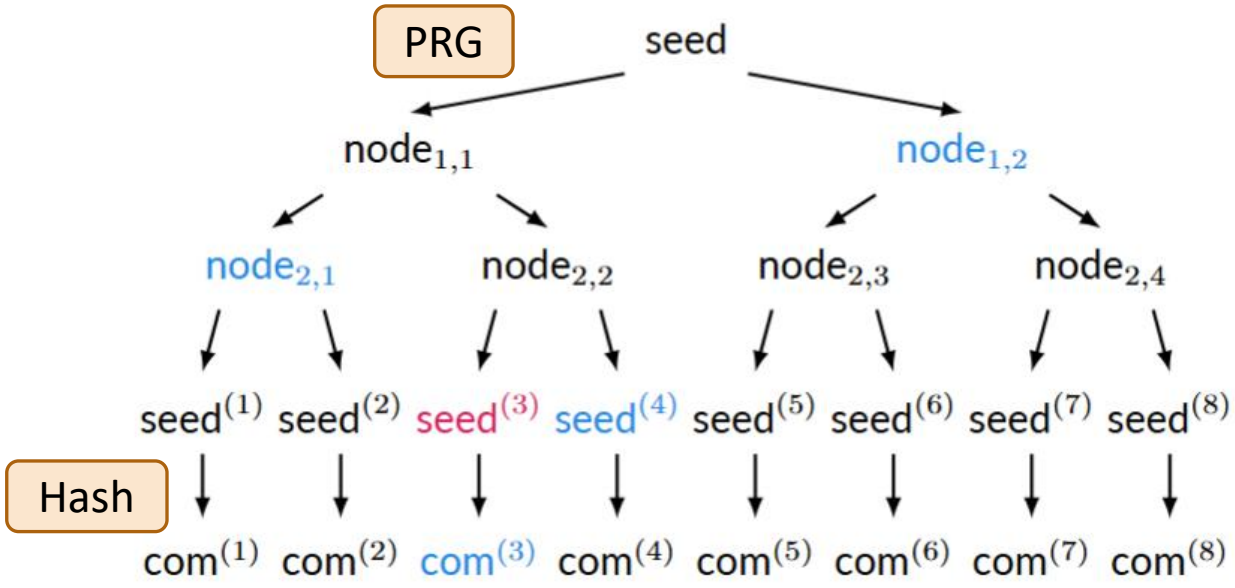


- $\mathrm{VC.Commit}(\mathrm{seed}) = (\mathrm{decom}, \mathrm{com})$
  - $\mathrm{com} := (\mathrm{com}^{(1)}, \ldots, \mathrm{com}^{(8)})$
- $\mathrm{VC.Open}(\mathrm{decom}, \bar{3}) = \mathrm{pdecom}$
  - $\mathrm{pdecom} := (\mathrm{node}_{1,2}, \mathrm{node}_{2,1}, \mathrm{seed}^{(4)}, \mathrm{com}^{(3)})$
- $\mathrm{VC.Verify}(\mathrm{com}, \mathrm{pdecom}, \bar{3}) = (\mathrm{seed}^{(i)})_{i \neq 3}$ or $\perp$

$$\mathrm{PRG}(\mathrm{seed}^{(i)}) = \left(\underbrace{w_1^{(i)}, \ldots, w_C^{(i)}}, \underbrace{a_1^{(i)}, \ldots, a_C^{(i)}, b_1^{(i)}, \ldots, b_C^{(i)}, c^{(i)}}\right)$$

Additive shares of wires of circuit — Additive shares of beaver triples

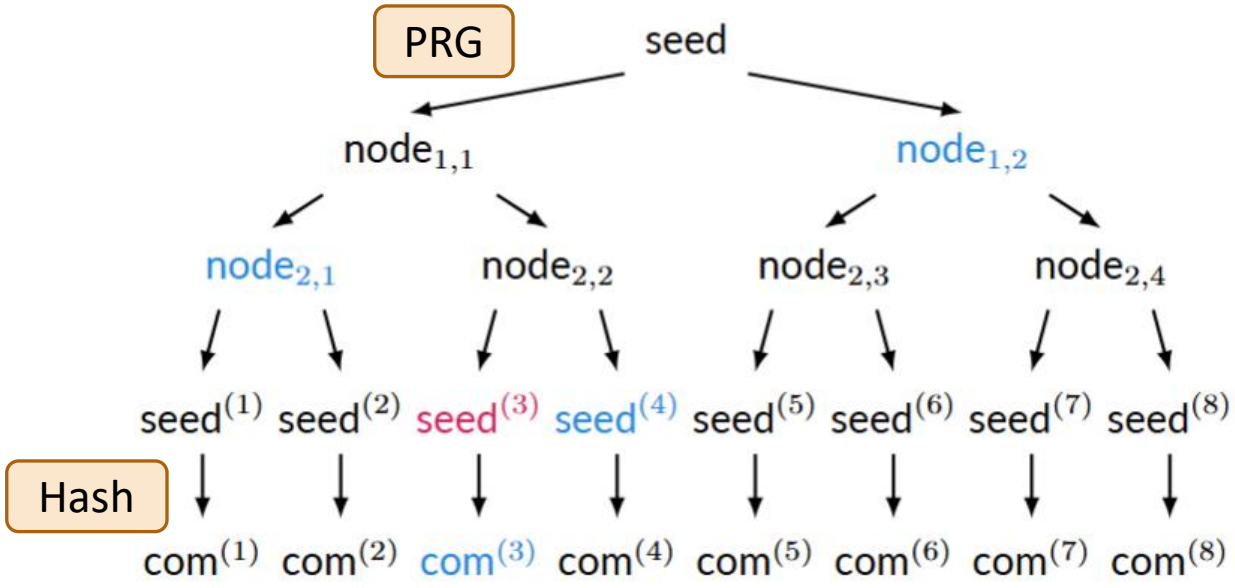
Vector Commitments (VC)



- 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 $\left|\mathrm{com}^{(i)}\right| \geq \lambda$

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

Vector Semi-Commitments (VSC)

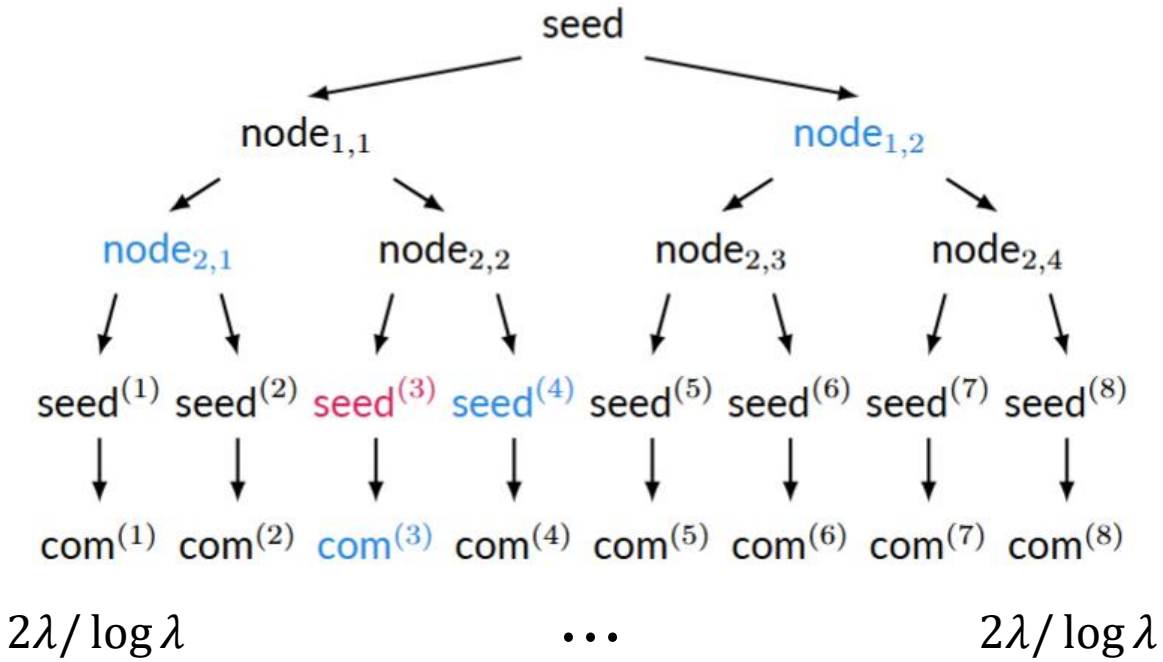


- VC is u-semi-binding
  - $\left(\mathrm{com}^{(i)}\right)_{i\in[N]}$ binds few (=u) of $\left(\mathrm{seed}^{(i)}\right)_{i\in[N]}$
  - One cannot find large multi-collisions of Hash
- Balls-into-Bins Game
  - If Q balls are randomly assigned into 2^{λ} bins

$$\Pr\left[\max\text{–load} \geq \frac{2\lambda}{\log\lambda}\right] \leq 0\left(\frac{Q}{2^{\lambda}}\right)$$

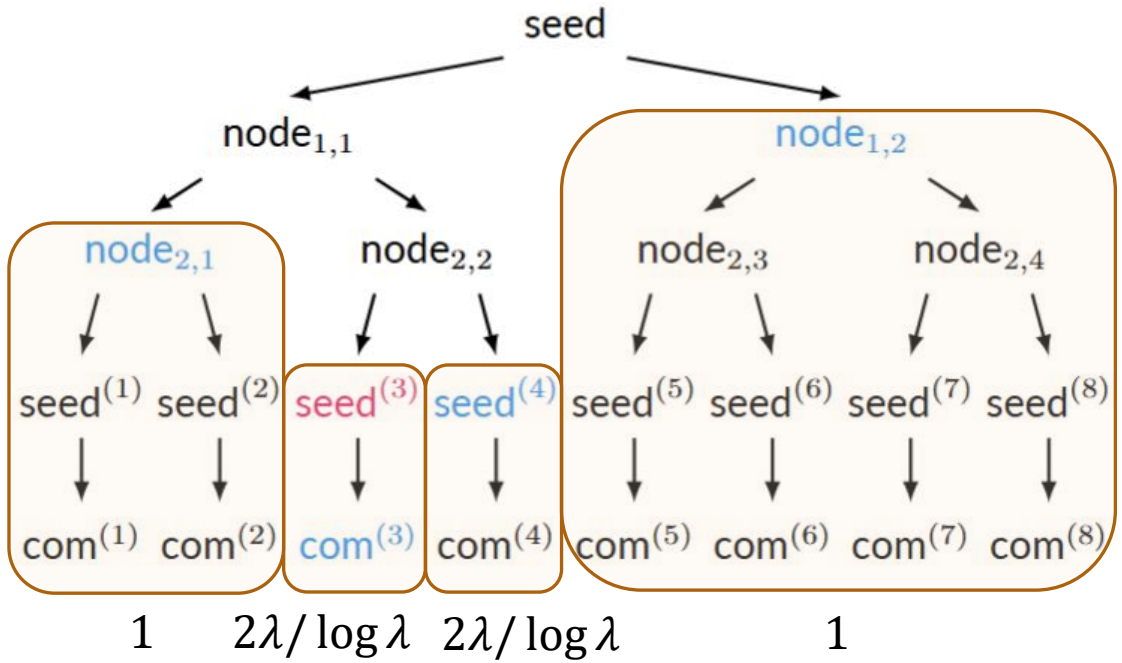
  - Set $\left|\mathrm{com}^{(i)}\right| = \lambda$ then $u = ??$

Vector Semi-Commitments (VSC)



- 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

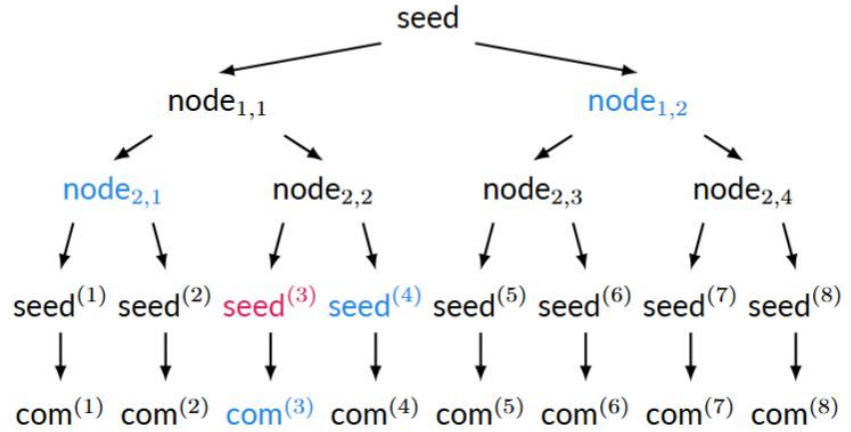
Vector Semi-Commitments (VSC)

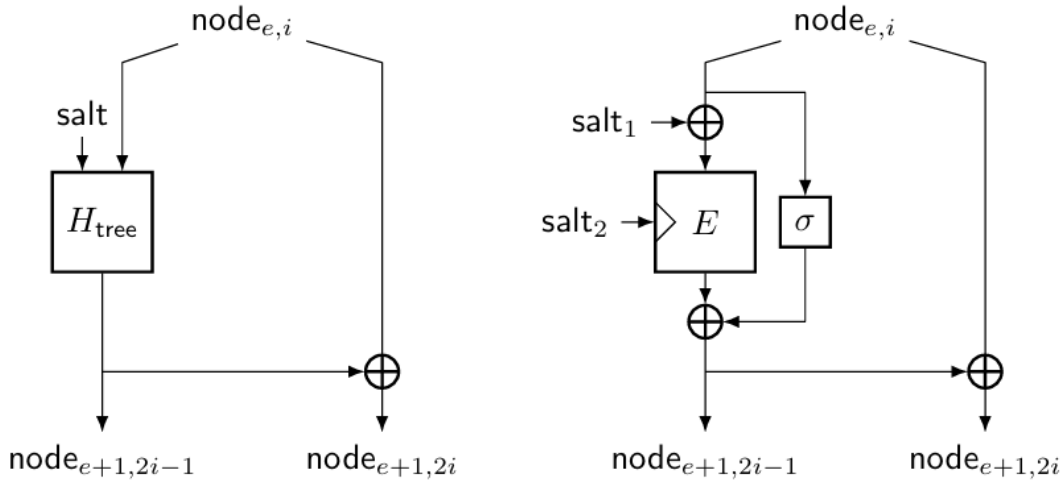


- # of $\left(\text{seed}^{(i)}\right)_{i\in[N]}$ with valid pdecom: $u = \frac{N}{2}\cdot\left(\frac{2\lambda}{\log\lambda}\right)^2$ ➔ VSC is u-semi-binding

Vector Semi-Commitments (VSC)

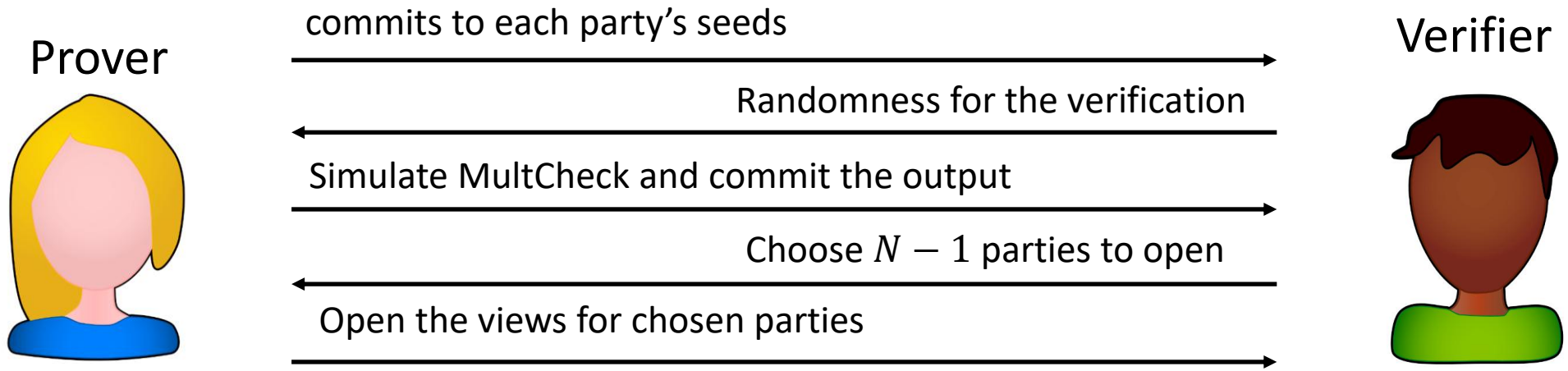
- Halved commit size by relaxing binding property
  - Reduce $\tau \cdot \lambda$ bits of signature size
- Applied Correlated GGM (cGGM) optimization
  - Use first λ-bits of witness as root seed
  - Further reduce $\tau \cdot \lambda$ bits of signature size
- Two instantiations: RO-VSC and IC-VSC
  - For IC-VSC, we use fixed key AES for tree expansion
    ➔ a lot faster VSC evaluation
  - We provide security proof in ROM/ICM





Differences in Security Proofs

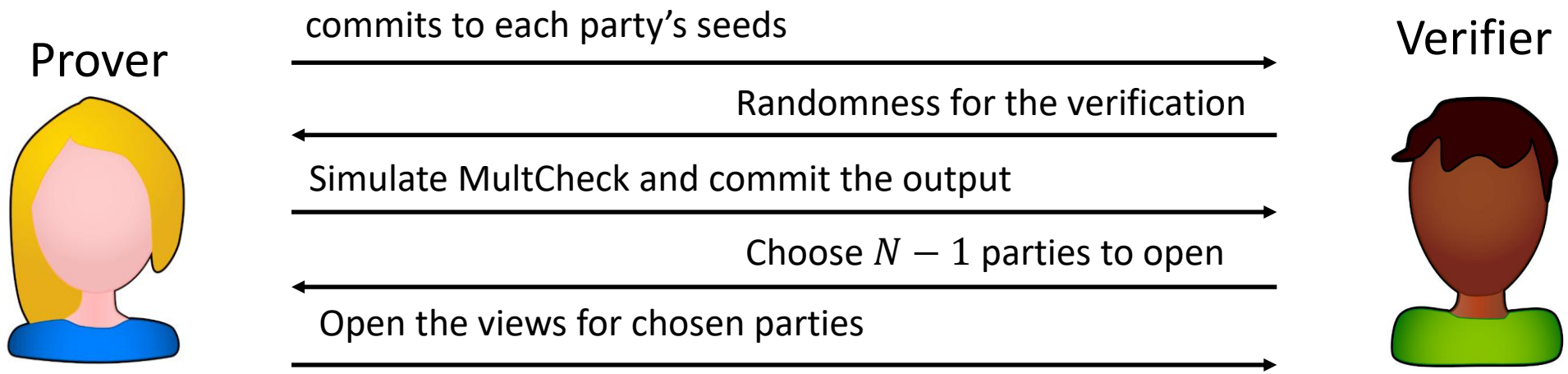
- The happy illusion in the beginning
  - VSC has u-semi-binding instead of binding(=1-semi-binding)
  - Soundness error of multiplication check becomes u-times larger
  - EUF-CMA to EUF-KO reduction would be same



Differences in Security Proofs

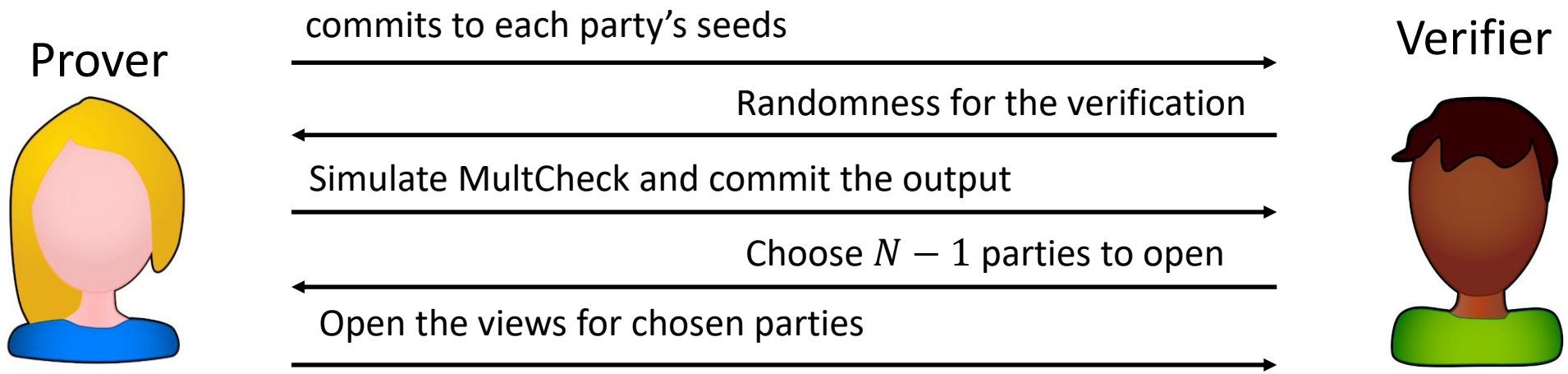
- The happy illusion in the beginning
  - VSC has u-semi-binding instead of binding(=1-semi-binding)
  - Soundness error of multiplication check becomes u-times larger
  - EUF-CMA to EUF-KO reduction would be same

But the world was not so simple



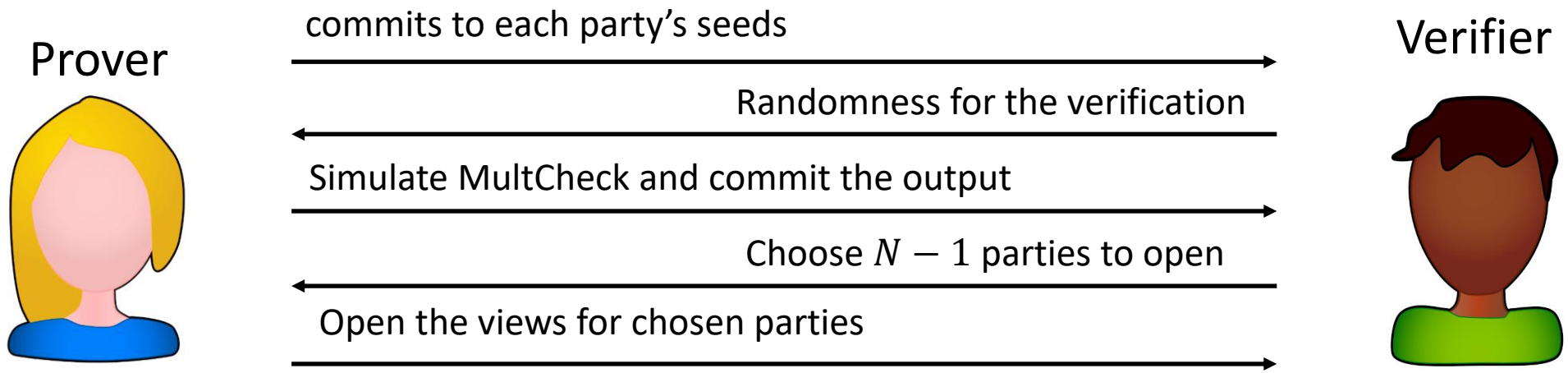
Differences in Security Proofs (EUF-KO)

- The reality is quite complicated
- Soundness error of multiplication check becomes u-times larger and



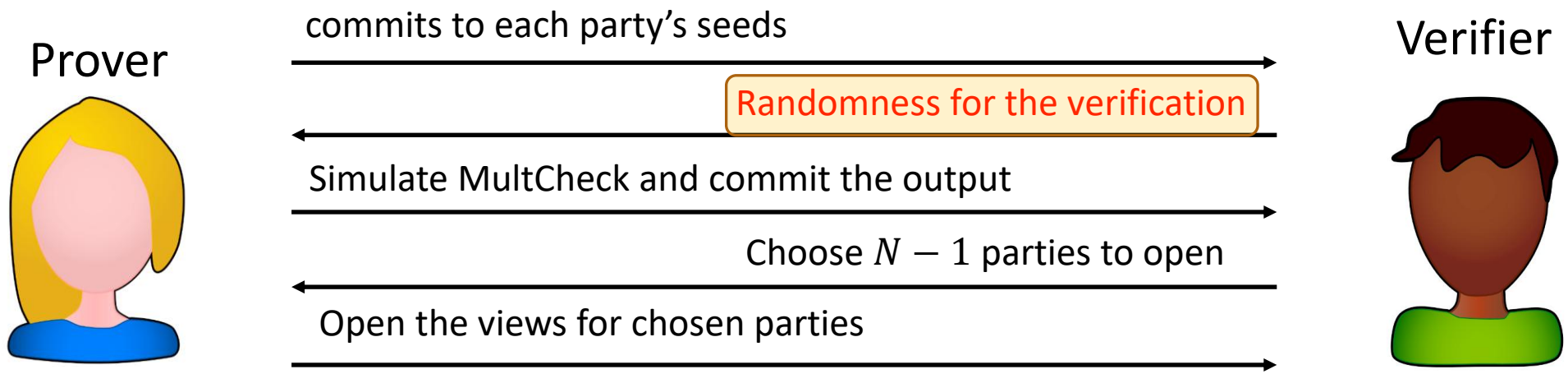
Differences in Security Proofs (EUF-KO)

- The reality is quite complicated
- Soundness error of multiplication check becomes u-times larger and
- Malicious prover can find new seeds those are consistent to previously generated commitments



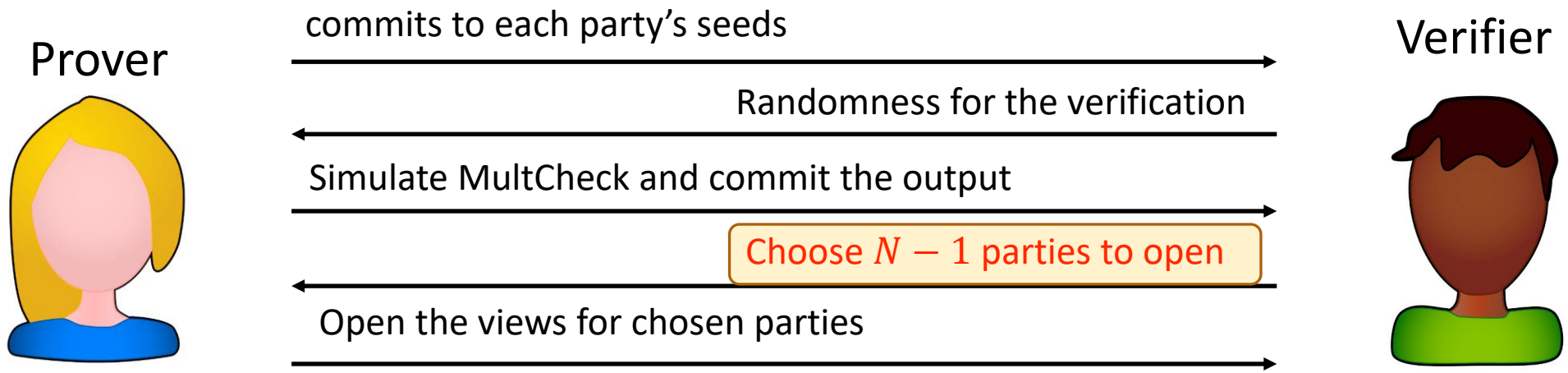
Differences in Security Proofs (EUF-KO)

- The reality is quite complicated
- Soundness error of multiplication check becomes u-times larger and
- Malicious prover can find new seeds those are consistent to previously generated commitments
  - Even after randomness for the verification is known



Differences in Security Proofs (EUF-KO)

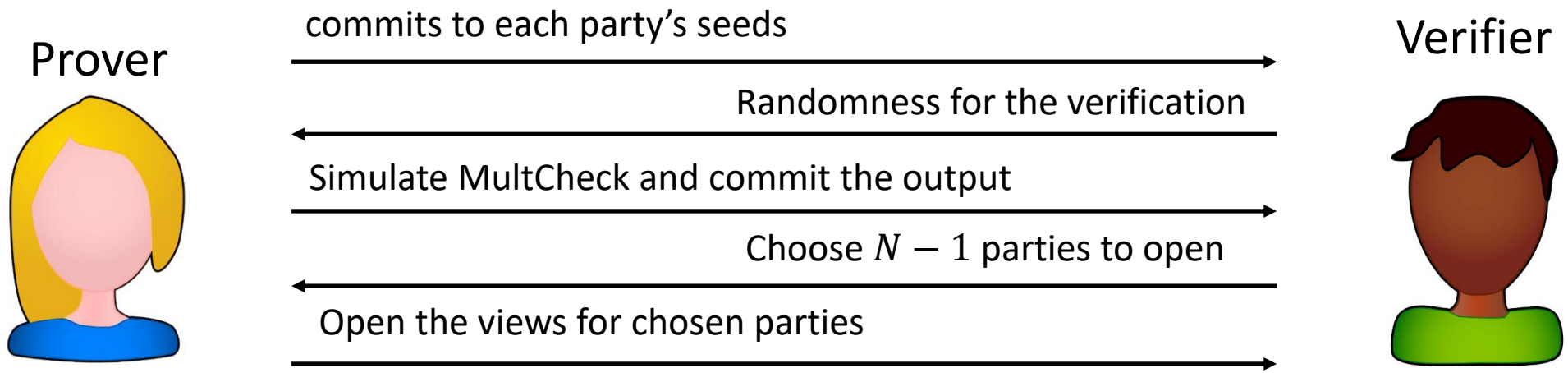
- The reality is quite complicated
- Soundness error of multiplication check becomes u-times larger and
- Malicious prover can find new seeds those are consistent to previously generated commitments
  - Even after randomness for the verification is known
  - Even after opening parties are known



Differences in Security Proofs (EUF-KO)

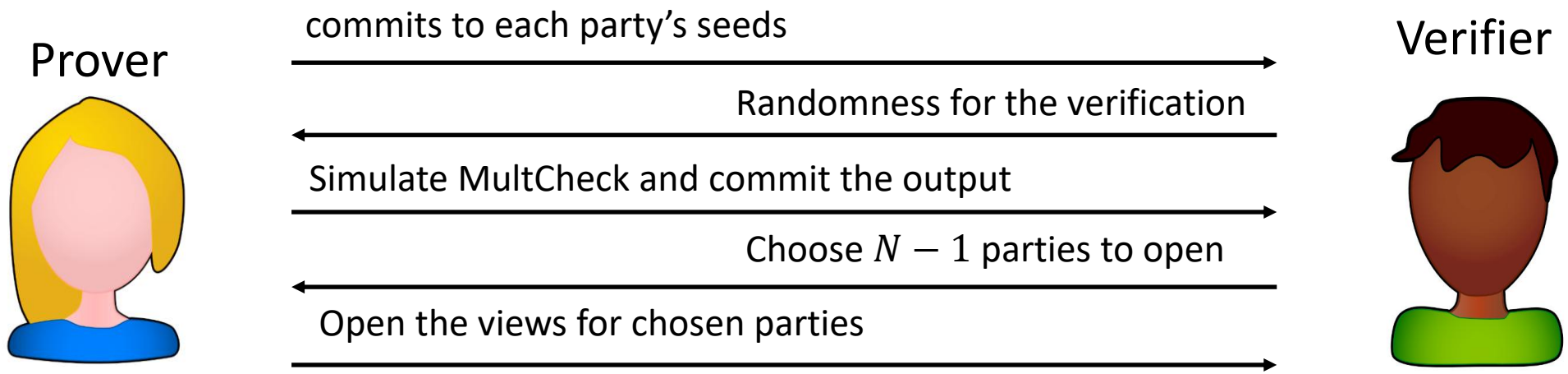
So, we should prove followings (for EUF-KO)

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 multiplication check protocol



Differences in Security Proofs (EUF-CMA)

- In VC, the output distribution of VC.Commit and VC.Open are independent to the secret key
- As VSC utilizes cGGM and inserts secret key into the root, we should prove that
  - Outputs of VSC.Commit and VSC.Open are indistinguishable to random
- Since we use IC, we should consider all input collisions between
  - Tree expansion, Seed hashing, PRG evaluation



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 + Key injection with cGGM ➔ Shorter signature size
  - Use cGGM with fixed key AES ➔ Less PRG/IC calls with faster evaluation

Result

Scheme	$\|pk\|$ (B)	$\|sig\|$ (B)	Sign (Kc)	Verify (Kc)
Dilithium2	1,312	2,420	162	57
SPHINCS$^+$-128f*	32	17,088	38,216	2,158
SPHINCS$^+$-128s*	32	7,856	748,053	799
SDitH-Hypercube-gf256	132	8,496	20,820	10,935
FAEST-v1-128f	32	6,336	2,387	2,344
FAEST-v1-128s	32	5,006	20,926	20,936
AIMer-v2.0-128f	32	5,888	788	752
AIMer-v2.0-128s	32	4,160	5,926	5,812
rAIMer-128f	32	4,848	421	395
rAIMer-128s	32	3,632	2,826	2,730

- By utilizing VSC, rAIMer has 18% shorter signatures and 112% faster signing speed

Conclusion

- Vector semi-commitment (VSC)
  - relaxing binding property of vector commitment
  - further optimized by utilizing correlated GGM tree
  - VSC makes signatures shorter and faster

- Future Works
  - VOLE-in-the-Head with VSC? ➔ Yes we can! (will be available soon)
  - VSC based on PRG assumption ➔ Useful for Quantum proofs

Thank you

Q&A : byghak.lee@samsung.com