RSAConference™2024
San Francisco | May 6 – 9 | Moscone Center

THE ART OF POSSIBLE

SESSION ID: CRYP-M06B

History-Free Sequential Aggregation of Hash-and-Sign Signatures

#RSAC

Edoardo Signorini
Cryptographer
Telsy
@Edoars

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference™ or any other co-sponsors. RSA Conference does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

Aggregate Signatures

$\overline{pk} = (pk_1, \ldots, pk_n)$

$(sk_i, pk_i) \leftarrow KeyGen(1^\lambda)$

$\sigma_i \leftarrow Sign(sk_i, m_i)$

Aggregate Signatures

$\overline{pk} = (pk_1, \ldots, pk_n)$

$(sk_i, pk_i) \leftarrow KeyGen(1^\lambda)$

$\sigma_i \leftarrow Sign(sk_i, m_i)$

$\Sigma \leftarrow AggSign(\overline{m}, \overline{\sigma})$

$\overline{m}, \Sigma$

$\{\checkmark, \times\} \leftarrow AggVf(\overline{pk}, \overline{m}, \Sigma)$

Goal

Combine multiple σ_i in a single Σ such that $|\Sigma| \ll |\sigma_1| + \cdots + |\sigma_n|$

Aggregate Signatures

$\overline{pk} = (pk_1, \ldots, pk_n)$

$(sk_i, pk_i) \leftarrow KeyGen(1^\lambda)$

$\sigma_i \leftarrow Sign(sk_i, m_i)$

$\Sigma \leftarrow AggSign(\overline{m}, \overline{\sigma})$

$\overline{m}, \Sigma$

$\{\checkmark, \times\} \leftarrow AggVf(\overline{pk}, \overline{m}, \Sigma)$

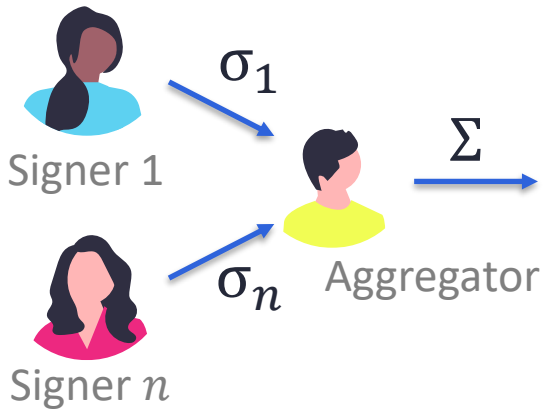
- Reduce bandwidth consumption
- Constrained devices
- Certificate chains
- Blockchains

Goal

Combine multiple σ_i in a single Σ such that $|\Sigma| \ll |\sigma_1| + \cdots + |\sigma_n|$

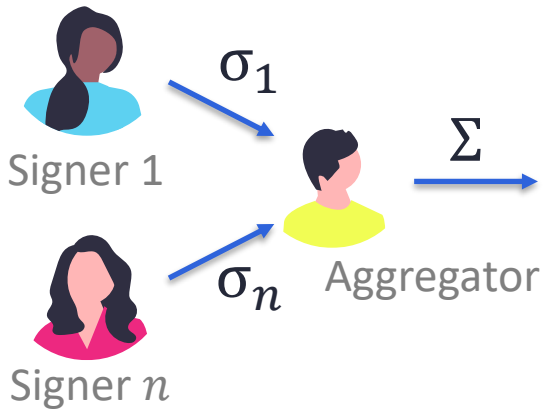
Types of Aggregate Signature

- General Aggregate Signature (AS)
  - Public aggregation by third party
  - No interaction required by signers
  - Construction based on bilinear pairings [BGLS03]

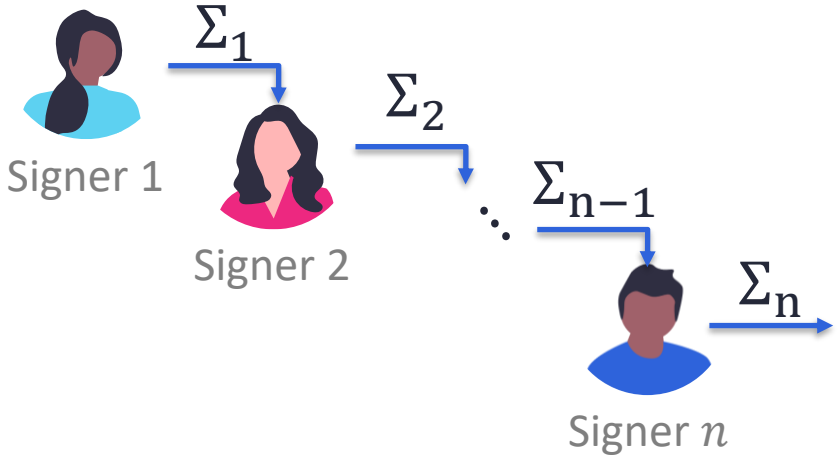


Types of Aggregate Signature

- General Aggregate Signature (AS)
  - Public aggregation by third party
  - No interaction required by signers
  - Construction based on bilinear pairings [BGLS03]

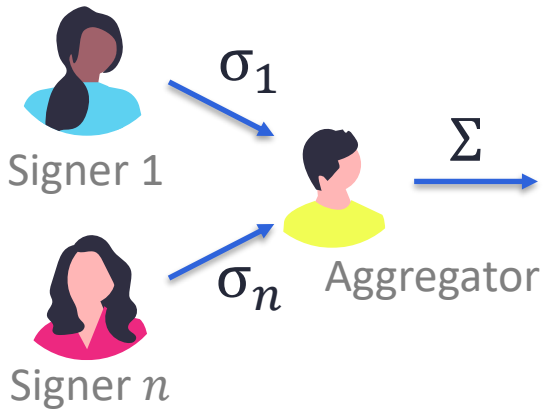


- Sequential Aggregate Signature (SAS)
  - Signatures are iteratively aggregated
  - Can be built from trapdoor permutation [LMRS04; Nev08; BGR12; GOR18]

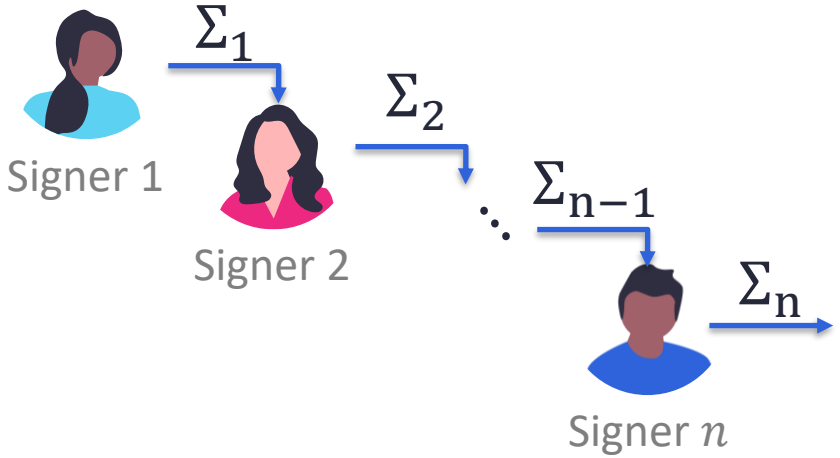


Types of Aggregate Signature

- General Aggregate Signature (AS)
  - Public aggregation by third party
  - No interaction required by signers
  - Construction based on bilinear pairings [BGLS03]



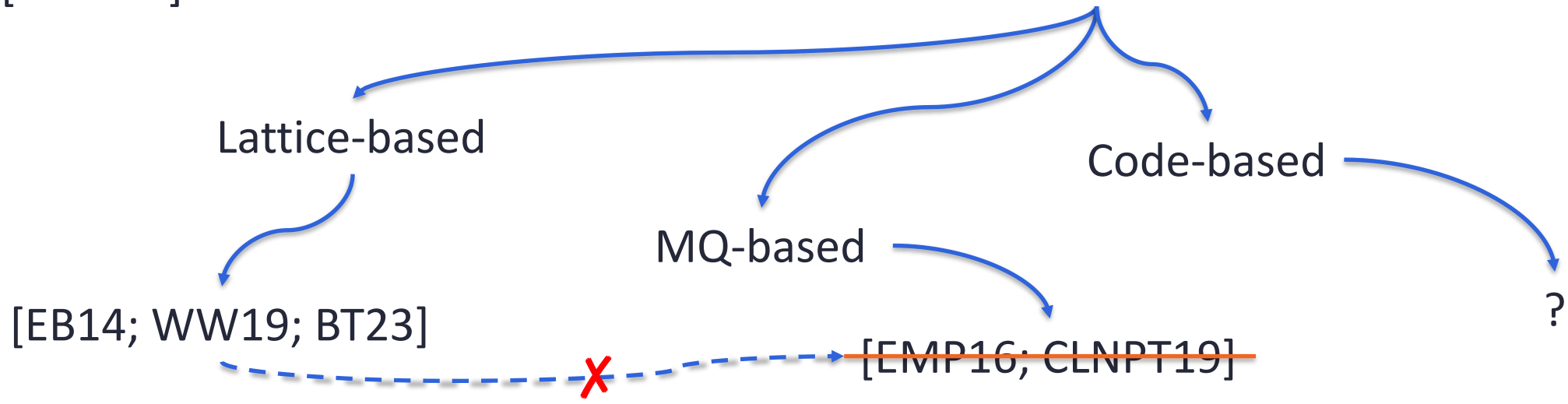
- Sequential Aggregate Signature (SAS)
  - Signatures are iteratively aggregated
  - Can be built from trapdoor permutation [LMRS04; Nev08; BGR12; GOR18]



Can (S)AS be built from post-quantum assumptions?

Types of Aggregate Signature

- General Aggregate Signature (AS)
  - Public aggregation by third party
  - No interaction required by signers
  - Construction based on bilinear pairings [BGLS03]
- **Sequential Aggregate Signature** (SAS)
  - Signatures are iteratively aggregated
  - Can be built from trapdoor permutation [LMRS04; Nev08; BGR12; GOR18]



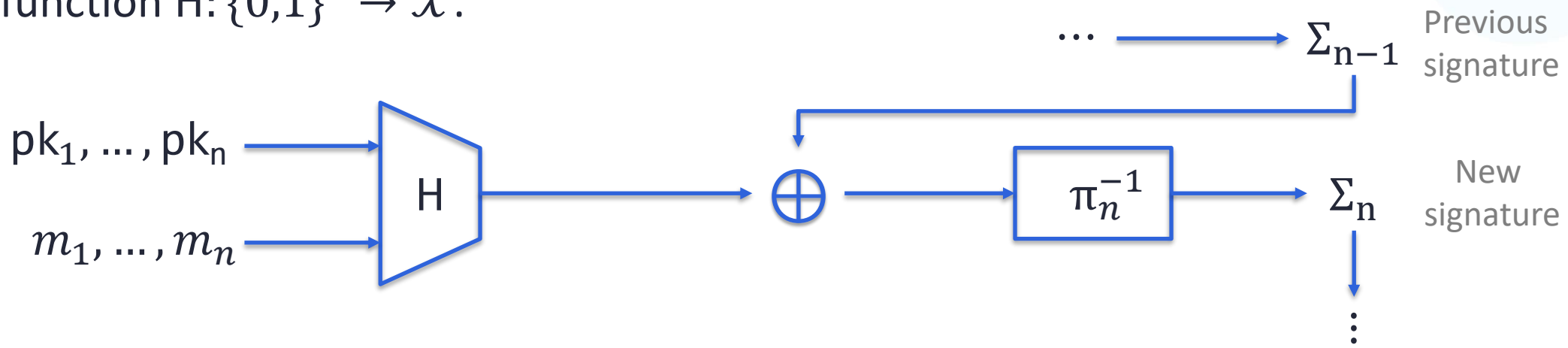
Can (S)AS be built from post-quantum assumptions? Yes, from lattices

SAS from Trapdoor Permutation

Full Domain Hash (FDH) signature from trapdoor permutation $\pi: \mathcal{X} \rightarrow \mathcal{X}$ and a suitable hash function $H: \{0,1\}^* \rightarrow \mathcal{X}$.

SAS from Trapdoor Permutation

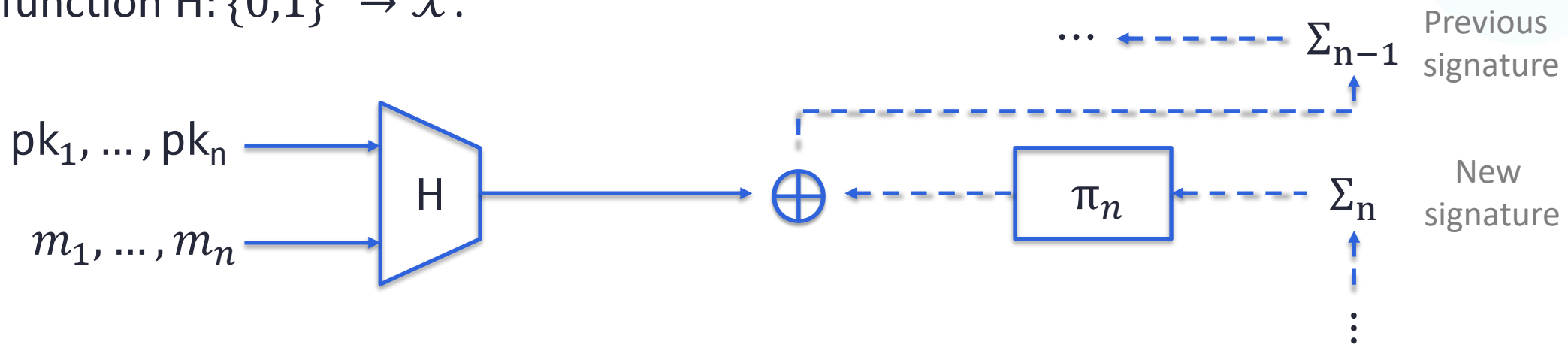
Full Domain Hash (FDH) signature from trapdoor permutation $\pi: \mathcal{X} \to \mathcal{X}$ and a suitable hash function $H: \{0,1\}^* \to \mathcal{X}$.



- **Aggregation** (simplified) [LMRS04, Nev08]: embed the previous aggregate signature into the new data to be signed.

SAS from Trapdoor Permutation

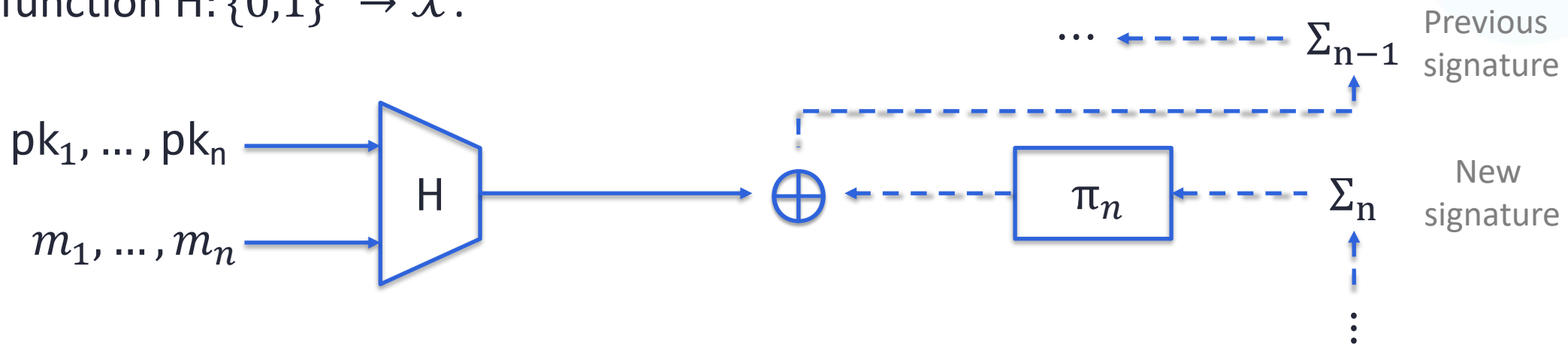
Full Domain Hash (FDH) signature from trapdoor permutation $\pi: \mathcal{X} \to \mathcal{X}$ and a suitable hash function $H: \{0,1\}^* \to \mathcal{X}$.



- **Aggregation** (simplified) [LMRS04, Nev08]: embed the previous aggregate signature into the new data to be signed.
- **Verification**: recover each intermediate signature. Requires n steps of verification.

SAS from Trapdoor Permutation

Full Domain Hash (FDH) signature from trapdoor permutation $\pi: \mathcal{X} \to \mathcal{X}$ and a suitable hash function $H: \{0,1\}^* \to \mathcal{X}$.



- **Aggregation** (simplified) [LMRS04, Nev08]: embed the previous aggregate signature into the new data to be signed.
- **Verification**: recover each intermediate signature. Requires n steps of verification.

Rigid transposition of FDH approach to post-quantum assumptions is impractical

Trapdoor Functions

A trapdoor function (TDF) is a tuple of three algorithm $(\mathrm{TrapGen}, \mathrm{F}, \mathrm{I})$:

- $\mathrm{TrapGen}(1^{\lambda})$: takes as input a security parameter 1^{λ} and generates an efficiently computable function $\mathrm{F}: \mathcal{X} \to \mathcal{Y}$ and a trapdoor I that allow to invert F.

- $\mathrm{F}(x)$: takes as input $x \in \mathcal{X}$ and outputs $\mathrm{F}(x) \in \mathcal{Y}$.
- $\mathrm{I}(y)$: takes as input $y \in \mathcal{Y}$ and outputs $x \in \mathcal{X}$ such that $\mathrm{F}(x) = y$ or it fails by returning $\perp$.

Trapdoor Functions

A trapdoor function (TDF) is a tuple of three algorithm $(\mathrm{TrapGen}, \mathrm{F}, \mathrm{I})$:

- $\mathrm{TrapGen}(1^{\lambda})$: takes as input a security parameter 1^{λ} and generates an efficiently computable function $\mathrm{F}: \mathcal{X} \to \mathcal{Y}$ and a trapdoor I that allow to invert F.

- $\mathrm{F}(x)$: takes as input $x \in \mathcal{X}$ and outputs $\mathrm{F}(x) \in \mathcal{Y}$.
- $\mathrm{I}(y)$: takes as input $y \in \mathcal{Y}$ and outputs $x \in \mathcal{X}$ such that $\mathrm{F}(x) = y$ or it fails by returning $\perp$.

- When F is a permutation, the security of the FDH scheme is reduced to the one-wayness (OW) of F.
- Generic trapdoor functions lose uniformity properties and provable security with FDH.

Trapdoor Functions

A trapdoor function (TDF) is a tuple of three algorithm $(\text{TrapGen}, \text{F}, \text{I})$:

- $\text{TrapGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates an efficiently computable function $\text{F}: \mathcal{X} \to \mathcal{Y}$ and a trapdoor I that allow to invert F.

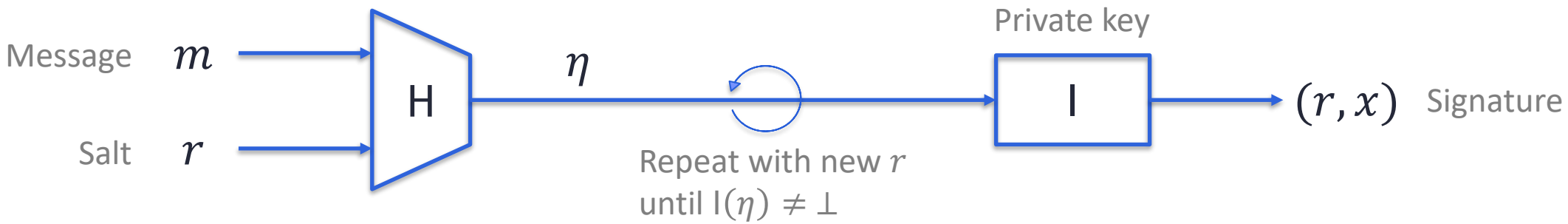
- $\text{F}(x)$: takes as input $x \in \mathcal{X}$ and outputs $\text{F}(x) \in \mathcal{Y}$.
- $\text{I}(y)$: takes as input $y \in \mathcal{Y}$ and outputs $x \in \mathcal{X}$ such that $\text{F}(x) = y$ or it fails by returning $\perp$.

- When F is a permutation, the security of the FDH scheme is reduced to the one-wayness (OW) of F.
- Generic trapdoor functions lose uniformity properties and provable security with FDH.

We can regain provable security using the probabilistic hash-and-sign with retry approach.

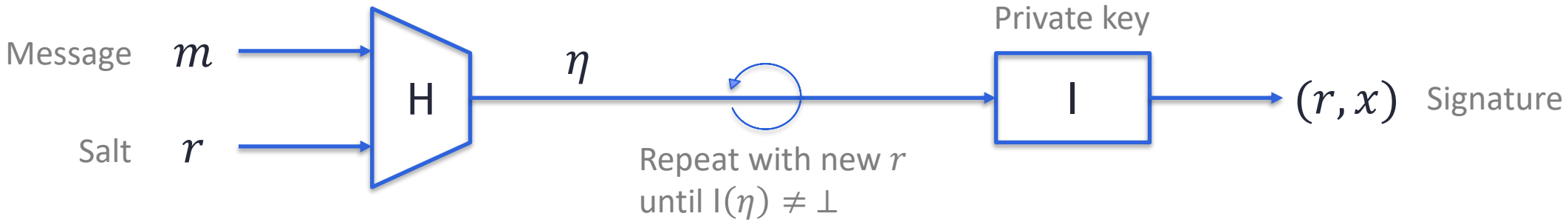
Probabilistic Hash-and-Sign with Retry

Signature from trapdoor function (F, I) and a suitable hash function $\mathsf{H}: \mathcal{X} \to \mathcal{Y}$.



Probabilistic Hash-and-Sign with Retry

Signature from trapdoor function (F, I) and a suitable hash function H: $\mathcal{X} \to \mathcal{Y}$.



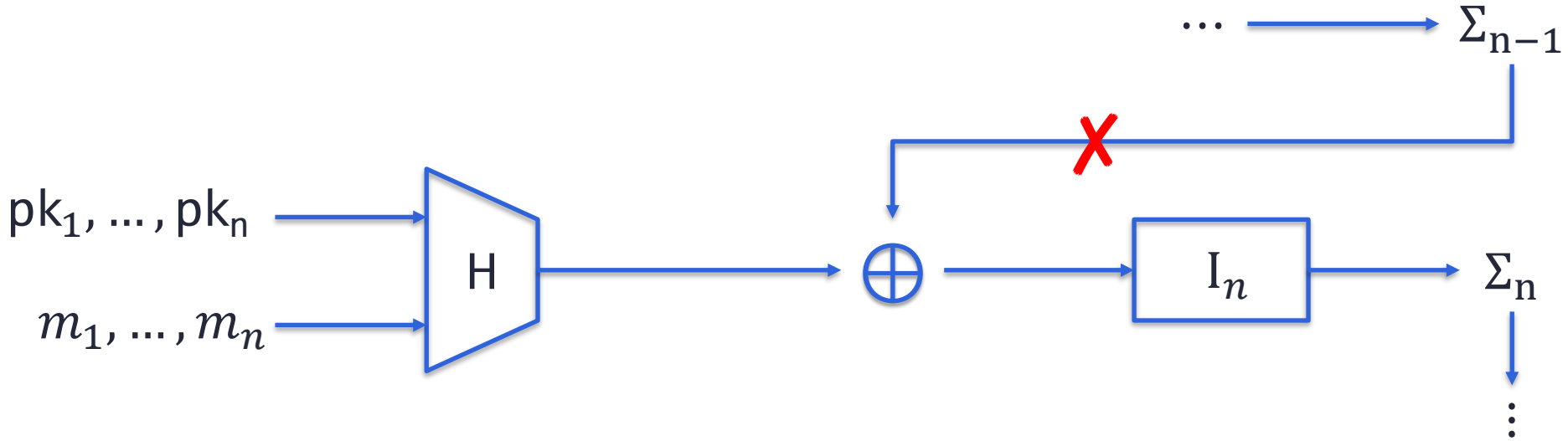
The security of the scheme is based on the one-wayness of F and the following additional property [KX24]:

The output of the signing algorithm (r,x) is such that:

1. The salt r is indistinguishable from $r \leftarrow_R \{0,1\}^{\lambda}$.
2. The signature x is indistinguishable from $x \leftarrow_R \mathcal{X}$.

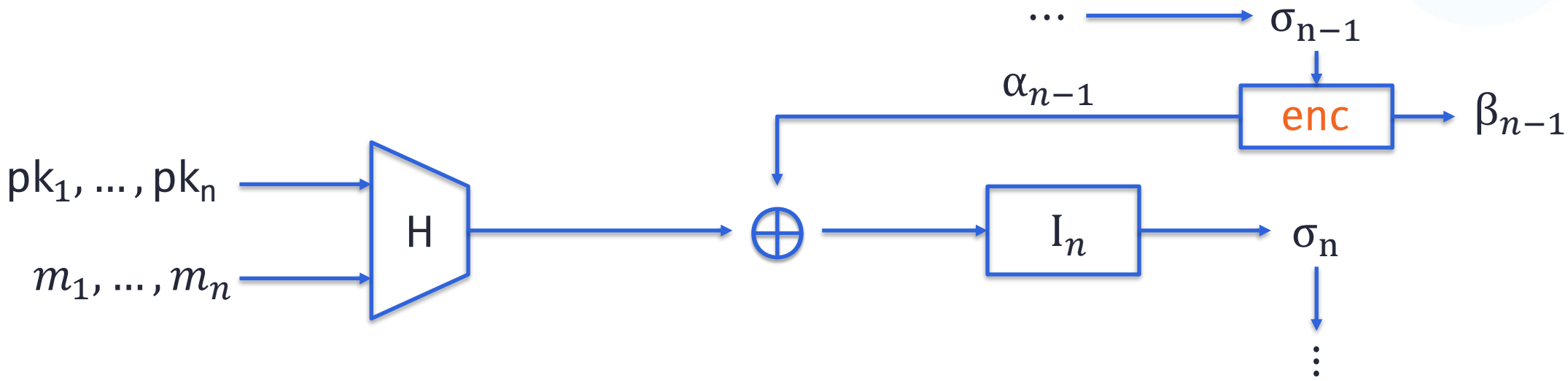
Generalize SAS Schemes

- Consider a generic trapdoor function (F, I) with F: $\mathcal{X} \rightarrow \mathcal{Y}$.



Generalize SAS Schemes

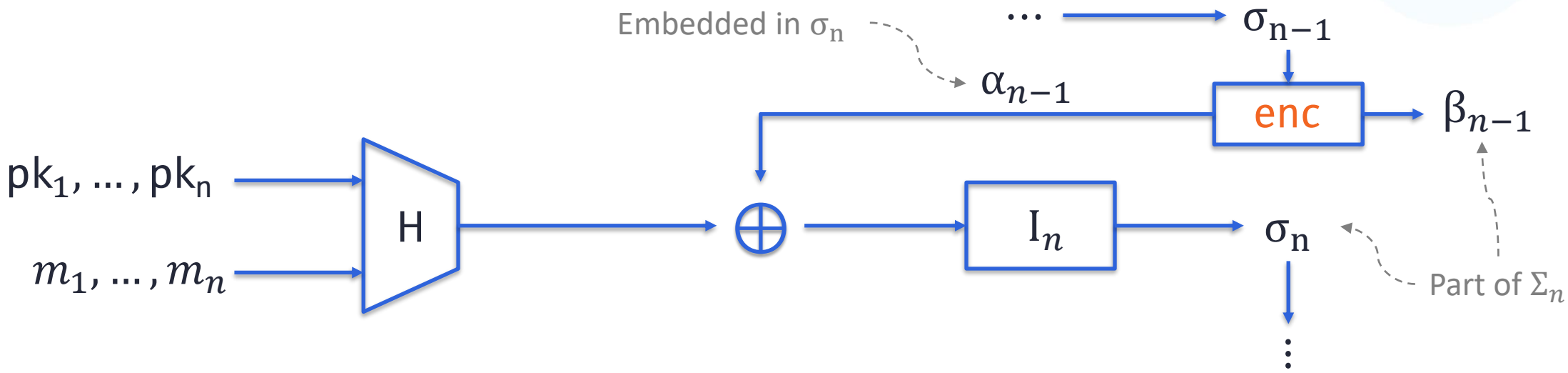
- Consider a generic trapdoor function (F, I) with F: $\mathcal{X} \to \mathcal{Y}$.



- Use an *efficient* encoding function enc: $\mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ that splits σ_i as (α_i, β_i) [Nev08].

Generalize SAS Schemes

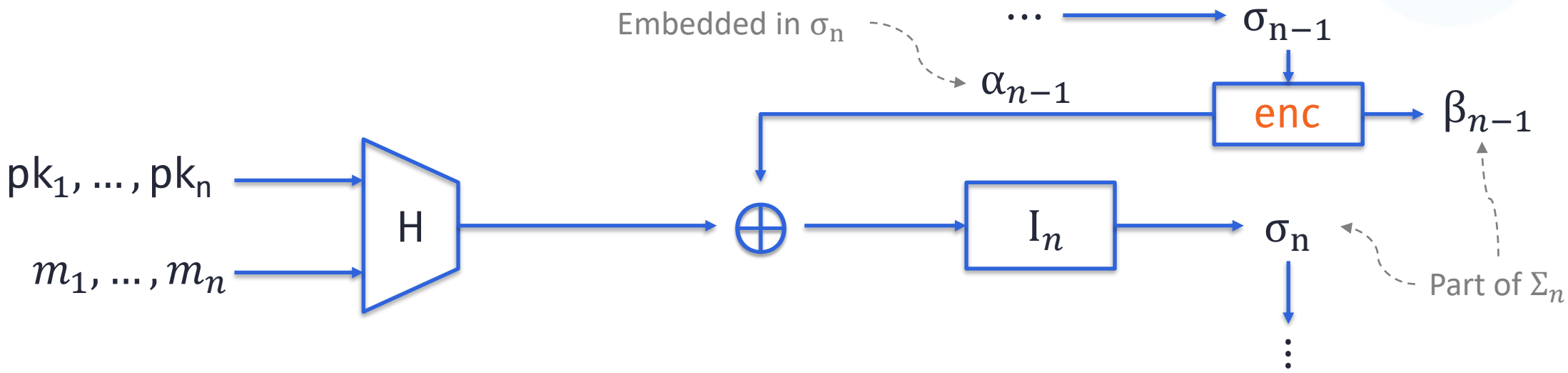
- Consider a generic trapdoor function (F, I) with F: $\mathcal{X} \to \mathcal{Y}$.



- Use an *efficient* encoding function enc: $\mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ that splits σ_i as (α_i, β_i) [Nev08].
- The aggregate signature is given by $\Sigma_n = (\beta_1, \ldots, \beta_{n-1}, \sigma_n)$.
- This construction is claimed secure with every multivariate HaS scheme [EMP16; CLNPT19].

Generalize SAS Schemes

- Consider a generic trapdoor function (F, I) with F: $\mathcal{X} \to \mathcal{Y}$.

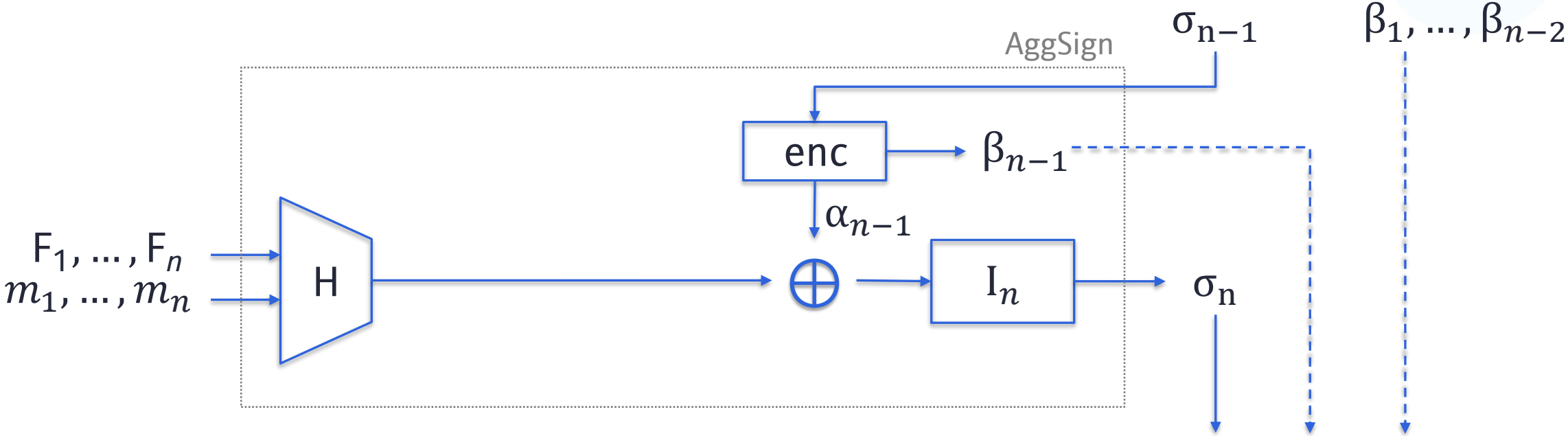


- Use an *efficient* encoding function enc: $\mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ that splits σ_i as (α_i, β_i) [Nev08].
- The aggregate signature is given by $\Sigma_n = (\beta_1, \ldots, \beta_{n-1}, \sigma_n)$.
- This construction is claimed secure with ~~every~~ multivariate HaS scheme [EMP16; CLNPT19].

False in general!

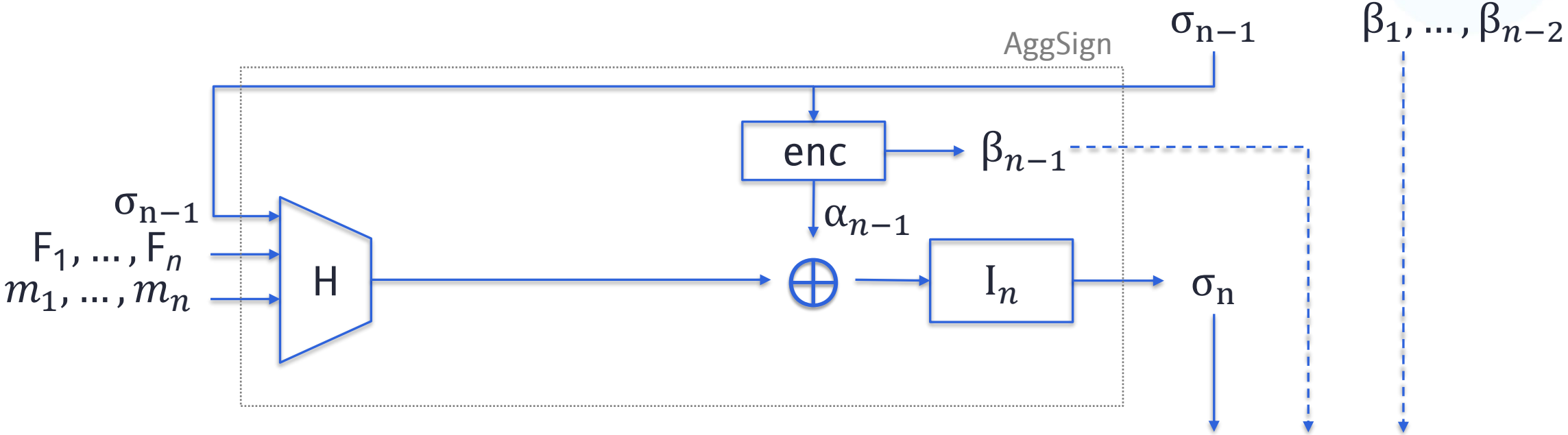
Steps to Provable Security

The following aggregate scheme is **not** provably secure (and sometimes provably insecure) with generic TDF.



Steps to Provable Security

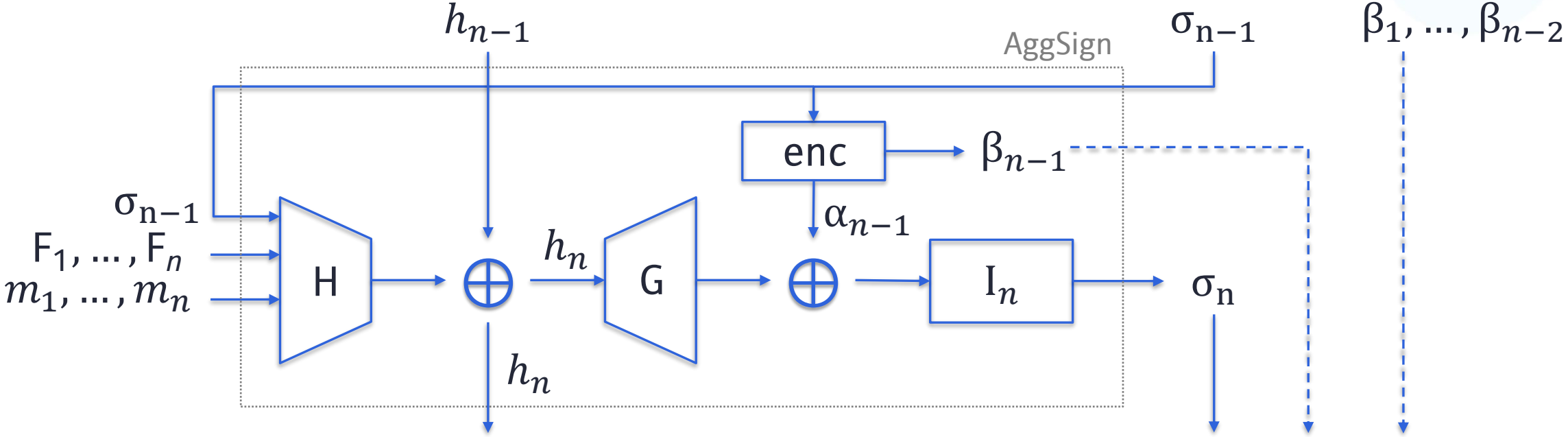
The following aggregate scheme is **not** provably secure (and sometimes provably insecure) with generic TDF.



- F_i is not injective and aggregate signatures are not unique on fixed input.

Steps to Provable Security

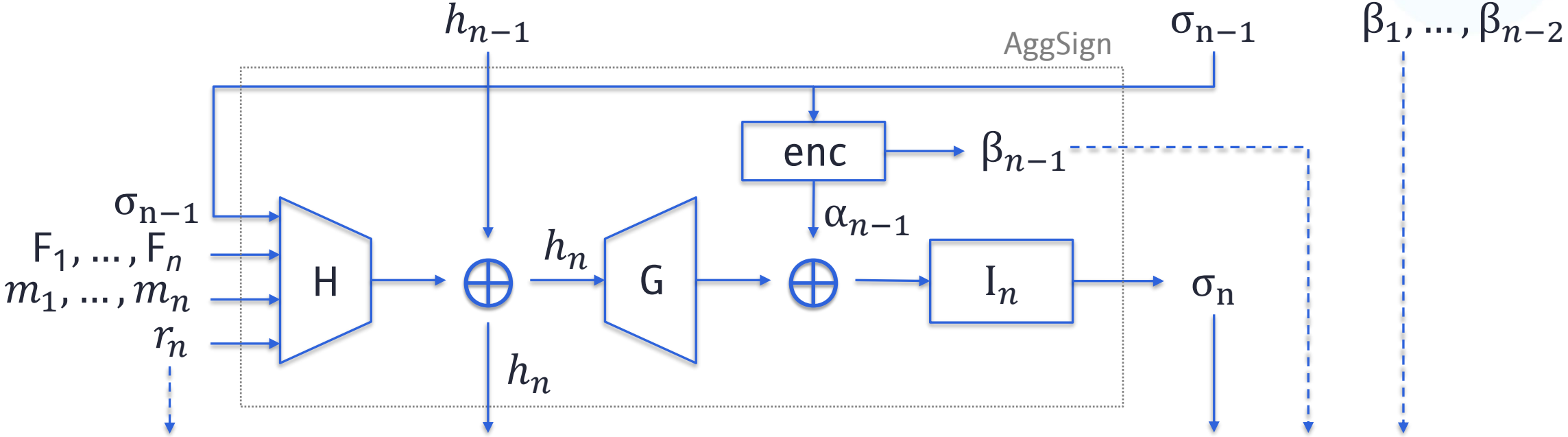
The following aggregate scheme is **not** provably secure (and sometimes provably insecure) with generic TDF.



- F_i is not injective and aggregate signatures are not unique on fixed input.
- If σ_{n-1} is part of the input to H it is not possible to directly retrieve it during verification.

Steps to Provable Security

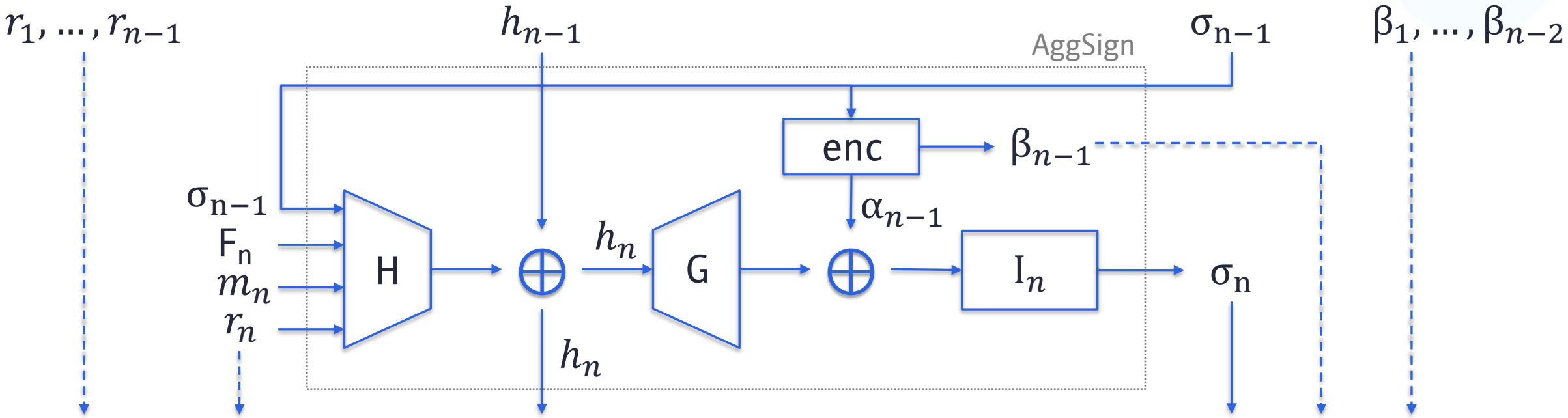
The following aggregate scheme is **not** provably secure (and sometimes provably insecure) with generic TDF.



- F_i is not injective and aggregate signatures are not unique on fixed input.
- If σ_{n-1} is part of the input to H it is not possible to directly retrieve it during verification.
- Failure on I_n may leak information.

A Secure SAS Scheme

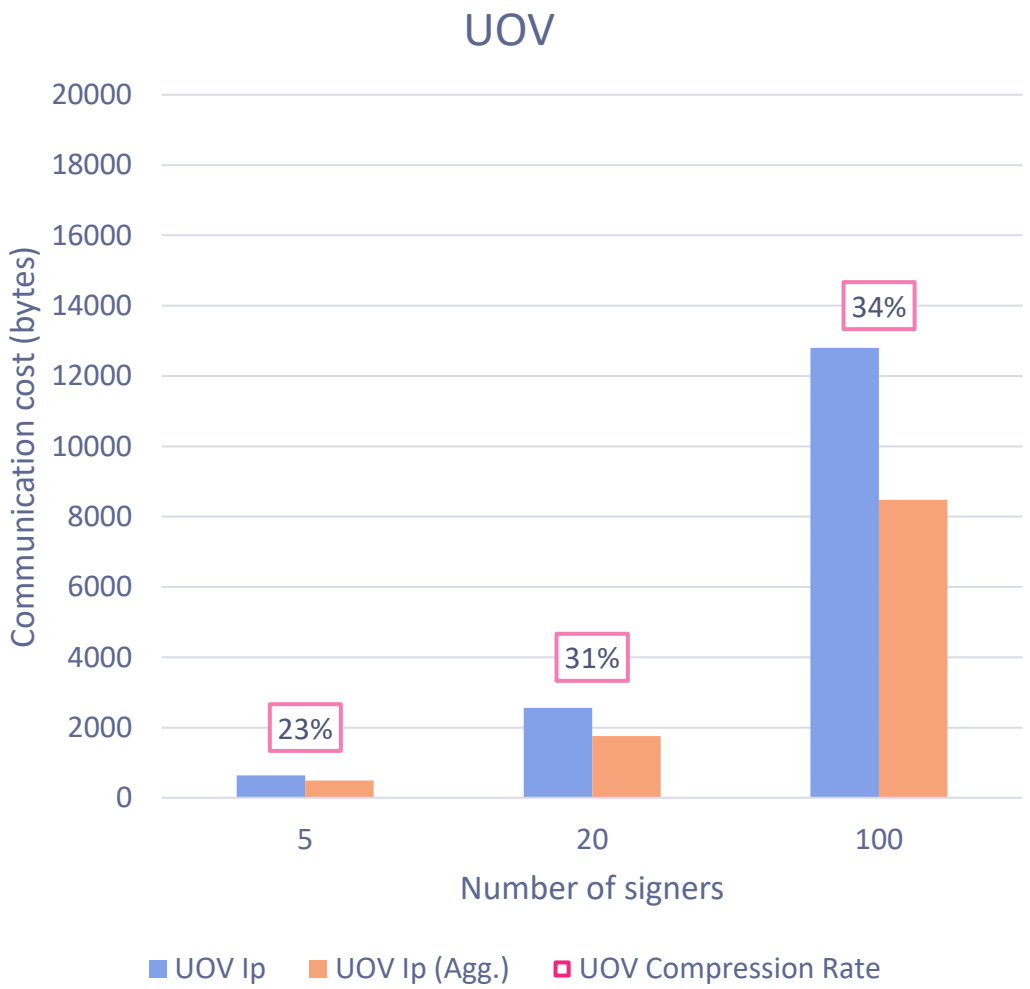
The following aggregate scheme is provably secure in the ROM with generic TDF.

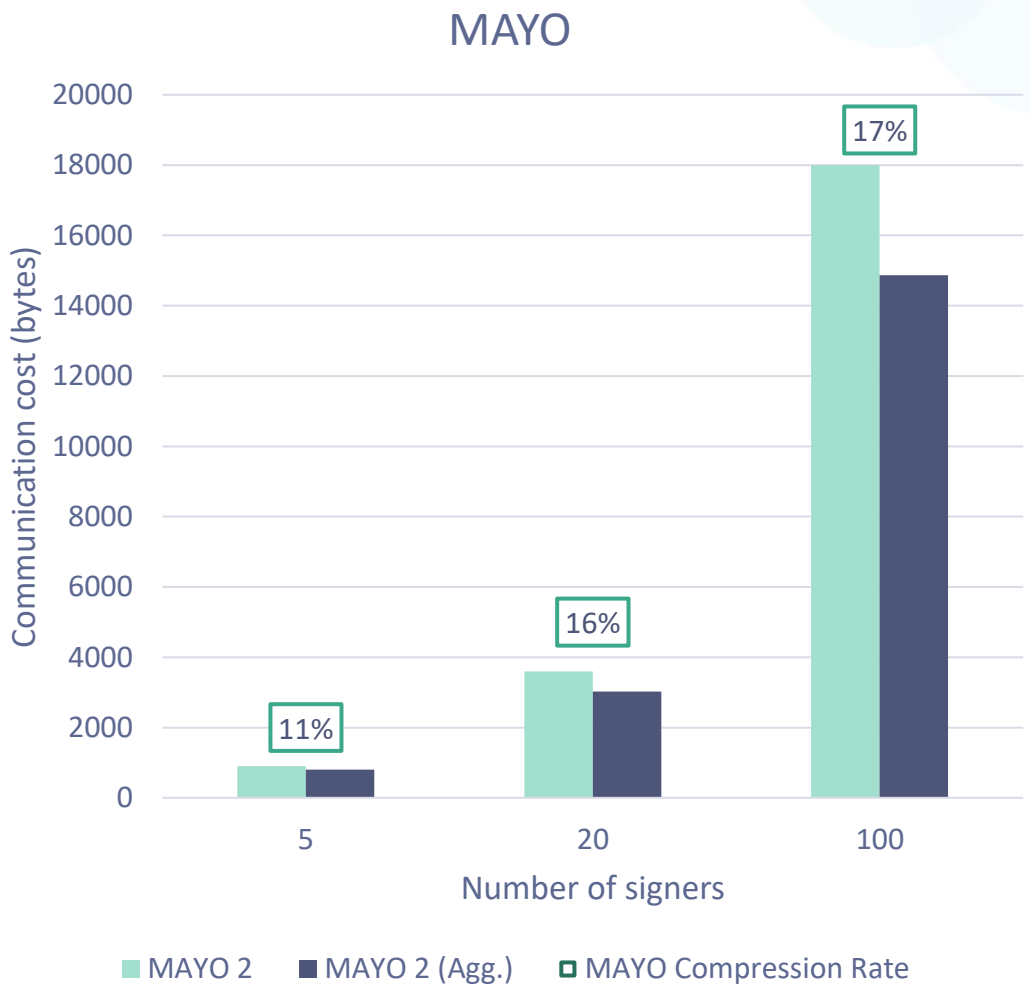


Compared with the previous construction

- Good: is provable secure (but not fully *black-box*).
- Good: is an **history-free** SAS scheme.
- Bad: the full n party signature has an overhead of length $2\lambda + n\lambda$.

Benchmarking





Conclusion

Hash-and-Sign Aggregation

- Many post-quantum trapdoor signatures are built from the hash-and-sign with retry approach.
- The same issues regarding provable security are also encountered for aggregated signatures.

Our Protocol

- Generalizes existing constructions for non-trapdoor functions.
- Recovers probable security with only a small overhead.
- The effectiveness of aggregation varies depending on the scheme and is generally not optimal.

Thank you for your attention!

References I

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps”. EUROCRYPT 2003.

[BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. “Sequential Aggregate Signatures with Lazy Verification from Trapdoor Permutations”. ASIACRYPT 2012.

[BT23] Katharina Boudgoust and Akira Takahashi. “Sequential half-aggregation of lattice-based signatures”. ESORICS 2023.

[CLNPT19] Jiahui Chen, Jie Ling, Jianting Ning, Zhiniang Peng, and Yang Tan. “MQ Aggregate Signature Schemes with Exact Security Based on UOV Signature”. Inscrypt 2019.

[EB14] Rachid El Bansarkhani and Johannes Buchmann. “Towards Lattice Based Aggregate Signatures”. AFRICACRYPT 2014.

[EMP16] Rachid El Bansarkhani, Mohamed Saied Emam Mohamed, and Albrecht Petzoldt. “MQSAS - A Multivariate Sequential Aggregate Signature Scheme”. ISC 2016.

References II

[GOR18] Craig Gentry, Adam O'Neill, and Leonid Reyzin. "A Unified Framework for Trapdoor-Permutation-Based Sequential Aggregate Signatures". PKC 2018.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. "Sequential Aggregate Signatures from Trapdoor Permutations". EUROCRYPT 2004.

[KX24] Haruhisa Kosuge and Keita Xagawa. "Probabilistic hash-and-sign with retry in the quantum random oracle model". PKC 2024.

[Nev08] Gregory Neven. "Efficient Sequential Aggregate Signed Data". EUROCRYPT 2008.

[WW19] Zhipeng Wang and Qianhong Wu. "A Practical Lattice-Based Sequential Aggregate Signature". ProvSec 2019.