Faster Signatures from MPC-in-the-Head

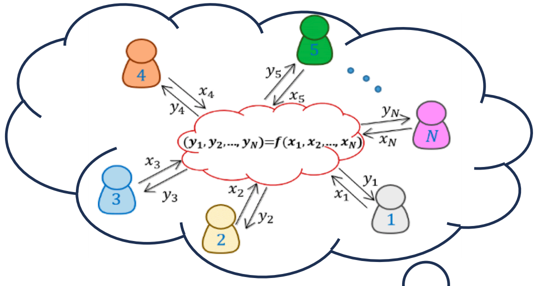
Dung Bui (IRIF), Eliana Carozza (IRIF), Geoffroy Couteau (IRIF,CNRS),
Dahmun Goudarzi (Quarkslab), and Antoine Joux (CISPA)

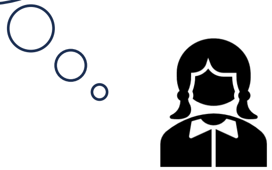
Kolkata, 10/12/2024

MPC-in-the-Head Paradigm (MPCitH)

MPC-in-the-Head Paradigm (MPCitH)

A compiler that transfers MPC protocol into HVZK proof for arbitrary circuits



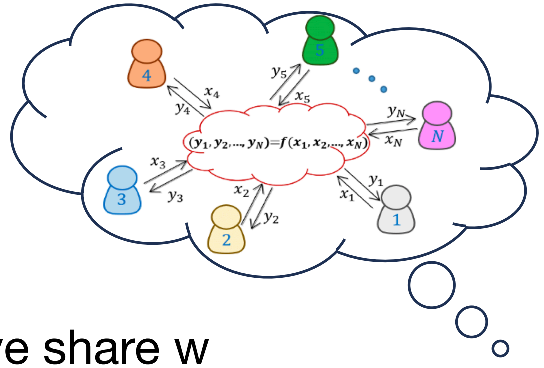


$\mathrm{w},\ C(\mathrm{w}) = 1$

Prover:
- Additive share w
- Execute MPC protocol in the Head
- Get the views of N parties

MPC-in-the-Head Paradigm (MPCitH)

A compiler that transfers MPC protocol into HVZK proof for arbitrary circuits



Prover:
- Additive share w
- Execute MPC protocol in the Head
- Get the views of N parties

$\mathrm{w},\ \mathrm{C(w)} = 1$

Commit to all $(\mathrm{view_i})_{i \in [N]}$ →

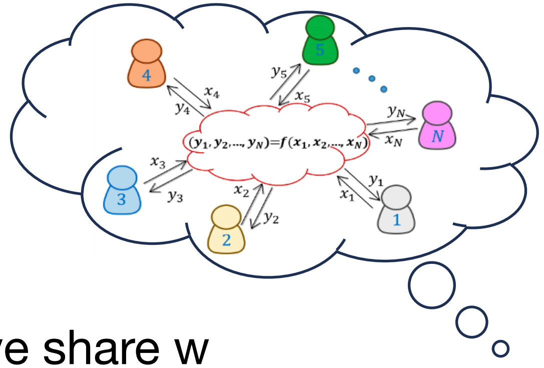
$\mathrm{P} \in [\mathrm{N}]$ ←

$(\mathrm{view_i})_{i \neq P}$ →

Verifier:
- Check the valid output of MPC protocol
- Verify views are consistent with commitments

MPC-in-the-Head Paradigm (MPCitH)

A compiler that transfers MPC protocol into HVZK proof for arbitrary circuits



Prover:
- Additive share w
- Execute MPC protocol in the Head
- Get the views of N parties

$\mathrm{w},\ \mathrm{C}(\mathrm{w}) = 1$

Commit to all $(\mathrm{view}_\mathrm{i})_{i\in[N]}$

$\mathrm{P} \in [\mathrm{N}]$

$(\mathrm{view}_\mathrm{i})_{i\neq P}$

Verifier:
- Check the valid output of MPC protocol
- Verify views are consistent with commitments

Security:

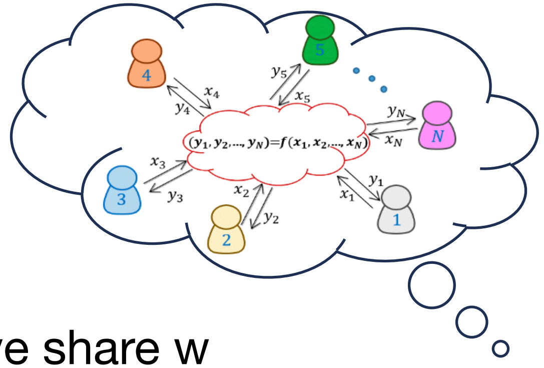
HVZK: MPC is secure against N -1 corrupted parties

There exists a simulator SimP that simulates the views of all parties except for P

Soundness error: 1/N

MPC-in-the-Head Paradigm (MPCitH)

A compiler that transfers MPC protocol into HVZK proof for arbitrary circuits



Prover:
- Additive share w
- Execute MPC protocol in the Head
- Get the views of N parties

$\mathrm{w},\ C(\mathrm{w}) = 1$

Commit to all $\left(\mathrm{view_i}\right)_{i \in [N]}$ →

$P \in [N]$ ←

$\left(\mathrm{view_i}\right)_{i \neq P}$ →

Verifier:
- Check the valid output of MPC protocol
- Verify views are consistent with commitments

Security:

HVZK: MPC is secure against N -1 corrupted parties

There exists a simulator SimP that simulates the views of all parties except for P

Soundness error: 1/N

Efficiency:

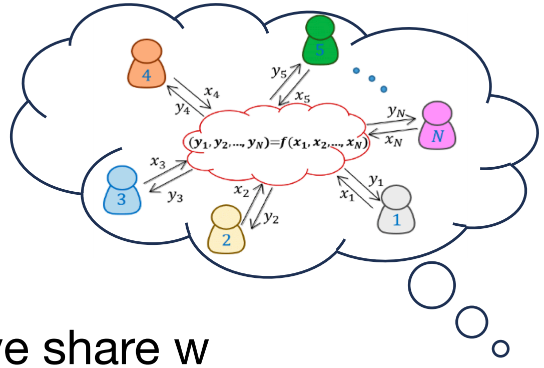
Computation: Underlying MPC protocol with N parties

Soundness is amplified with parallel repetitions

Communication: manner to open views

MPC-in-the-Head Paradigm (MPCitH)

A compiler that transfers MPC protocol into HVZK proof for arbitrary circuits



Prover:
- Additive share w
- Execute MPC protocol in the Head
- Get the views of N parties

$\mathrm{w},\ \mathrm{C}(\mathrm{w}) = 1$

Commit to all $(\mathrm{view_i})_{i \in [N]}$ →

← $\mathrm{P} \in [\mathrm{N}]$

$(\mathrm{view_i})_{i \neq P}$ →

Verifier:
- Check the valid output of MPC protocol
- Verify views are consistent with commitments

Security:

HVZK: MPC is secure against N -1 corrupted parties
There exists a simulator SimP that simulates the views of all parties except for P
Soundness error: 1/N

Efficiency:

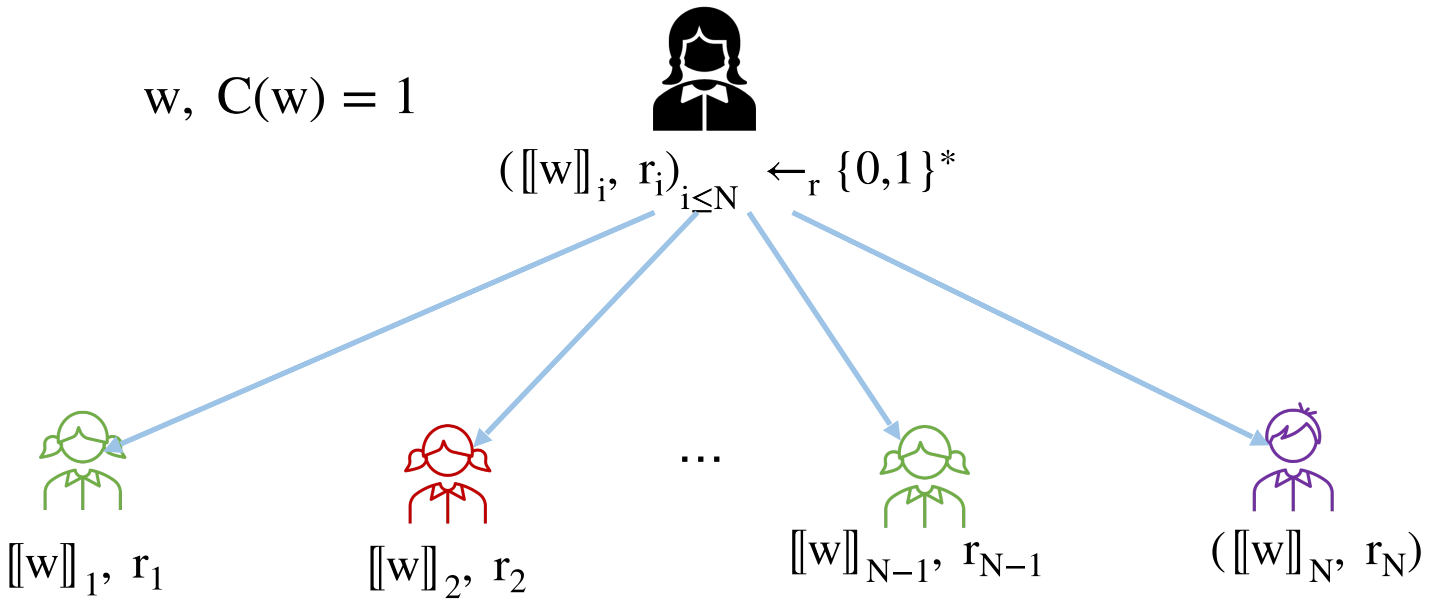
Computation: Underlying MPC protocol with N parties
Soundness is amplified with parallel repetitions
Communication: manner to open views

Get a signature from any OWF using Fiat-Shamir

Open all-but-one Views in MPCitH

MPCitH protocol:

- Generate shares of witness with shares of preprocessing material
- All shares can be considered as **random values**

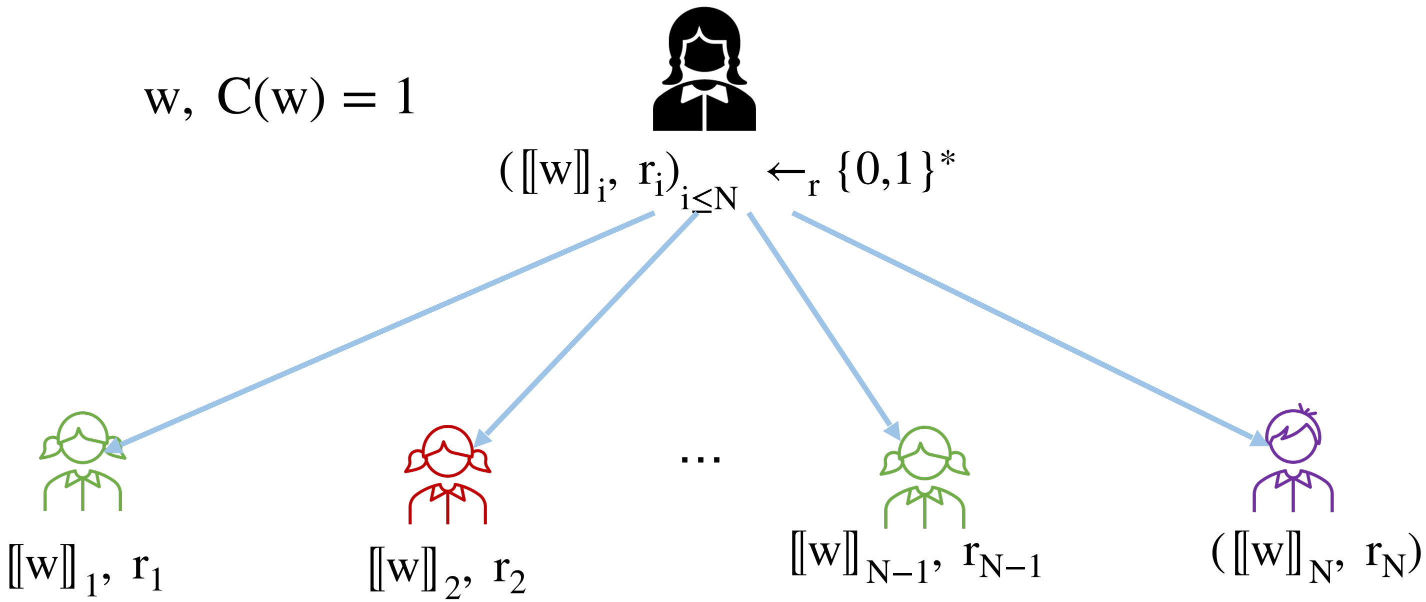


- Open N-1 out of N **random** shares $(r_i)_{i \leq N}$

Open all-but-one Views in MPCitH

MPCitH protocol:

- Generate shares of witness with shares of preprocessing material
- All shares can be considered as **random values**



- Open N-1 out of N **random** shares $(r_i)_{i \le N}$

Requirements in MPCitH-based signatures:

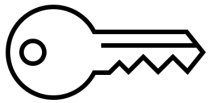
Unforgeability security

Parallel repetitions

Puncturable PRF (PPRF)

Puncturable PRF (PPRF): PRF **F** with master key K: $\mathbf{F}(K, x) \rightarrow y$

Punctured point: $P \leftarrow_r \{0,1\}^{\lambda}$



msk K

Punctured key K{P}

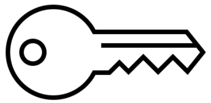
- K{P} is succinct
- Compute **F**(K, x) for all x except P
- **F**(K,P) is pseudorandom given K{P}

Puncturable PRF (PPRF)

Puncturable PRF (PPRF):

PRF **F** with master key K: $\mathbf{F}(K, x) \rightarrow y$

Punctured point: $P \leftarrow_r \{0,1\}^\lambda$



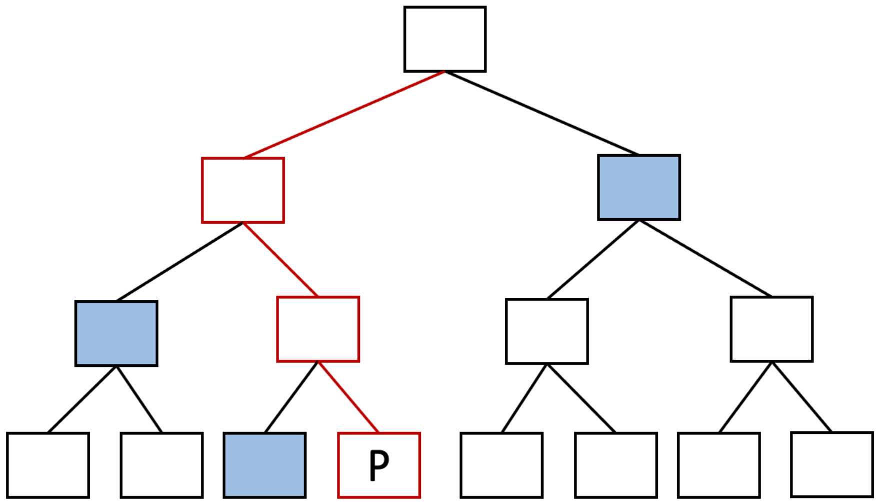
msk K

Punctured key K{P}

- K{P} is succinct
- Compute **F**(K, x) for all x except P
- **F**(K,P) is pseudorandom given K{P}

Instantiation PPRF: GGM tree

Length-doubling PRG: $F(x) \rightarrow \left(x_0, x_1\right) \in \{0,1\}^{2\lambda}$



Communication is reduced from $(N-1)\cdot\lambda$ to $\log N\cdot\lambda$

N leaves

Contribution 1: Multi-instance PPRF

Contribution 1: Multi-instance PPRF

(Q, τ)-multi-instance PPRF:

- Handle the security related to τ-repetitions in each signature
- Drop-in replacement of all PPRF in MPCitH-based signatures as considering Q the is number of queries from AdvA to signing oracle in unforgeability game

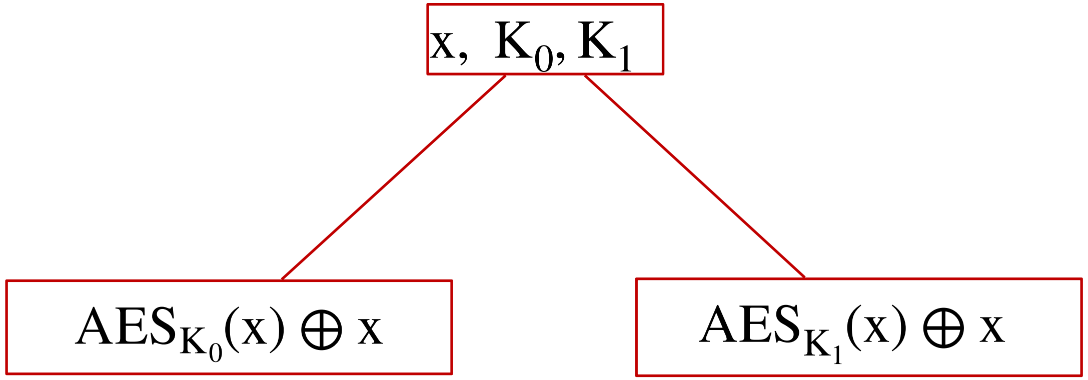
Contribution 1: Multi-instance PPRF

(Q, τ)-multi-instance PPRF:

- Handle the security related to τ-repetitions in each signature
- Drop-in replacement of all PPRF in MPCitH-based signatures as considering Q the is number of queries from AdvA to signing oracle in unforgeability game

New construction of PRG:

Davies-Meyer function $$F(x, salt) = \left(F_0(x,\ salt),\ F_1(x, salt)\right)$$



For τ-repetitions, (K_0, K_1) is used across all PPRF trees

Instantiate by fixed-key AES-NI (takes only 1.3 cycles per Byte)

Efficient, 12x to 55x speed improvement when plugging in the state of art ([C:JouHut24])

Contribution 1: Multi-instance PPRF

Contribution 1: Multi-instance PPRF

Construction of PRG:

$$F(x, K_0, K_1) = (AES_{K_0}(x) \oplus x, AES_{K_1}(x) \oplus x)$$

Security:

is proved in the **ideal cipher** using H-coefficient technique

(Q, τ)-instance (t, ϵ)-secure PRG

$(x_i, K_{0,i}, K_{1,i})_{i \le Q}$

$b \leftarrow_r \{0,1\}$

$(K_{0,i}, K_{1,i})_{i \le Q}$ →

$b = 0, F(x_i,\ K_i)$ →

$b = 1$, random

Guess b

Q instances, each instance repeats τ-times using **same salt**

Contribution 1: Multi-instance PPRF

Construction of PRG:

$$\mathrm{F}(\mathrm{x}, \mathrm{K}_0, \mathrm{K}_1) = (\mathrm{AES}_{\mathrm{K}_0}(\mathrm{x}) \oplus \mathrm{x}, \mathrm{AES}_{\mathrm{K}_1}(\mathrm{x}) \oplus \mathrm{x})$$

Security:

is proved in the **ideal cipher** using H-coefficient technique

$\left(Q, \tau\right)$-instance (t, ϵ)-secure PRG

$(\mathrm{x}_\mathrm{i}, \mathrm{K}_{0,\mathrm{i}}, \mathrm{K}_{1,\mathrm{i}})_{\mathrm{i} \leq \mathrm{Q}}$

$\mathrm{b} \leftarrow_r \{0,1\}$

$(\mathrm{K}_{0,\mathrm{i}}, \mathrm{K}_{1,\mathrm{i}})_{\mathrm{i} \leq \mathrm{Q}}$ →

$\mathrm{b} = 0, \mathrm{F}(\mathrm{x}_\mathrm{i},\ \mathrm{K}_\mathrm{i})$ →

$\mathrm{b} = 1$, random

Guess b

Q instances, each instance repeats τ-times using **same salt**

Security of PPRF: for a GGM tree of N = 2^D leaves

(t, ϵ)-secure PRG → $\left(Q, \tau\right)$-instance $(t, D.\epsilon)$-secure PPRF

Security loss 5 bits
$(\mathrm{D} = 16,\ \tau = 8, \lambda = 128)$

Contribution 2: New MPCitH-based signature

Assumption: Regular syndrome decoding (RSD),

Sample a matrix $\mathrm{H} \in \mathbb{F}_2^{k \times K}$, $\mathbf{x} \in \mathbb{F}_2^{K}$ such that **x** is w-regular noise vector, set $\mathbf{y} = \mathrm{H} \cdot \mathbf{x}$

Given (H,**y**), it is hard to find **x**

Contribution 2: New MPCitH-based signature

Assumption: Regular syndrome decoding (RSD),

Sample a matrix $\mathrm{H} \in \mathbb{F}_2^{k \times K}$, $\mathbf{x} \in \mathbb{F}_2^{K}$ such that **x** is w-regular noise vector, set $\mathbf{y} = \mathrm{H} \cdot \mathbf{x}$

Given (H,**y**), it is hard to find **x**

Intuition:

Denote bs = K/w,

	bs				
Vector **x**:	001..000	000...100	.....	100...000	010...000
Position of noises:	i_1	i_2	.....	i_{w-1}	i_w

Compressed vector of **x**: **comp(x)** = $(i_1, i_2, \ldots, i_{w-1}, i_w) \in (\mathbb{Z}_{bs})^w$

Instead of sharing over $\mathbb{F}_2^K$, we share over $\mathbb{Z}_{bs}$ → $\mathbf{w} \cdot \mathbf{log}(\mathbf{bs})$ bits

Contribution 2: New MPCitH-based signature

Contribution 2: New MPCitH-based signature

Key idea:

$\mathbf{x} \in \mathbb{F}_2^K$ is w-regular noise: sample $\mathbf{r} \leftarrow_r \mathbb{F}_2^K$ is w-regular noise vector

$\mathbf{z}$ = comp($\mathbf{x}$) – comp($\mathbf{r}$) $\in (\mathbb{Z}_{bs})^w$ (positions of noise)

→ $\mathbf{x}$ = $\mathbf{r}$ shifted by $\mathbf{z}$

$\mathbf{r}$ can be considered as a mask of $\mathbf{x}$

Contribution 2: New MPCitH-based signature

Key idea:

$\mathbf{x} \in \mathbb{F}_2^K$ is w-regular noise: sample $\mathbf{r} \leftarrow_r \mathbb{F}_2^K$ is w-regular noise vector

z = comp(**x**) – comp(**r**) $\in (\mathbb{Z}_{bs})^w$ (positions of noise)

→ **x** = **r** shifted by **z**

r can be considered as a mask of **x**

MPCitH protocol:

Parties holds shares of **r** $\in \mathbb{F}_2^K$ (w-regular noise); comp(**x**), comp(**r**) $\in (\mathbb{Z}_{bs})^w$:

- All parties broadcast their shares of **z** = comp(**x**) – comp(**r**) and reconstruct **z**
- All parties locally shift cyclically their share of **r** by **z**

Contribution 2: New MPCitH-based signature

Key idea:

$\mathbf{x} \in \mathbb{F}_2^K$ is w-regular noise: sample $\mathbf{r} \leftarrow_r \mathbb{F}_2^K$ is w-regular noise vector

z = comp(**x**) – comp(**r**) $\in (\mathbb{Z}_{bs})^w$ (positions of noise)

→ **x** = **r** shifted by **z**

r can be considered as a mask of **x**

MPCitH protocol:

Parties holds shares of **r** $\in \mathbb{F}_2^K$ (w-regular noise); comp(**x**), comp(**r**) $\in (\mathbb{Z}_{bs})^w$:

- All parties broadcast their shares of **z** = comp(**x**) – comp(**r**) and reconstruct **z**
- All parties locally shift cyclically their share of **r** by **z**

Soundness:

Prover can cheat in generating a preprocessing

→ verifier adds a permutation π on **r**, i.e., **z** = comp(**x**) – π(comp(**r**))

Takeaway

New multi-instance PPRF
- Efficient based on AES-NI
- Used to bootstrap any MPCitH-based signatures
- Benchmark by plugging into [C:JouHut24]

New MPCitH-based signature
- Based on RSD
- Outperform [EC:CCJ23], competitive efficiency

https://ia.cr/2024/252

References: [C:JouHut24] *MPC in the head using the subfield bilinear collision problem, by J. Huth and A. Joux, in CRYPTO 2024*
[EC:CCJ23] *Short signatures from regular syndrome decoding in the head, by E. Carozza, G. Couteau, and A. Joux, in EUROCRYPT 2023*

Takeaway

New multi-instance PPRF
- Efficient based on AES-NI
- Used to bootstrap any MPCitH-based signatures
- Benchmark by plugging into [C:JouHut24]

New MPCitH-based signature
- Based on RSD
- Outperform [EC:CCJ23], competitive efficiency

https://ia.cr/2024/252

Thank you

References: [C:JouHut24] *MPC in the head using the subfield bilinear collision problem, by J. Huth and A. Joux, in CRYPTO 2024*
[EC:CCJ23] *Short signatures from regular syndrome decoding in the head, by E. Carozza, G. Couteau, and A. Joux, in EUROCRYPT 2023*