Sequential Half-Aggregation of Lattice-Based Signatures

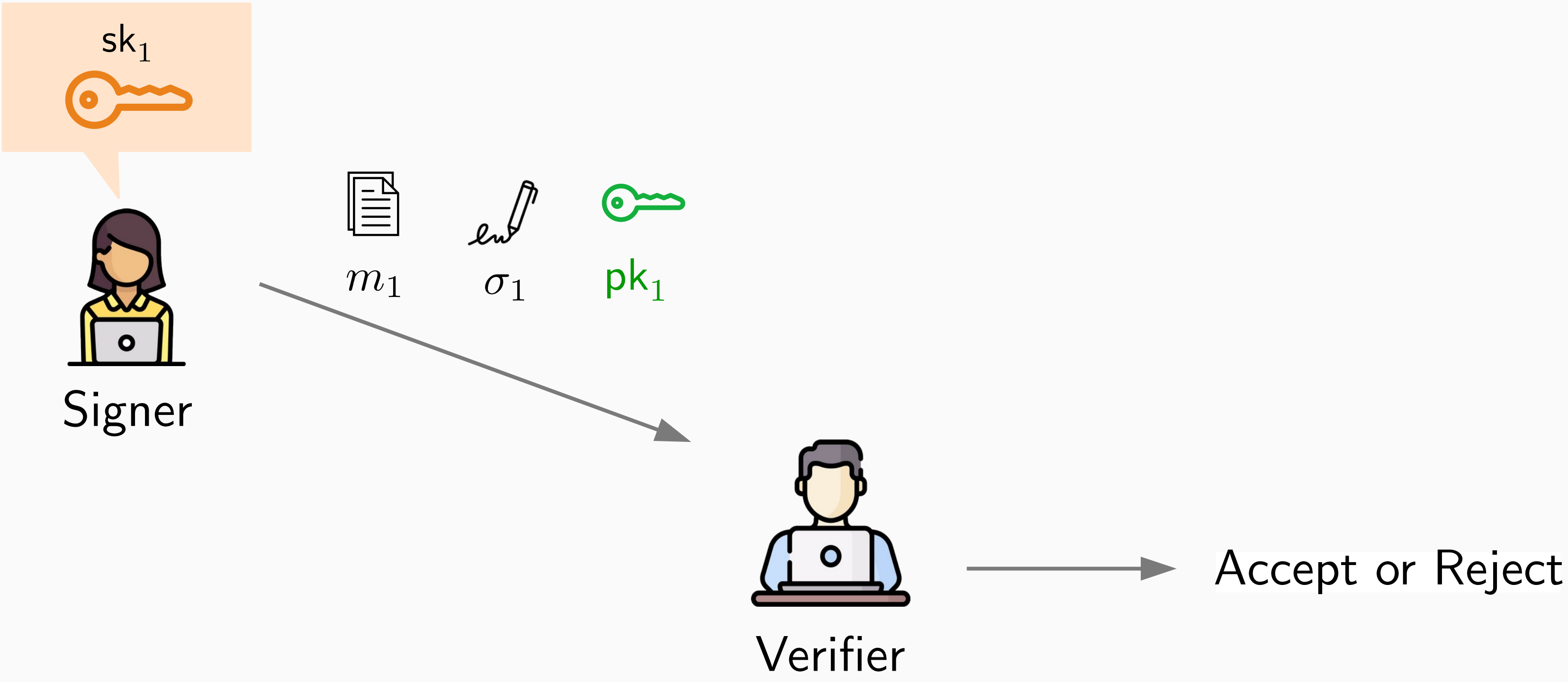
ESORICS 2023

Katharina Boudgoust and Akira Takahashi

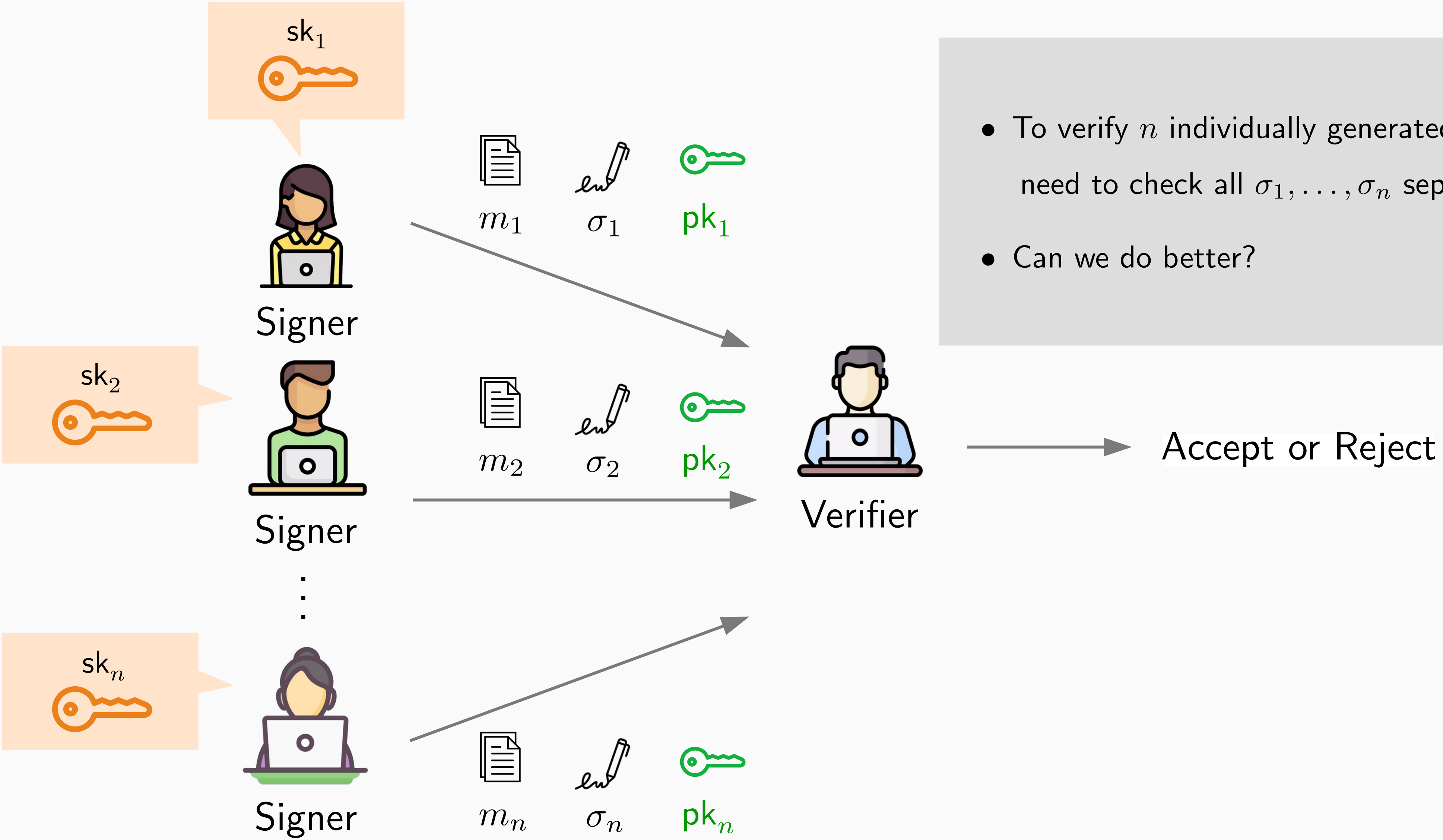
AARHUS UNIVERSITY

THE UNIVERSITY of EDINBURGH

Digital Signatures

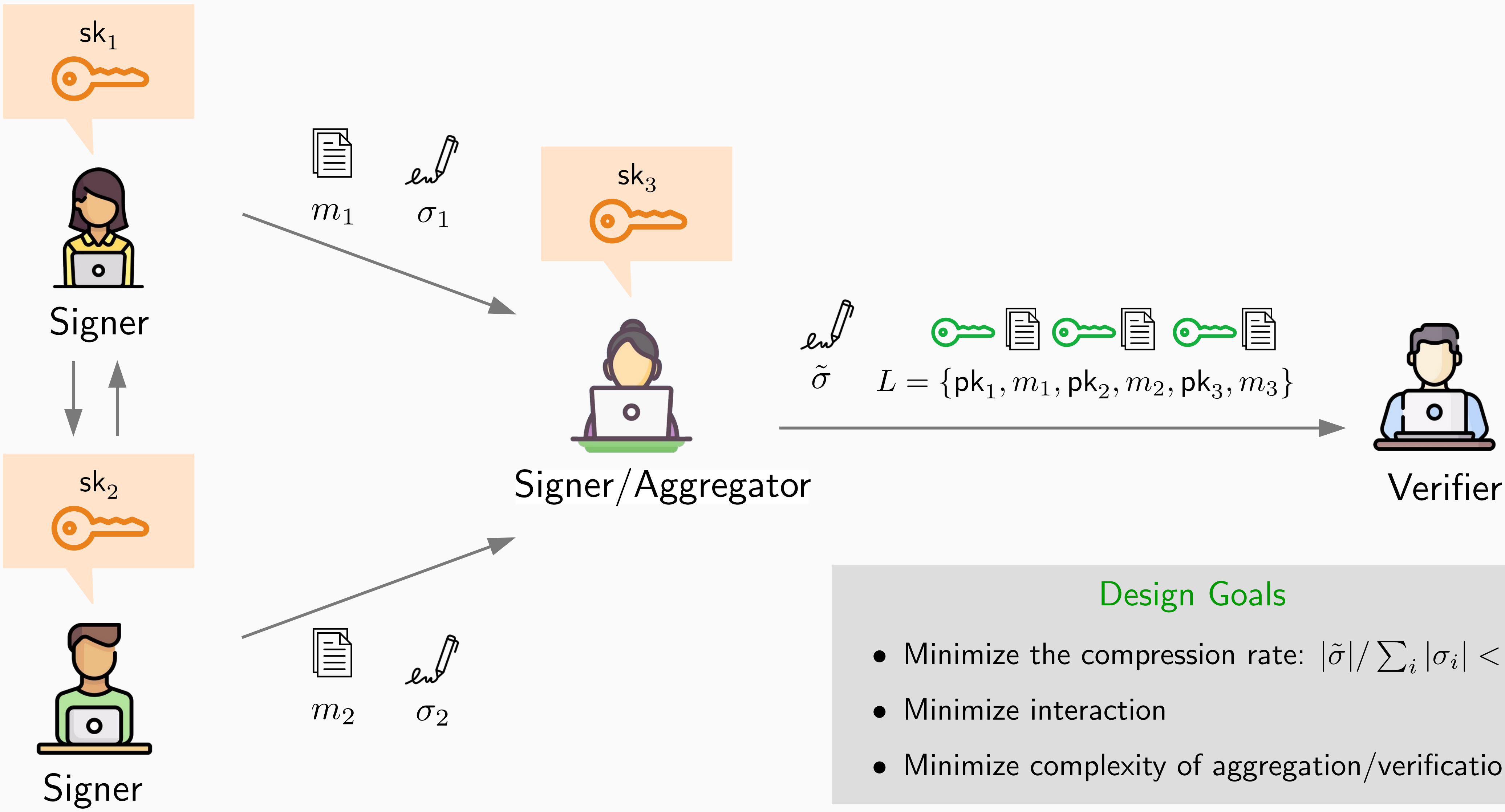


Digital Signatures



- To verify n individually generated signatures, need to check all $\sigma_1, \ldots, \sigma_n$ separately
- Can we do better?

Aggregate Signature [BGLS03]



Design Goals

- Minimize the compression rate: $|\tilde{\sigma}| / \sum_i |\sigma_i| < 1$
- Minimize interaction
- Minimize complexity of aggregation/verification

Motivation

Hash-then-Sign

FALCON

Fast-Fourier Lattice-based Compact Signatures over NTRU

Fiat-Shamir

- **Lattice-based Cryptography**: Popular paradigm for post-quantum cryptography
- NIST is standardizing two lattice-based signatures: Falcon and Dilithium
- Limited solutions to aggregate Fiat-Shamir signatures (including Dilithium):
  - Expensive generic solutions
  - Or need several rounds of interaction
  - Or larger signature than the naive concatenation!

Our Results

Hash-then-Sign

FALCON
Fast-Fourier Lattice-based
Compact Signatures over NTRU

Fiat-Shamir

1. **Forgery attacks** on two aggregate signatures based on the NIST candidates
   - Falcon-based sequential (half-)aggregate signature [WW19, ProvSec]
   - Dilithium-based interactive multi/aggregate signature [FH20, ProvSec]

2. **New** sequential Fiat-Shamir (half-)aggregate signature
   - With a signature size < naive concatenation (caveat: low compression rate)
   - Without invoking generic solutions

3-Move Identification from Module Lattices

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

$\xrightarrow{\mathbf{u}}$

Challenge

$c \leftarrow_\$ C \subset R$

$\xleftarrow{c}$

Response

$\mathbf{z} := c \cdot \mathsf{sk} + \mathbf{r}$

If $\mathsf{RejSamp}(\mathbf{z}) = 0$:

Abort

$\xrightarrow{\mathbf{z}}$

Accept iff

$\mathbf{A}\mathbf{z} = c \cdot \mathsf{pk} + \mathbf{u}$

$\wedge \|\mathbf{z}\| \leq B$

Prover(sk)

Verifier($\mathsf{pk} = \mathbf{A} \cdot \mathsf{sk}$)

- Defined in $R = \mathbb{Z}[X]/(f(X))$ and $R_q = R/qR$
- Random public matrix $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$
- "Small" secret $\mathsf{sk} \in S_\eta^{\ell+k}$

3-Move Identification from Module Lattices

- $\textsf{sk}, \mathbf{r}$, and c are small $\rightsquigarrow \mathbf{z}$ is also small
- RejSamp forces $\mathbf{z}$ to be independent of $\textsf{sk}$

- Defined in $R = \mathbb{Z}[X]/(f(X))$ and $R_q = R/qR$
- Random public matrix $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$
- "Small" secret $\textsf{sk} \in S_\eta^{\ell+k}$

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

$\xrightarrow{\mathbf{u}}$

Challenge

$c \leftarrow_{\$} C \subset R$

$\xleftarrow{c}$

Response

$\mathbf{z} := c \cdot \textsf{sk} + \mathbf{r}$

If $\text{RejSamp}(\mathbf{z}) = 0$:

Abort

$\xrightarrow{\mathbf{z}}$

Accept iff

$\mathbf{A}\mathbf{z} = c \cdot \textsf{pk} + \mathbf{u}$

$\wedge \|\mathbf{z}\| \leq B$

Prover($\textsf{sk}$)

Verifier($\textsf{pk} = \mathbf{A} \cdot \textsf{sk}$)

3-Move Identification from Module Lattices

Short Integer Solution (SIS) Problem

Given $\bar{\mathbf{A}} \leftarrow_{\$} R_q^{k\times\ell}$, find $\mathbf{x} \in R^{\ell+k}$

s.t. $\|\mathbf{x}\| \le \beta \wedge [\bar{\mathbf{A}}|\mathbf{I}]\mathbf{x} = \mathbf{0} \bmod q$

- sk, $\mathbf{r}$, and c are small $\rightsquigarrow$ $\mathbf{z}$ is also small
- RejSamp forces $\mathbf{z}$ to be independent of sk

- Defined in $R = \mathbb{Z}[X]/(f(X))$ and $R_q = R/qR$
- Random public matrix $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$
- "Small" secret sk $\in S_\eta^{\ell+k}$

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

$\xrightarrow{\mathbf{u}}$

Challenge

$c \leftarrow_{\$} C \subset R$

$\xleftarrow{c}$

Response

$\mathbf{z} := c \cdot \mathsf{sk} + \mathbf{r}$

If RejSamp$(\mathbf{z}) = 0$:

Abort

$\xrightarrow{\mathbf{z}}$

Accept iff

$\mathbf{A}\mathbf{z} = c \cdot \mathsf{pk} + \mathbf{u}$

$\wedge \|\mathbf{z}\| \le B$

Prover(sk)

Verifier(pk $= \mathbf{A} \cdot$ sk)

Fiat-Shamir Signature from Module Lattices

Short Integer Solution (SIS) Problem

Given $\bar{\mathbf{A}} \leftarrow_{\$} R_q^{k\times\ell}$, find $\mathbf{x} \in R^{\ell+k}$

s.t. $\|\mathbf{x}\| \leq \beta \wedge [\bar{\mathbf{A}}|\mathbf{I}]\mathbf{x} = \mathbf{0} \bmod q$

- sk, $\mathbf{r}$, and c are small $\rightsquigarrow$ $\mathbf{z}$ is also small
- RejSamp forces $\mathbf{z}$ to be independent of sk

- Defined in $R = \mathbb{Z}[X]/(f(X))$ and $R_q = R/qR$
- Random public matrix $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$
- "Small" secret sk $\in S_\eta^{\ell+k}$

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

Hash

$c := \mathsf{H}(\mathbf{u}, m, \mathsf{pk})$

Response

$\mathbf{z} := c \cdot \mathsf{sk} + \mathbf{r}$

If RejSamp$(\mathbf{z}) = 0$:

Abort

$\sigma = (c, \mathbf{z})$ $\longrightarrow$

$\mathbf{u} := \mathbf{A}\mathbf{z} - c \cdot \mathsf{pk}$

Accept iff

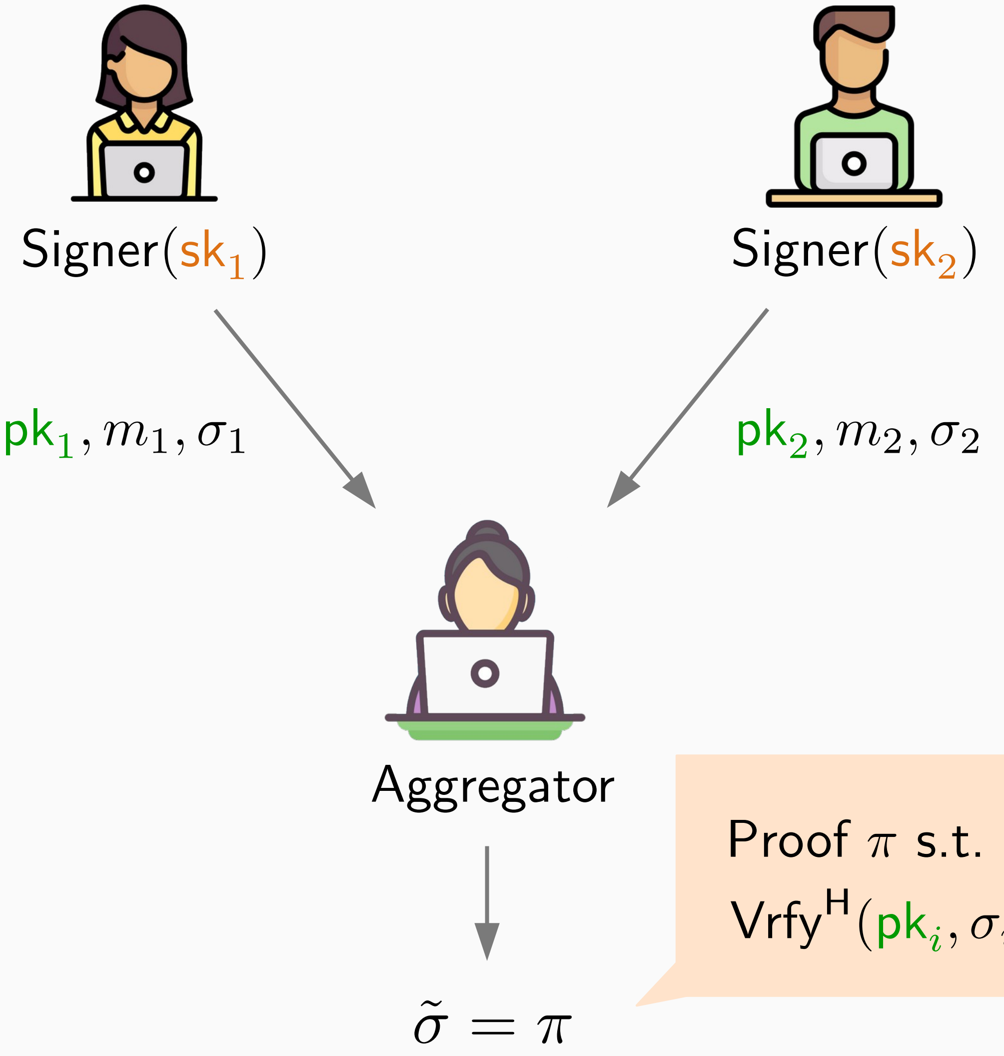
$c = \mathsf{H}(\mathbf{u}, m, \mathsf{pk})$

$\wedge \|\mathbf{z}\| \leq B$

Signer(sk)

Verifier($\mathsf{pk} = \mathbf{A} \cdot \mathsf{sk}$)

Approaches to Aggregate FS Lattice Signatures

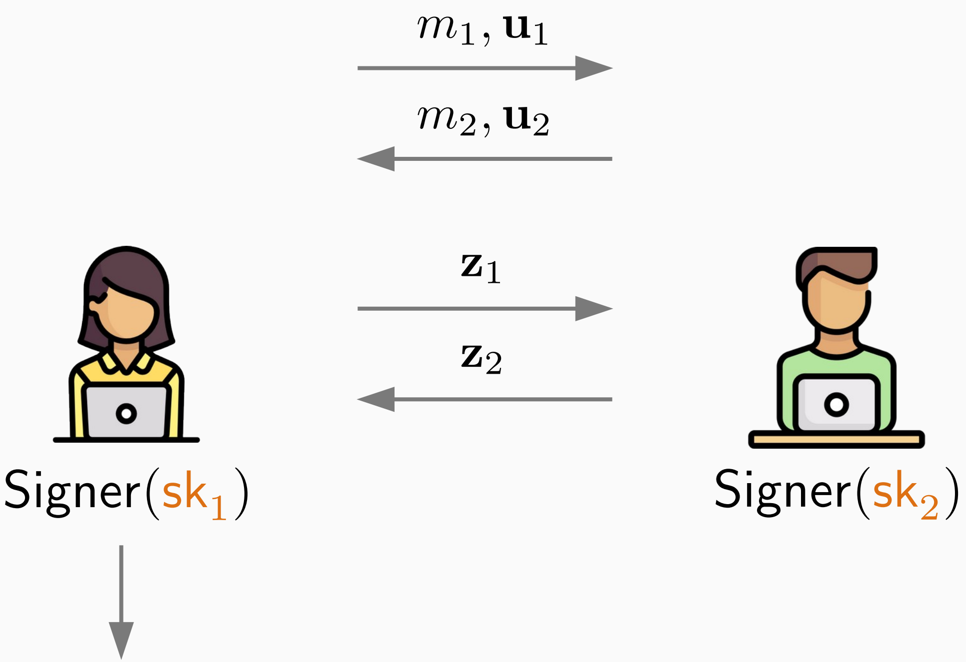


Proof π s.t. $\forall i$

$\mathsf{Vrfy}^{\mathsf{H}}(\mathsf{pk}_i, \sigma_i, m_i) = 1$

SNARK/BARG [DGKV22,ACL+22]	
Pros	Cons
• Generic solution • Compact $\tilde{\sigma}$ • No modification to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

Approaches to Aggregate FS Lattice Signatures



$\tilde{\sigma} = (\tilde{c}, \tilde{\mathbf{z}})$

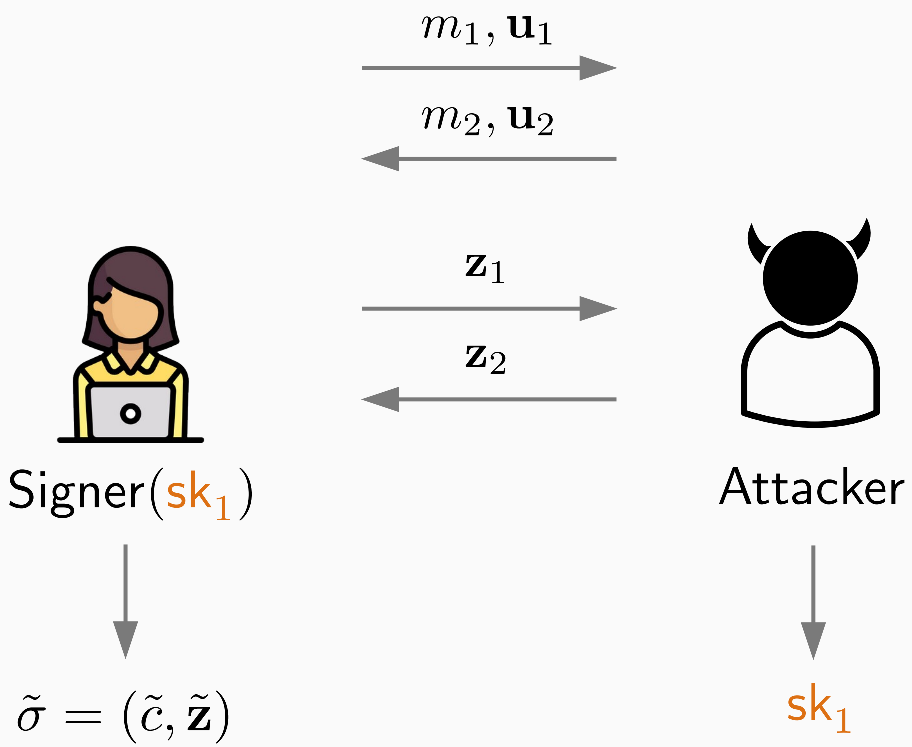
$$\tilde{c} = \mathsf{H}(\mathbf{u}_1 + \mathbf{u}_2, m_1, m_2, \mathsf{pk}_1, \mathsf{pk}_2)$$

$$\tilde{\mathbf{z}} = \mathbf{z}_1 + \mathbf{z}_2$$

SNARK/BARG [DGKV22,ACL+22]	
Pros	**Cons**
• Generic solution • Compact $\tilde{\sigma}$ • No modification to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

Interactive Aggregation [ES16,BK20,DOTT21,FH20,..]	
Pros	**Cons**
• $\lvert\tilde{\sigma}\rvert = O(\log^c n)$	• 2-3 rounds of interaction between all signers • Leaks sk$_1$ if instantiated with Dilithium (this work)

Approaches to Aggregate FS Lattice Signatures



$$\tilde{c} = \mathsf{H}(\mathbf{u}_1 + \mathbf{u}_2, m_1, m_2, \mathsf{pk}_1, \mathsf{pk}_2)$$

$$\tilde{\mathbf{z}} = \mathbf{z}_1 + \mathbf{z}_2$$

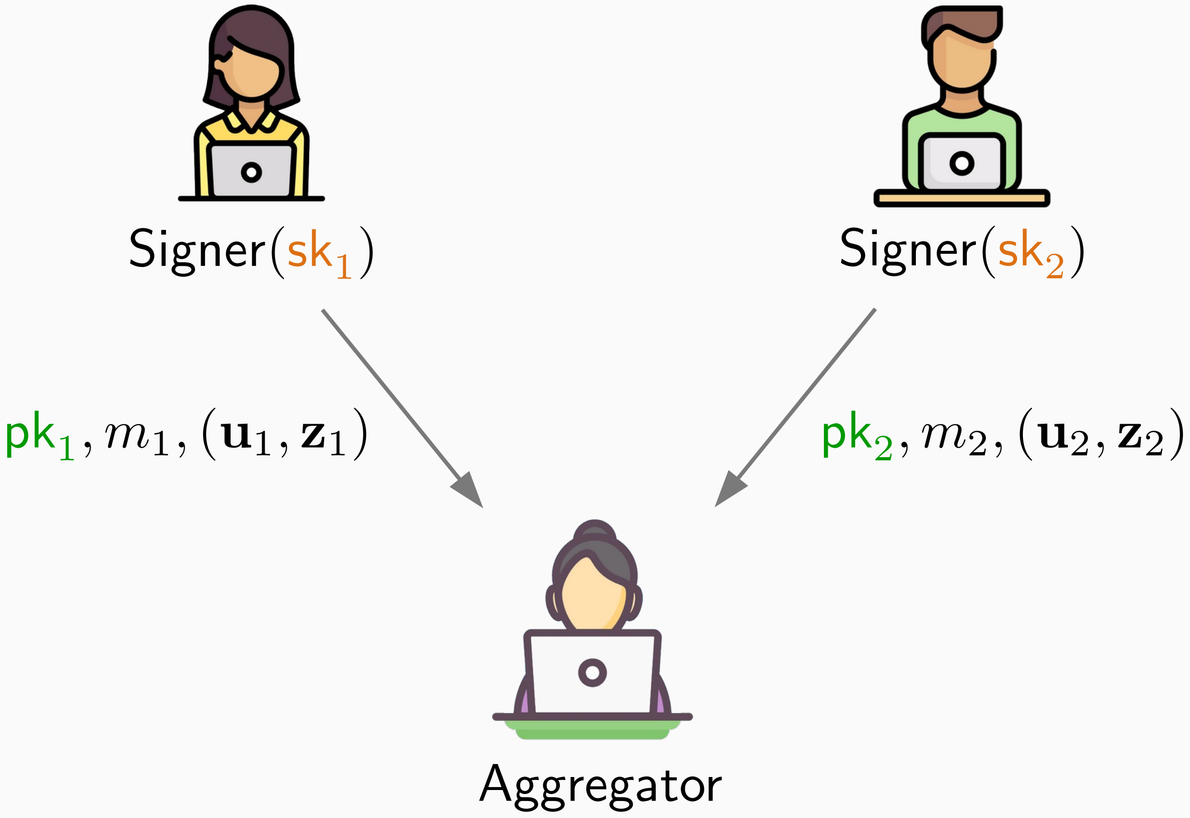
SNARK/BARG [DGKV22,ACL+22]

Pros	Cons
• Generic solution • Compact $\tilde{\sigma}$ • No modification to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

Interactive Aggregation [ES16,BK20,DOTT21,FH20,..]

Pros	Cons
• $\lvert\tilde{\sigma}\rvert = O(\log^c n)$	• 2-3 rounds of interaction between all signers • Leaks sk_1 if instantiated with Dilithium (this work)

Approaches to Aggregate FS Lattice Signatures



$$c_i = \mathsf{H}(\mathbf{u}_i, m_i, \mathsf{pk}_i), \forall i$$
$$e_i = \mathsf{H}_e(c_1, c_2, i), \forall i$$
$$\tilde{\mathbf{z}} = e_1\mathbf{z}_1 + e_2\mathbf{z}_2$$

Compressed!

$$\tilde{\sigma} = (\mathbf{u}_1, \mathbf{u}_2, \boxed{\tilde{\mathbf{z}}})$$

SNARK/BARG [DGKV22,ACL+22]

Pros	Cons
• Generic solution • Compact $\tilde{\sigma}$ • No modification to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

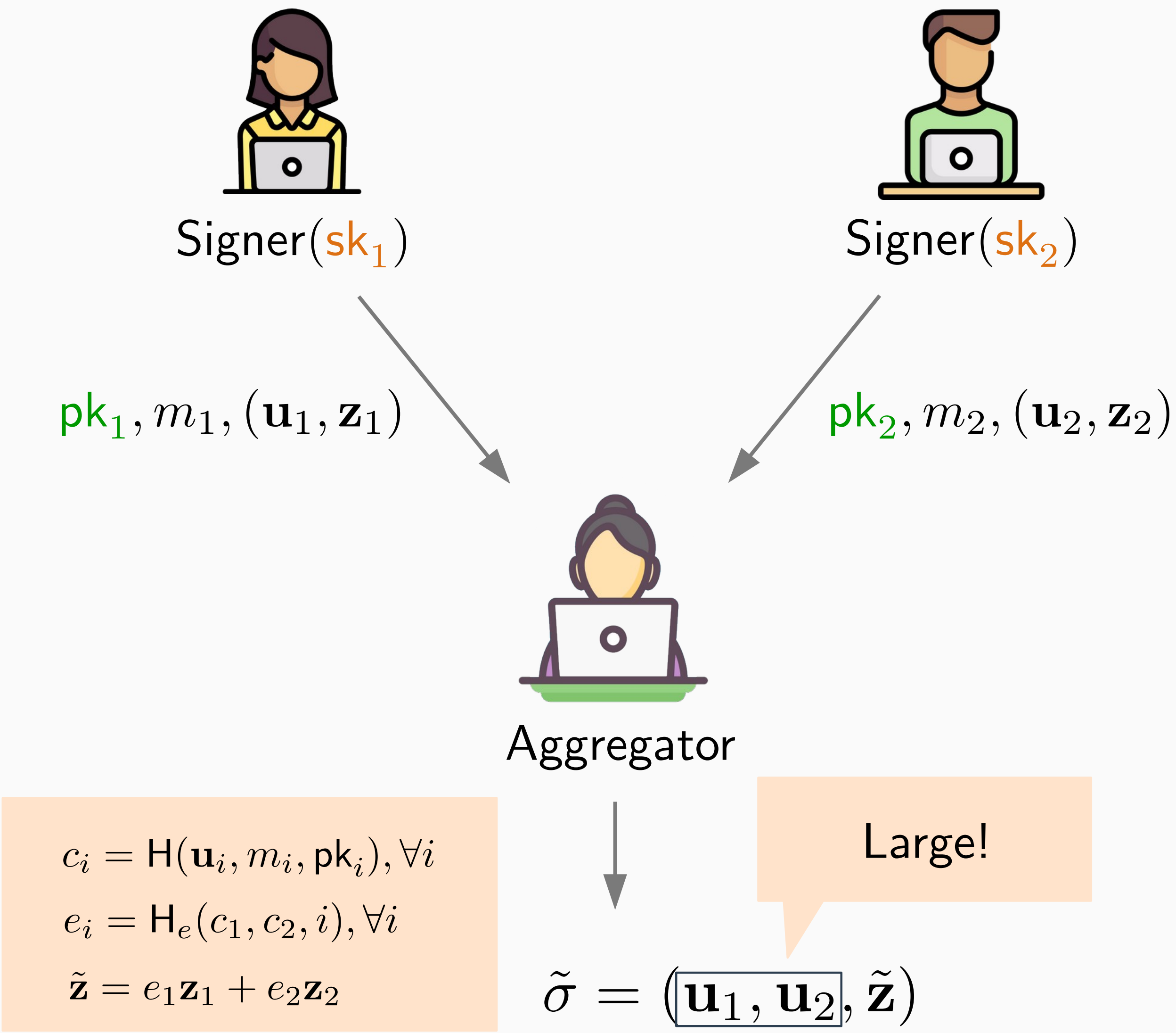
Interactive Aggregation [ES16,BK20,DOTT21,FH20,..]

Pros	Cons
• $\lvert\tilde{\sigma}\rvert = O(\log^c n)$	• 2-3 rounds of interaction between all signers • Leaks sk_1 if instantiated with Dilithium (this work)

Half-Aggregation: Combine the $\mathbf{z}_i$-parts [BR23]

Pros	Cons
• Fast Aggregator • No modification to Sign	• Larger $\tilde{\sigma}$ than the trivial concatenation! $\lvert\tilde{\sigma}\rvert > \lvert(c_1, \mathbf{z}_1, c_2, \mathbf{z}_2)\rvert$

Approaches to Aggregate FS Lattice Signatures



SNARK/BARG [DGKV22,ACL+22]

Pros	Cons
• Generic solution • Compact $\tilde{\sigma}$ • No modification to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

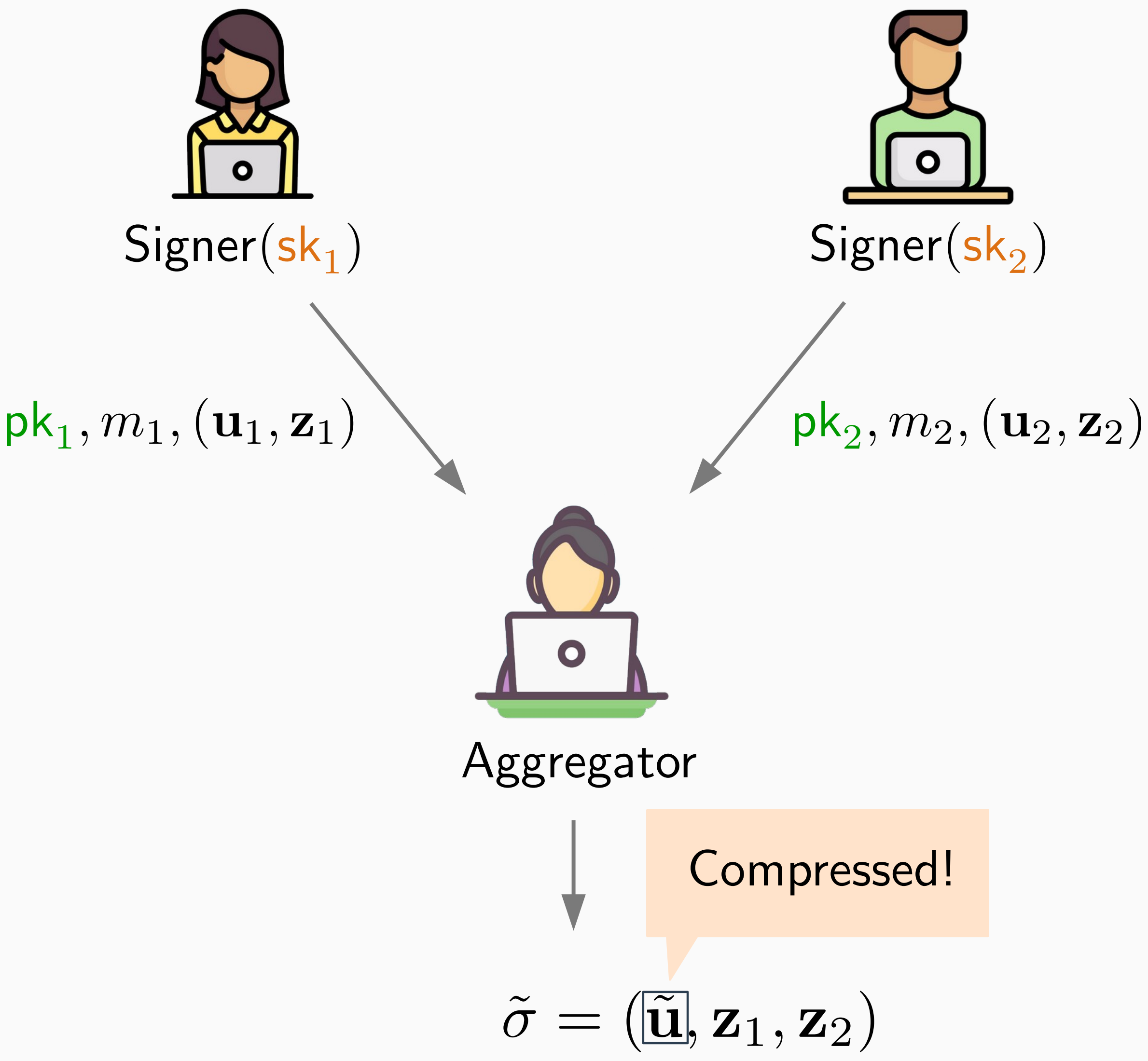
Interactive Aggregation [ES16,BK20,DOTT21,FH20,..]

Pros	Cons
• $\lvert\tilde{\sigma}\rvert = O(\log^c n)$	• 2-3 rounds of interaction between all signers • Leaks sk_1 if instantiated with Dilithium (this work)

Half-Aggregation: Combine the $\mathbf{z}_i$-parts [BR23]

Pros	Cons
• Fast Aggregator • No modification to Sign	• Larger $\tilde{\sigma}$ than the trivial concatenation! $\lvert\tilde{\sigma}\rvert > \lvert(c_1, \mathbf{z}_1, c_2, \mathbf{z}_2)\rvert$

Can We Aggregate the $\mathbf{u}_i$-parts?

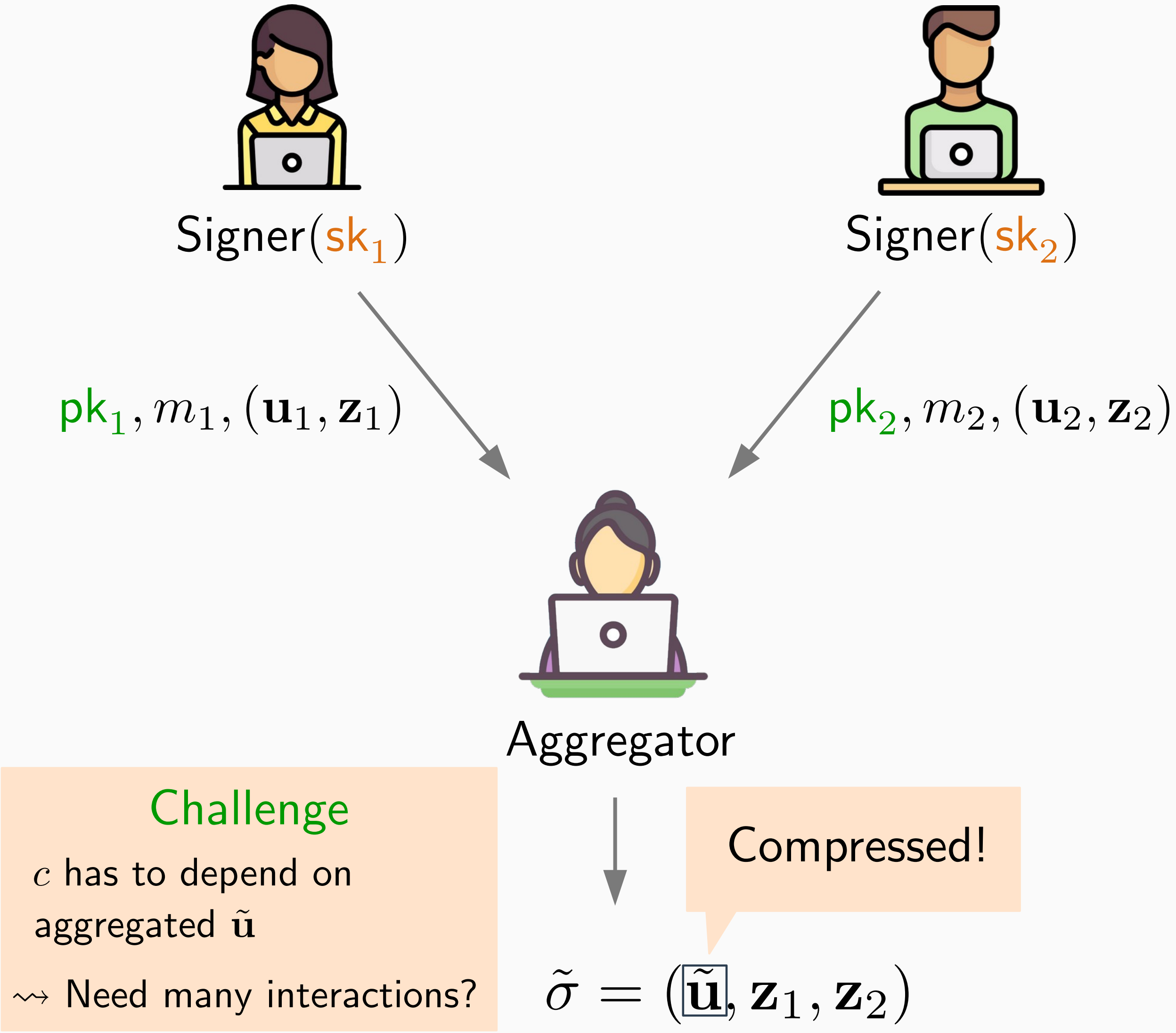


SNARK/BARG [DGKV22,ACL+22]	
Pros	**Cons**
• Generic solution • Compact $\tilde{\sigma}$ • No modificatoin to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

Interactive Aggregation [ES16,BK20,DOTT21,FH20,..]	
Pros	**Cons**
• $\lvert\tilde{\sigma}\rvert = O(\log^c n)$	• 2-3 rounds of interaction between all signers • Leaks sk_1 if instantiated with Dilithium (this work)

Half-Aggregation: Combine the $\mathbf{z}_i$-parts [BR23]	
Pros	**Cons**
• Fast Aggregator • No modification to Sign	• Larger $\tilde{\sigma}$ than the trivial concatenation! $\lvert\tilde{\sigma}\rvert > \lvert(c_1, \mathbf{z}_1, c_2, \mathbf{z}_2)\rvert$

Can We Aggregate the $\mathbf{u}_i$-parts?



SNARK/BARG [DGKV22,ACL+22]

Pros	Cons
• Generic solution • Compact $\tilde{\sigma}$ • No modificatoin to Sign	• Expensive Aggregator • Heuristic security proof if Vrfy calls the RO H

Interactive Aggregation [ES16,BK20,DOTT21,FH20,..]

Pros	Cons
• $\lvert\tilde{\sigma}\rvert = O(\log^c n)$	• 2-3 rounds of interaction between all signers • Leaks sk_1 if instantiated with Dilithium (this work)

Half-Aggregation: Combine the $\mathbf{z}_i$-parts [BR23]

Pros	Cons
• Fast Aggregator • No modification to Sign	• Larger $\tilde{\sigma}$ than the trivial concatenation! $\lvert\tilde{\sigma}\rvert > \lvert(c_1, \mathbf{z}_1, c_2, \mathbf{z}_2)\rvert$

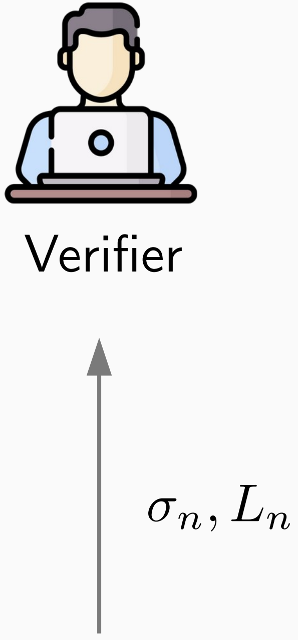
Sequential Aggregate Signature (SAS) [LMRS04]

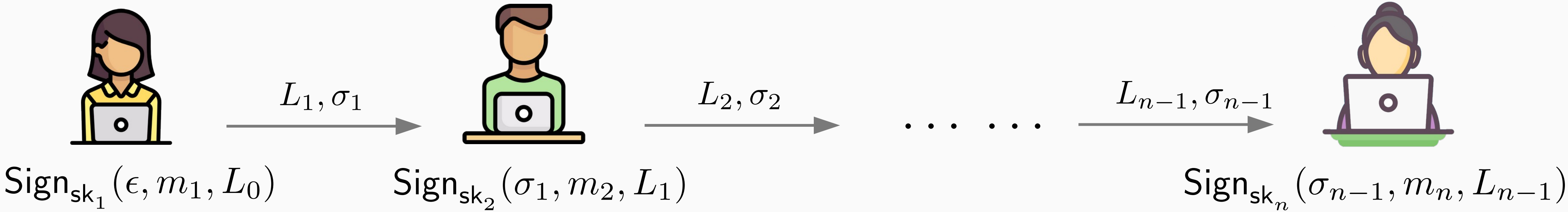
- Signers sequentially update σ_i cf. cert chain, BGPsec, etc.
- An evolving ordered set L_i

$$L_i = \{(m_1, \mathsf{pk}_1), \ldots, (m_i, \mathsf{pk}_i))\}$$

- Known constructions from trapdoors, pairing, etc.
- Recent Schnorr-based SAS (Chen-Zhao, ESORICS'22)

$\rightsquigarrow$ Can we adapt it to lattice-based FS?





Our SAS from Lattice-based FS

Commit

$\mathbf{r}_1 \leftarrow D$
$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$
$\tilde{\mathbf{u}}_1 := \mathbf{u}_1$

Hash

$L_1 := (m_1, \mathsf{pk}_1)$
$c_1 := \mathsf{H}(\mathbf{u}_1, L_1, \epsilon)$

Response

$\mathbf{z}_1 := c_1 \cdot \mathsf{sk}_1 + \mathbf{r}_1$
If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:
Go to Commit
$\sigma_1 := (\tilde{\mathbf{u}}_1, \mathbf{z}_1)$

Verifier

σ_n, L_n

$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m_1, L_0)$

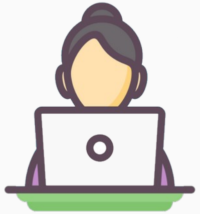
L_1, σ_1

$\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m_2, L_1)$

L_2, σ_2

$\cdots \quad \cdots$

L_{n-1}, σ_{n-1}



$\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m_n, L_{n-1})$

Our SAS from Lattice-based FS

Commit

$\mathbf{r}_1 \leftarrow D$

$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$

$\tilde{\mathbf{u}}_1 := \mathbf{u}_1$

Hash

$L_1 := (m_1, \mathsf{pk}_1)$

$c_1 := \mathsf{H}(\mathbf{u}_1, L_1, \epsilon)$

Response

$\mathbf{z}_1 := c_1 \cdot \mathsf{sk}_1 + \mathbf{r}_1$

If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:

Go to Commit

$\sigma_1 := (\tilde{\mathbf{u}}_1, \mathbf{z}_1)$

Commit

$\mathbf{r}_2 \leftarrow D$

$\mathbf{u}_2 := \mathbf{A}\mathbf{r}_2$

$\boxed{\tilde{\mathbf{u}}_2 := \tilde{\mathbf{u}}_1 + \mathbf{u}_2}$

Hash

$\boxed{L_2 := L_1 || (m_2, \mathsf{pk}_2)}$

$c_2 := \mathsf{H}(\tilde{\mathbf{u}}_2, L_2, \boxed{\mathbf{z}_1})$

Response

$\mathbf{z}_2 := c_2 \cdot \mathsf{sk}_2 + \mathbf{r}_2$

If $\mathsf{RejSamp}(\mathbf{z}_2) = 0$:

Go to Commit

$\sigma_2 := (\tilde{\mathbf{u}}_2, \mathbf{z}_1, \mathbf{z}_2)$

Verifier

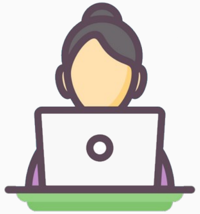
σ_n, L_n

$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m_1, L_0)$

$\xrightarrow{L_1, \sigma_1}$

$\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m_2, L_1)$

$\xrightarrow{L_2, \sigma_2}$ . . . . . . $\xrightarrow{L_{n-1}, \sigma_{n-1}}$



$\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m_n, L_{n-1})$

Our SAS from Lattice-based FS

Commit
$\mathbf{r}_1 \leftarrow D$
$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$
$\tilde{\mathbf{u}}_1 := \mathbf{u}_1$

Hash
$L_1 := (m_1, \mathsf{pk}_1)$
$c_1 := \mathsf{H}(\mathbf{u}_1, L_1, \epsilon)$

Response
$\mathbf{z}_1 := c_1 \cdot \mathsf{sk}_1 + \mathbf{r}_1$
If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:
Go to Commit
$\sigma_1 := (\tilde{\mathbf{u}}_1, \mathbf{z}_1)$

$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m_1, L_0)$

L_1, σ_1

Commit
$\mathbf{r}_2 \leftarrow D$
$\mathbf{u}_2 := \mathbf{A}\mathbf{r}_2$
$\boxed{\tilde{\mathbf{u}}_2 := \tilde{\mathbf{u}}_1 + \mathbf{u}_2}$

Hash
$\boxed{L_2 := L_1 || (m_2, \mathsf{pk}_2)}$
$c_2 := \mathsf{H}(\tilde{\mathbf{u}}_2, L_2, \boxed{\mathbf{z}_1})$

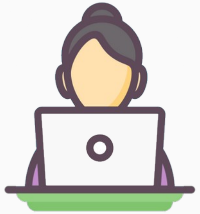
Response
$\mathbf{z}_2 := c_2 \cdot \mathsf{sk}_2 + \mathbf{r}_2$
If $\mathsf{RejSamp}(\mathbf{z}_2) = 0$:
Go to Commit
$\sigma_2 := (\tilde{\mathbf{u}}_2, \mathbf{z}_1, \mathbf{z}_2)$

$\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m_2, L_1)$

L_2, σ_2

. . . . . .

L_{n-1}, σ_{n-1}



$\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m_n, L_{n-1})$

σ_n, L_n

Parse $\sigma_n = (\tilde{\mathbf{u}}_n, \mathbf{z}_1, \ldots, \mathbf{z}_n)$

For $i = n, \ldots, 1$:
Check $\|\mathbf{z}_i\| \leq B$
$c_i := \mathsf{H}(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$
$\mathbf{u}_i := \mathbf{A}\mathbf{z}_i - c_i \cdot \mathsf{pk}_i$
$\tilde{\mathbf{u}}_{i-1} := \tilde{\mathbf{u}}_i - \mathbf{u}_i$

Check $\tilde{\mathbf{u}}_0 = \mathbf{0}$

Verifier

Commit

$\mathbf{r}_1 \leftarrow D$

$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$

$\tilde{\mathbf{u}}_1 := \mathbf{u}_1$

Hash

$L_1 := (m_1, \mathsf{pk}_1)$

$c_1 := \mathsf{H}(\mathbf{u}_1, L_1, \epsilon)$

Response

$\mathbf{z}_1 := c_1 \cdot \mathsf{sk}_1 + \mathbf{r}_1$

If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:

Go to Commit

$\sigma_1 := (\tilde{\mathbf{u}}_1, \mathbf{z}_1)$

Commit

$\mathbf{r}_2 \leftarrow D$

$\mathbf{u}_2 := \mathbf{A}\mathbf{r}_2$

$\boxed{\tilde{\mathbf{u}}_2 := \tilde{\mathbf{u}}_1 + \mathbf{u}_2}$

Hash

$\boxed{L_2 := L_1 || (m_2, \mathsf{pk}_2)}$

$c_2 := \mathsf{H}(\tilde{\mathbf{u}}_2, L_2, \boxed{\mathbf{z}_1})$

Response

$\mathbf{z}_2 := c_2 \cdot \mathsf{sk}_2 + \mathbf{r}_2$

If $\mathsf{RejSamp}(\mathbf{z}_2) = 0$:

Go to Commit

$\sigma_2 := (\tilde{\mathbf{u}}_2, \mathbf{z}_1, \mathbf{z}_2)$

Provable Security

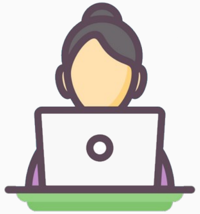
- If the single-user FS lattice signature is EUF-CMA secure then our SAS is also secure (in the ROM)
- Tight security reduction
- No need to change the original parameters

L_1, σ_1

L_2, σ_2

. . . . . .

L_{n-1}, σ_{n-1}



$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m_1, L_0)$ $\quad$ $\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m_2, L_1)$ $\quad$ $\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m_n, L_{n-1})$

Security and Performance Estimates

Commit

$\mathbf{r}_1 \leftarrow D$

$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$

$\tilde{\mathbf{u}}_1 := \mathbf{u}_1$

Hash

$L_1 := (m_1, \mathsf{pk}_1)$

$c_1 := \mathsf{H}(\mathbf{u}_1, L_1, \epsilon)$

Response

$\mathbf{z}_1 := c_1 \cdot \mathsf{sk}_1 + \mathbf{r}_1$

If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:

Go to Commit

$\sigma_1 := (\tilde{\mathbf{u}}_1, \mathbf{z}_1)$

Commit

$\mathbf{r}_2 \leftarrow D$

$\mathbf{u}_2 := \mathbf{A}\mathbf{r}_2$

$\boxed{\tilde{\mathbf{u}}_2 := \tilde{\mathbf{u}}_1 + \mathbf{u}_2}$

Hash

$\boxed{L_2 := L_1 || (m_2, \mathsf{pk}_2)}$

$c_2 := \mathsf{H}(\tilde{\mathbf{u}}_2, L_2, \boxed{\mathbf{z}_1})$

Response

$\mathbf{z}_2 := c_2 \cdot \mathsf{sk}_2 + \mathbf{r}_2$

If $\mathsf{RejSamp}(\mathbf{z}_2) = 0$:

Go to Commit

$\sigma_2 := (\tilde{\mathbf{u}}_2, \mathbf{z}_1, \mathbf{z}_2)$

Provable Security

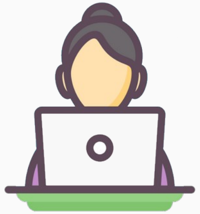
- If the single-user FS lattice signature is EUF-CMA secure then our SAS is also secure (in the ROM)
- Tight security reduction
- No need to change the original parameters

Performance Estimates

- Better than the naive concatenation, but not dramatically
- Saves $\approx 1\%$ of the signature size (w/ Dilithium)
- Open question: optimal compression rate (i.e. 50%) with other FS lattice schemes?

$\mathsf{Sign}_{\mathsf{sk}_1}(\epsilon, m_1, L_0)$ $\xrightarrow{L_1, \sigma_1}$

$\mathsf{Sign}_{\mathsf{sk}_2}(\sigma_1, m_2, L_1)$ $\xrightarrow{L_2, \sigma_2}$ $\cdots \quad \cdots$ $\xrightarrow{L_{n-1}, \sigma_{n-1}}$



$\mathsf{Sign}_{\mathsf{sk}_n}(\sigma_{n-1}, m_n, L_{n-1})$

Wrapping up

- Constructed SAS tailored to Fiat-Shamir lattice signatures, with concrete size estimates based on the Dilithium parameter sets

- Paper compares with an existing hash-then-sign SAS
    - Turned out a Falcon-based SAS also only saves ~ 3%

- Paper describes forgery attacks against existing AS schemes from Falcon and Dilithium
    - Illustrates the sensitivity of various optimization techniques

- Take away: Non-trivial aggregation of lattice-based signatures is **hard**!

- Open questions:
    - Can we improve the compression rate for FS-based (S)AS?
    - How efficient can the generic solutions be for aggregating Falcon?

Wrapping up

- Constructed SAS tailored to Fiat-Shamir lattice signatures, with concrete size estimates based on the Dilithium parameter sets
- Paper compares with an existing hash-then-sign SAS
  - Turned out a Falcon-based SAS also only saves ~ 3%
- Paper describes forgery attacks against existing AS schemes from Falcon and Dilithium
  - Illustrates the sensitivity of various optimization techniques
- Take away: Non-trivial aggregation of lattice-based signatures is **hard**!
- Open questions:
  - Can we improve the compression rate for FS-based (S)AS?
  - How efficient can the generic solutions be for aggregating Falcon?

Thank you!

ePrint 2023/159 for details

Insecurity of Interactive Aggregation of Dilithium

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

$\boxed{\mathbf{u}' := \mathsf{HighBit}(\mathbf{u})}$

Hash

$c := \mathsf{H}(\mathbf{u}', m, \mathsf{pk})$

Response

$\mathbf{z} := c \cdot \mathbf{s}_1 + \mathbf{r}$

If $\mathsf{RejSamp}(\mathbf{z}) = 0$:

Go to Commit

$\sigma := (c, \mathbf{z})$

Signer$(\mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2), \mathsf{pk} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$

Optimization Technique in Dilithium

- Drop the lower bits of $\mathbf{u}$ while preserving correctness:

$$\begin{aligned}\mathsf{HighBit}(\mathbf{A}\mathbf{z} - c \cdot \mathsf{pk}) &= \mathsf{HighBit}(\mathbf{A}\mathbf{r} - c \cdot \mathbf{s}_2)\\ &\approx \mathsf{HighBit}(\mathbf{u})\end{aligned}$$

Insecurity of Interactive Aggregation of Dilithium

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

$\mathbf{u}' := \text{HighBit}(\mathbf{u} + \mathbf{u}^*)$

Hash

$c := \mathsf{H}(\mathbf{u}', m, m^*, \mathsf{pk}, \mathsf{pk}^*)$

Response

$\mathbf{z} := c \cdot \mathbf{s}_1 + \mathbf{r}$

If $\text{RejSamp}(\mathbf{z}) = 0$:

Go to Commit

$\sigma := (c, \mathbf{z})$

$\xrightarrow{m \ \mathbf{u}}$

$\xleftarrow{m^* \mathbf{u}^*}$

$\xrightarrow{\mathbf{z}}$

$\xleftarrow{\mathbf{z}^*}$

Signer$(\mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2), \mathsf{pk} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$

Optimization Technique in Dilithium

- Drop the lower bits of $\mathbf{u}$ while preserving correctness:

$$\begin{aligned}\text{HighBit}(\mathbf{A}\mathbf{z} - c \cdot \mathsf{pk}) &= \text{HighBit}(\mathbf{A}\mathbf{r} - c \cdot \mathbf{s}_2) \\ &\approx \text{HighBit}(\mathbf{u})\end{aligned}$$

Interactive Dilithium Aggregation

- First round of interaction reveals $\mathbf{u}$ in the clear
- If $\mathbf{u}$ is known, $\mathbf{s}_2$ can be recovered!

$$\mathbf{u} - (\mathbf{A}\mathbf{z} - c \cdot \mathsf{pk}) = c \cdot \mathbf{s}_2$$

- $\mathbf{A}$ is tall $\rightsquigarrow$ recovering $\mathbf{s}_1$ is easy

Insecurity of Interactive Aggregation of Dilithium

Commit

$\mathbf{r} \leftarrow D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$

$\mathbf{u}' := \mathsf{HighBit}(\mathbf{u} + \mathbf{u}^*)$

Hash

$c := \mathsf{H}(\mathbf{u}', m, m^*, \mathsf{pk}, \mathsf{pk}^*)$

Response

$\mathbf{z} := c \cdot \mathbf{s}_1 + \mathbf{r}$

If $\mathsf{RejSamp}(\mathbf{z}) = 0$:

Go to Commit

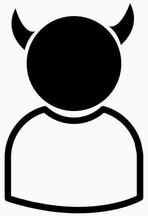
$\sigma := (c, \mathbf{z})$

$\xrightarrow{m \ \mathbf{u}}$

$\xleftarrow{m^* \mathbf{u}^*}$

$\xrightarrow{\mathbf{z}}$

$\xleftarrow{\mathbf{z}^*}$



Signer$(\mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2), \mathsf{pk} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$

Optimization Technique in Dilithium

- Drop the lower bits of $\mathbf{u}$ while preserving correctness:

$$\mathsf{HighBit}(\mathbf{A}\mathbf{z} - c \cdot \mathsf{pk}) = \mathsf{HighBit}(\mathbf{A}\mathbf{r} - c \cdot \mathbf{s}_2) \approx \mathsf{HighBit}(\mathbf{u})$$

Interactive Dilithium Aggregation

- First round of interaction reveals $\mathbf{u}$ in the clear
- If $\mathbf{u}$ is known, $\mathbf{s}_2$ can be recovered!

$$\mathbf{u} - (\mathbf{A}\mathbf{z} - c \cdot \mathsf{pk}) = c \cdot \mathbf{s}_2$$

- $\mathbf{A}$ is tall $\rightsquigarrow$ recovering $\mathbf{s}_1$ is easy

References

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. EUROCRYPT 2003.

[LMRS04] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. EUROCRYPT 2004

[CZ22] Y. Chen and Y. Zhao. Half-aggregation of schnorr signatures with tight reductions. ESORICS 2022.

[WW19] Z. Wang and Q. Wu. A practical lattice-based sequential aggregate signature. ProvSec 2019.

[ES16] R. El Bansarkhani and J. Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. CANS 2016.

[FH20] M. Fukumitsu and S. Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. ProvSec 2020.

[BR23] K. Boudgoust and A. Roux-Langlois. Compressed linear aggregate signatures based on module lattices. The Computer Journal 2023.

[DOTT21] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. PKC 2021 & Journal of Cryptology.

[BK20] D. Boneh and S. Kim. One-time and interactive aggregate signatures from lattices.

Icons are designed by Freepik. http://www.freepik.com/