Signature Schemes based on the MPC-in-the-Head Paradigm

Seongkwang Kim
Samsung SDS

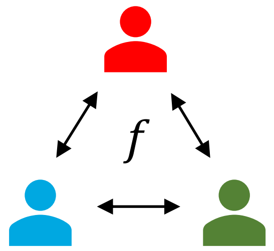
SAMSUNG SDS

Ewha-KMS IWC 2023

MPC-in-the-Head Paradigm

Secure Multiparty Computation

- Multiparty computation (MPC) enables a computation while preserving privacy
  - Yao's garbled circuit
  - **Additive secret sharing** (GMW, Beaver triple)
  - Shamir secret sharing

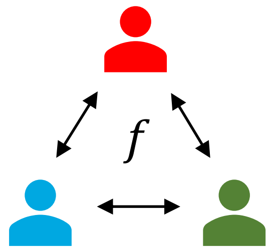


Secure Multiparty Computation

- Multiparty computation (MPC) enables a computation while preserving privacy
  - Yao's garbled circuit
  - **Additive secret sharing** (GMW, Beaver triple)
  - Shamir secret sharing
- Additive secret sharing
  - Secret is shared additively: $x = \sum_i x^{(i)}$
  - Addition is naturally compatible with shares

$$x + y = \sum_i x^{(i)} + \sum_i y^{(i)} = \sum_i (x^{(i)} + y^{(i)})$$

  - Multiplication needs a Beaver triple $\left\{\left(a^{(i)}, b^{(i)}, c^{(i)}\right)\right\}_i$ s.t. $c = ab$
    1. Compute $A^{(i)} = x^{(i)} + a^{(i)}, B^{(i)} = y^{(i)} + b^{(i)}$ and Open them
    2. Locally compute $z^{(i)} = Ay^{(i)} - Ba^{(i)} + c^{(i)} = (x + a)y^{(i)} - (y + b)a^{(i)} + c^{(i)} = xy^{(i)}$

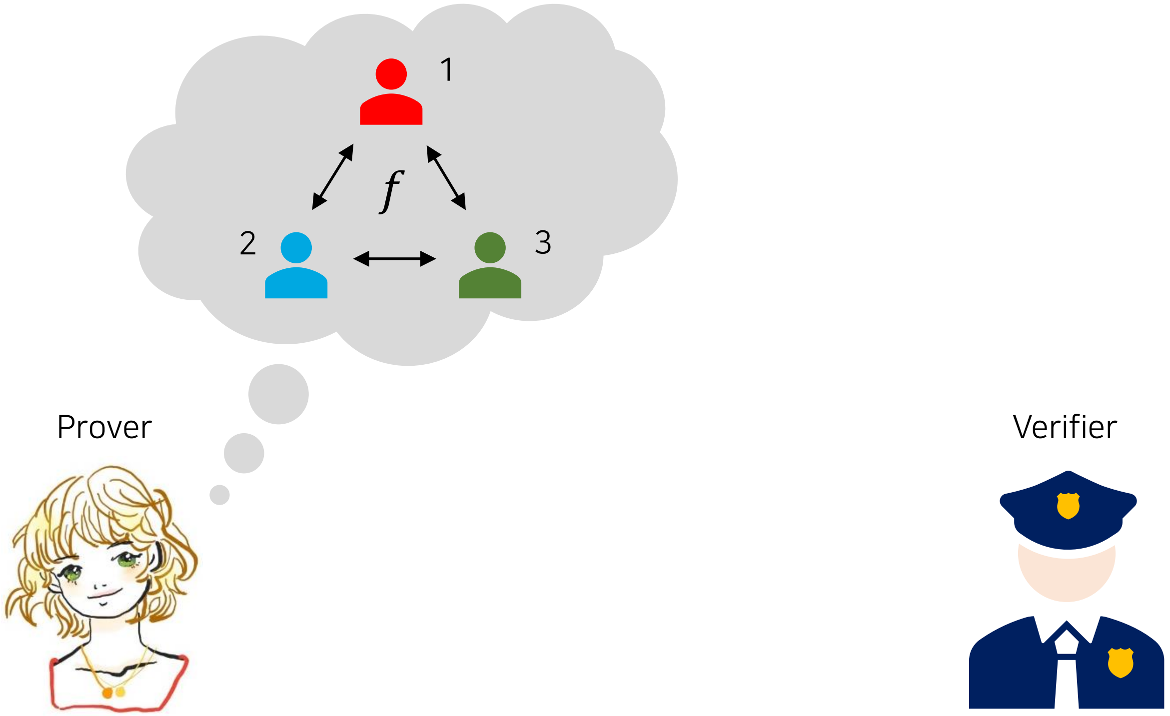


MPC-in-the-Head Paradigm

- Ishai et al. proposed a generic conversion from MPC to ZKP
- Prover simulates a multiparty computation in her head

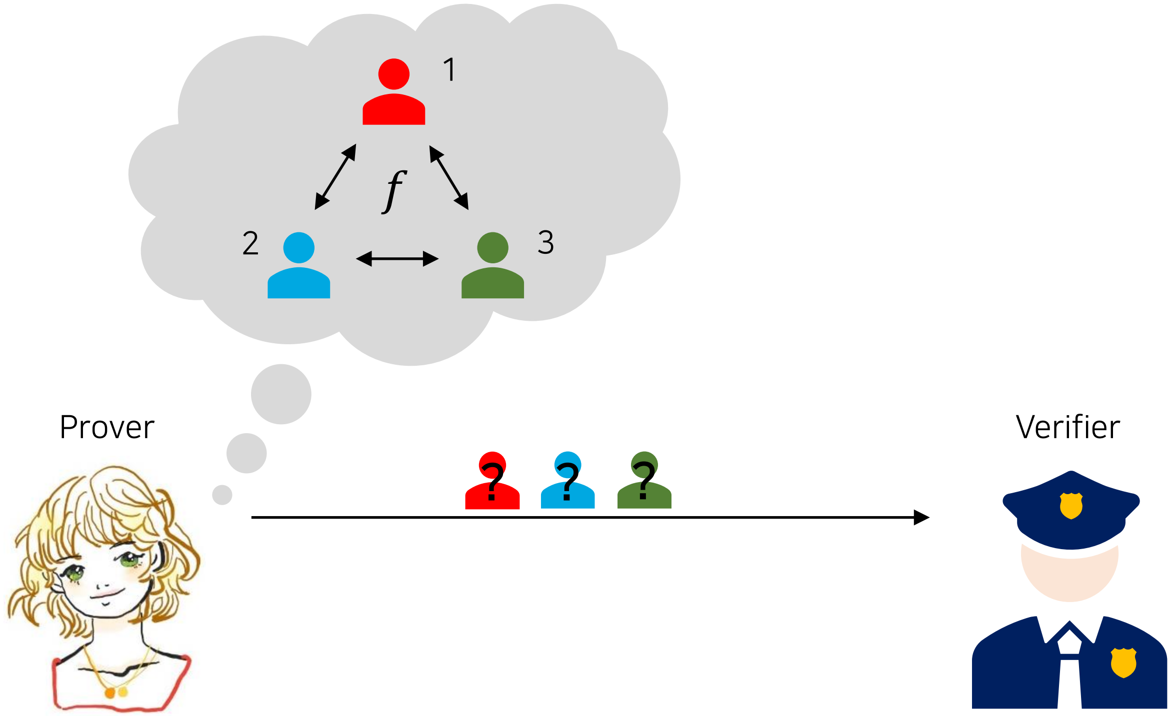
MPC-in-the-Head Paradigm

- Ishai et al. proposed a generic conversion from MPC to ZKP
- Prover simulates a multiparty computation in her head
    1. Prover simulates a multiparty computation of a function f



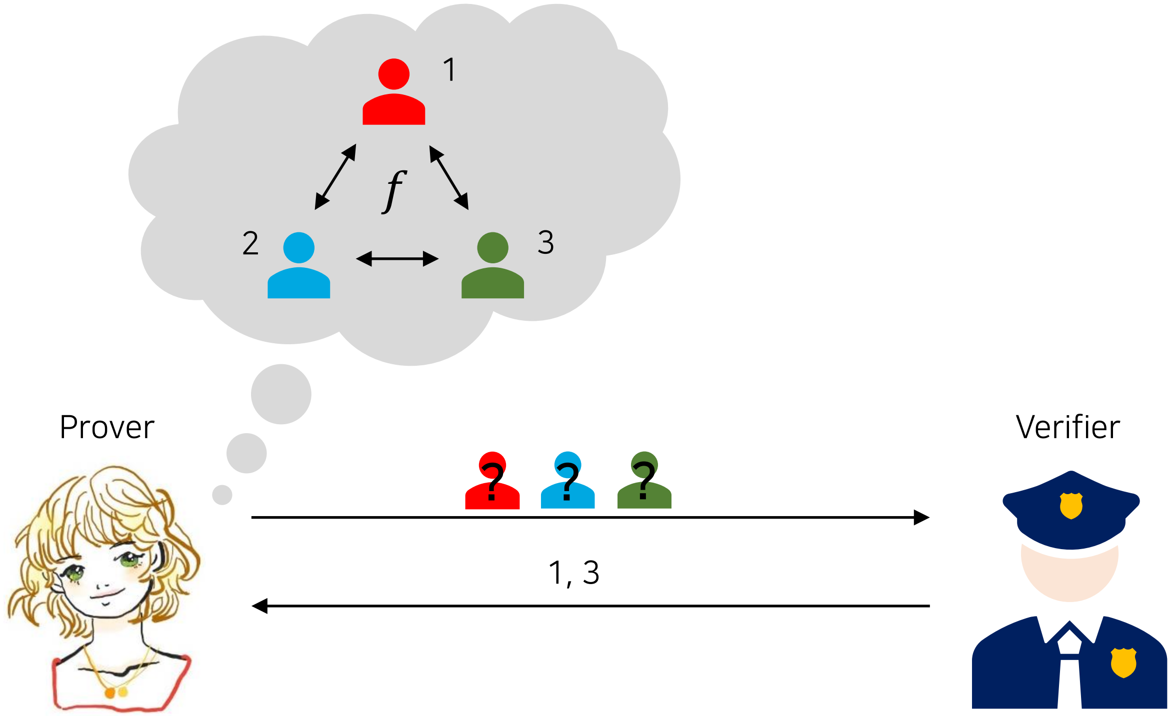
MPC-in-the-Head Paradigm

- Ishai et al. proposed a generic conversion from MPC to ZKP
- Prover simulates a multiparty computation in her head
    1. Prover simulates a multiparty computation of a function f
    2. Prover commits to all the views of the parties



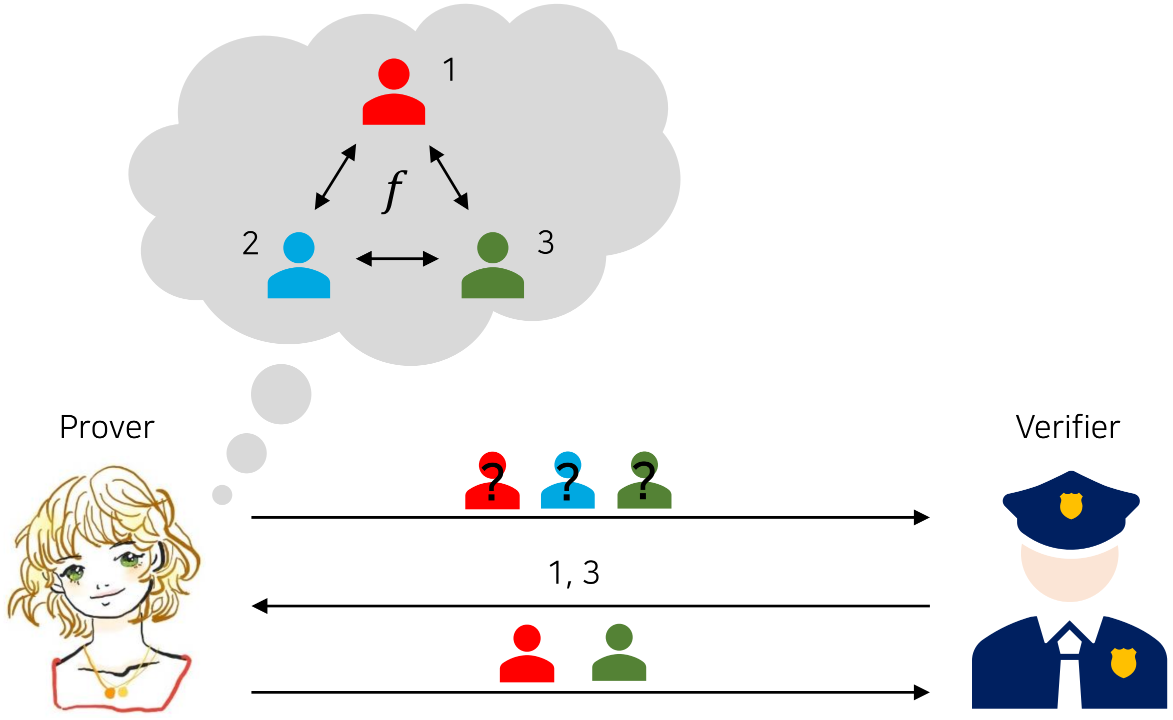
MPC-in-the-Head Paradigm

- Ishai et al. proposed a generic conversion from MPC to ZKP
- Prover simulates a multiparty computation in her head
    1. Prover simulates a multiparty computation of a function f
    2. Prover commits to all the views of the parties
    3. Verifier sends a random challenge



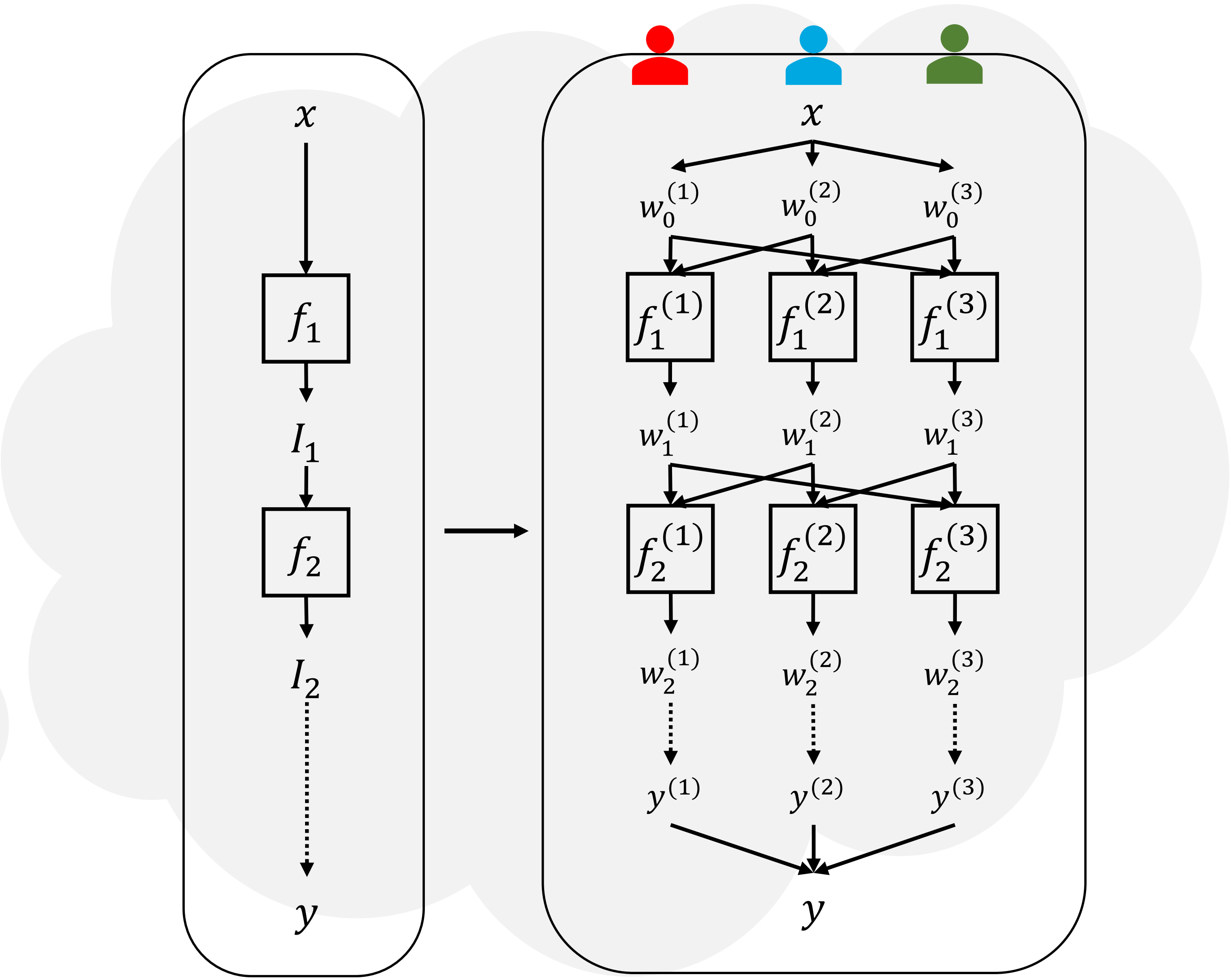
MPC-in-the-Head Paradigm

- Ishai et al. proposed a generic conversion from MPC to ZKP
- Prover simulates a multiparty computation in her head
    1. Prover simulates a multiparty computation of a function f
    2. Prover commits to all the views of the parties
    3. Verifier sends a random challenge
    4. Prover opens the challenged view
    5. Verifier checks consistency



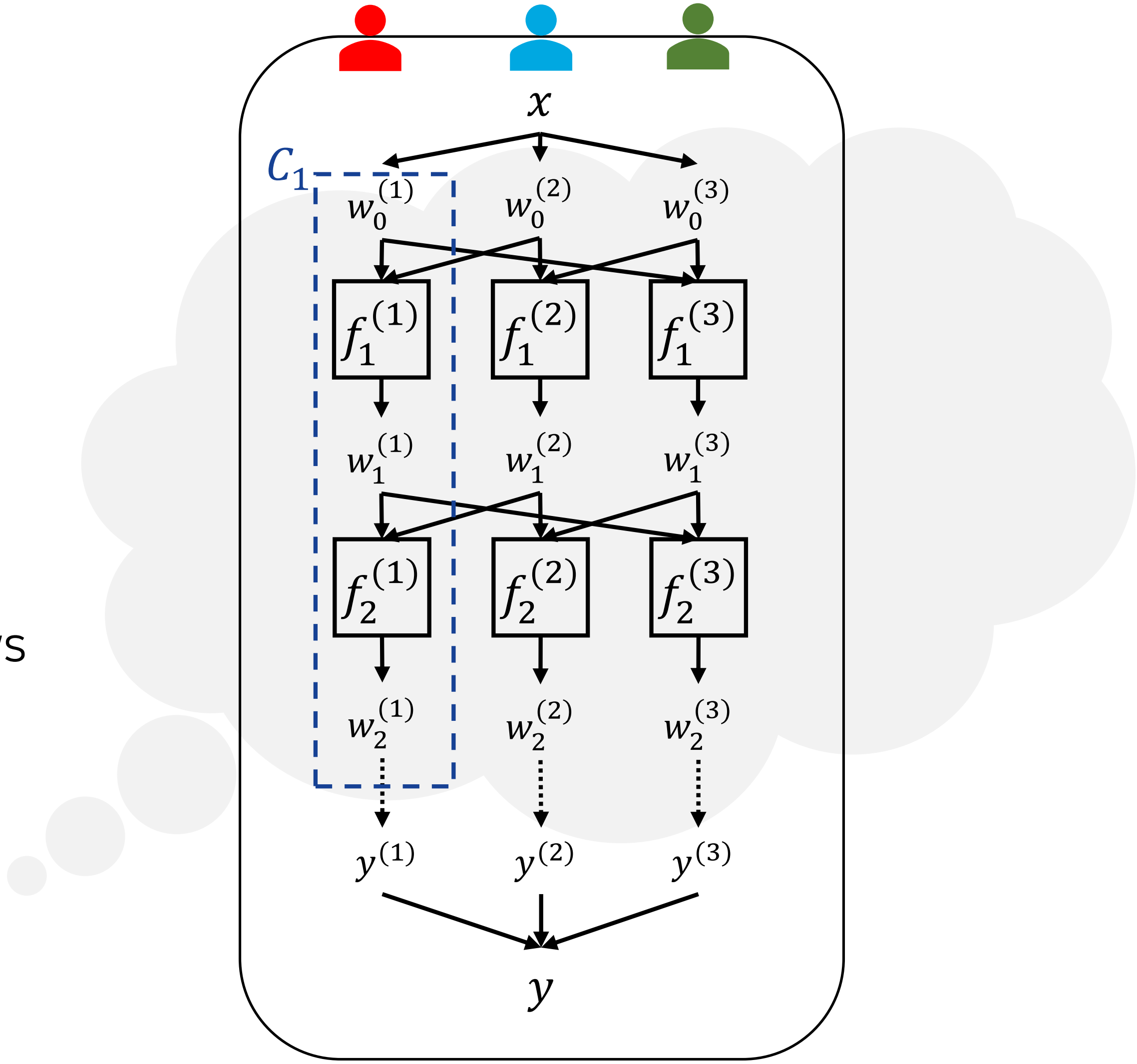
MPC-in-the-Head Paradigm (Simplified)

Want to prove a knowledge of x such that $f(x) = y$



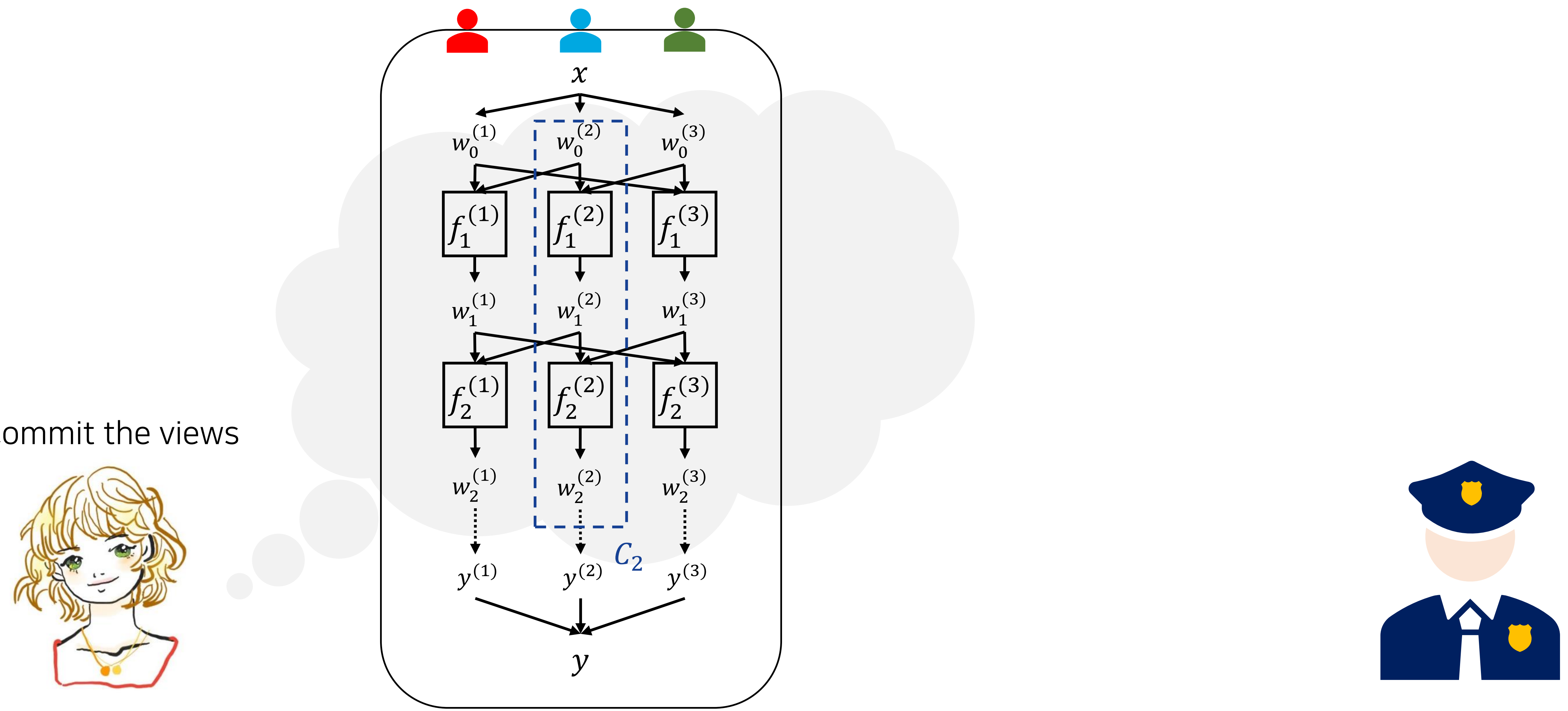
MPC-in-the-Head Paradigm (Simplified)

Commit the views



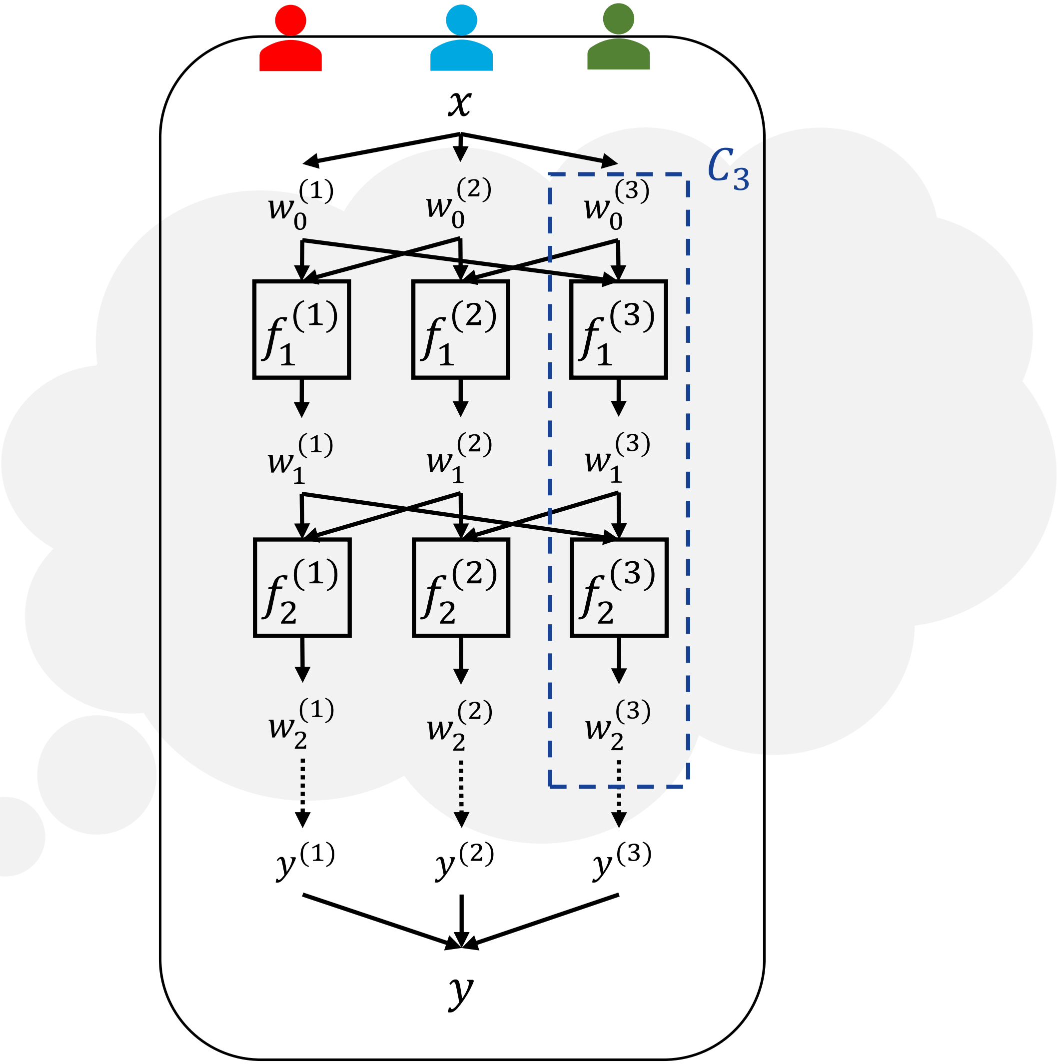
MPC-in-the-Head Paradigm (Simplified)

Commit the views

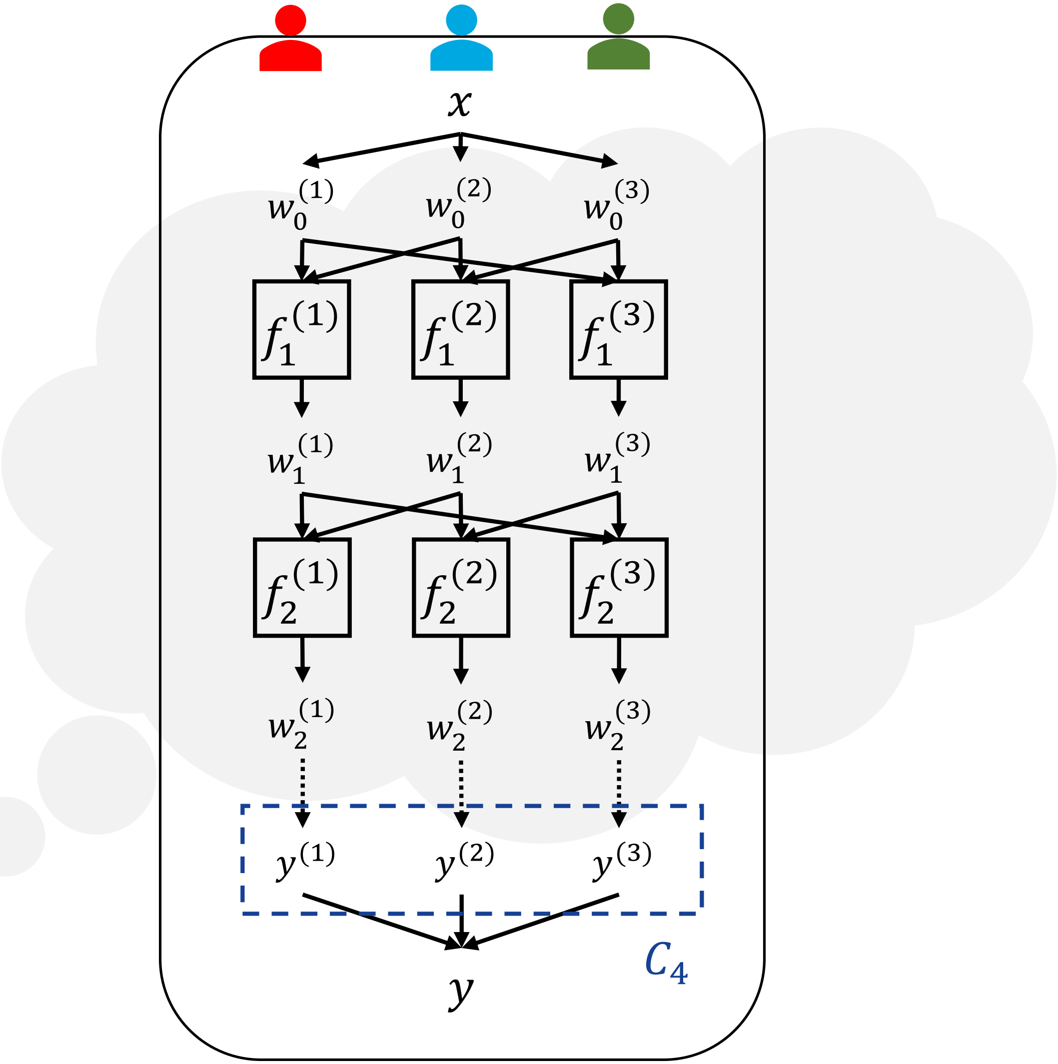


MPC-in-the-Head Paradigm (Simplified)

Commit the views



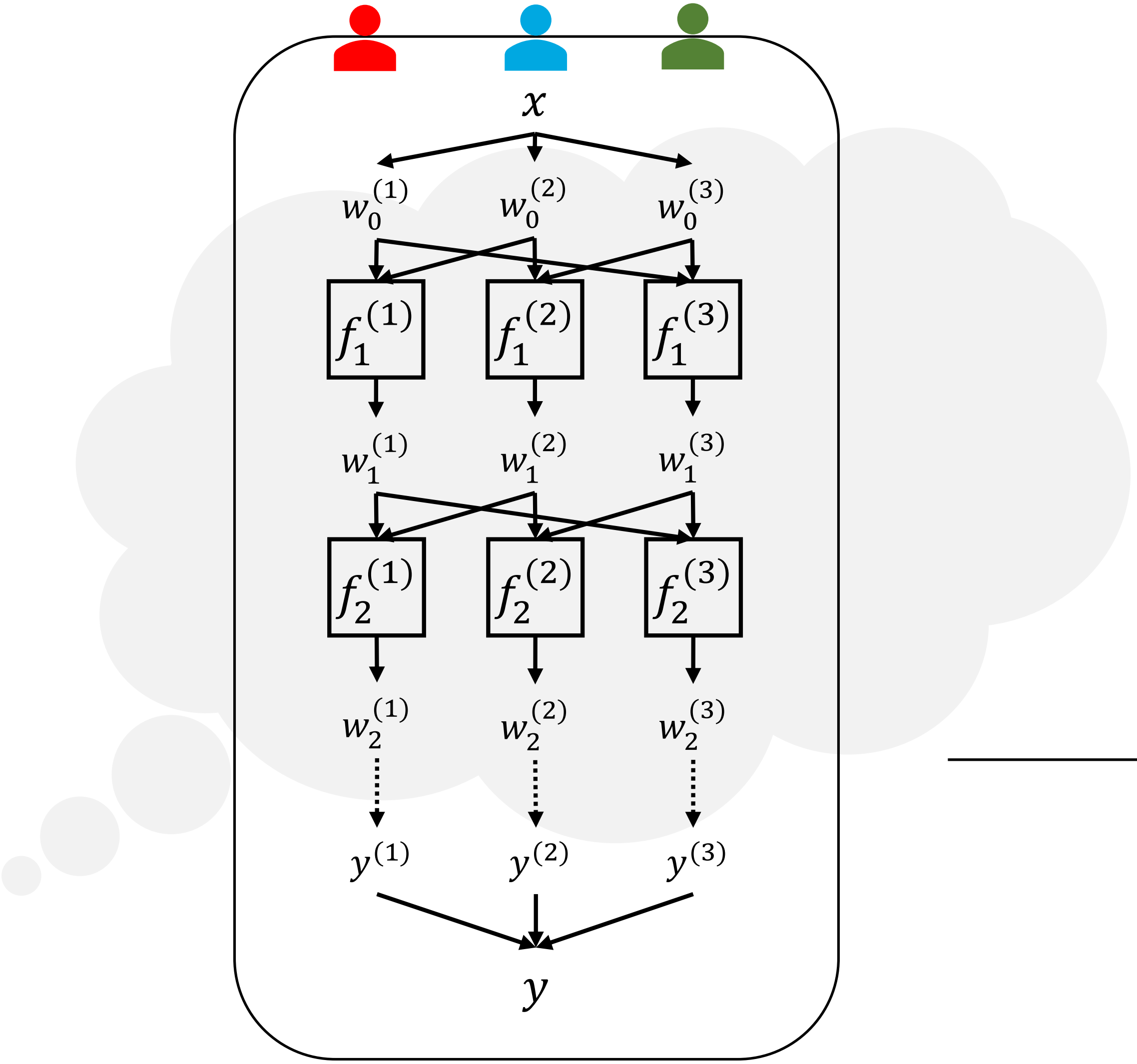
MPC-in-the-Head Paradigm (Simplified)



Commit the output shares

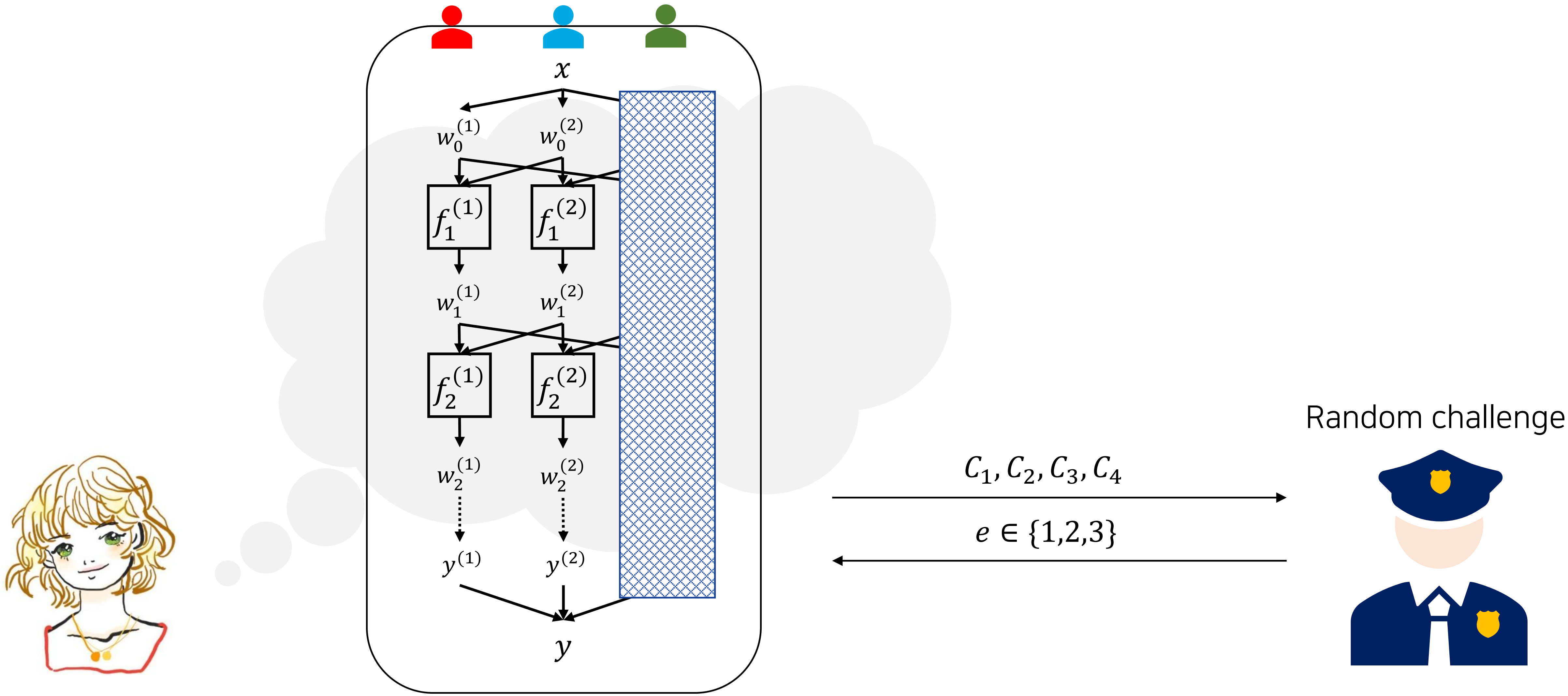
MPC-in-the-Head Paradigm (Simplified)

Send commits

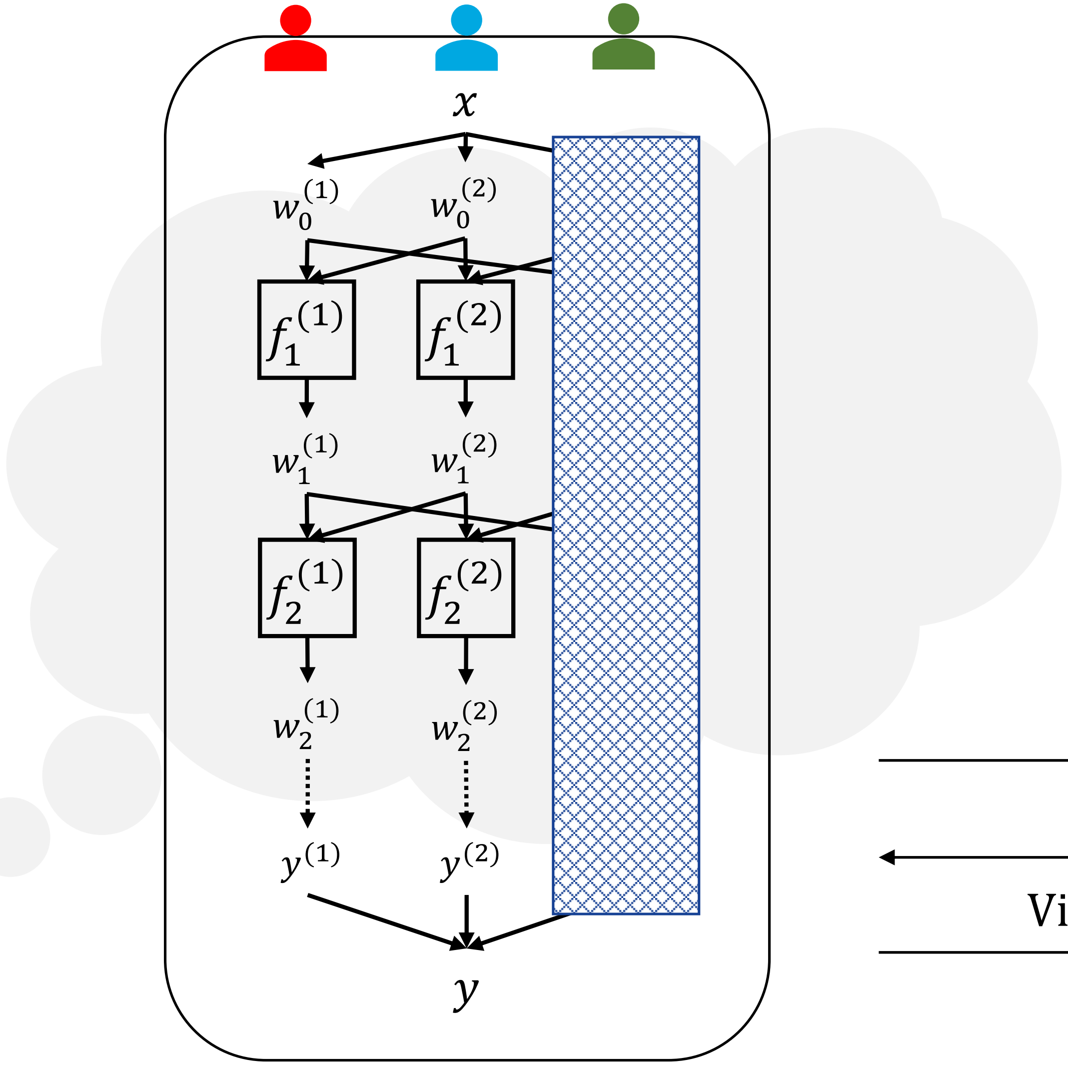


$$\xrightarrow{C_1, C_2, C_3, C_4}$$

MPC-in-the-Head Paradigm (Simplified)



MPC-in-the-Head Paradigm (Simplified)



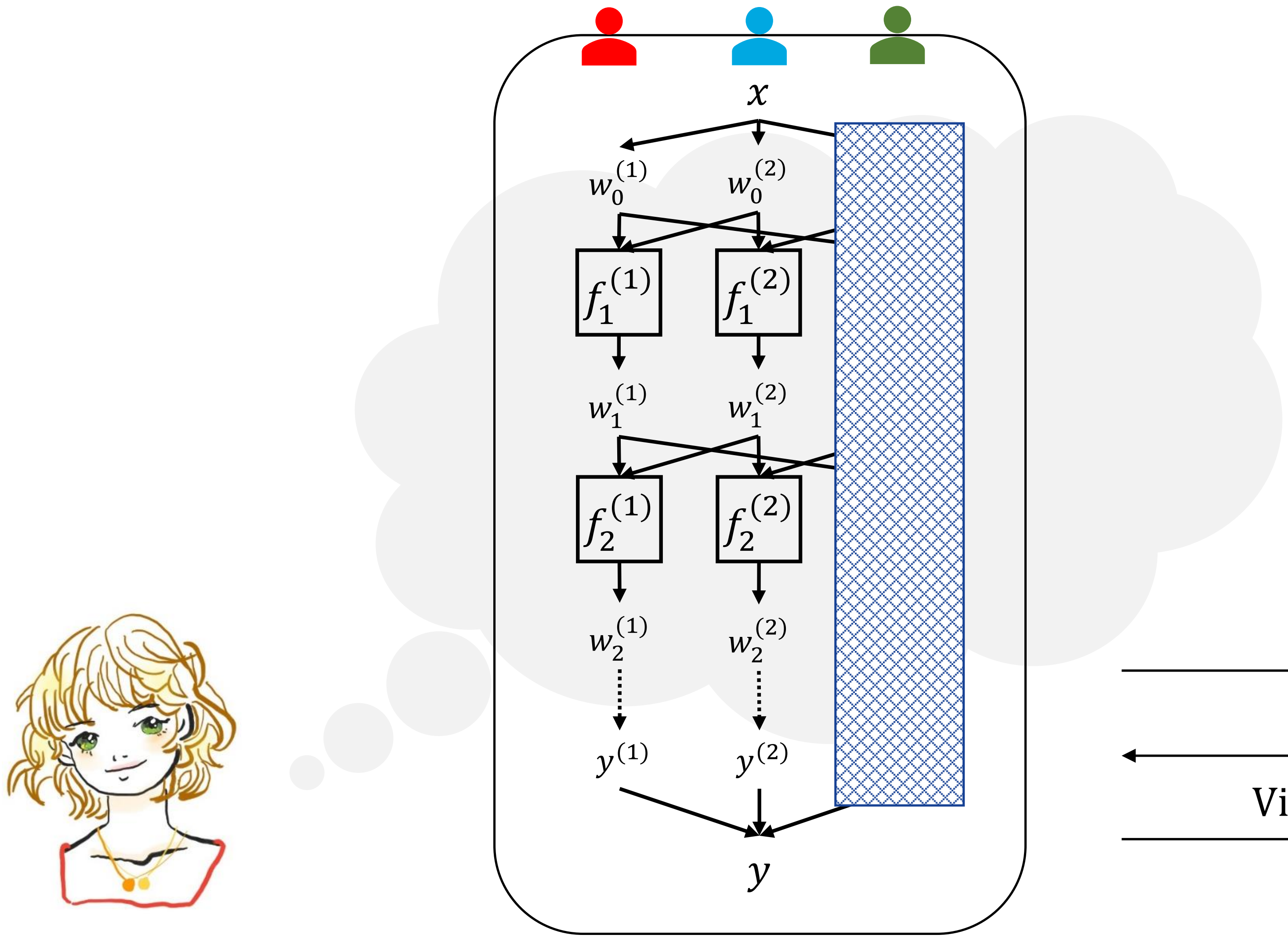
Send views

C_1, C_2, C_3, C_4 $\longrightarrow$

$e \in \{1,2,3\}$ $\longleftarrow$

$\text{View}_{e+1}, \text{View}_{e+2}$ $\longrightarrow$

MPC-in-the-Head Paradigm (Simplified)



C_1, C_2, C_3, C_4 →

$e \in \{1,2,3\}$ ←

$\text{View}_{e+1}, \text{View}_{e+2}$ →

Check Consistency

$\text{Commit}(\text{View}_{e+1}) = C_{e+1}$

$\text{Commit}(\text{View}_{e+2}) = C_{e+2}$

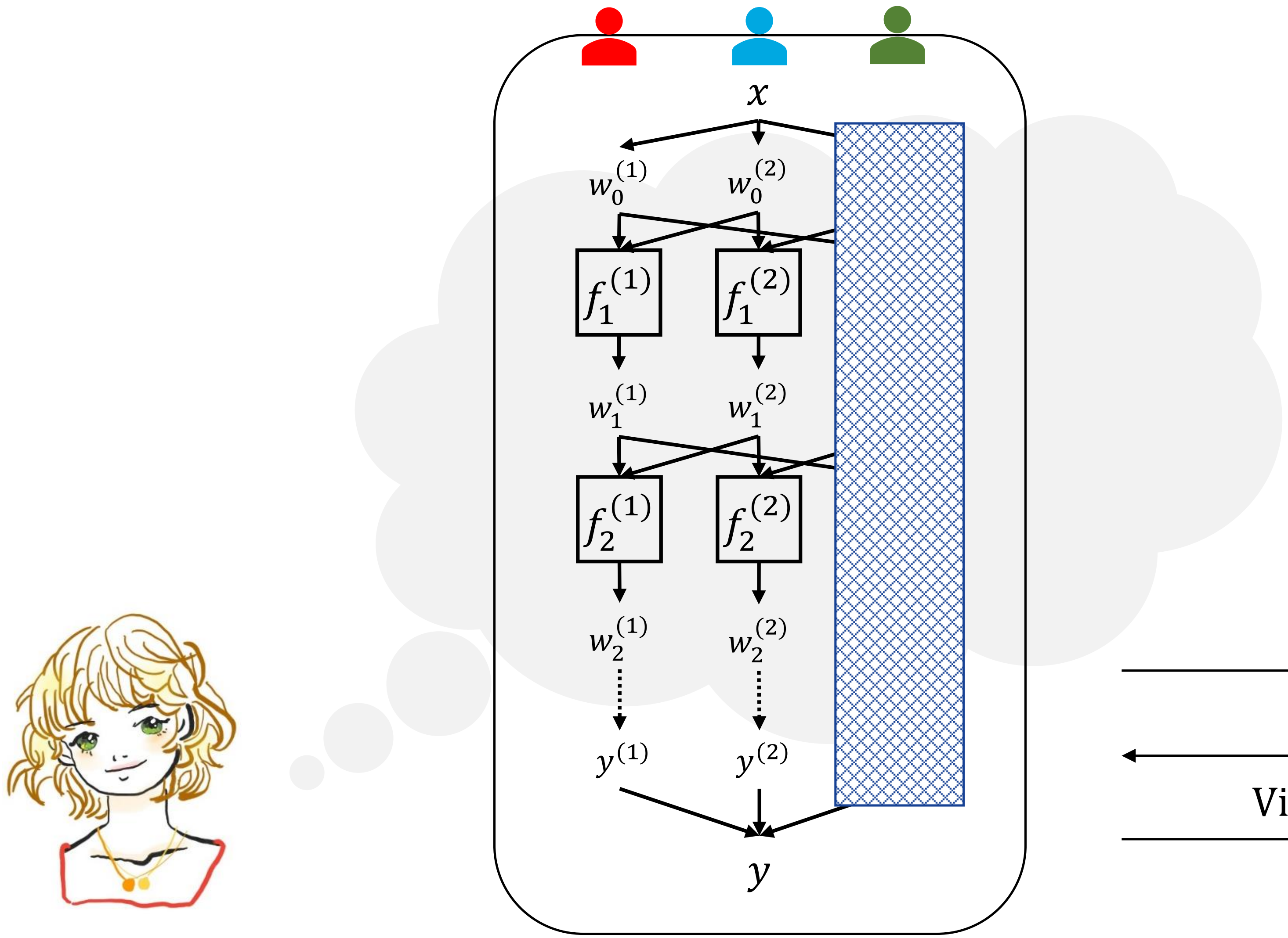
$\text{View}_{e+1} \to y^{(e+1)}$

$\text{View}_{e+2} \to y^{(e+2)}$

$y^{(e)} = y - y^{(e+1)} - y^{(e+2)}$

$\text{Commit}\left(y^{(1)}, y^{(2)}, y^{(3)}\right) = C_4$

MPC-in-the-Head Paradigm (Simplified)



C_1, C_2, C_3, C_4 →

$e \in \{1,2,3\}$ ←

$\mathrm{View}_{e+1}, \mathrm{View}_{e+2}$ →

Check Consistency

$\mathrm{Commit}(\mathrm{View}_{e+1}) = C_{e+1}$

$\mathrm{Commit}(\mathrm{View}_{e+2}) = C_{e+2}$

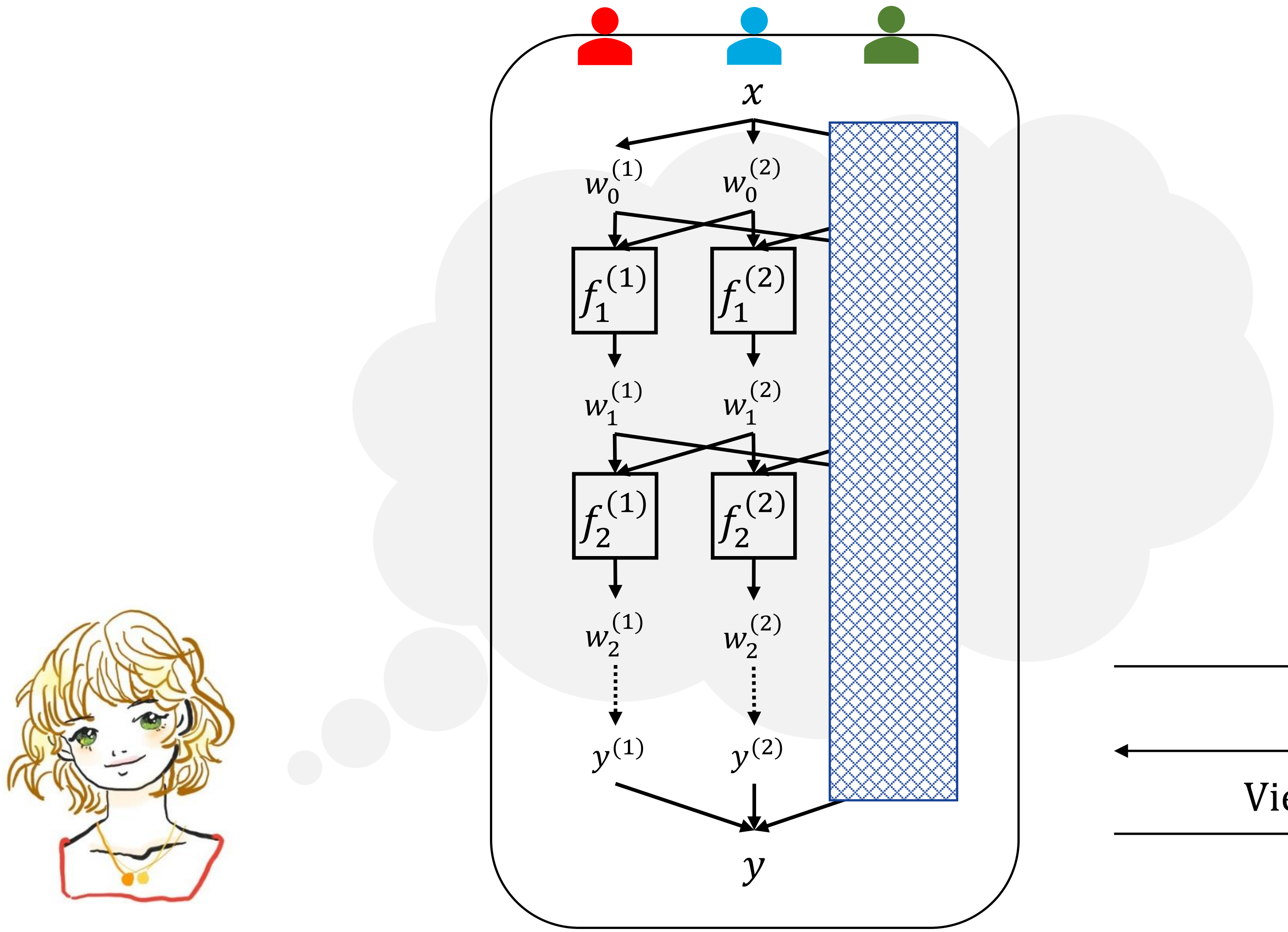
$\mathrm{View}_{e+1} \to y^{(e+1)}$

$\mathrm{View}_{e+2} \to y^{(e+2)}$

$y^{(e)} = y - y^{(e+1)} - y^{(e+2)}$

$\mathrm{Commit}\left(y^{(1)}, y^{(2)}, y^{(3)}\right) = C_4$

MPC-in-the-Head Paradigm (Simplified)



Check Consistency

$\mathrm{Commit}(\mathrm{View}_{e+1}) = C_{e+1}$

$\mathrm{Commit}(\mathrm{View}_{e+2}) = C_{e+2}$

$\mathrm{View}_{e+1} \to y^{(e+1)}$

$\mathrm{View}_{e+2} \to y^{(e+2)}$

$y^{(e)} = y - y^{(e+1)} - y^{(e+2)}$

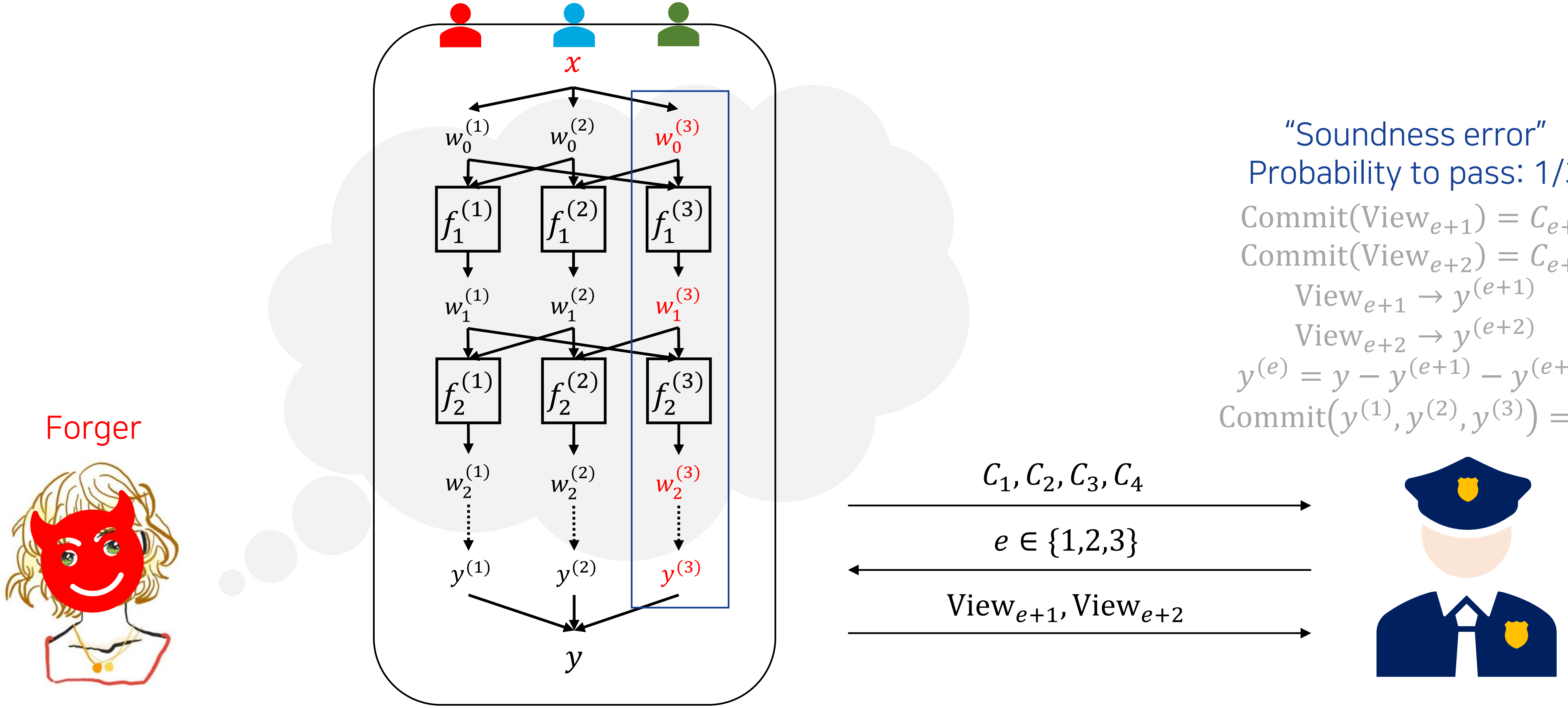
$\mathrm{Commit}\left(y^{(1)}, y^{(2)}, y^{(3)}\right) = C_4$

C_1, C_2, C_3, C_4 →

← $e \in \{1,2,3\}$

$\mathrm{View}_{e+1}, \mathrm{View}_{e+2}$ →

MPC-in-the-Head Paradigm (Simplified)



"Soundness error"
Probability to pass: 1/3

$\mathrm{Commit}(\mathrm{View}_{e+1}) = C_{e+1}$

$\mathrm{Commit}(\mathrm{View}_{e+2}) = C_{e+2}$

$\mathrm{View}_{e+1} \to y^{(e+1)}$

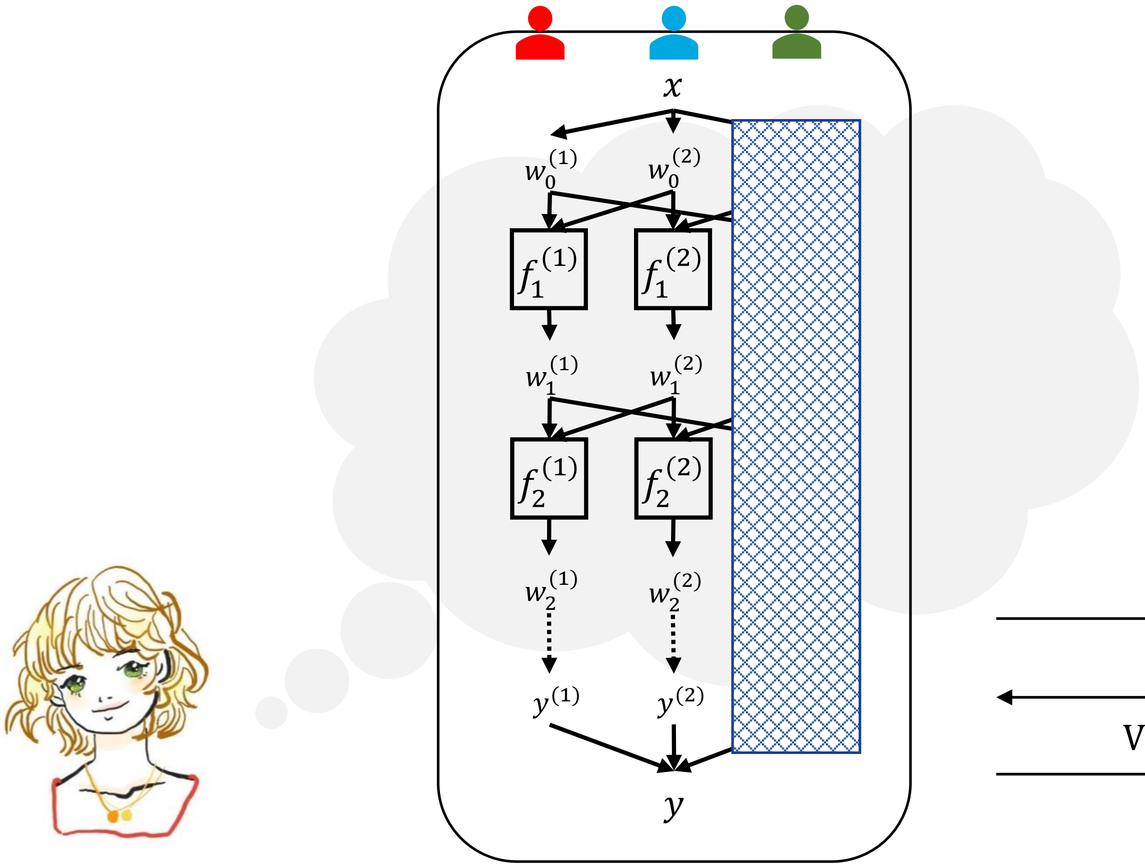
$\mathrm{View}_{e+2} \to y^{(e+2)}$

$y^{(e)} = y - y^{(e+1)} - y^{(e+2)}$

$\mathrm{Commit}\left(y^{(1)}, y^{(2)}, y^{(3)}\right) = C_4$

MPC-in-the-Head Paradigm (Simplified)

Repeat several times for security



Soundness error: 1/3

$\mathrm{Commit}(\mathrm{View}_{e+1}) = C_{e+1}$

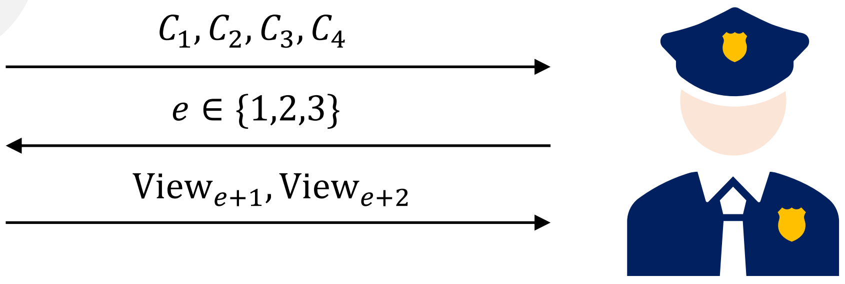
$\mathrm{Commit}(\mathrm{View}_{e+2}) = C_{e+2}$

$\mathrm{View}_{e+1} \to y^{(e+1)}$

$\mathrm{View}_{e+2} \to y^{(e+2)}$

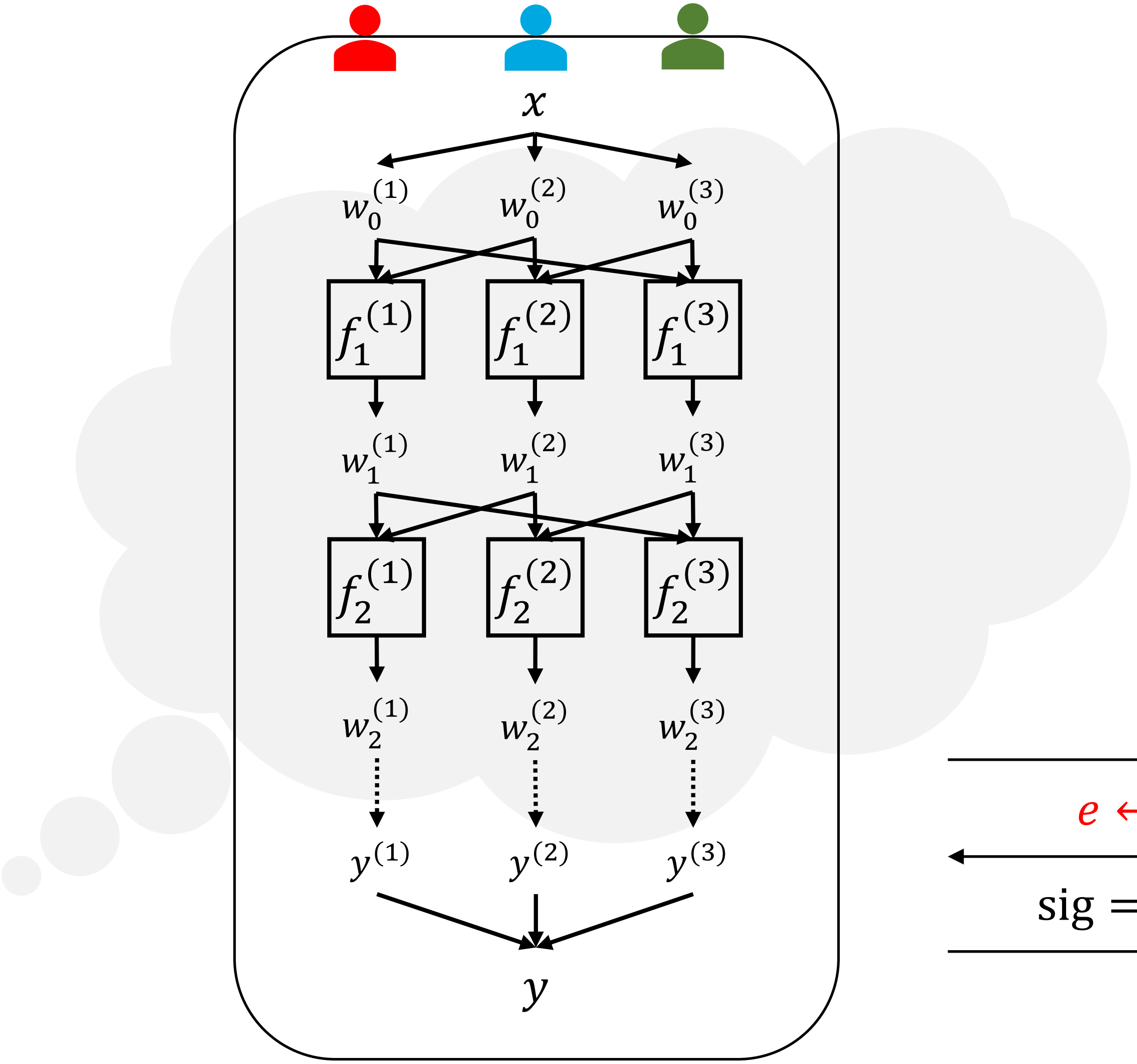
$y^{(e)} = y - y^{(e+1)} - y^{(e+2)}$

$\mathrm{Commit}\left(y^{(1)}, y^{(2)}, y^{(3)}\right) = C_4$



MPCitH-based Signature (Simplified)

$f(K) = E_K(m)$



C_1, C_2, C_3, C_4 $\longrightarrow$

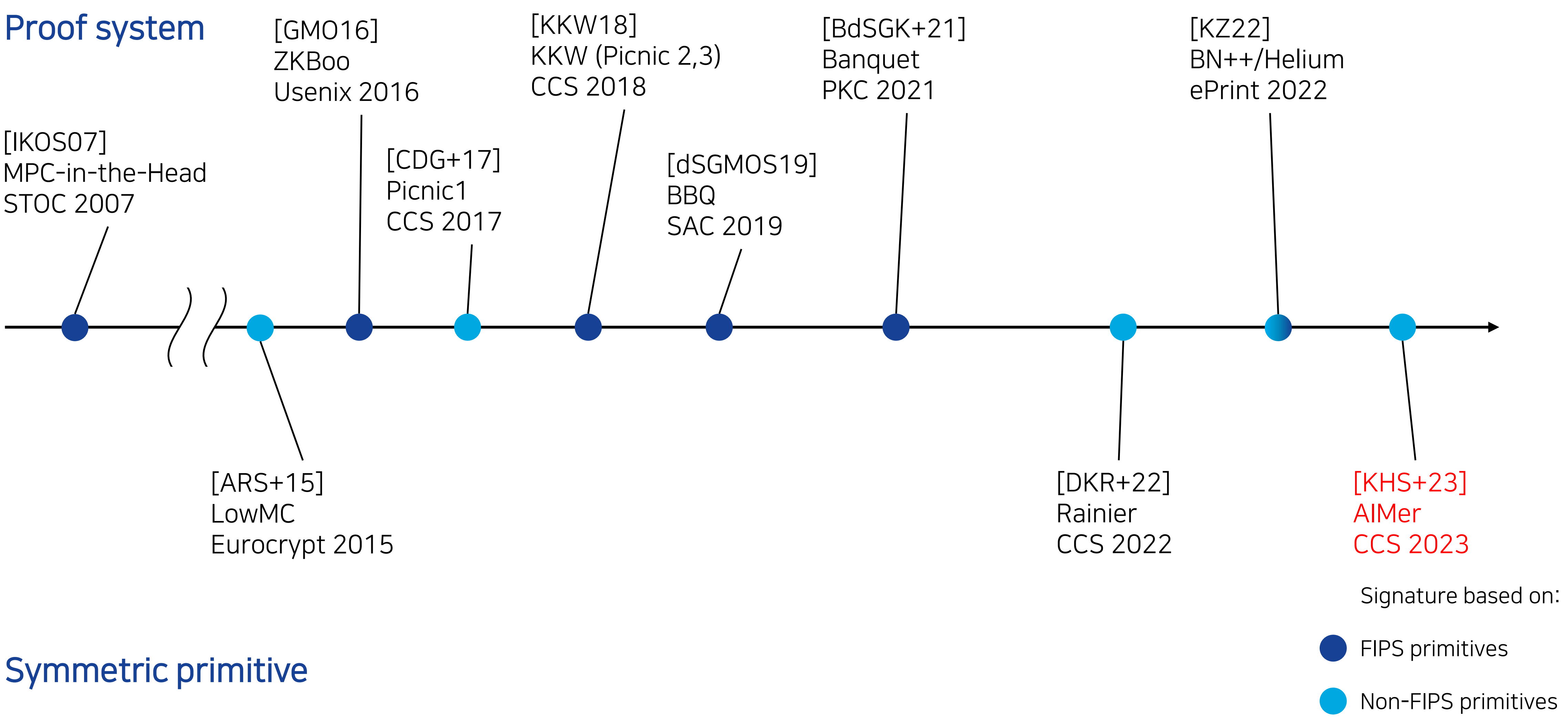
$e \leftarrow H(C_1, C_2, C_3, C_4)$ $\longleftarrow$

$\mathrm{sig} = (\mathrm{View}_{e+1}, \mathrm{View}_{e+2})$ $\longrightarrow$

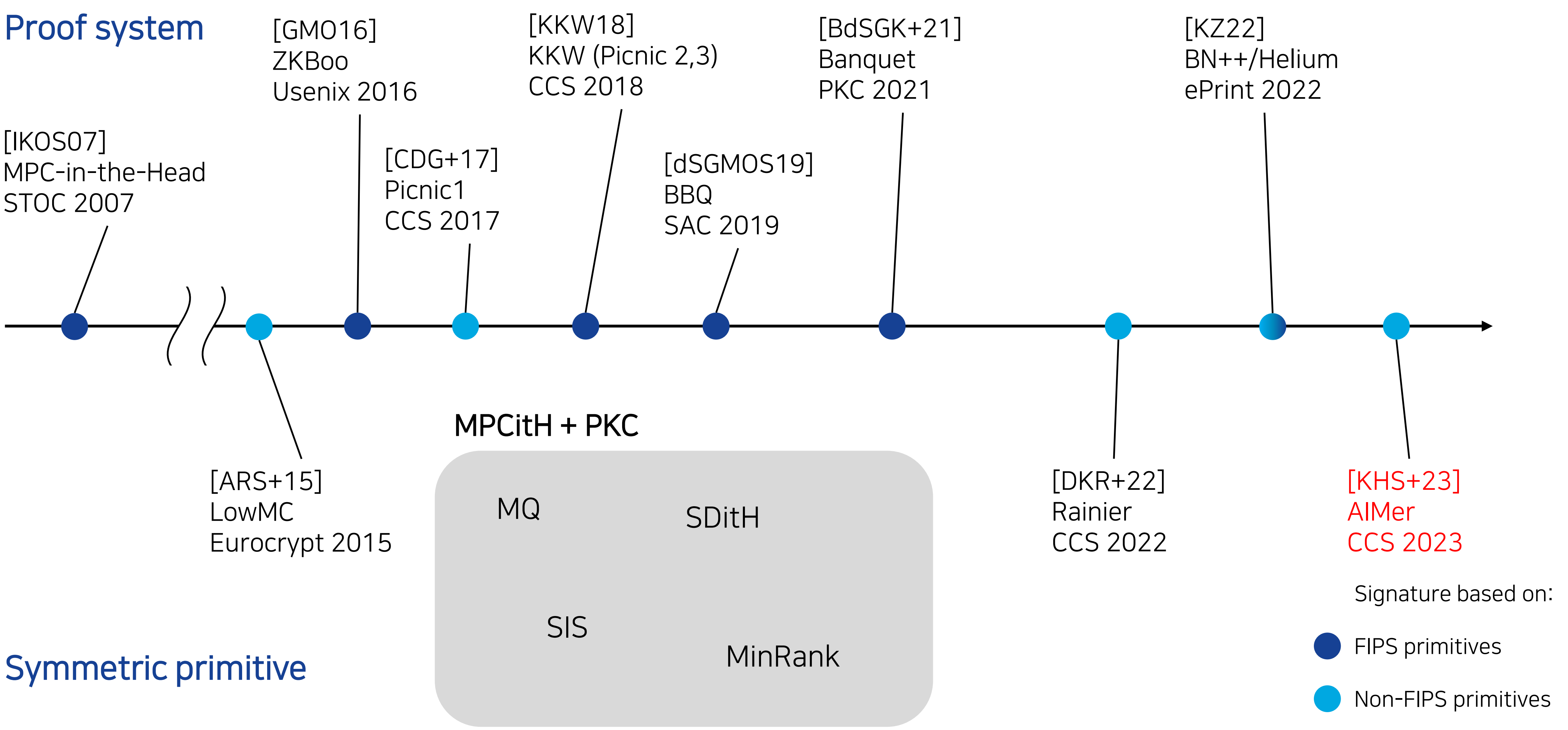
Fiat-Shamir transform

Previous Works

Brief History



Brief History

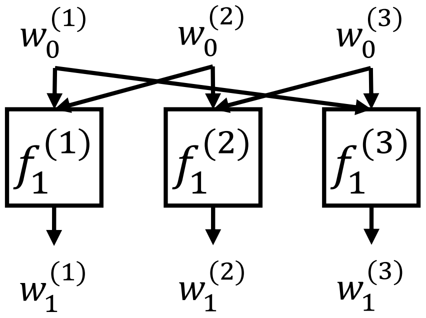


Picnic1

- Picnic1 = ZKB++ (optimized ZKBoo) + Fiat-Shamir transform + LowMC

ZKB++

- (2,3)-circuit decomposition
- No multiplication triple
- 3-party fixed, large number of repetition

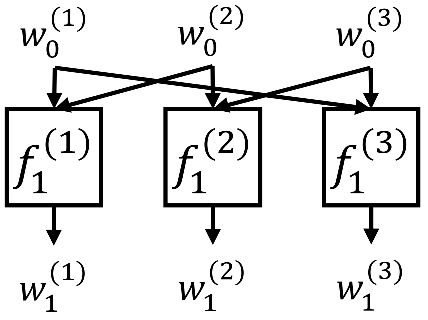


Picnic1

- Picnic1 = ZKB++ (optimized ZKBoo) + Fiat-Shamir transform + LowMC

ZKB++
- (2,3)-circuit decomposition
- No multiplication triple
- 3-party fixed, large number of repetition

FS transform
- Interactive ZK → NIZK
- QROM security is later proved: Unruh → FS



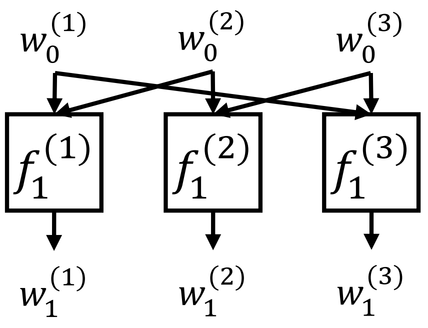
Picnic1

- Picnic1 = ZKB++ (optimized ZKBoo) + Fiat-Shamir transform + LowMC

ZKB++
- (2,3)-circuit decomposition
- No multiplication triple
- 3-party fixed, large number of repetition

FS transform
- Interactive ZK → NIZK
- QROM security is later proved: Unruh → FS

LowMC
- Cipher for MPC/FHE/ZKP
- Low number of AND gates
- 3-bit S-box, random affine
- Reduced parameter sets



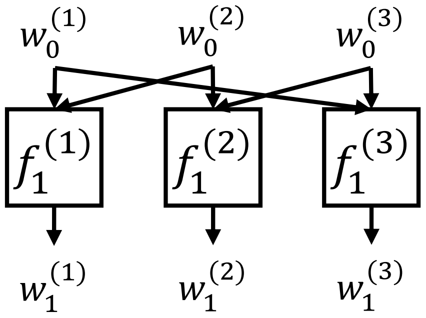
Picnic1

- Picnic1 = ZKB++ (optimized ZKBoo) + Fiat-Shamir transform + LowMC

ZKB++
- (2,3)-circuit decomposition
- No multiplication triple
- 3-party fixed, large number of repetition

FS transform
- Interactive ZK → NIZK
- QROM security is later proved: Unruh → FS

LowMC
- Cipher for MPC/FHE/ZKP
- Low number of AND gates
- 3-bit S-box, random affine
- Reduced parameter sets



Performance

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Picnic1-L1-full	32	30925	1.16	0.91

KKW Proof System (Picnic3)

- Picnic3 = KKW NIZK proof of knowledge + LowMC
- Poor soundness of 3-party → use preprocessing model to simulate N parties!

KKW Proof System (Picnic3)

- Picnic3 = KKW NIZK proof of knowledge + LowMC
- Poor soundness of 3-party → use preprocessing model to simulate N parties!

MPCitH with (2,3)-decomposition

- 2-party secure channel model
- No multiplication triple needed
- 3-party fixed, large number of repetition

MPCitH with preprocessing

- N-party broadcast model
- Prover generates multiplication triples and commit to them
- Checking consistency by opening some of triples
- #parties↑, #repetitions↓

KKW Proof System (Picnic3)

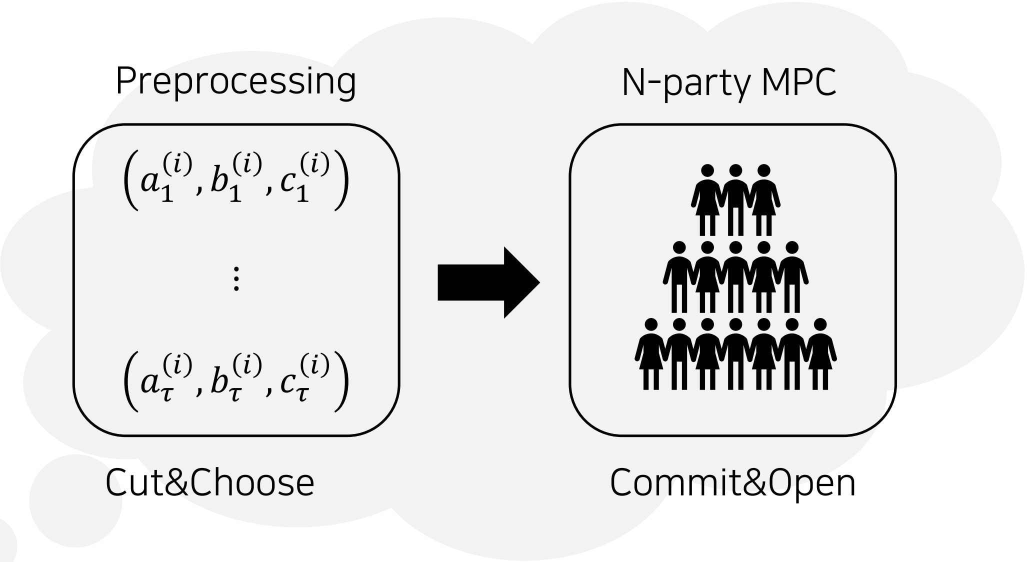
- Picnic3 = KKW NIZK proof of knowledge + LowMC
- Poor soundness of 3-party → use preprocessing model to simulate N parties!

MPCitH with (2,3)-decomposition

- 2-party secure channel model
- No multiplication triple needed
- 3-party fixed, large number of repetition

MPCitH with preprocessing

- N-party broadcast model
- Prover generates multiplication triples and commit to them
- Checking consistency by opening some of triples
- #parties↑, #repetitions↓

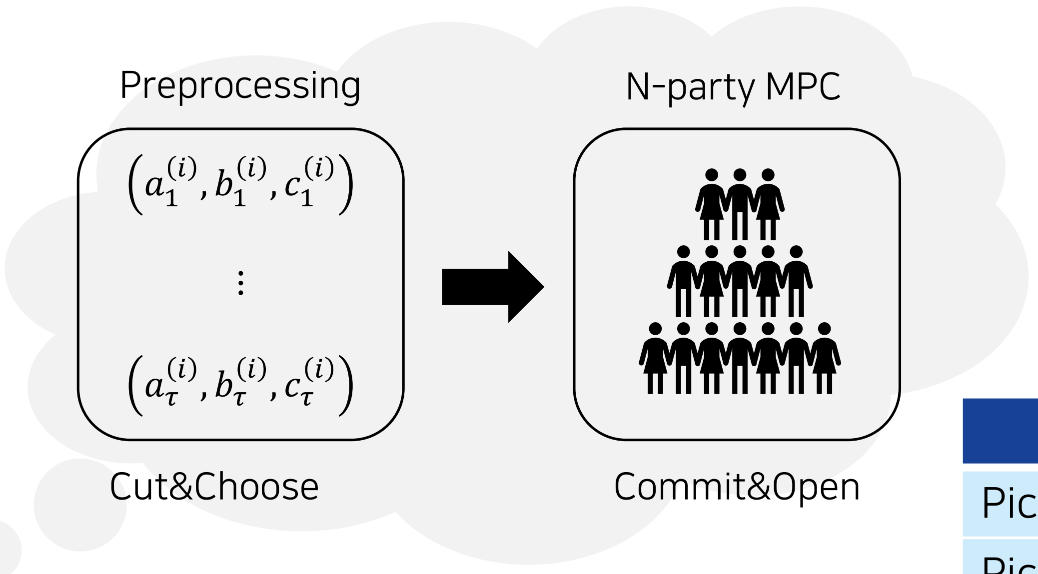


KKW Proof System (Picnic3)

- Picnic3 = KKW NIZK proof of knowledge + LowMC
- Poor soundness of 3-party → use preprocessing model to simulate N parties!

MPCitH with (2,3)-decomposition

- 2-party secure channel model
- No multiplication triple needed
- 3-party fixed, large number of repetition



MPCitH with preprocessing

- N-party broadcast model
- Prover generates multiplication triples and commit to them
- Checking consistency by opening some of triples
- #parties↑, #repetitions↓

Performance

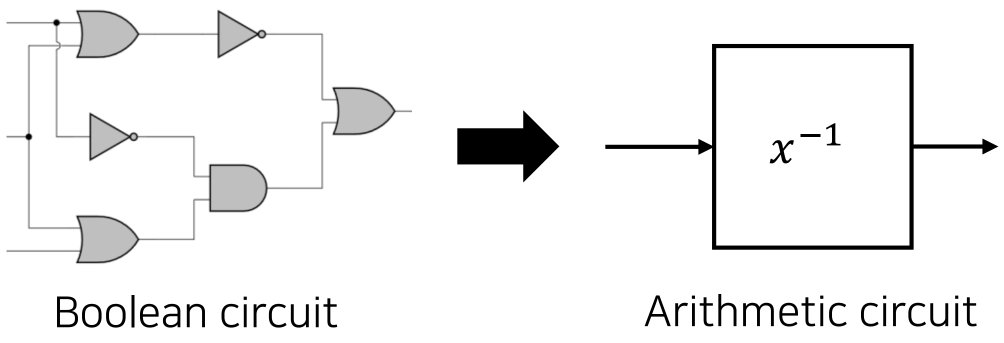
Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Picnic1-L1-full	32	30925	1.16	0.91
Picnic3	32	12463	5.83	4.24

BBQ Signature Scheme

- BBQ = KKW with $\mathbb{F}_{2^8}$ multiplication triples + AES

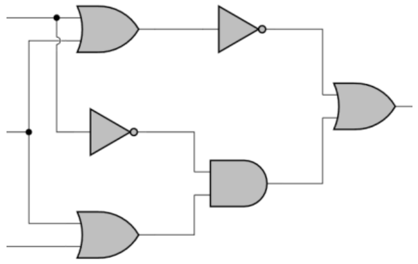
BBQ Signature Scheme

- BBQ = KKW with $\mathbb{F}_{2^8}$ multiplication triples + AES
- Motivation
  - LowMC is not solid compared to AES
  - AES has too much ANDs (LowMC = 600 ANDs, AES = 6400 ANDs)
  - Arithmetic inversion leads to 40% smaller signature size

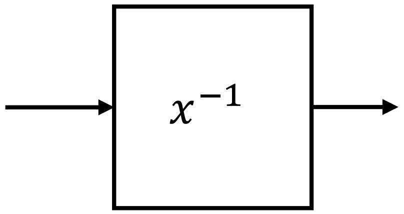


BBQ Signature Scheme

- BBQ = KKW with $\mathbb{F}_{2^8}$ multiplication triples + AES
- Motivation
  - LowMC is not solid compared to AES
  - AES has too much ANDs (LowMC = 600 ANDs, AES = 6400 ANDs)
  - Arithmetic inversion leads to 40% smaller signature size



Boolean circuit



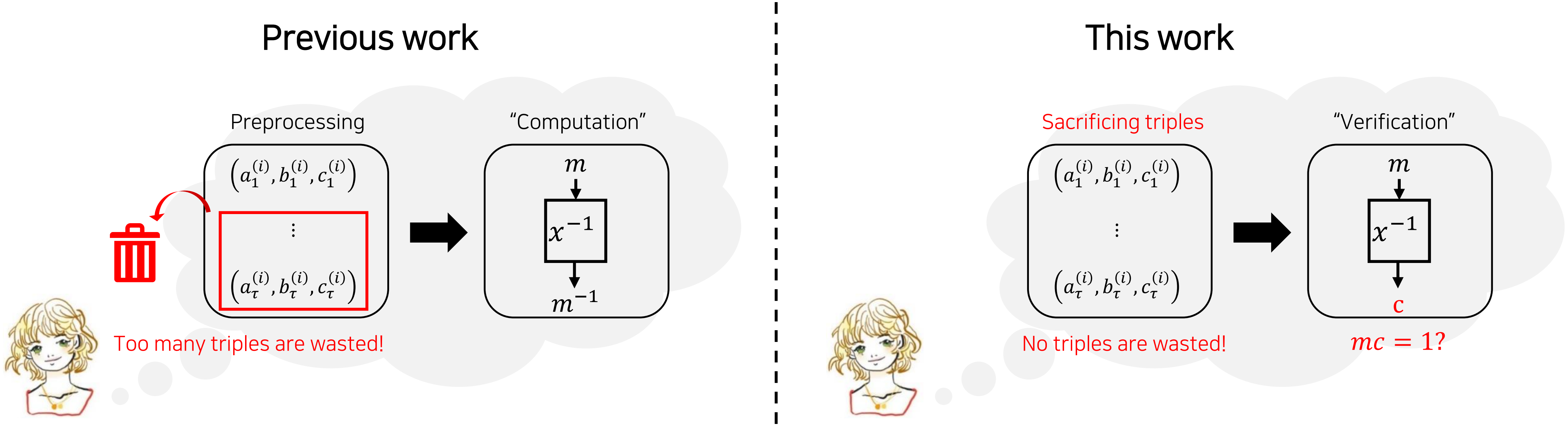
Arithmetic circuit

Performance

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Picnic1-L1-full	32	30925	1.16	0.91
Picnic3	32	12463	5.83	4.24
BBQ	32	31568	unknown	unknown

Banquet Signature Scheme

- Banquet = Multiplication-checking protocol + AES
- Idea
  - Cut-and-choose → Sacrificing technique with inverse injection



Banquet Signature Scheme

- Banquet = Multiplication-checking protocol + AES
- Idea
  - Cut-and-choose → Sacrificing technique with inverse injection
  - Batching verification

Soundness error = $2m/|\mathbb{F} - m|$

$(s_1, t_1, 1)$
$\vdots$
$(s_m, t_m, 1)$

Sacrifice to verify

(a_1, b_1, c_1)
$\vdots$
(a_m, b_m, c_m)

$3m$ elements

$S(1) = s_1, T(1) = t_1$
$\vdots$
$S(m) = s_m, T(1) = t_m$
$P = S \cdot T$

(Kind of) Sacrifice
half of $P(X)$
to verify
$P(R) - S(R)T(R) = 0$

$m + 1$ elements

Banquet Signature Scheme

- Banquet = Multiplication-checking protocol + AES
- Idea
  - Cut-and-choose → Sacrificing technique with inverse injection
  - Batching verification

Soundness error = $2m/|\mathbb{F} - m|$

$(s_1, t_1, 1)$
$\vdots$
$(s_m, t_m, 1)$

Sacrifice to verify ←

(a_1, b_1, c_1)
$\vdots$
(a_m, b_m, c_m)

$3m$ elements

$S(1) = s_1, T(1) = t_1$
$\vdots$
$S(m) = s_m, T(1) = t_m$
$P = S \cdot T$

←

(Kind of) Sacrifice half of $P(X)$ to verify
$P(R) - S(R)T(R) = 0$

$m + 1$ elements

Performance

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Picnic1-L1-full	32	30925	1.16	0.91
Picnic3	32	12463	5.83	4.24
Banquet	32	19776	7.09	5.24

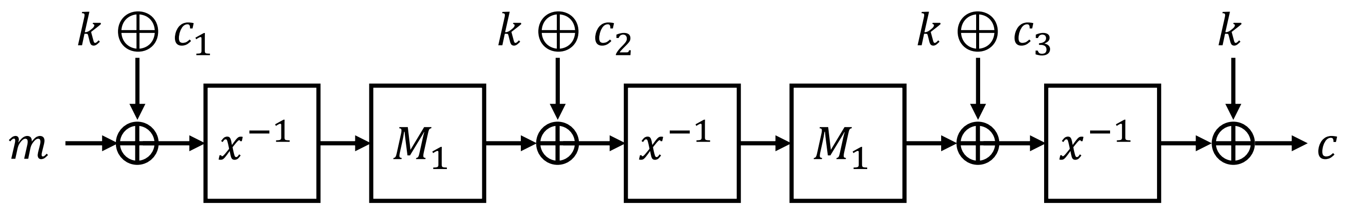
Rainier Signature Scheme

- Rainier = Modified Banquet proof + New symmetric primitive Rain
- Motivation
  - AES uses a small field, which occurs poor soundness
  - Banquet already lifts $\mathbb{F}_{2^8}$ to $\mathbb{F}_{2^{32}}$ for soundness
  - Inverse on a large field is not expensive in MPC

Rainier Signature Scheme

- Rainier = Modified Banquet proof + New symmetric primitive Rain
- Motivation
  - AES uses a small field, which occurs poor soundness
  - Banquet already lifts $\mathbb{F}_{2^8}$ to $\mathbb{F}_{2^{32}}$ for soundness
  - Inverse on a large field is not expensive in MPC
- Cryptanalytic characteristic
  - Large inverse and random matrix are used for algebraic attacks
  - Statistical attacks are not much of our interest

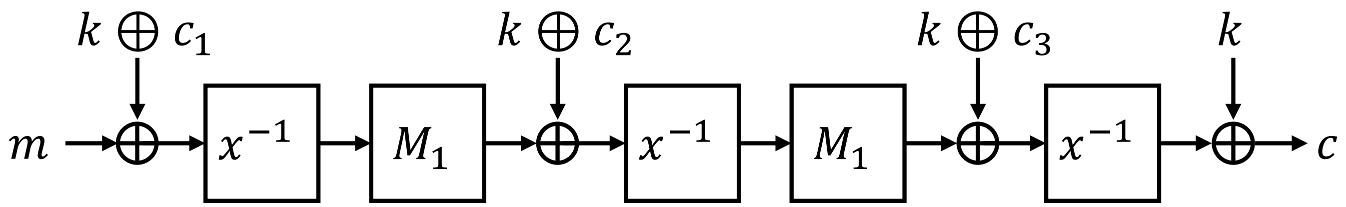
Rain$_3$



Rainier Signature Scheme

- Rainier = Modified Banquet proof + New symmetric primitive Rain
- Motivation
  - AES uses a small field, which occurs poor soundness
  - Banquet already lifts $\mathbb{F}_{2^8}$ to $\mathbb{F}_{2^{32}}$ for soundness
  - Inverse on a large field is not expensive in MPC
- Cryptanalytic characteristic
  - Large inverse and random matrix are used for algebraic attacks
  - Statistical attacks are not much of our interest

Rain$_3$



Performance

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Picnic1-L1-full	32	30925	1.16	0.91
Picnic3	32	12463	5.83	4.24
Banquet	32	19776	7.09	5.24
Rainier$_3$	32	8544	0.97	0.89

BN++/Helium Proof System

- BN++: Optimization of BN protocol
  - BN20: Sacrificing-based interactive proof protocol
  - Remove needless broadcasts
  - Repeated multiplier
  - Known output share

$$\boxed{x} \cdot y = z$$

BN++/Helium Proof System

- BN++: Optimization of BN protocol
  - BN20: Sacrificing-based interactive proof protocol
  - Remove needless broadcasts
  - Repeated multiplier
  - Known output share

$$x \cdot y = \boxed{z}$$

BN++/Helium Proof System

- BN++: Optimization of BN protocol
  - BN20: Sacrificing-based interactive proof protocol
  - Remove needless broadcasts
  - Repeated multiplier
  - Known output share

$$x \cdot y = z$$

- Helium: BN++ with RMFE (Reverse Multiplication-Friendly Embedding)
  - Small field arithmetic has high soundness error
  - Batch small field operations to a large field one

BN++/Helium Proof System

- BN++: Optimization of BN protocol
  - BN20: Sacrificing-based interactive proof protocol
  - Remove needless broadcasts
  - Repeated multiplier
  - Known output share

$$x \cdot y = z$$

- Helium: BN++ with RMFE (Reverse Multiplication-Friendly Embedding)
  - Small field arithmetic has high soundness error
  - Batch small field operations to a large field one

Performance

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Picnic1-L1-full	32	30925	1.16	0.91
Picnic3	32	12463	5.83	4.24
Banquet	32	19776	7.09	5.24
Rainier_3	32	8544	0.97	0.89
BN++Rain_3	32	6432	0.83	0.77
Helium-AES	32	9888	16.53	16.47

The AIMer Signature Scheme

Motivation

- MPC(itH)-friendly symmetric primitives are advanced in directions of:
  - S-boxes on large field
  - Low multiplicative complexity

Motivation

- MPC(itH)-friendly symmetric primitives are advanced in directions of:
  - S-boxes on large field
  - Low multiplicative complexity

- Some symmetric primitives based on large S-boxes have been broken by algebraic attacks
  - MiMC (AC 16, AC 20)
  - Agrasta (C 18, AC 21)
  - Jarvis/Friday (ePrint 18, AC 19)
  - Chaghri (CCS 22, EC 23)

Motivation

- MPC(itH)-friendly symmetric primitives are advanced in directions of:
  - S-boxes on large field
  - Low multiplicative complexity

- Some symmetric primitives based on large S-boxes have been broken by algebraic attacks
  - MiMC (AC 16, AC 20)
  - Agrasta (C 18, AC 21)
  - Jarvis/Friday (ePrint 18, AC 19)
  - Chaghri (CCS 22, EC 23)

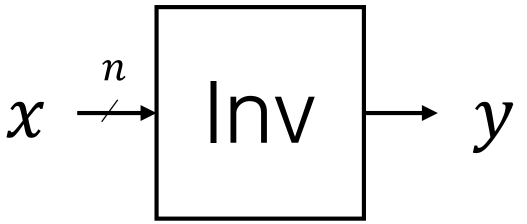
Sufficient security against algebraic attacks + Best performance when combined to BN++

Inverse S-box

- Inverse S-box ($x \mapsto x^{-1}$) is widely used in MPC/ZKP-friendly ciphers
  - High degree, but quadratic relation ($xy = 1$)
  - Invertible
  - Nice DC/LC resistance
  - But, produces many linearly independent quadratic equations

Inverse S-box

- Inverse S-box ($x \mapsto x^{-1}$) is widely used in MPC/ZKP-friendly ciphers
  - High degree, but quadratic relation ($xy = 1$)
  - Invertible
  - Nice DC/LC resistance
  - But, produces many linearly independent quadratic equations



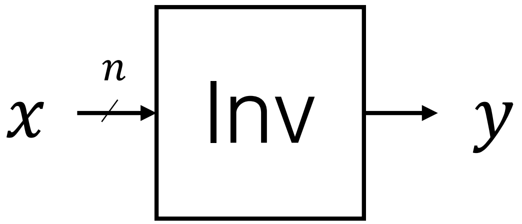
$$\begin{cases} f_1(x_1, \dots, x_n, y_1, \dots, y_n) = 0 \\ \vdots \\ f_{5n}(x_1, \dots, x_n, y_1, \dots, y_n) = 0 \end{cases}$$

$5n$ quadratic equations

c.f. optimally n equations

Inverse S-box

- Inverse S-box ($x \mapsto x^{-1}$) is widely used in MPC/ZKP-friendly ciphers
  - High degree, but quadratic relation ($xy = 1$)
  - Invertible
  - Nice DC/LC resistance
  - But, produces many linearly independent quadratic equations



$$\begin{cases} f_1(x_1, \dots, x_n, y_1, \dots, y_n) = 0 \\ \vdots \\ f_{5n}(x_1, \dots, x_n, y_1, \dots, y_n) = 0 \end{cases}$$

$5n$ quadratic equations

c.f. optimally n equations

More equations lead to a weaker resistance against algebraic attacks!

Candidates of Appropriate S-box

- Niho exponent
  - $x \mapsto x^{2^s+2^{s/2}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s + 1$
  - n equations, high-degree
  - 2 multiplications, odd-length field

Candidates of Appropriate S-box

- Niho exponent
  - $x \mapsto x^{2^s+2^{s/2}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s + 1$
  - n equations, high-degree
  - 2 multiplications, odd-length field

- NGG exponent (Nawaz et al., 2009)
  - $x \mapsto x^{2^{s+1}+2^{s-1}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s$
  - $2n$ equations, even-length field, good DC/LC resistance
  - 2 multiplications

Candidates of Appropriate S-box

- Niho exponent
  - $x \mapsto x^{2^s+2^{s/2}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s+1$
  - n equations, high-degree
  - 2 multiplications, odd-length field
- NGG exponent (Nawaz et al., 2009)
  - $x \mapsto x^{2^{s+1}+2^{s-1}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s$
  - $2n$ equations, even-length field, good DC/LC resistance
  - 2 multiplications
- Mersenne exponent
  - $x \mapsto x^{2^s-1}$ over $\mathbb{F}_{2^n}$
  - $3n$ equations, even-length field, single multiplication
  - moderate DC/LC resistance

Candidates of Appropriate S-box

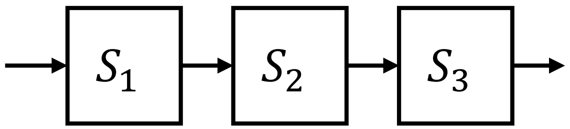
- Niho exponent
  - $x \mapsto x^{2^s+2^{s/2}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s + 1$
  - n equations, high-degree
  - 2 multiplications, odd-length field
- NGG exponent (Nawaz et al., 2009)
  - $x \mapsto x^{2^{s+1}+2^{s-1}-1}$ over $\mathbb{F}_{2^n}$, $n = 2s$
  - $2n$ equations, even-length field, good DC/LC resistance
  - 2 multiplications
- Mersenne exponent
  - $x \mapsto x^{2^s-1}$ over $\mathbb{F}_{2^n}$
  - $3n$ equations, even-length field, single multiplication
  - moderate DC/LC resistance
- Gold exponent
  - $x \mapsto x^{2^s+1}$ over $\mathbb{F}_{2^n}$
  - Even-length field, single multiplication, good DC/LC resistance
  - $4n$ equations

Repetitive Structure for BN++

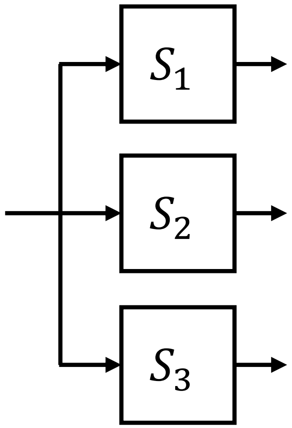
- Repeated multiplier technique (in BN++)
  - If prover needs to check multiple multiplications with a same multiplier,
    - e.g. $x_1 \cdot y = z_1, x_2 \cdot y = z_2$
  - Then, the prover can prove them in a batched way
  - More same multiplier → Smaller signature size

Repetitive Structure for BN++

- Repeated multiplier technique (in BN++)
  - If prover needs to check multiple multiplications with a same multiplier,
    - e.g. $x_1 \cdot y = z_1, x_2 \cdot y = z_2$
  - Then, the prover can prove them in a batched way
  - More same multiplier → Smaller signature size

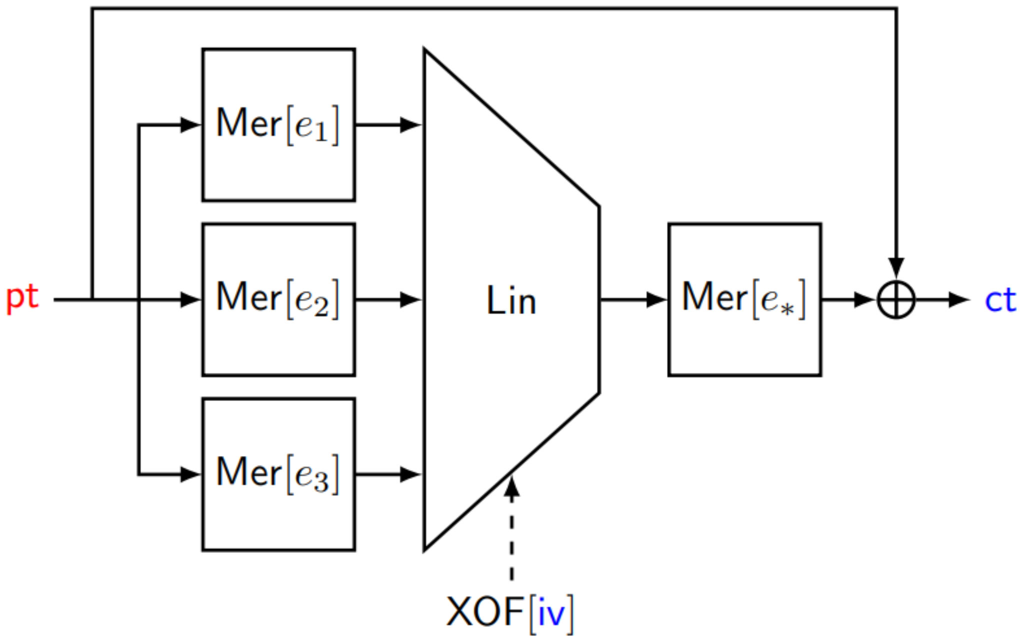


Serial S-box
(Limited application of repeated multiplier)



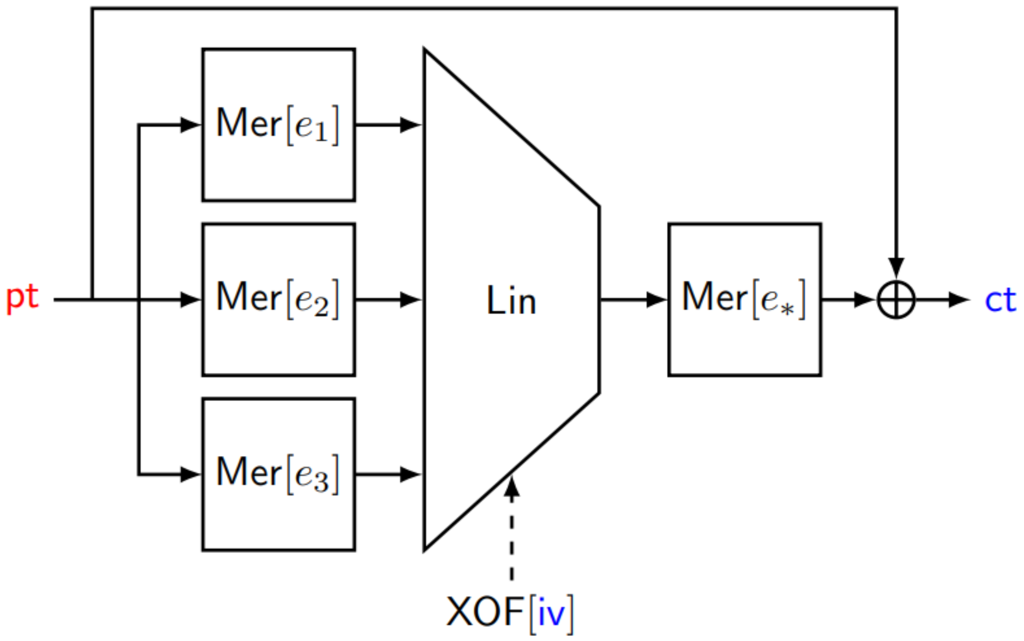
Parallel S-box
(Full application of repeated multiplier)

Symmetric Primitive AIM



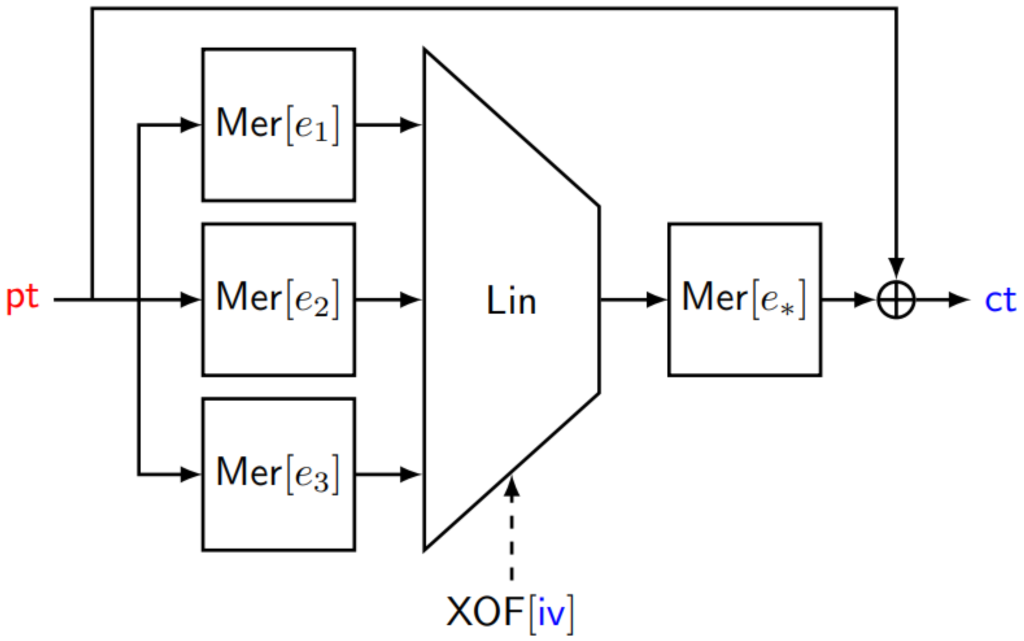
- Mersenne S-box
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
  - Moderate DC/LC resistance

Symmetric Primitive AIM



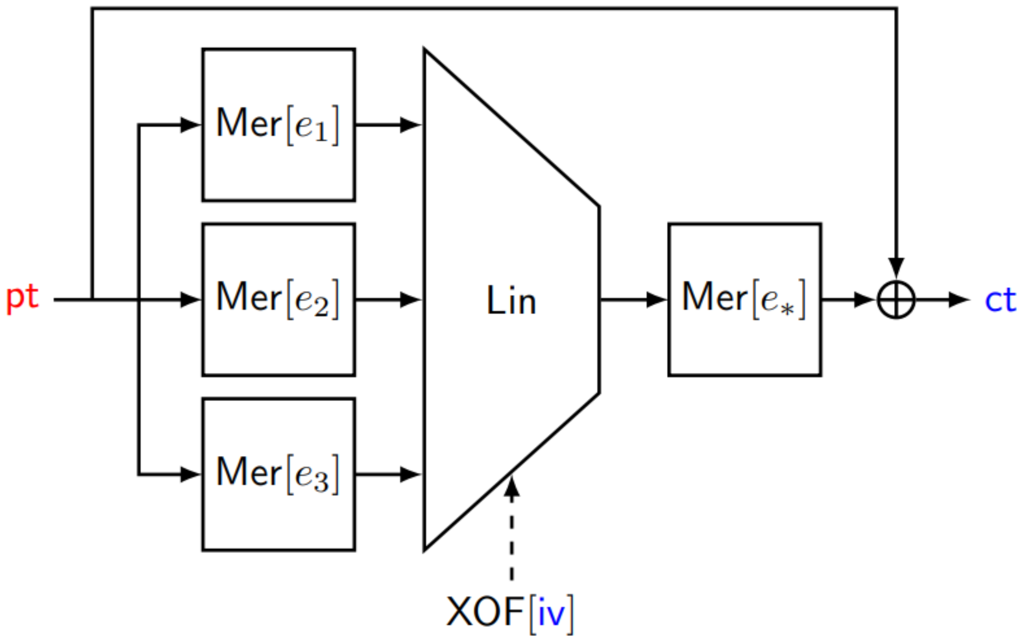
- Mersenne S-box
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
  - Moderate DC/LC resistance
- Repetitive structure
  - Parallel application of S-boxes
  - Feed-forward construction
  - Fully exploit the BN++ optimizations
  - Locally-computable output share

Symmetric Primitive AIM



- Mersenne S-box
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
  - Moderate DC/LC resistance
- Repetitive structure
  - Parallel application of S-boxes
  - Feed-forward construction
  - Fully exploit the BN++ optimizations
  - Locally-computable output share
- Randomized structure
  - Affine layer is generated from XOF

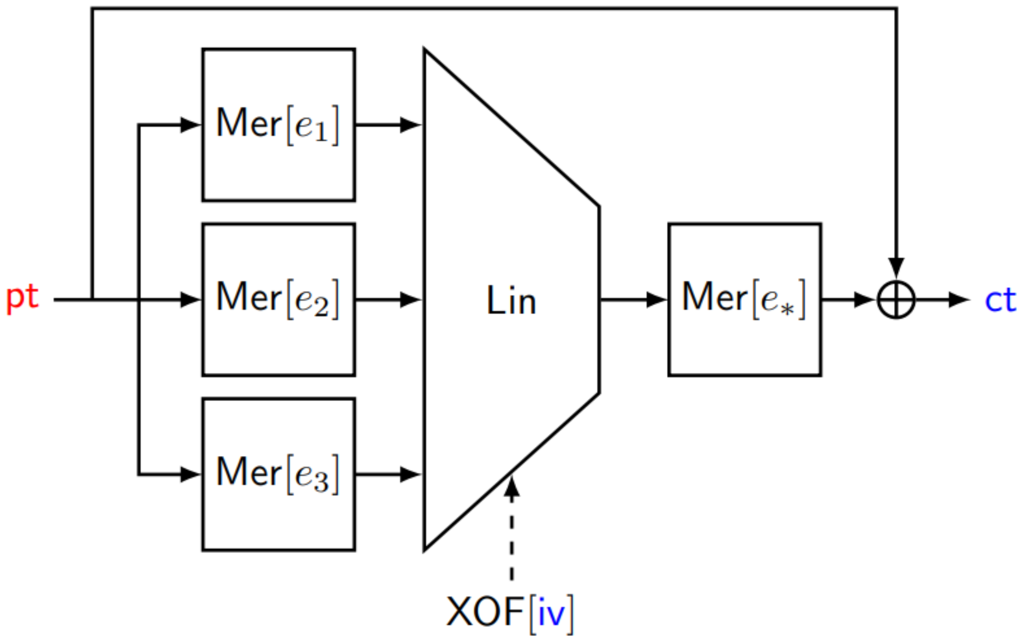
Symmetric Primitive AIM



Scheme	λ	n	ℓ	e_1	e_2	e_3	e_*
AIM-I	128	128	2	3	27	-	5
AIM-III	192	192	2	5	29	-	7
AIM-V	256	256	3	3	53	7	5

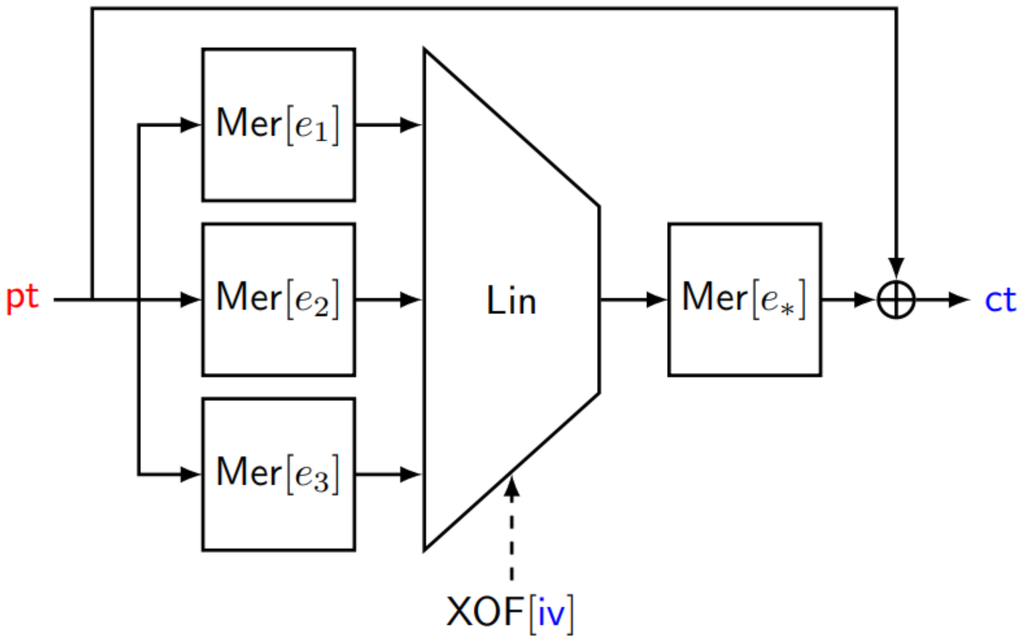
- Mersenne S-box
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
  - Moderate DC/LC resistance
- Repetitive structure
  - Parallel application of S-boxes
  - Feed-forward construction
  - Fully exploit the BN++ optimizations
  - Locally-computable output share
- Randomized structure
  - Affine layer is generated from XOF

Cryptanalytic Scenario



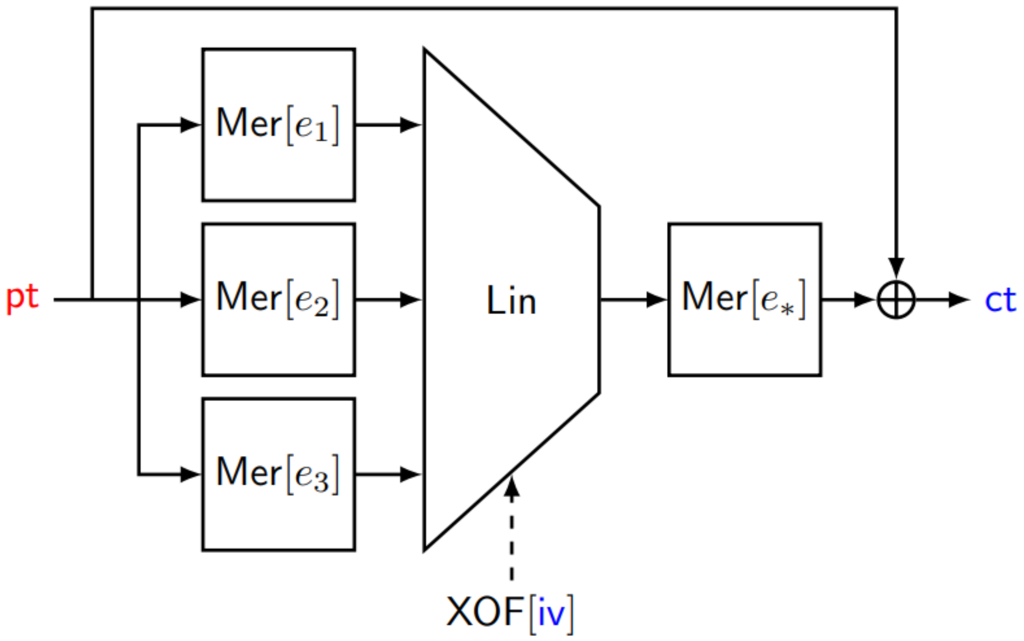
- Single-user setting
  - For a random $(\mathrm{pt}, \mathrm{iv}) \in \mathbb{F}_{2^n} \times \{0,1\}^n$, a single pair $(\mathrm{iv}, \mathrm{ct})$ is given
  - Finding $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that $\mathrm{AIM}[\mathrm{iv}](\mathrm{pt}^*) = \mathrm{ct}$

Cryptanalytic Scenario



- Single-user setting
  - For a random $(\mathrm{pt}, \mathrm{iv}) \in \mathbb{F}_{2^n} \times \{0,1\}^n$, a single pair $(\mathrm{iv}, \mathrm{ct})$ is given
  - Finding $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that $\mathrm{AIM}[\mathrm{iv}](\mathrm{pt}^*) = \mathrm{ct}$
- Multi-user setting
  - For random pairs $(\mathrm{pt}_i, \mathrm{iv}_i) \in \mathbb{F}_{2^n} \times \{0,1\}^n$, multiple pairs $(\mathrm{iv}_i, \mathrm{ct}_i)$ are given
  - Finding $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that $\mathrm{AIM}[\mathrm{iv}_i](\mathrm{pt}^*) = \mathrm{ct}_i$ for some i

Cryptanalytic Scenario



- Single-user setting
  - For a random $(\mathrm{pt}, \mathrm{iv}) \in \mathbb{F}_{2^n} \times \{0,1\}^n$, a single pair $(\mathrm{iv}, \mathrm{ct})$ is given
  - Finding $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that $\mathrm{AIM}[\mathrm{iv}](\mathrm{pt}^*) = \mathrm{ct}$
- Multi-user setting
  - For random pairs $(\mathrm{pt}_i, \mathrm{iv}_i) \in \mathbb{F}_{2^n} \times \{0,1\}^n$, multiple pairs $(\mathrm{iv}_i, \mathrm{ct}_i)$ are given
  - Finding $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that $\mathrm{AIM}[\mathrm{iv}_i](\mathrm{pt}^*) = \mathrm{ct}_i$ for some i
- IV misuse setting
  - For some chosen iv_i, multiple pairs $(\mathrm{iv}_i, \mathrm{ct}_i)$ are given
  - Finding $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that $\mathrm{AIM}[\mathrm{iv}_i](\mathrm{pt}^*) = \mathrm{ct}_i$ for some i
  - Expected to be birthday-bound secure

(General) Cryptanalytic Results

Attack	Log of Complexity			Remark
	AIM-I	AIM-III	AIM-V	
Brute-force	149	214.4	280	Gate-count
Algebraic	137.3	194.1	260.1	Details in the next slide
LC	240	360	496	**Impossible**
DC	125	187	253	**Impossible**
Quantum	159.8	225.2	291.7	Depth * Complexity
Provable security	126.4	190.4	254.4	Everywhere preimage resistance in the random permutation model

(Algebraic) Cryptanalytic Results

Scheme	#Var	(#Eqs, Deg)	Grobner Basis		XL		Dinur's Algorithm	
			Deg. of reg.	Time	D	Time	Time	Memory
AIM-I	n	$(3n, 10)$	51	300.8	52	244.8	**137.3**	138.3
	$2n$	$(3n, 2) + (3n, 4)$	22	**214.9**	14	150.4	248.3	253.7
	$3n$	$(9n, 2)$	20	222.8	12	**148.0**	330.1	346.3
AIM-III	n	$(3n, 14)$	82	474.0	84	375.3	**202.1**	203.3
	$2n$	$(3n, 2) + (3n, 6)$	31	**310.6**	18	203.0	377.5	382.9
	$3n$	$(9n, 2)$	27	310.8	15	**194.1**	487.7	512.1
AIM-V	n	$(3n, 12)$	100	601.1	101	489.7	**264.1**	265.9
	$2n$	$(3n, 2) + (3n, 8)$	40	**406.2**	26	289.5	506.3	511.7
	$3n$	$(6n, 2) + (3n, 4)$	47	510.4	20	**260.6**	716.1	732.3
	$4n$	$(12n, 2)$	45	530.3	19	266.1	854.4	897.7

Performance Comparison

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Dilithium2	1312	2420	0.10	0.03
Falcon-512	897	690	0.27	0.04
$SPHINCS^{+}$-128s	32	7856	315.74	0.35
$SPHINCS^{+}$-128f	32	17088	16.32	0.97
Picnic1-L1-full	32	30925	1.16	0.91
Picnic3	32	12463	5.83	4.24
Banquet	32	19776	7.09	5.24
$Rainier_3$	32	8544	0.97	0.89
BN++$Rain_3$	32	6432	0.83	0.77
AIMer-L1 (Updated)	32	5904	0.59	0.53
AIMer-L1 (Updated)	32	3840	22.29	21.09

Some Remarks

- Remark
  - We submitted AIMer to KpqC and NIST PQC competition
  - Our homepage: https://aimer-signature.org
  - We are waiting for **third-party analysis**!

- Future work
  - QROM security of AIMer
  - More optimization on BN++

Thank you!

Check out aimer-signature.org

Question?