Post-Quantum Signatures from Legendre and Power Residue PRFs

Xinyu Zhang
Email: Xinyu.Zhang2@monash.edu

2025.07.11

Content

- Legendre and Power Residue PRFs

Content

- Legendre and Power Residue PRFs
- How to Build Signature Schemes based on These PRFs?

Content

- Legendre and Power Residue PRFs
- How to Build Signature Schemes based on These PRFs?
  - Identification Scheme based on MPC-in-the-Head

Content

- Legendre and Power Residue PRFs
- How to Build Signature Schemes based on These PRFs?
  - Identification Scheme based on MPC-in-the-Head
  - SNARK-Friendly Signature and its Applications

Content

- Legendre and Power Residue PRFs
- How to Build Signature Schemes based on These PRFs?
  - Identification Scheme based on MPC-in-the-Head
  - SNARK-Friendly Signature and its Applications
  - Ring Signature

Content

- Legendre and Power Residue PRFs
- How to Build Signature Schemes based on These PRFs?
  - Identification Scheme based on MPC-in-the-Head
  - SNARK-Friendly Signature and its Applications
  - Ring Signature
- Future Works

Content

- Legendre and Power Residue PRFs
- How to Build Signature Schemes based on These PRFs?
  - Identification Scheme based on MPC-in-the-Head
  - SNARK-Friendly Signature and its Applications
  - Ring Signature
- Future Works
- Conclusion

Legendre and Power Residue PRFs

Legendre Symbol

- Given an odd prime p, the Legendre symbol of $a \in \mathbb{Z}_p$ is written as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a = b^2 \bmod p \text{ for some } b \in \mathbb{Z}_p \text{ (i.e., } a \text{ is quadratic residue (QR))} \\ 0 & \text{if } a = 0 \bmod p \\ -1 & \text{otherwise (i.e., } a \text{ is quadratic non-residue (QNR))} \end{cases}$$

Legendre Symbol

- Given an odd prime p, the Legendre symbol of $a \in \mathbb{Z}_p$ is written as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a = b^2 \bmod p \text{ for some } b \in \mathbb{Z}_p \text{ (i.e., } a \text{ is quadratic residue (QR))} \\ 0 & \text{if } a = 0 \bmod p \\ -1 & \text{otherwise (i.e., } a \text{ is quadratic non-residue (QNR))} \end{cases}$$

Euler's Criterion

Let p be an odd prime and $a \in \mathbb{Z}_p^*$, then

1. $a^{(p-1)/2} = \pm 1$
2. If a is QR, then $a^{(p-1)/2} = 1$.
3. If a is QNR, then $a^{(p-1)/2} = -1$.

Legendre Symbol

- Given an odd prime p, the Legendre symbol of $a \in \mathbb{Z}_p$ is written as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a = b^2 \bmod p \text{ for some } b \in \mathbb{Z}_p \text{ (i.e., } a \text{ is quadratic residue (QR))} \\ 0 & \text{if } a = 0 \bmod p \\ -1 & \text{otherwise (i.e., } a \text{ is quadratic non-residue (QNR))} \end{cases}$$

Euler's Criterion

Let p be an odd prime and $a \in \mathbb{Z}_p^*$, then

1. $a^{(p-1)/2} = \pm 1$
2. If a is QR, then $a^{(p-1)/2} = 1$.
3. If a is QNR, then $a^{(p-1)/2} = -1$.

Multiplicative Homomorphism

$$\left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right) = a^{\frac{p-1}{2}} \cdot b^{\frac{p-1}{2}} = (a \cdot b)^{\frac{p-1}{2}} = \left(\frac{ab}{p}\right)$$

Pseudo-Randomness of the Legendre Symbol

- In 1988, Damgård [Dam90] conjectured pseudo-randomness properties of the Legendre symbol sequence

$$\left(\frac{a}{p}\right), \left(\frac{a+1}{p}\right), \left(\frac{a+2}{p}\right), \ldots$$

where a is sampled from $\mathbb{Z}_p^*$ uniformly at random.

Pseudo-Randomness of the Legendre Symbol

- In 1988, Damgård [Dam90] conjectured pseudo-randomness properties of the Legendre symbol sequence

$$\left(\frac{a}{p}\right), \left(\frac{a+1}{p}\right), \left(\frac{a+2}{p}\right), \dots$$

where a is sampled from $\mathbb{Z}_p^*$ uniformly at random.

It follows the Weil bound [Wei48] that the number of occurrences of a fixed pattern of l non-zero Legendre symbols among the integers $1,2,\dots,p-1$ is $\frac{p}{2^l} + O\left(\sqrt{p}\right)$, as $p \to \infty$.

Legendre PRF

Given an input $a \in \mathbb{F}_p$, compute the Legendre PRF as

$$\mathcal{L}_0^2(a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{a}{p}\right)\right)\right\rfloor = \begin{cases} 0 & \text{if } a \text{ is QR mod } p \text{ or } a = 0 \ mod \ p \\ 1 & \text{if } a \text{ is QNR mod } p \end{cases}$$

Legendre PRF

Given an input $a \in \mathbb{F}_p$, compute the Legendre PRF as

$$\mathcal{L}_0^2(a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{a}{p}\right)\right) \right\rfloor = \begin{cases} 0 & \text{if } a \text{ is QR mod } p \text{ or } a = 0 \; mod \; p \\ 1 & \text{if } a \text{ is QNR mod } p \end{cases}$$

Multiplicative Homomorphism

$$\mathcal{L}_0^2(ab) = \mathcal{L}_0^2(a) + \mathcal{L}_0^2(b)$$

Legendre PRF

Given an input $a \in \mathbb{F}_p$, compute the Legendre PRF as

$$\mathcal{L}_0^2(a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{a}{p}\right)\right)\right\rfloor = \begin{cases} 0 & \text{if } a \text{ is QR mod } p \text{ or } a = 0 \; mod \; p \\ 1 & \text{if } a \text{ is QNR mod } p \end{cases}$$

Multiplicative Homomorphism

$$\mathcal{L}_0^2(ab) = \mathcal{L}_0^2(a) + \mathcal{L}_0^2(b)$$

Pseudo-randomness

$\mathcal{L}_0^2(a), \mathcal{L}_0^2(a+1), \ldots$ can be seen as a sequence of pseudo-random bits

Legendre PRF with Hidden Shift

Given a hidden shift $K \in \mathbb{F}_p$ and an input $a \in \mathbb{F}_p$, the Legendre PRF with the hidden shift is defined as

$$\mathcal{L}_K^2(a) = \mathcal{L}_0^2(K + a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{K + a}{p}\right)\right)\right\rfloor$$

Legendre PRF with Hidden Shift

Given a hidden shift $K \in \mathbb{F}_p$ and an input $a \in \mathbb{F}_p$, the Legendre PRF with the hidden shift is defined as

$$\mathcal{L}_K^2(a) = \mathcal{L}_0^2(K + a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{K + a}{p}\right)\right)\right\rfloor$$

Shifted Legendre Symbol (SLS) Problem

Let K be uniformly sampled from $\mathbb{F}_p$ and define O_{Leg} to be an oracle that takes $a \in \mathbb{F}_p$ and outputs $\mathcal{L}_K^2(a)$. Then the SLS problem is to find K, with non-negligible probability.

Legendre PRF with Hidden Shift

Given a hidden shift $K \in \mathbb{F}_p$ and an input $a \in \mathbb{F}_p$, the Legendre PRF with the hidden shift is defined as

$$\mathcal{L}_K^2(a) = \mathcal{L}_0^2(K + a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{K + a}{p}\right)\right)\right\rfloor$$

Shifted Legendre Symbol (SLS) Problem

Let K be uniformly sampled from $\mathbb{F}_p$ and define O_{Leg} to be an oracle that takes $a \in \mathbb{F}_p$ and outputs $\mathcal{L}_K^2(a)$. Then the SLS problem is to find K, with non-negligible probability.

Is SLS problem secure against quantum attacks?

Legendre PRF with Hidden Shift

If the adversary can query O_{Leg} in a **superposition**, then NO…

There exists a polynomial time quantum algorithm that finds the hidden shift K with just a single query [vDH00].

Legendre PRF with Hidden Shift

If the adversary can query O_{Leg} in a **superposition**, then NO…

There exists a polynomial time quantum algorithm that finds the hidden shift K with just a single query [vDH00].

If the adversary can only query O_{Leg} **classically**, the attack is still **exponential** in the security parameter.

e.g., [Kho19] shows a memoryless collision search algorithm with $O\left(\sqrt{p}\log p\right)$ Legendre symbol evaluations with $\sqrt{p}\log p$ queries to O_{Leg}.

Legendre PRF based Public Key System

Let the hidden shift $K \in \mathbb{F}_p$ be a secret key, for a public list $\mathcal{I} = (I_1, \dots, I_L) \in_R \mathbb{F}_p^L$, define the Legendre-based public key as:

$$pk = (pk_1, \dots, pk_L) \in \mathbb{Z}_2^L : pk_i = \mathcal{L}_K^2(I_i) \text{ for } i \in [L].$$

Legendre PRF based Public Key System

Let the hidden shift $K \in \mathbb{F}_p$ be a secret key, for a public list $\mathcal{I} = (I_1, \dots, I_L) \in_R \mathbb{F}_p^L$, define the Legendre-based public key as:

$$pk = (pk_1, \dots, pk_L) \in \mathbb{Z}_2^L : pk_i = \mathcal{L}_K^2(I_i) \text{ for } i \in [L].$$

L must be sufficiently large such that the sequence can uniquely identify the hidden shift K.

Legendre PRF based Public Key System

Let the hidden shift $K \in \mathbb{F}_p$ be a secret key, for a public list $\mathcal{I} = (I_1, \dots, I_L) \in_R \mathbb{F}_p^L$, define the Legendre-based public key as:

$$pk = (pk_1, \dots, pk_L) \in \mathbb{Z}_2^L : \; pk_i = \mathcal{L}_K^2(I_i) \text{ for } i \in [L].$$

L must be sufficiently large such that the sequence can uniquely identify the hidden shift K.

Legendre-based Key Recovery Problem

Let K be uniformly sampled from $\mathbb{F}_p$ and given L random uncontrollable PRF evaluations (i.e., public key) to the function $\mathcal{L}_K^2(\cdot): \mathbb{F}_p \mapsto \mathbb{Z}_2$, the Legendre-based key recovery problem is to find K, with non-negligible probability.

Legendre PRF based Public Key System

Let the hidden shift $K \in \mathbb{F}_p$ be a secret key, for a public list $\mathcal{I} = (I_1, \dots, I_L) \in_R \mathbb{F}_p^L$, define the Legendre-based public key as:

$$pk = (pk_1, \dots, pk_L) \in \mathbb{Z}_2^L : \; pk_i = \mathcal{L}_K^2(I_i) \text{ for } i \in [L].$$

This problem is at least as hard as the SLSP, and the adversary loses the ability to query the O_{Leg}.

Legendre-based Key Recovery Problem

Let K be uniformly sampled from $\mathbb{F}_p$ and given L random uncontrollable PRF evaluations (i.e., public key) to the function $\mathcal{L}_K^2(\cdot): \mathbb{F}_p \mapsto \mathbb{Z}_2$, the Legendre-based key recovery problem is to find K, with non-negligible probability.

Power Residue PRF

Key Problem with the Legendre PRF:

- PRF output size is just one bit

Power Residue PRF

Key Problem with the Legendre PRF:

- PRF output size is just one bit

The number of symbols in each public key is very large ($L = 2^{15}$ for signature schemes based on the Legendre PRF with 128-bit security)

Power Residue PRF

Key Problem with the Legendre PRF:

- PRF output size is just one bit

The number of symbols in each public key is very large ($L = 2^{15}$ for signature schemes based on the Legendre PRF with 128-bit security)

Generalize the notion: t-th power residue symbol, where for $a \in \mathbb{F}_p$ and $t \in \mathbb{Z}$ with $t|p-1$,

$$\left(\frac{a}{p}\right)_t = a^{\frac{p-1}{t}} mod\ p \in \mathbb{Z}_t$$

Power Residue PRF

Key Problem with the Legendre PRF:

- PRF output size is just one bit

The number of symbols in each public key is very large ($L = 2^{15}$ for signature schemes based on the Legendre PRF with 128-bit security)

Generalize the notion: t-th power residue symbol, where for $a \in \mathbb{F}_p$ and $t \in \mathbb{Z}$ with $t|p-1$,

$$\left(\frac{a}{p}\right)_t = a^{\frac{p-1}{t}} mod\ p \in \mathbb{Z}_t$$

The multiplicative homomorphism also exists in t-th power residue (i.e., $\mathcal{L}_0^t(ab) = \mathcal{L}_0^t(a) + \mathcal{L}_0^t(b)$)

Power Residue PRF

Key Problem with the Legendre PRF:

- PRF output size is just one bit

The number of symbols in each public key is very large ($L = 2^{15}$ for signature schemes based on the Legendre PRF with 128-bit security)

Generalize the notion: t-th power residue symbol, where for $a \in \mathbb{F}_p$ and $t \in \mathbb{Z}$ with $t|p-1$,

$$\left(\frac{a}{p}\right)_t = a^{\frac{p-1}{t}} mod\ p \in \mathbb{Z}_t$$

Keyed Power Residue PRF

$$\mathcal{L}_K^t(a) = \begin{cases} i & \text{If } (a+K)/g^i \equiv h^t\ mod\ p \text{ for some } h \in \mathbb{F}_p^* \\ 0 & \text{If } (a+K) \equiv 0\ mod\ p \end{cases}$$

where g is a fixed generator of $\mathbb{F}_p^*$

The multiplicative homomorphism also exists in t-th power residue (i.e., $\mathcal{L}_0^t(ab) = \mathcal{L}_0^t(a) + \mathcal{L}_0^t(b)$)

Signature Schemes Based on Legendre and Power Residue PRFs

MONASH University

Identification Scheme

Initial Idea:

- Given $pk = (pk_1, \dots, pk_L) = \left(\mathcal{L}_K^t(I_1), \dots, \mathcal{L}_K^t(I_L)\right)$, applying a generic zero-knowledge proof (e.g., MPC-in-the-head), to prove for all $i \in [L]$:

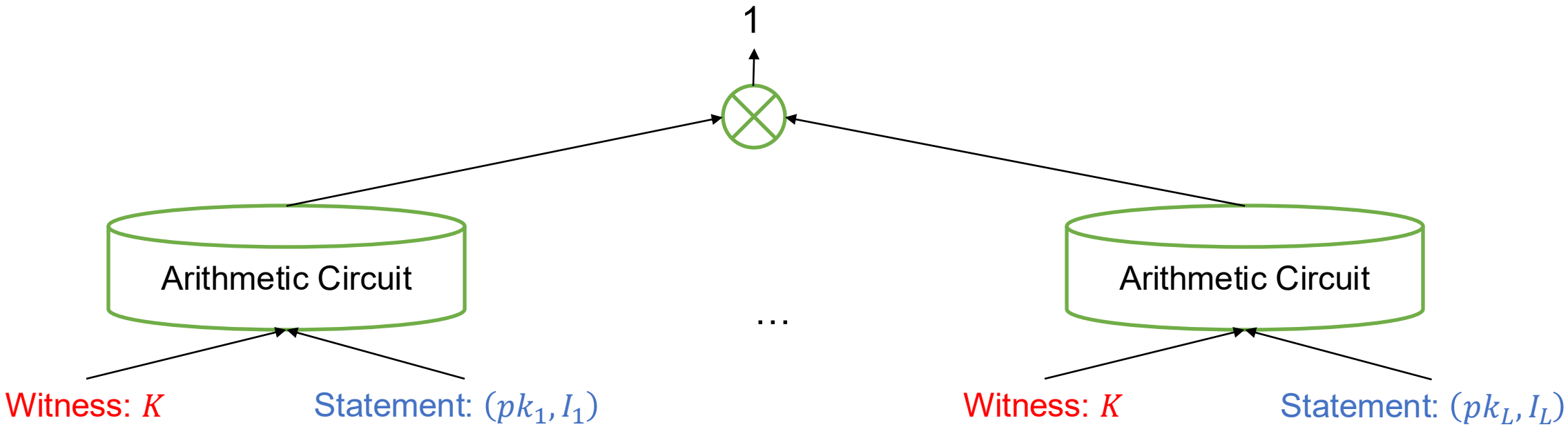
$$pk_i = (K + I_i)^{\frac{p-1}{t}} \; mod \; p$$

Identification Scheme

Initial Idea:

- Given $pk = (pk_1, \ldots, pk_L) = \left(\mathcal{L}_K^t(I_1), \ldots, \mathcal{L}_K^t(I_L)\right)$, applying a generic zero-knowledge proof (e.g., MPC-in-the-head), to prove for all $i \in [L]$:

$$pk_i = (K + I_i)^{\frac{p-1}{t}} \; mod \; p$$

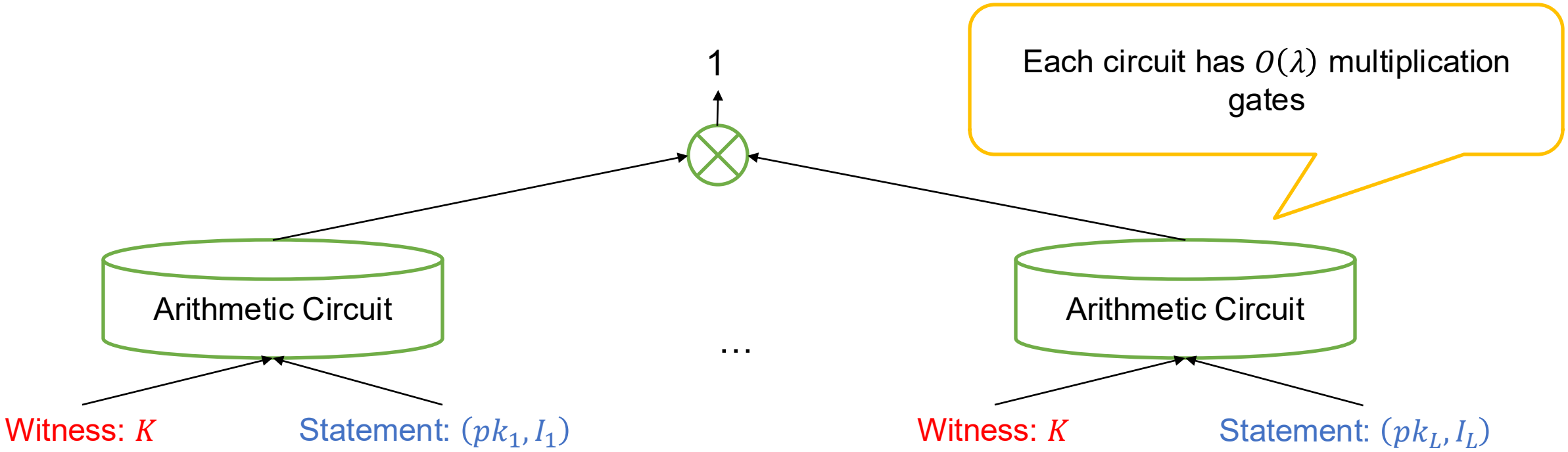


Identification Scheme

Initial Idea:

- Given $pk = (pk_1, \dots, pk_L) = \left(\mathcal{L}_K^t(I_1), \dots, \mathcal{L}_K^t(I_L)\right)$, applying a generic zero-knowledge proof (e.g., MPC-in-the-head), to prove for all $i \in [L]$:

$$pk_i = (K + I_i)^{\frac{p-1}{t}} \; mod \; p$$

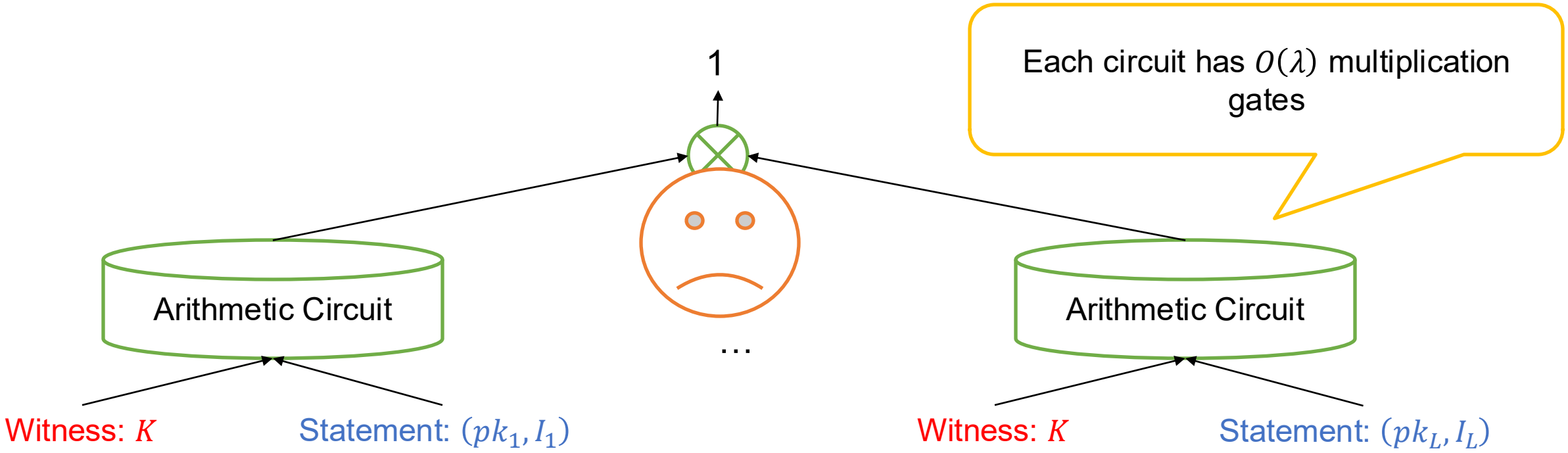


Identification Scheme

Initial Idea:

- Given $pk = (pk_1, \dots, pk_L) = \left(\mathcal{L}_K^t(I_1), \dots, \mathcal{L}_K^t(I_L)\right)$, applying a generic zero-knowledge proof (e.g., MPC-in-the-head), to prove for all $i \in [L]$:

$$pk_i = (K + I_i)^{\frac{p-1}{t}} \; mod \; p$$



Identification Scheme

Improvement based on [GRR+16]

- Based on the multiplicative homomorphism of Legendre and power residue PRFs, instead of computing $pk_i = (K + I_i)^{\frac{p-1}{t}} \, mod \, p$ in the circuit, we prove for all $i \in [L]$:

$$o_i = (K + I_i)r_i \text{ for some random } r_i \in_R \mathbb{F}_p$$

Identification Scheme

Improvement based on [GRR+16]

- Based on the multiplicative homomorphism of Legendre and power residue PRFs, instead of computing $pk_i = (K + I_i)^{\frac{p-1}{t}} \, mod \, p$ in the circuit, we prove for all $i \in [L]$:

$$o_i = (K + I_i)r_i \text{ for some random } r_i \in_R \mathbb{F}_p$$

- Then, the prover sends $\left(\left(\mathcal{L}_0^t(r_1), \ldots, \mathcal{L}_0^t(r_L)\right), (o_1, \ldots, o_L)\right)$ in the clear, and the verifier can check

$$\mathcal{L}_0^t(o_i) = \mathcal{L}_0^t\big((K + I_i)r_i\big) = \mathcal{L}_0^t(K + I_i) + \mathcal{L}_0^t(r_i) = pk_i + \mathcal{L}_0^t(r_i)$$

$$pk = (pk_i)_{i\in[L]} = \left(\mathcal{L}_K^t(I_i)\right)_{i\in[L]} = \left(\mathcal{L}_0^t(K + I_i)\right)_{i\in[L]}$$

Identification Scheme

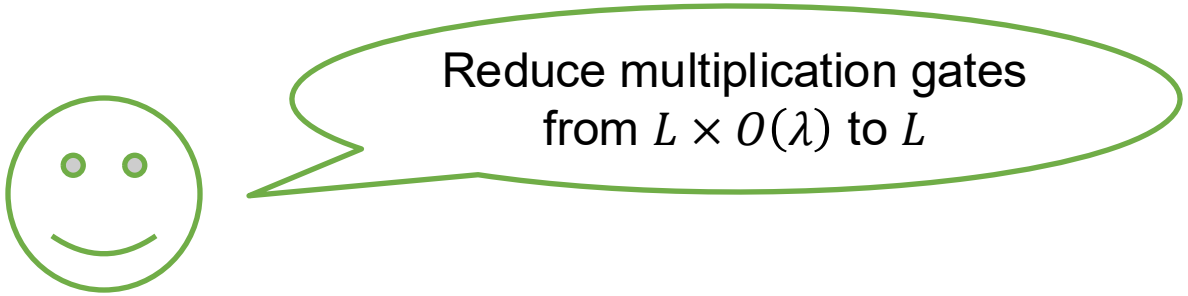
Improvement based on [GRR+16]

- Based on the multiplicative homomorphism of Legendre and power residue PRFs, instead of computing $pk_i = (K + I_i)^{\frac{p-1}{t}} \, mod \, p$ in the circuit, we prove for all $i \in [L]$:

$$o_i = (K + I_i)r_i \text{ for some random } r_i \in_R \mathbb{F}_p$$

- Then, the prover sends $\left(\left(\mathcal{L}_0^t(r_1), \ldots, \mathcal{L}_0^t(r_L)\right), (o_1, \ldots, o_L)\right)$ in the clear, and the verifier can check

$$\mathcal{L}_0^t(o_i) = \mathcal{L}_0^t\big((K + I_i)r_i\big) = \mathcal{L}_0^t(K + I_i) + \mathcal{L}_0^t(r_i) = pk_i + \mathcal{L}_0^t(r_i)$$



Identification Scheme

Improvement based on [GRR+16]

- Based on the multiplicative homomorphism of Legendre and power residue PRFs, instead of computing $pk_i = (K + I_i)^{\frac{p-1}{t}} \, mod \, p$ in the circuit, we prove for all $i \in [L]$:

$$o_i = (K + I_i)r_i \text{ for some random } r_i \in_R \mathbb{F}_p$$

- Then, the prover sends $\left(\left(\mathcal{L}_0^t(r_1), \ldots, \mathcal{L}_0^t(r_L)\right), (o_1, \ldots, o_L)\right)$ in the clear, and the verifier can check

$$\mathcal{L}_0^t(o_i) = \mathcal{L}_0^t\left((K + I_i)r_i\right) = \mathcal{L}_0^t(K + I_i) + \mathcal{L}_0^t(r_i) = pk_i + \mathcal{L}_0^t(r_i)$$

Reduce multiplication gates from $L \times O(\lambda)$ to L

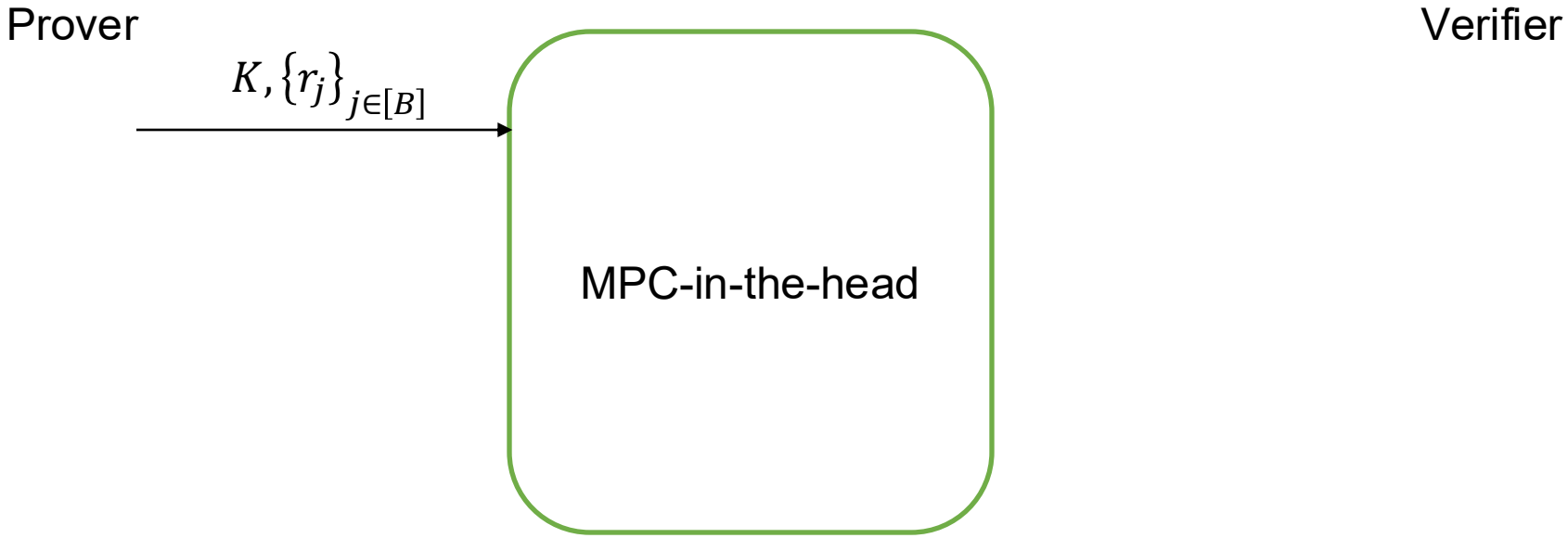
But L is still large…

LegRoast [BG20]: MPCitH-based Signature

- Instead of proving all L public key symbols, it suffices to prove a random subset of these symbols.
- Let $B \leq L$ be an integer

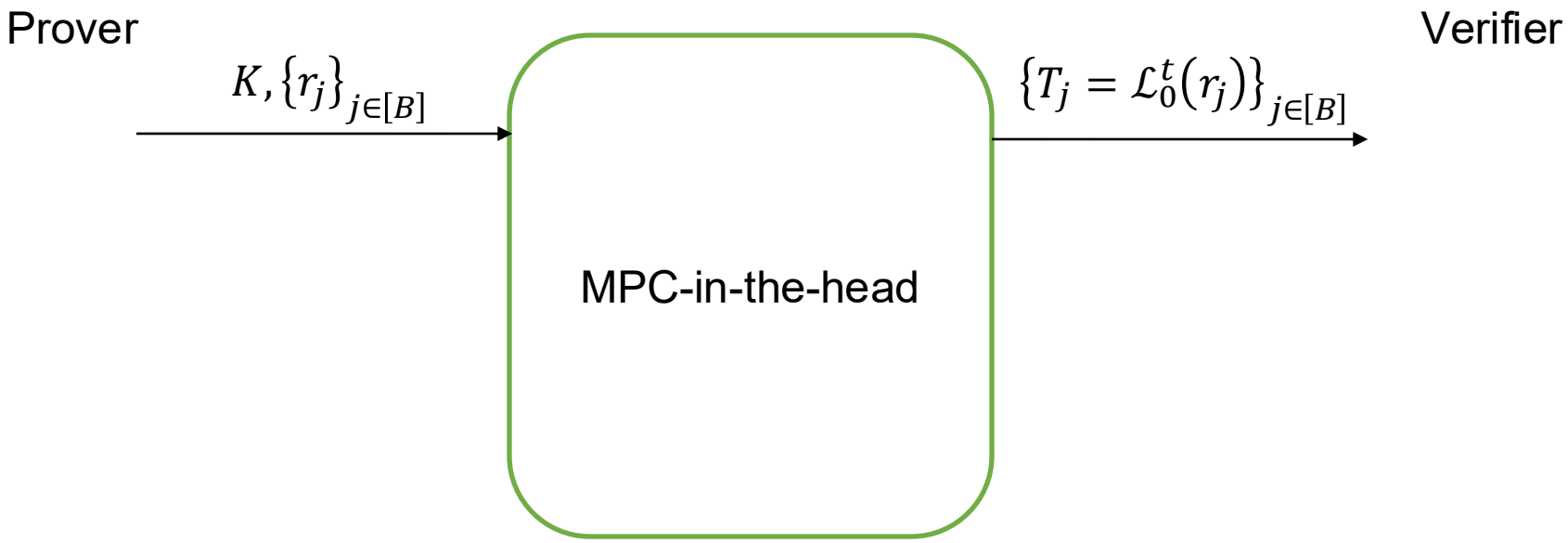
LegRoast [BG20]: MPCitH-based Signature

- Instead of proving all L public key symbols, it suffices to prove a random subset of these symbols.
- Let $B \leq L$ be an integer



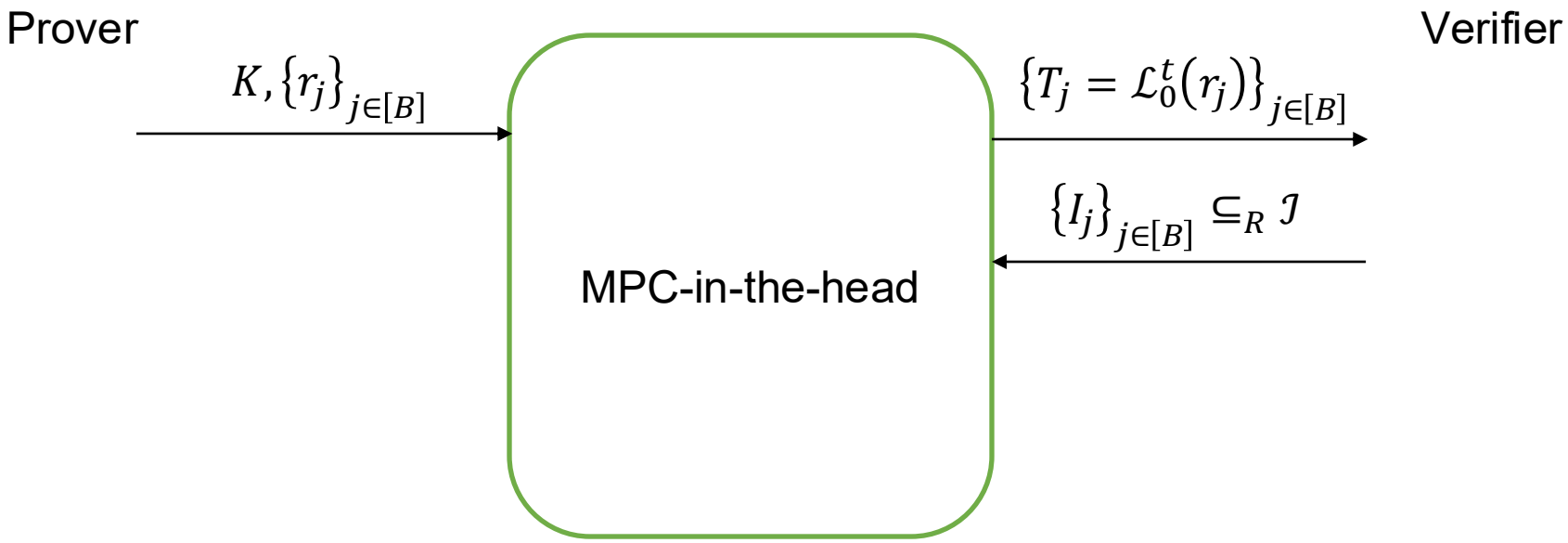
LegRoast [BG20]: MPCitH-based Signature

- Instead of proving all L public key symbols, it suffices to prove a random subset of these symbols.
- Let $B \leq L$ be an integer



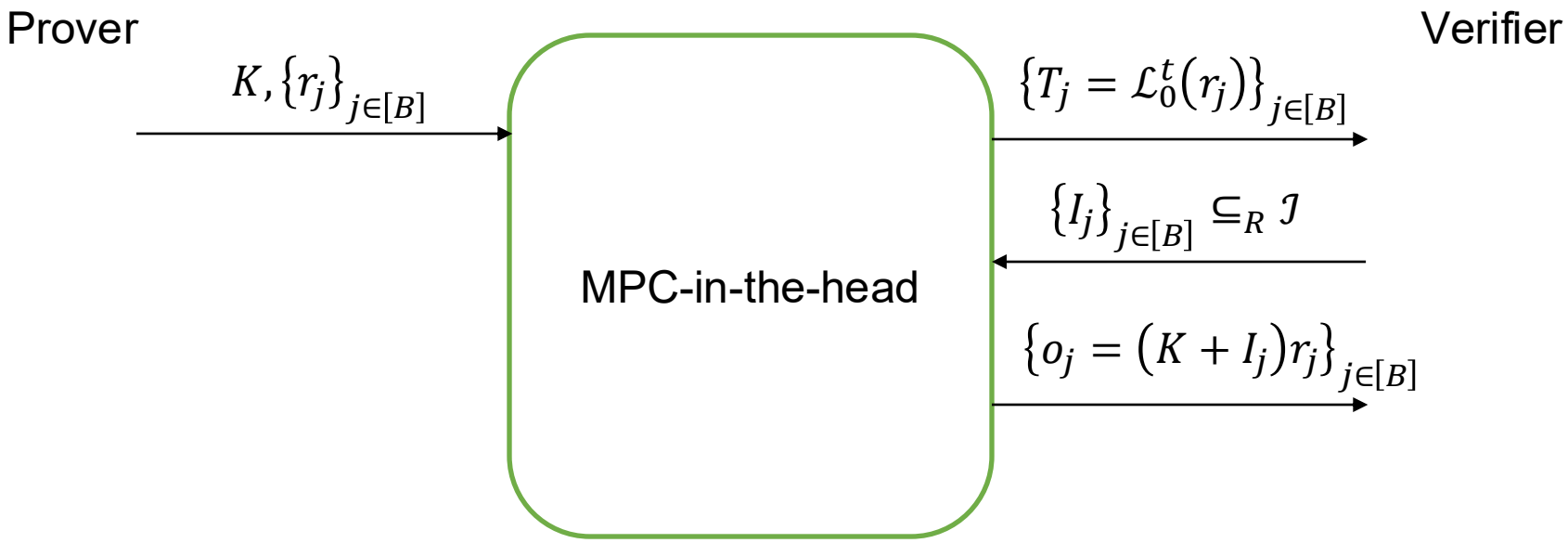
LegRoast [BG20]: MPCitH-based Signature

- Instead of proving all L public key symbols, it suffices to prove a random subset of these symbols.
- Let $B \leq L$ be an integer



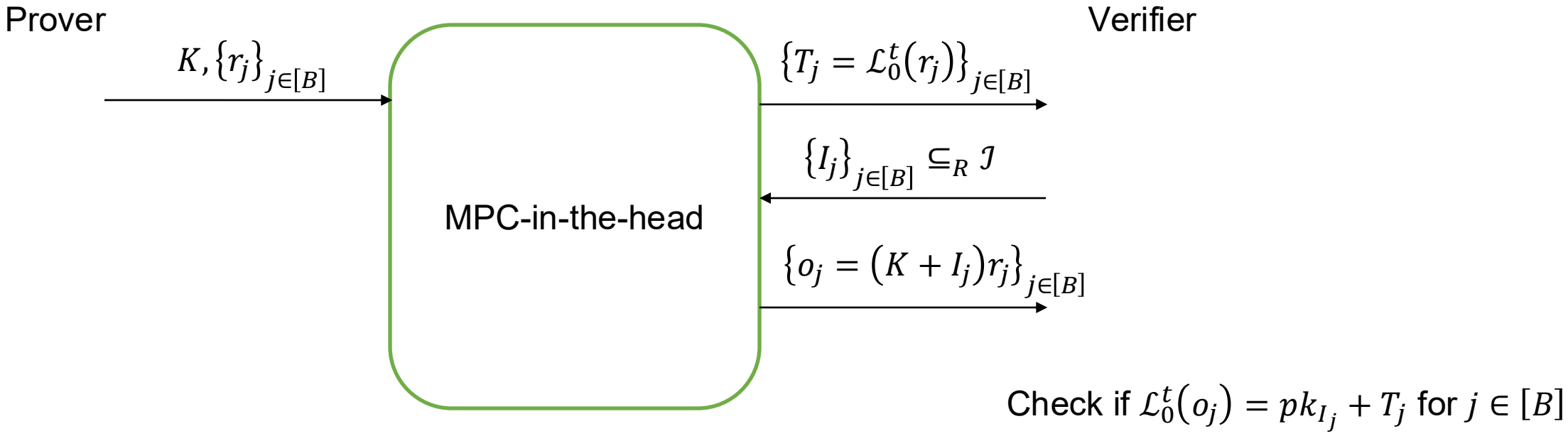
LegRoast [BG20]: MPCitH-based Signature

- Instead of proving all L public key symbols, it suffices to prove a random subset of these symbols.
- Let $B \leq L$ be an integer



LegRoast [BG20]: MPCitH-based Signature

- Instead of proving all L public key symbols, it suffices to prove a random subset of these symbols.
- Let $B \leq L$ be an integer



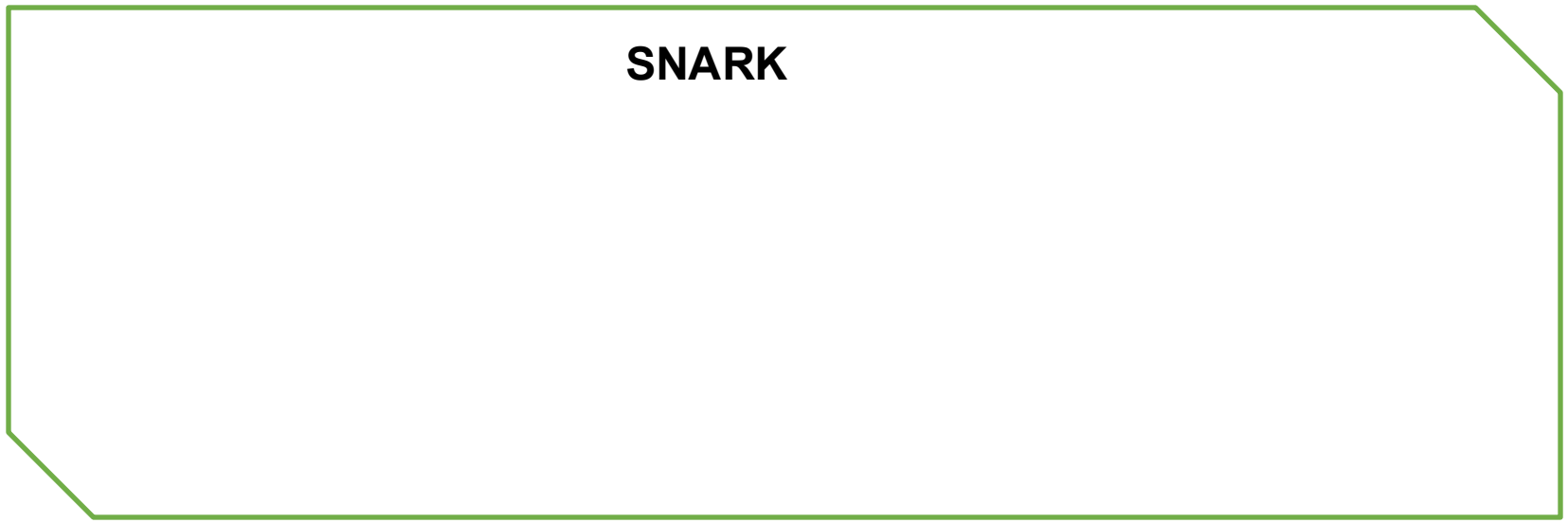
Check if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$ for $j \in [B]$

SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?

SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?



SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?

SNARK

- **Succinct**: Proof size (and verification) is sublinear in the witness size.

SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?

SNARK

- **Succinct**: Proof size (and verification) is sublinear in the witness size.
- **Non-interactive**: Prover transmits a single message (the proof) to the verifier

SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?

SNARK

- **Succinct**: Proof size (and verification) is sublinear in the witness size.
- **Non-interactive**: Prover transmits a single message (the proof) to the verifier
- **Argument of Knowledge**: A valid proof convinces the verifier that a *computationally bounded* prover knows the non-deterministic witness.

SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?

Program

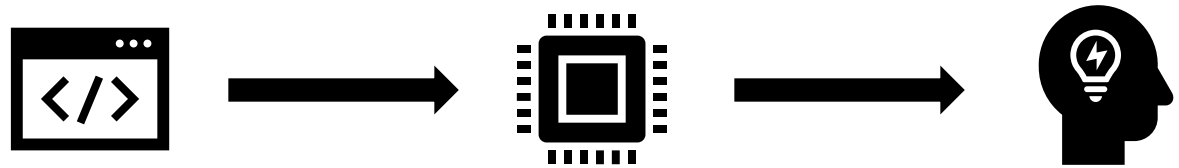
SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?

Program Circuit

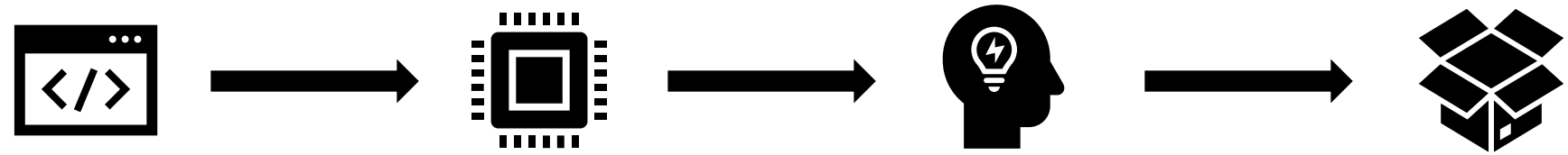
SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?



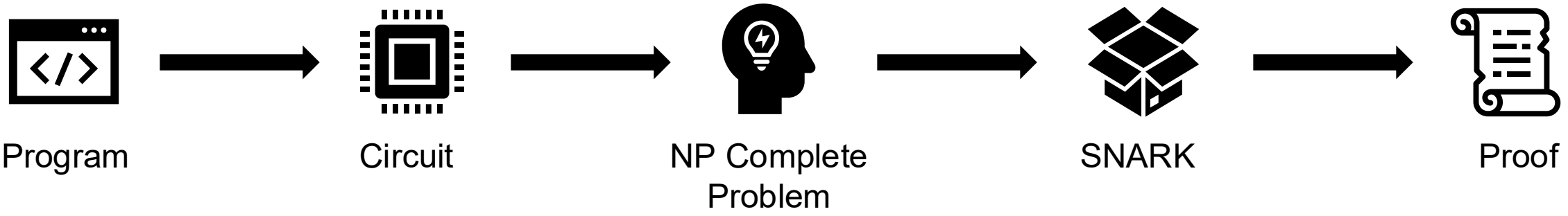
SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?



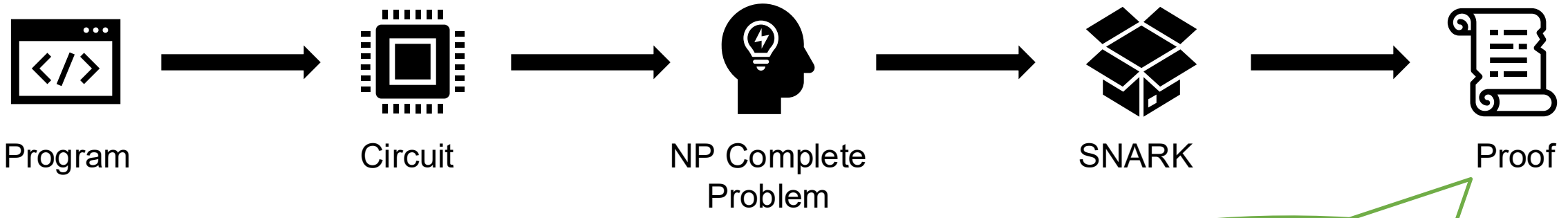
SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?



SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?



SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?

Signature
Verification

SNARK-Friendly Signature [ZSE+24]

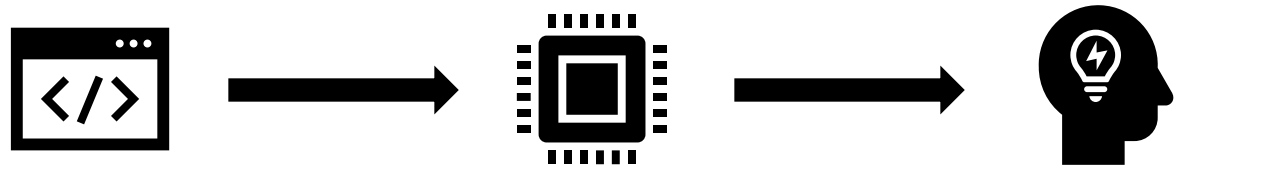
What does it mean by “SNARK-friendly signature”?

Signature Verification

Arithmetic Circuit

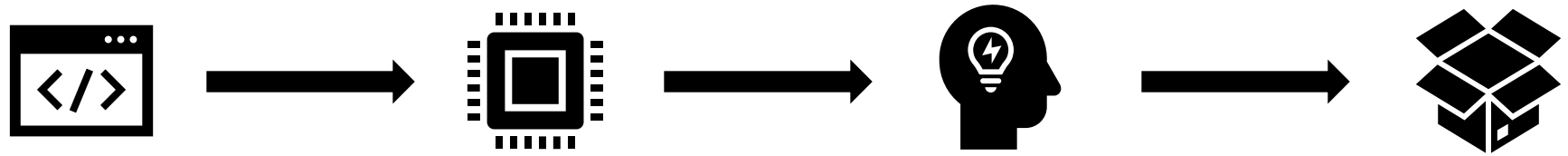
SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?



SNARK-Friendly Signature [ZSE+24]

What does it mean by “SNARK-friendly signature”?

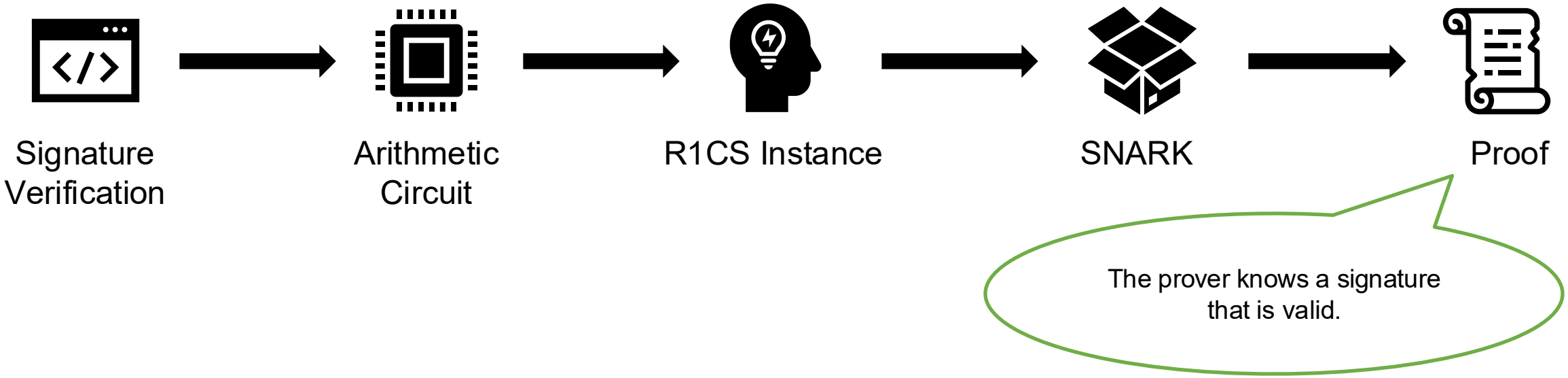


SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?

SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?



SNARK-Friendly Signature [ZSE+24]

What does it mean by "SNARK-friendly signature"?

The prover knows a signature that is valid.

SNARK-Friendly Signature
The verification algorithm of the signature scheme can be translated to a relatively small circuit

SNARK-Friendly Signature [ZSE+24]

SK-based Post-Quantum Signature are Normally NOT SNARK-Friendly

Hash-Tree

MPC-in-the-Head

SNARK-Friendly Signature [ZSE+24]

SK-based Post-Quantum Signature are Normally NOT SNARK-Friendly

Hash-Tree

- Circuit of a standard hash function is prohibitively large → algebraic hash
- A huge slow down in key gen and sign due to algebraic hash

MPC-in-the-Head

SNARK-Friendly Signature [ZSE+24]

SK-based Post-Quantum Signature are Normally NOT SNARK-Friendly

Hash-Tree

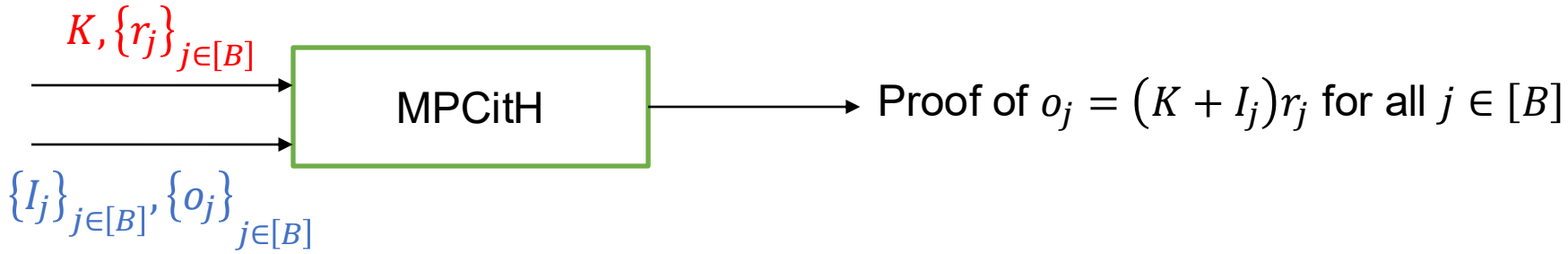
- Circuit of a standard hash function is prohibitively large → algebraic hash
- A huge slow down in key gen and sign due to algebraic hash

MPC-in-the-Head

- Circuit of a standard hash function is prohibitively large → algebraic hash
- Slightly slower performances
- Still prohibitively large verification circuit due to “non-succinctness” of MPCitH

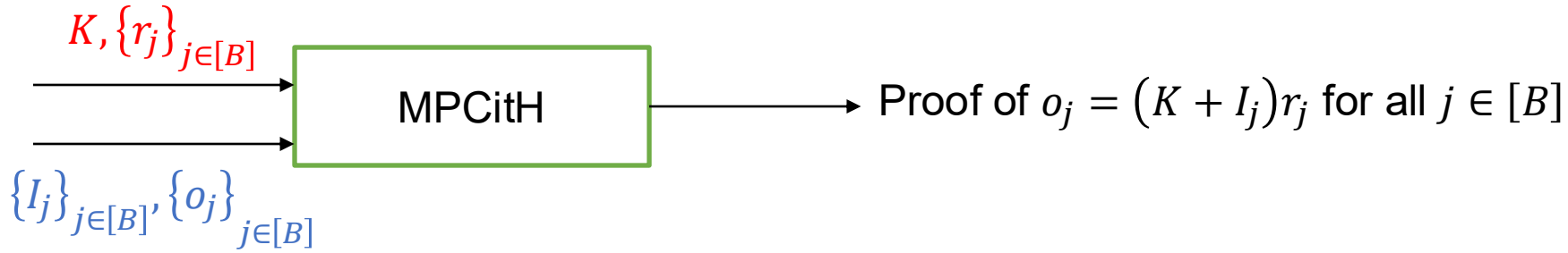
SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck



SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

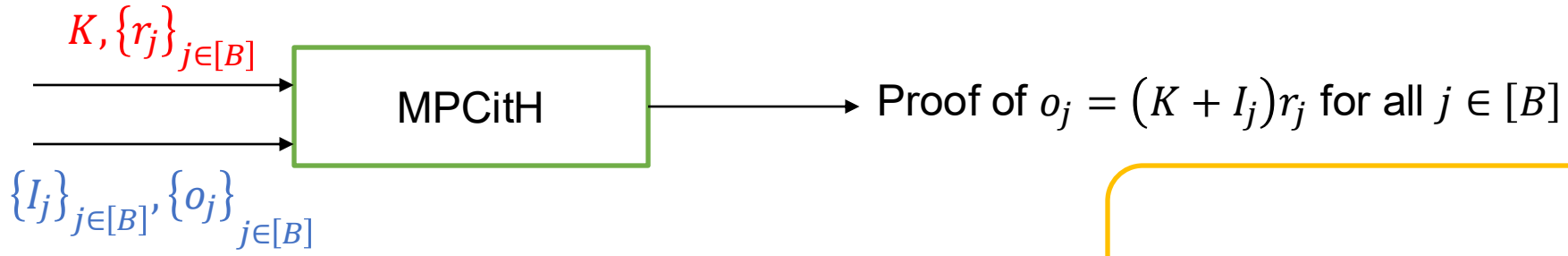


Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck



This is actually a polynomial commitment!

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.

But SK-based polynomial commitment is expensive, and we need to compute B of them!

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.
- From linear univariate polynomial to linear multivariate polynomial:

$$\hat{c}(x_1, \ldots, x_B) = Kr_1 + r_1x_1 + \cdots + Kr_B + r_Bx_B$$

Is this enough?

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.
- From linear univariate polynomial to linear multivariate polynomial:

$$\hat{c}(x_1, \ldots, x_B) = Kr_1 + r_1x_1 + \cdots + Kr_B + r_Bx_B$$

The polynomial has constant term $\sum_{j\in[B]} Kr_j$ instead of individual Kr_j coefficients!

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.
- From linear univariate polynomial to linear multivariate polynomial:

$$\hat{c}(x_1, \dots, x_B) = Kr_1 + r_1x_1 + \cdots + Kr_B + r_Bx_B$$

- From linear multivariate polynomial to quadratic multivariate polynomial:

$$\hat{c}(x_1, \dots, x_B, y_1, \dots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Observation

- Before learning $\{I_j\}_{j\in[B]}$, $\left(K, \{r_j\}_{j\in[B]}\right)$ are committed by the prover
- View I_j as an unknown, and (K, r_j) as coefficients, we can construct a linear univariate polynomial $\hat{o}_j(x) = Kr_j + xr_j$.
- From linear univariate polynomial to linear multivariate polynomial:

$$\hat{c}(x_1, \dots, x_B) = Kr_1 + r_1x_1 + \cdots + Kr_B + r_Bx_B$$

- From linear multivariate polynomial to quadratic multivariate polynomial:

$$\hat{c}(x_1, \dots, x_B, y_1, \dots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

$$\hat{c}(I_1, \dots, I_B, \lambda_1, \dots, \lambda_B) = Kr_1\lambda_1 + r_1I_1\lambda_1 + \cdots + Kr_B\lambda_B + r_BI_B\lambda_B = \sum_{j\in[B]} \lambda_j(K + I_j)r_j = \sum_{j\in[B]} \lambda_j o_j$$

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

A signature based on multivariate polynomial commitment:

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

A signature based on multivariate polynomial commitment:

- Prover commits to multivariate polynomial coefficients (i.e., $(Kr_1, r_1, \ldots, Kr_B, r_B)$)
- Sending the commitment to the verifier

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

A signature based on multivariate polynomial commitment:

- Prover commits to multivariate polynomial coefficients (i.e., $(Kr_1, r_1, \ldots, Kr_B, r_B)$)
- Sending the commitment to the verifier
- Verifier sends the challenges $\{I_j\}_{j\in[B]}$.
- Prover sends $\{o_j\}_{j\in[B]}$.

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

A signature based on multivariate polynomial commitment:

- Prover commits to multivariate polynomial coefficients (i.e., $(Kr_1, r_1, \ldots, Kr_B, r_B)$)
- Sending the commitment to the verifier
- Verifier sends the challenges $\{I_j\}_{j\in[B]}$.
- Prover sends $\{o_j\}_{j\in[B]}$.
- Verifier sends scalar challenges $\{\lambda_j\}_{j\in[B]}$.
- Prover proves the committed multivariate polynomial $\sum_{j\in[B]} \lambda_j o_j = \hat{c}(I_1, \ldots, I_B, \lambda_1, \ldots, \lambda_B)$.

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

From multivariate polynomial commitment to univariate sumcheck [ZXZS20]:

- The evaluation of $\hat{c}(I_1, \ldots, I_B, \lambda_1, \ldots, \lambda_B)$ can be written as an inner-product between two vectors:
  - Coefficient vector: $(Kr_1, r_1, \ldots, Kr_B, r_B)$
  - Evaluation vector: $(\lambda_1, I_1\lambda_1, \ldots, \lambda_B, I_B\lambda_B)$

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

From multivariate polynomial commitment to univariate sumcheck [ZXZS20]:

- The evaluation of $\hat{c}(I_1, \ldots, I_B, \lambda_1, \ldots, \lambda_B)$ can be written as an inner-product between two vectors:
  - Coefficient vector: $(Kr_1, r_1, \ldots, Kr_B, r_B)$
  - Evaluation vector: $(\lambda_1, I_1\lambda_1, \ldots, \lambda_B, I_B\lambda_B)$
- Each vector can be used to compute a univariate polynomial based on some pre-defined set H with $|H| = 2B$.
  - Coefficient polynomial: $\hat{f} \coloneqq Interpolate\big(H, (Kr_1, r_1, \ldots, Kr_B, r_B)\big)$
  - Evaluation polynomial: $\hat{q} \coloneqq Interpolate\big(H, (\lambda_1, I_1\lambda_1, \ldots, \lambda_B, I_B\lambda_B)\big)$

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

From multivariate polynomial commitment to univariate sumcheck [ZXZS20]:

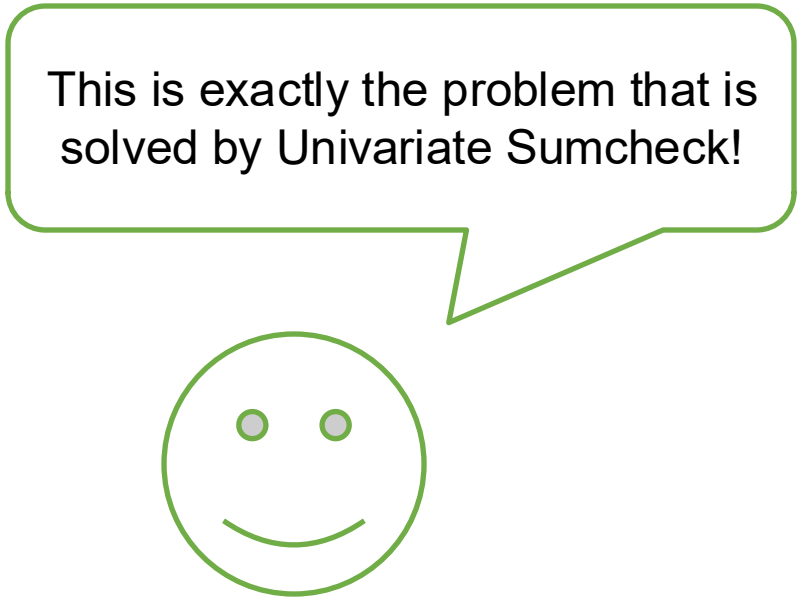
- The evaluation of $\hat{c}(I_1, \ldots, I_B, \lambda_1, \ldots, \lambda_B)$ can be written as an inner-product between two vectors:
  - Coefficient vector: $(Kr_1, r_1, \ldots, Kr_B, r_B)$
  - Evaluation vector: $(\lambda_1, I_1\lambda_1, \ldots, \lambda_B, I_B\lambda_B)$
- Each vector can be used to compute a univariate polynomial based on some pre-defined set H with $|H| = 2B$.
  - Coefficient polynomial: $\hat{f} \coloneqq Interpolate\big(H, (Kr_1, r_1, \ldots, Kr_B, r_B)\big)$
  - Evaluation polynomial: $\hat{q} \coloneqq Interpolate\big(H, (\lambda_1, I_1\lambda_1, \ldots, \lambda_B, I_B\lambda_B)\big)$
- Checking if $\sum_{j\in[B]} \lambda_j o_j = \hat{c}(I_1, \ldots, I_B, \lambda_1, \ldots, \lambda_B)$ can be transformed to $\sum_{j\in[B]} \lambda_j o_j = \sum_{a\in H} \hat{f}(a) \cdot \hat{q}(a)$

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \ldots, x_B, y_1, \ldots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

- Checking if $\sum_{j\in[B]} \lambda_j o_j = \hat{c}(I_1, \ldots, I_B, \lambda_1, \ldots, \lambda_B)$ can be transformed to $\sum_{j\in[B]} \lambda_j o_j = \sum_{a\in H} \hat{f}(a) \cdot \hat{q}(a)$



SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \dots, x_B, y_1, \dots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

- Checking if $\sum_{j\in[B]} \lambda_j o_j = \hat{c}(I_1, \dots, I_B, \lambda_1, \dots, \lambda_B)$ can be transformed to $\sum_{j\in[B]} \lambda_j o_j = \sum_{a\in H} \hat{f}(a) \cdot \hat{q}(a)$

Univariate Sumcheck

Given a finite field $\mathbb{F}$, a pre-defined coset H, a degree $d \in \mathbb{N}$, a committed univariate polynomial $\hat{h}$, and a claimed sum $\mu \in \mathbb{F}$, univariate sumcheck is a protocol that convinces the verifier:

- $\mu = \sum_{a\in H} \hat{h}(a)$
- $\deg \hat{h} \leq d$

This is exactly the problem that is solved by Univariate Sumcheck!

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

$$\hat{c}(x_1, \dots, x_B, y_1, \dots, y_B) = Kr_1y_1 + r_1x_1y_1 + \cdots + Kr_By_B + r_Bx_By_B$$

- Checking if $\sum_{j\in[B]} \lambda_j o_j = \hat{c}(I_1, \dots, I_B, \lambda_1, \dots, \lambda_B)$ can be transformed to $\sum_{j\in[B]} \lambda_j o_j = \sum_{a\in H} \hat{f}(a) \cdot \hat{q}(a)$

Univariate Sumcheck

Given a finite field $\mathbb{F}$, a pre-defined coset H, a degree $d \in \mathbb{N}$, a committed univariate polynomial $\hat{h}$, and a claimed sum $\mu \in \mathbb{F}$, univariate sumcheck is a protocol that convinces the verifier:

- $\mu = \sum_{a\in H} \hat{h}(a)$
- $\deg \hat{h} \leq d$

The verification of univariate sumcheck proof is succinct!

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Features of Loquat

- Balance SNARK-friendliness and signature performance

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Features of Loquat

- Balance SNARK-friendliness and signature performance
- 57KB signature size (128-bit security)

Larger than MPCitH-based signatures but smaller than hash-based zkSNARKs

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

Key Features of Loquat

- Balance SNARK-friendliness and signature performance
- 57KB signature size (128-bit security)
- 148,825 R1CS constraints for each signature verification

Up to 9× smaller than SPHINCS+'s verification and up to 175× smaller than MPCitH-based signature verification.

SNARK-Friendly Signature [ZSE+24]

Loquat: Legendre and Power Residue PRFs + Univariate Sumcheck

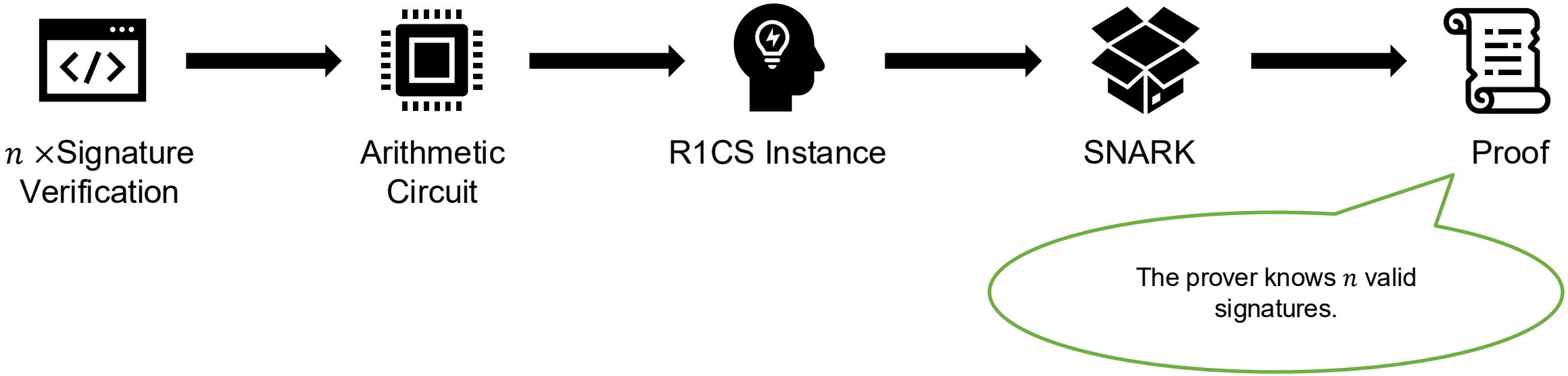
Key Features of Loquat

- Balance SNARK-friendliness and signature performance
- 57KB signature size (128-bit security)
- 148,825 R1CS constraints for each signature verification
- 5.54s sign and 0.21s verify (standard hash + Python)

105s sign and 11s verify (algebraic hash + Python)

SNARK-Friendly Signature [ZSE+24]

Applications of SNARK-Friendly Signatures: SNARK-based Aggregate Signature



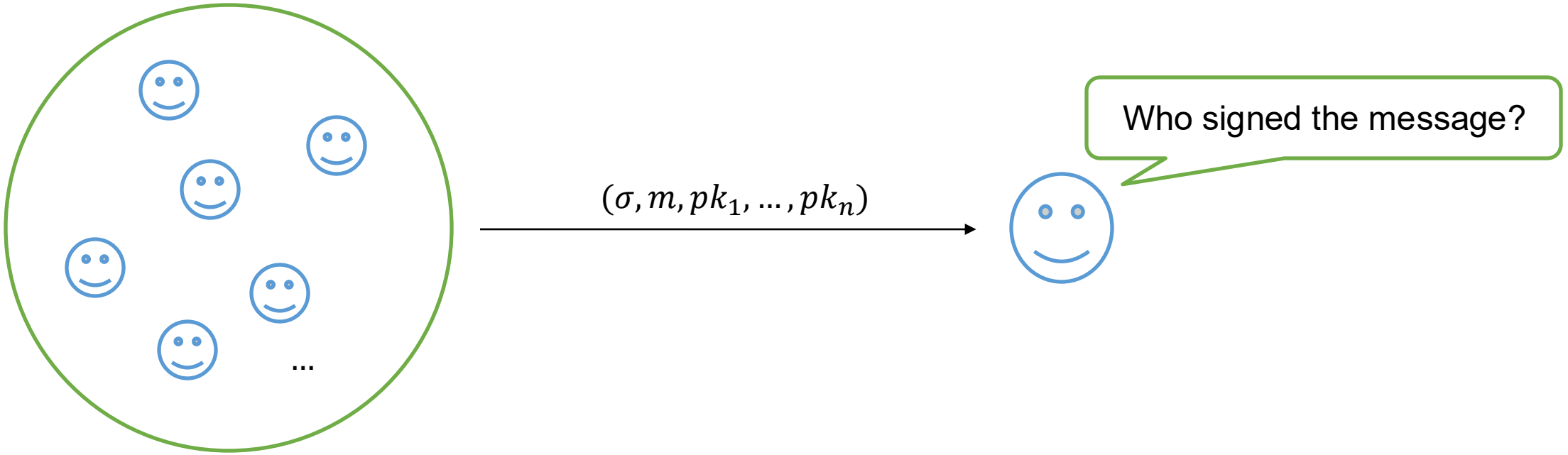
SNARK-Friendly Signature [ZSE+24]

Applications of SNARK-Friendly Signatures: SNARK-based Aggregate Signature

- **Witness**: $(\sigma_1, \ldots, \sigma_n)$
- **Statement**: $\left(C, (pk_1, m_1), \ldots, (pk_n, m_n)\right)$ where C is the circuit description of n signature verifications
- **Proof:** the aggregate signature (due to succinctness, the signature size is sublinear in n)

The prover knows n valid signatures.

Ring Signature



Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

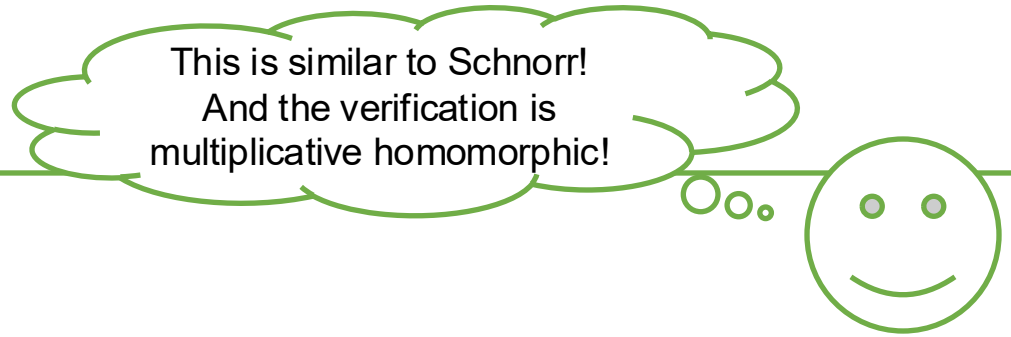
- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$

Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$

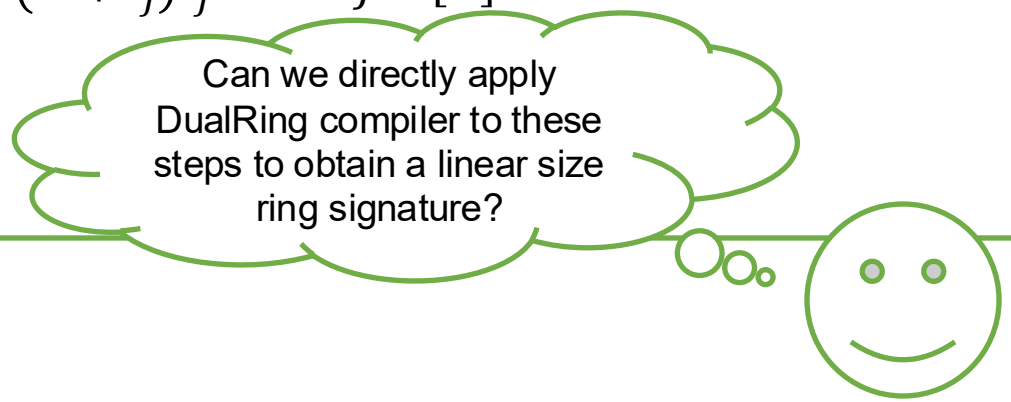


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$

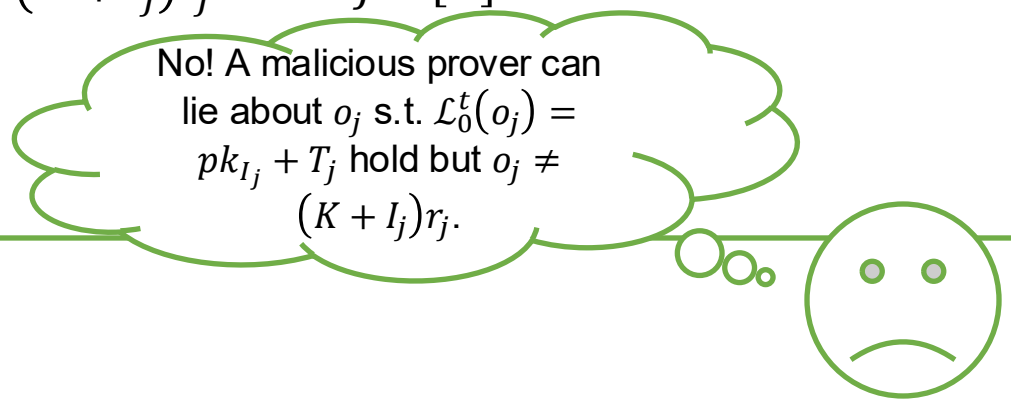


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$

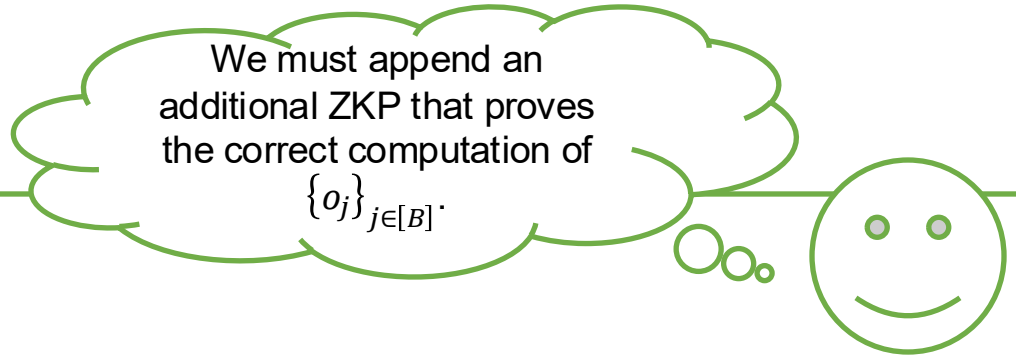


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$

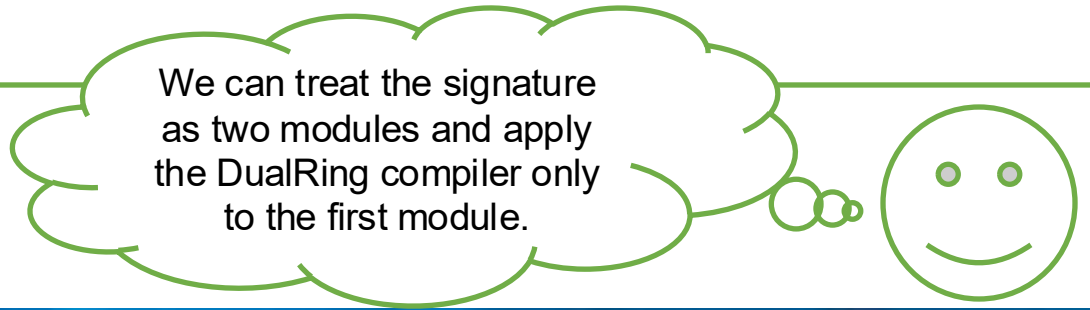


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$ and commits to the secret key K
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$ and the ZKP (e.g., MPCitH) which attests $o_j = (K + I_j)r_j$

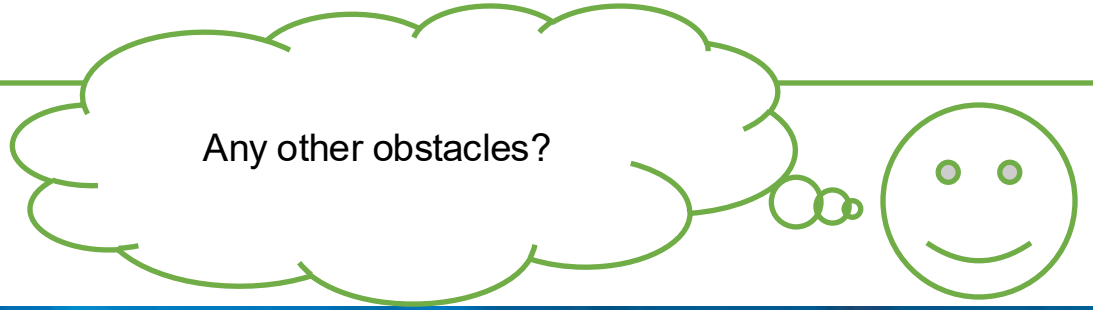


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$ and commits to the secret key K
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$ and the ZKP (e.g., MPCitH) which attests $o_j = (K + I_j)r_j$

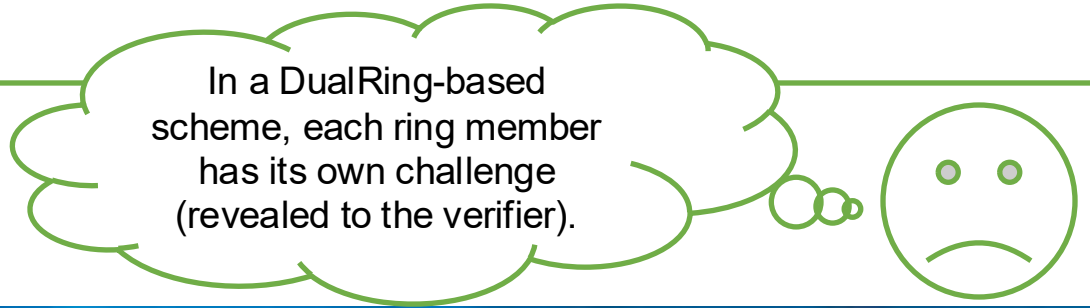


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$ and commits to the secret key K
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$ and the ZKP (e.g., MPCitH) which attests $o_j = (K + I_j)r_j$

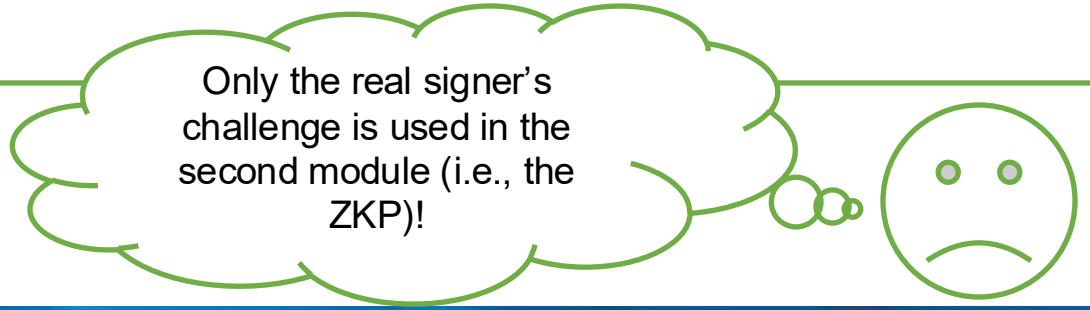


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$ and commits to the secret key K
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$ and the ZKP (e.g., MPCitH) which attests $o_j = (K + I_j)r_j$

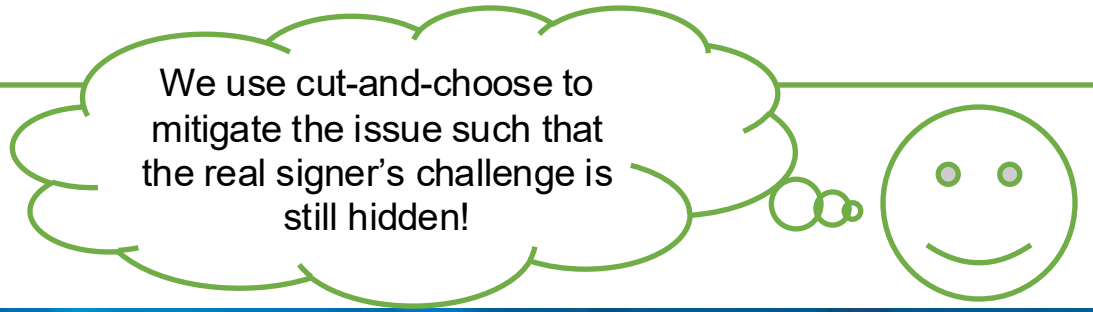


Ring Signature [ZSL+24]

Legendre/Power Residue PRF based Signature + DualRing [YEL+21]

Key Observation

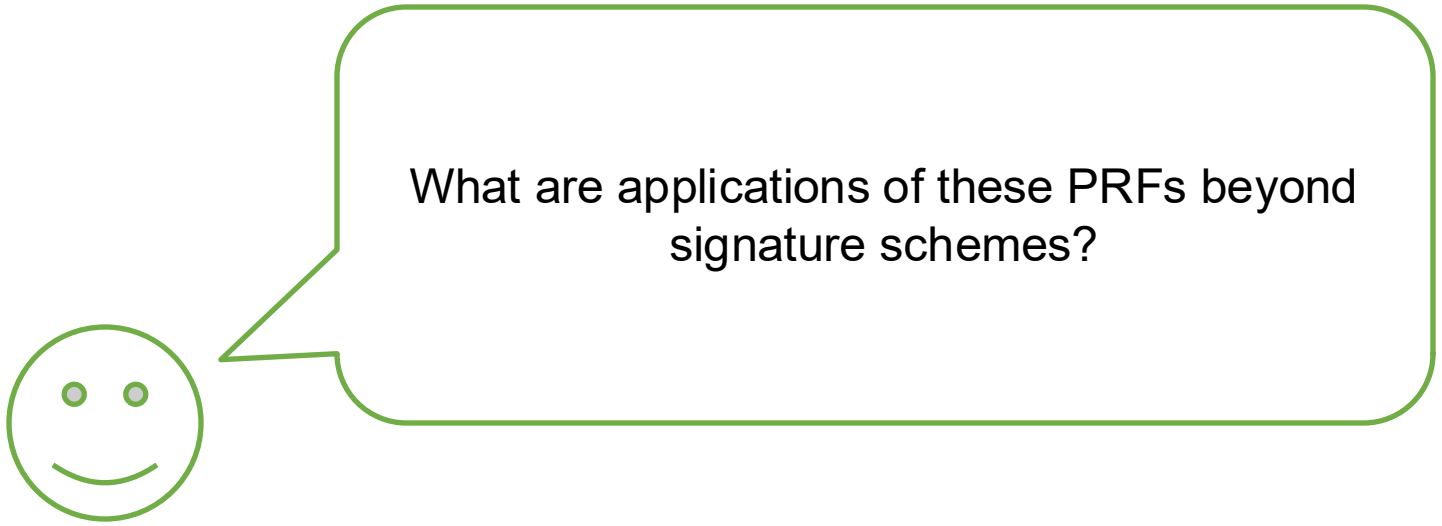
- There is a three-move identification scheme based on Legendre/power residue PRFs
  - Prover commits to randomness $\{r_j\}_{j\in[B]}$ by sending $T_j = \mathcal{L}_0^t(r_j)$ and commits to the secret key K
  - Verifier sends challenges $\{I_j\}_{j\in[B]}$
  - Prover sends the response $\{o_j\}_{j\in[B]}$, where $o_j = (K + I_j)r_j$ for all $j \in [B]$
  - Verifier checks if $\mathcal{L}_0^t(o_j) = pk_{I_j} + T_j$ and the ZKP (e.g., MPCitH) which attests $o_j = (K + I_j)r_j$



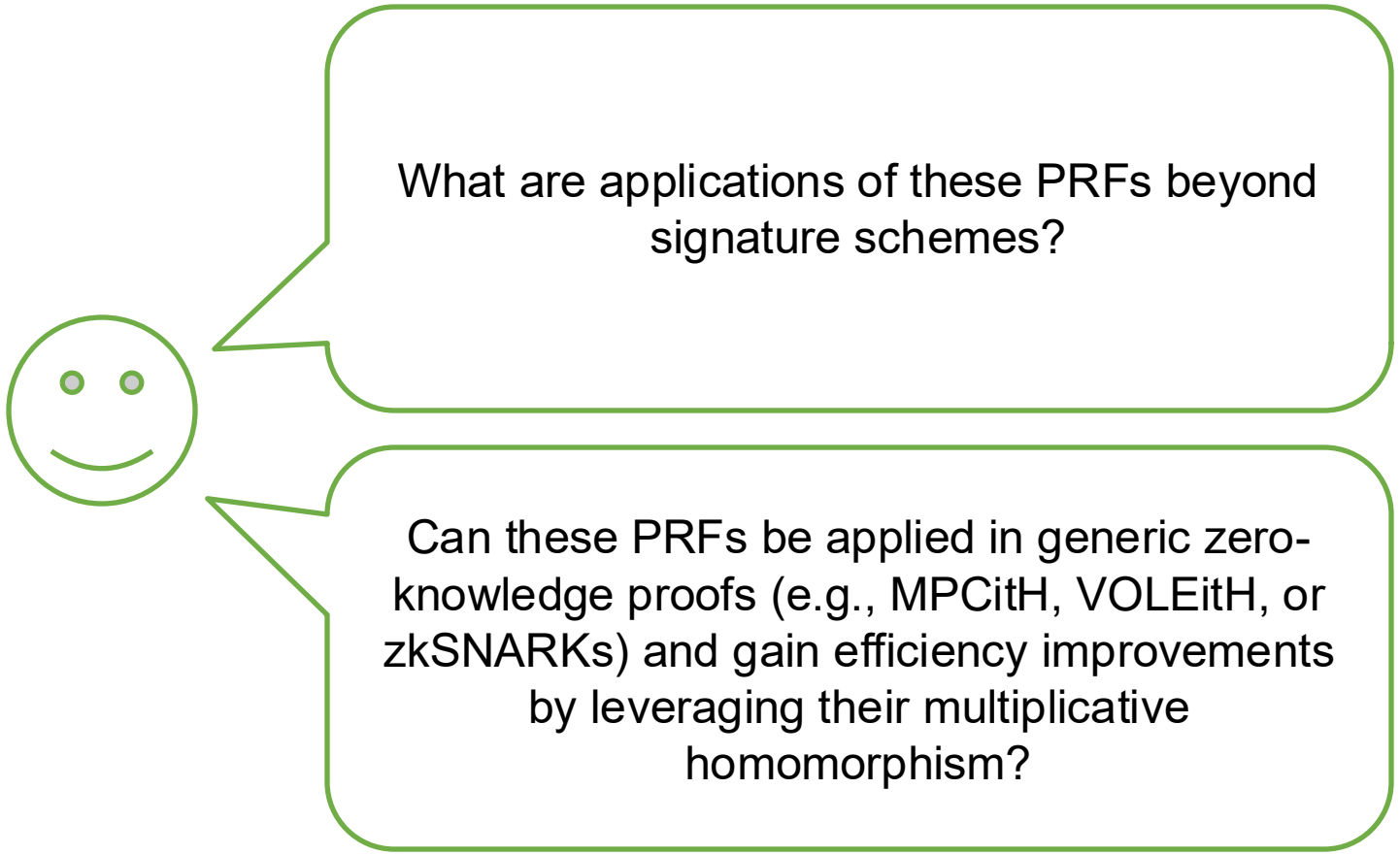
Future Works

MONASH University

Future Works



Future Works



Conclusion

Take Away

- Legendre and power residue PRFs (based on the conjectured pseudo-randomness of Legendre/power residue symbols)

Take Away

- Legendre and power residue PRFs (based on the conjectured pseudo-randomness of Legendre/power residue symbols)
- Signature schemes based on Legendre and power residue PRFs
  - Signature based on MPC-in-the-head [BG20]
  - Signature based on univariate sumcheck [ZSE+24] -> SNARK-friendly Signature
  - Ring signature [ZSL+24] based on DualRing compiler [YEL+21]

Take Away

- Legendre and power residue PRFs (based on the conjectured pseudo-randomness of Legendre/power residue symbols)
- Signature schemes based on Legendre and power residue PRFs
  - Signature based on MPC-in-the-head [BG20]
  - Signature based on univariate sumcheck [ZSE+24] -> SNARK-friendly Signature
  - Ring signature [ZSL+24] based on DualRing compiler [YEL+21]
- Future work -> applying Legendre and power residue PRFs beyond signature constructions?

Thank You!

Questions?

References

[Wei48] Weil, André. "On some exponential sums." *Proceedings of the National Academy of Sciences* 34, no. 5 (1948): 204-207.

[vDH00] van Dam, Wim, and Sean Hallgren. "Efficient quantum algorithms for shifted quadratic character problems." *arXiv preprint quant-ph/0011067* (2000).

[Kho19] Khovratovich, Dmitry. "Key recovery attacks on the Legendre PRFs within the birthday bound." *Cryptology ePrint Archive* (2019).

[GRR+16] Grassi, Lorenzo, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. "MPC-friendly symmetric key primitives." In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 430-443. 2016.

[BG20] Beullens, Ward, and Cyprien Delpech de Saint Guilhem. "LegRoast: Efficient post-quantum signatures from the Legendre PRF." In *International Conference on Post-Quantum Cryptography*, pp. 130-150. Cham: Springer International Publishing, 2020.

[ZXZS20] Zhang, Jiaheng, Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Transparent polynomial delegation and its applications to zero knowledge proof." In *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 859-876. IEEE, 2020.

[ZSE+24] Zhang, Xinyu, Ron Steinfeld, Muhammed F. Esgin, Joseph K. Liu, Dongxi Liu, and Sushmita Ruj. "Loquat: a SNARK-friendly post-quantum signature based on the legendre PRF with applications in ring and aggregate signatures." In *Annual International Cryptology Conference*, pp. 3-38. Cham: Springer Nature Switzerland, 2024.

MONASH University

References

[ZSL+24] Zhang, Xinyu, Ron Steinfeld, Joseph K. Liu, Muhammed F. Esgin, Dongxi Liu, and Sushmita Ruj. "DualRing-PRF: Post-quantum (Linkable) Ring Signatures from Legendre and Power Residue PRFs." In *Australasian Conference on Information Security and Privacy*, pp. 124-143. Singapore: Springer Nature Singapore, 2024.

[YEL+21] Yuen, Tsz Hon, Muhammed F. Esgin, Joseph K. Liu, Man Ho Au, and Zhimin Ding. "DualRing: generic construction of ring signatures with efficient instantiations." In *Annual International Cryptology Conference*, pp. 251-281. Cham: Springer International Publishing, 2021.

MONASH University