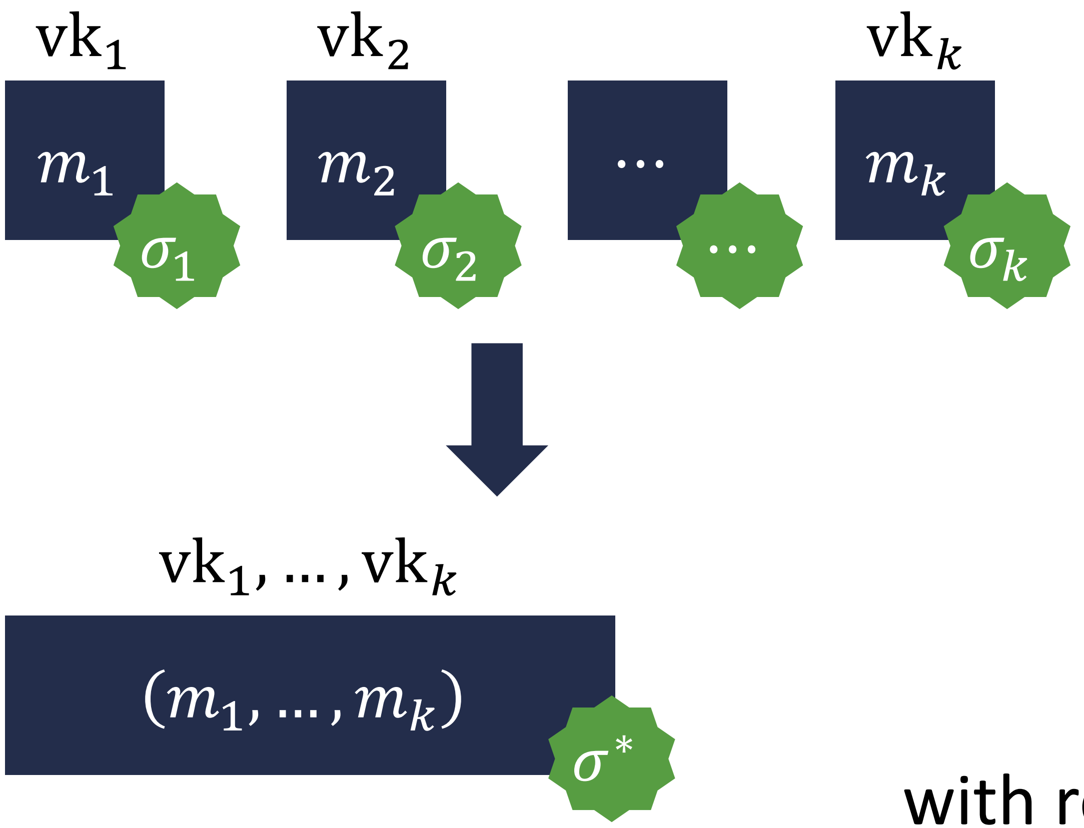
Pairing-Based Aggregate Signatures without Random Oracles

Susan Hohenberger, Brent Waters, and **David Wu**

Aggregate Signatures



Given k triples of verification keys vk_i, messages m_i, and signatures σ_i

Short signature σ^* on $(m_1, \dots, m_k)$:

$$|\sigma^*| = \mathrm{poly}(\lambda, \log k)$$

with respect to joint verification key $(\mathrm{vk}_1, \dots, \mathrm{vk}_k)$

Useful primitive whenever we need to communicate multiple signatures: e.g., certificate chains in TLS, consensus mechanisms, transactions on a blockchain

Assumptions for Aggregate Signatures

[BGLS03]: computational Diffie-Hellman in a bilinear group + random oracle model

What about constructions in the plain model?

[BCCT13]: succinct non-interactive arguments of knowledge (SNARKs) for NP
[HKW15]: indistinguishability obfuscation
[WW22, DGKV22]: batch arguments (BARGs) for NP

Relatively heavyweight tools (e.g., require non-black-box use of an underlying signature scheme)

Many different relaxations: multi-signatures [Ita83, OO99, MOR01, Bol03, …], synchronized aggregation [GR06, AGH10, LLY13, …], sequential aggregation [LMRS04, LOSSW06, BGOY07, …], and more

Assumptions for Aggregate Signatures

[BGLS03]: computational Diffie-Hellman in a bilinear group + random oracle model

What about constructions in the plain model?

[BCCT13]: succinct non-interactive arguments of knowledge (SNARKs) for NP
[HKW15]: indistinguishability obfuscation
[WW22, DGKV22]: batch arguments (BARGs) for NP

Relatively heavyweight tools (e.g., require non-black-box use of an underlying signature scheme)

This work: pairing-based aggregate signatures in the plain model (with falsifiable assumptions) that make black-box use of the group

comparable concrete efficiency as [BGLS03] aggregate signatures

This Work

This work: pairing-based aggregate signatures in the plain model (with falsifiable assumptions) that make black-box use of the group

Aggregate signatures are short: 2 group elements

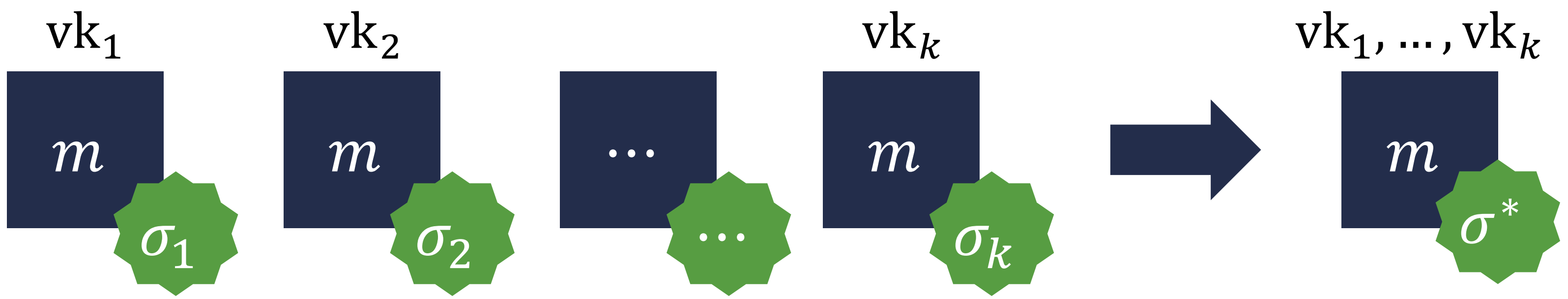
Unforgeability relies on (bilateral) CDH in a pairing group (in the plain model)

Two relaxations:

- bounded aggregation (i.e., scheme allows one to aggregate up to k signatures and we allow public parameters to scale with k)
- base signatures are long (size linear in k)

Starting Point: [LOSSW06] Multi-signatures

Multi-signature: supports same-message aggregation



Builds on the Boneh-Boyen [BB04] pairing-based signature scheme (derived from an identity-based encryption scheme)

sk: g^α $(\alpha \leftarrow \mathbb{Z}_p)$

vk: $(e(g,g)^\alpha, u, h)$

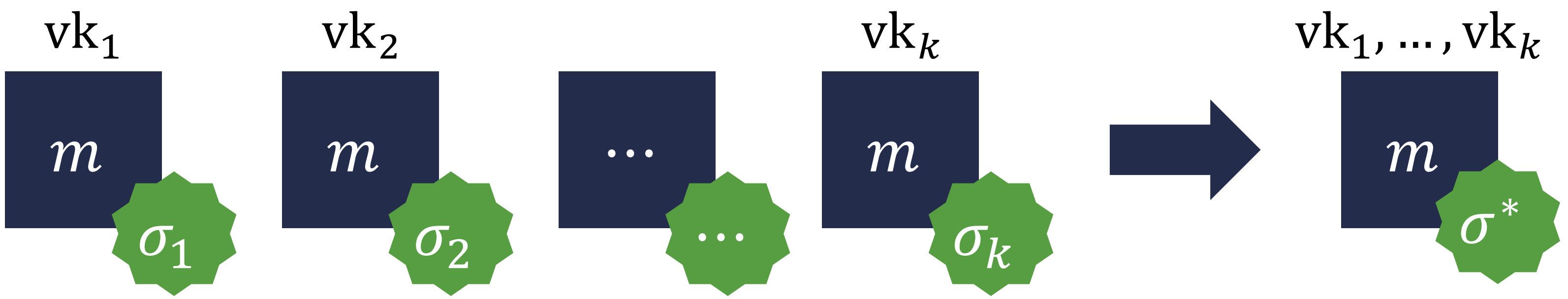
$u, h \leftarrow \mathbb{G}$

Conventions (this talk):

- Symmetric prime-order pairing group $(\mathbb{G}, \mathbb{G}_\mathrm{T})$
- Group order p
- Generator g
- Pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathrm{T}$

Starting Point: [LOSSW06] Multi-signatures

Multi-signature: supports same-message aggregation



Builds on the Boneh-Boyen [BB04] pairing-based signature scheme (derived from an identity-based encryption scheme)

sk: g^α $(\alpha \leftarrow \mathbb{Z}_p)$

vk: $(e(g,g)^\alpha, u, h)$

$u, h \leftarrow \mathbb{G}$

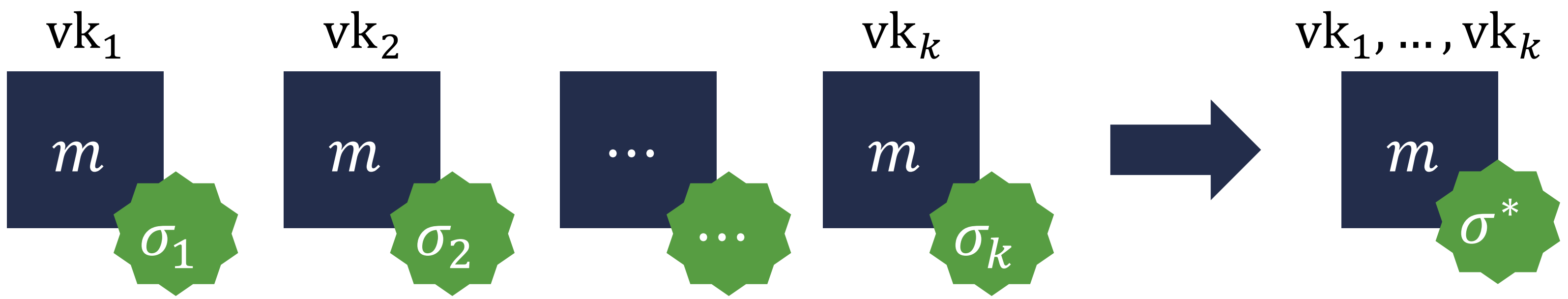
Sign message $m \in \mathbb{Z}_p$:

1. Sample $r \leftarrow \mathbb{Z}_p$
2. Compute "hash" of the message $u^m h$
3. Output $(g^\alpha (u^m h)^r, g^r)$

"encryption of the signing key g^α where the hash of the message $u^m h$ is the public key"

Starting Point: [LOSSW06] Multi-signatures

Multi-signature: supports same-message aggregation



Builds on the Boneh-Boyen [BB04] pairing-based signature scheme (derived from an identity-based encryption scheme)

sk: g^α $(\alpha \leftarrow \mathbb{Z}_p)$

vk: $(e(g,g)^\alpha, u, h)$

$u, h \leftarrow \mathbb{G}$

Signature on $m \in \mathbb{Z}_p$: $(g^\alpha (u^m h)^r, g^r)$

Verification: check that $e(g,g)^\alpha = \dfrac{e(g, g^\alpha (u^m h)^r)}{e(g^r, u^m h)}$

"decrypt in the target group via the pairing"

Starting Point: [LOSSW06] Multi-signatures

Scheme supports same-message aggregation (multi-signature)

sk: g^α

vk: $(e(g,g)^\alpha, u, h)$

Signature on $m \in \mathbb{Z}_p$: $(g^\alpha (u^m h)^r, g^r)$

Verification: check that $e(g,g)^\alpha = \frac{e(g, g^\alpha (u^m h)^r)}{e(g^r, u^m h)}$

Move (u, h) to global public parameters (i.e., same u, h used in all verification keys)

pp: (u, h) where $u, h \leftarrow \mathbb{G}$

sk: g^α

vk: $e(g,g)^\alpha$

$\sigma_1 = (g^{\alpha_1} (u^m h)^{r_1}, g^{r_1})$

$\sigma_2 = (g^{\alpha_2} (u^m h)^{r_2}, g^{r_2})$

Starting Point: [LOSSW06] Multi-signatures

Scheme supports same-message aggregation (multi-signature)

sk: g^α

vk: $(e(g,g)^\alpha, u, h)$

Signature on $m \in \mathbb{Z}_p$: $(g^\alpha (u^m h)^r, g^r)$

Verification: check that $e(g,g)^\alpha = \frac{e(g, g^\alpha (u^m h)^r)}{e(g^r, u^m h)}$

Move (u, h) to global public parameters (i.e., same u, h used in all verification keys)

pp: (u, h) where $u, h \leftarrow \mathbb{G}$

sk: g^α

vk: $e(g,g)^\alpha$

$$\sigma_1 = (g^{\alpha_1}(u^m h)^{r_1}, g^{r_1})$$

$$\sigma_2 = (g^{\alpha_2}(u^m h)^{r_2}, g^{r_2})$$

Observation: When the message m is the same, hashes are **the same**

Can aggregate by multiplying signatures together

Starting Point: [LOSSW06] Multi-signatures

Scheme supports same-message aggregation (multi-signature)

sk: g^α

vk: $(e(g,g)^\alpha, u, h)$

Signature on $m \in \mathbb{Z}_p$: $(g^\alpha (u^m h)^r, g^r)$

Verification: check that $e(g,g)^\alpha = \frac{e(g, g^\alpha (u^m h)^r)}{e(g^r, u^m h)}$

Move (u, h) to global public parameters (i.e., same u, h used in all verification keys)

pp: (u, h) where $u, h \leftarrow \mathbb{G}$

sk: g^α

vk: $e(g,g)^\alpha$

$$\sigma_1 = (g^{\alpha_1}(u^m h)^{r_1}, g^{r_1})$$

$$\sigma_2 = (g^{\alpha_2}(u^m h)^{r_2}, g^{r_2})$$

$$\sigma^* = (g^{\alpha_1+\alpha_2}(u^m h)^{r_1+r_2}, g^{r_1+r_2})$$

Can aggregate by multiplying signatures together

Signature with respect to verification key $e(g,g)^{\alpha_1+\alpha_2} = e(g,g)^{\alpha_1} \cdot e(g,g)^{\alpha_2}$ and randomness $r_1 + r_2$

Aggregating Signatures on Different Messages

pp: (u, h) where $u, h \leftarrow \mathbb{G}$

sk: g^{α}

vk: $e(g, g)^{\alpha}$

$$\sigma_1 = (g^{\alpha_1}(u^m h)^{r_1}, g^{r_1})$$

$$\sigma_2 = (g^{\alpha_2}(u^m h)^{r_2}, g^{r_2})$$

$$\sigma_{\text{agg}} = (g^{\alpha_1+\alpha_2}(u^m h)^{r_1+r_2}, g^{r_1+r_2})$$

What if messages were different?

Aggregating Signatures on Different Messages

pp: (u, h) where $u, h \leftarrow \mathbb{G}$

sk: g^α

vk: $e(g, g)^\alpha$

$$\sigma_1 = (g^{\alpha_1}(u^{m_1}h)^{r_1}, g^{r_1})$$

$$\sigma_2 = (g^{\alpha_2}(u^{m_2}h)^{r_2}, g^{r_2})$$

$$\sigma_{\text{agg}} = (g^{\alpha_1+\alpha_2}(u^{m_1}h)^{r_1}(u^{m_2}h)^{r_2}, g^{r_1+r_2})$$

Unclear how to verify given just $g^{r_1+r_2}$

What if messages were different?

Our Approach

Step 1: Introduce additional helper terms in each signature to facilitate aggregation

Captured via intermediate abstraction of **slotted** aggregate signature

Public parameters define a sequence of slots

Slot 1	Slot 2	$\cdots$	Slot k

Signatures are associated with a slot $i \in [k]$

$$\mathrm{Sign}(\mathrm{pp}, \mathrm{sk}, m, i) \to \sigma$$

Aggregation takes **one signature per slot** and aggregates them together

Caveats:
- Can only aggregate at most k signatures at once
- Allow signature size for each slot to scale with k

Our Approach

Step 1: Introduce additional helper terms in each signature to facilitate aggregation

Captured via intermediate abstraction of **slotted** aggregate signature

Leverages cross-term cancellation techniques like those used in pairing-based registration-based cryptography

[HLWW23, ZZGQ23, FFMMRV23, KMW23, …]

This talk

Step 2: Compile a slotted signature scheme into an unslotted scheme that supports bounded aggregation

Signatures no longer associated with slots

Supports aggregation on any collection of up to k signatures

Individual signatures are still long (scale with k), but aggregate signature is short

Similar technique used to lift a slotted scheme to an unslotted one in the setting of distributed broadcast encryption

[GLWW23]

See paper for details

Slotted Aggregate Signatures

Key idea: can also aggregate Boneh-Boyen signatures if they have common randomness

$$\sigma_1 = (g^{\alpha_1}(u^{m_1}h)^r, g^r) \qquad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$$

$$\sigma_2 = (g^{\alpha_2}(u^{m_2}h)^r, g^r) \qquad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$$

different message, same randomness

Aggregate signature: $\sigma^* = (g^{\alpha_1+\alpha_2}(u^{m_1}h)^r(u^{m_2}h)^r, g^r)$

"encryption of joint signing key $g^{\alpha_1+\alpha_2}$ under the public key $(u^{m_1}h)(u^{m_2}h)$ with randomness r"

Slotted Aggregate Signatures

Key idea: can also aggregate Boneh-Boyen signatures if they have common randomness

$$\sigma_1 = (g^{\alpha_1}(u^{m_1}h)^r, g^r) \qquad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$$

$$\sigma_2 = (g^{\alpha_2}(u^{m_2}h)^r, g^r) \qquad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$$

different message, same randomness

Aggregate signature: $\sigma^* = (g^{\alpha_1+\alpha_2}(u^{m_1}h)^r(u^{m_2}h)^r, g^r)$

Observation: verifier can compute $(u^{m_1}h)(u^{m_2}h)$ from u, h and m_1, m_2

Verification check:

$$\underbrace{e(g,g)^{\alpha_1}e(g,g)^{\alpha_2}}_{\text{product of verification keys}} = \frac{e(g, g^{\alpha_1+\alpha_2}(u^{m_1}h)^r(u^{m_2}h)^r)}{e\left((u^{m_1}h)(u^{m_2}h), g^r\right)} = \frac{e(g, \sigma_1^*)}{e\left((u^{m_1}h)(u^{m_2}h), \sigma_2^*\right)}$$

Slotted Aggregate Signatures

Key idea: can also aggregate Boneh-Boyen signatures if they have common randomness

$$\sigma_1 = (g^{\alpha_1}(u^{m_1}h)^r, g^r) \qquad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$$

$$\sigma_2 = (g^{\alpha_2}(u^{m_2}h)^r, g^r) \qquad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$$

different message, same randomness

Aggregate signature: $\sigma^* = (g^{\alpha_1+\alpha_2}(u^{m_1}h)^r(u^{m_2}h)^r, g^r)$

Observation: verifier can compute $(u^{m_1}h)$

Having common randomness allows us to just pair with g^r

Verification check:

$$e(g,g)^{\alpha_1}e(g,g)^{\alpha_2} = \frac{e(g, g^{\alpha_1+\alpha_2}(u^{m_1}h)^r(u^{m_2}h)^r)}{e\big((u^{m_1}h)(u^{m_2}h), g^r\big)} = \frac{e(g,\sigma_1^*)}{e\big((u^{m_1}h)(u^{m_2}h), \sigma_2^*\big)}$$

product of verification keys

Slotted Aggregate Signatures

Key idea: can also aggregate Boneh-Boyen signatures if they have common randomness

$$\sigma_1 = (g^{\alpha_1}(u^{m_1}h)^r, g^r) \qquad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$$

$$\sigma_2 = (g^{\alpha_2}(u^{m_2}h)^r, g^r) \qquad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$$

different message, same randomness

*How do we ensure **independent** signatures share common randomness?*

Note: using fixed randomness is insecure

Slotted Aggregate Signatures

Solution: give out helper terms for each slot to "recode" signatures with common randomness

Consider setting with $k = 2$ slots:

u_1, h_1	u_2, h_2
Slot 1	Slot 2

Each slot has a set of Boneh-Boyen public parameters

Signing/verification keys are unchanged: $\mathrm{sk} = g^\alpha$ and $\mathrm{vk} = e(g, g)^\alpha$

Signature for Slot 1: $\sigma_1 = (\underbrace{g^\alpha (u_1^m h_1)^r, \; g^r}_{\text{standard Boneh-Boyen signature with Slot 1 public parameters}} , \underbrace{u_2^r \; , \; h_2^r}_{\text{"cross-terms" associated with other slots}})$

standard Boneh-Boyen signature with Slot 1 public parameters

"cross-terms" associated with other slots

Slotted Aggregate Signatures

Solution: give out helper terms for each slot to "recode" signatures with common randomness

Suppose we have two signatures σ_1 and σ_2

Signature for Slot 1: $\sigma_1 = \left(g^{\alpha_1}\left(u_1^{m_1} h_1\right)^{r_1}, g^{r_1}, u_2^{r_1}, h_2^{r_1} \right)$ $\quad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$

Signature for Slot 2: $\sigma_2 = \left(g^{\alpha_2}\left(u_2^{m_2} h_2\right)^{r_2}, g^{r_2}, u_1^{r_2}, h_1^{r_2} \right)$ $\quad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$

Aggregator uses cross-terms to translate σ_1, σ_2 to use **common randomness**

$g^{\alpha_1}\left(u_1^{m_1} h_1\right)^{r_1}$

$u_1^{r_2}, h_1^{r_2} \xrightarrow{\text{using } m_1} \left(u_1^{r_2}\right)^{m_1} h_1^{r_2} = \left(u_1^{m_1} h_1\right)^{r_2}$

$\Rightarrow g^{\alpha_1}\left(u_1^{m_1} h_1\right)^{r_1+r_2}$

Slotted Aggregate Signatures

Solution: give out helper terms for each slot to "recode" signatures with common randomness

Suppose we have two signatures σ_1 and σ_2

Signature for Slot 1: $\sigma_1 = \left(g^{\alpha_1}(u_1^{m_1}h_1)^{r_1}, g^{r_1}, u_2^{r_1}, h_2^{r_1}\right)$ $\quad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$

Signature for Slot 2: $\sigma_2 = \left(g^{\alpha_2}(u_2^{m_2}h_2)^{r_2}, g^{r_2}, u_1^{r_2}, h_1^{r_2}\right)$ $\quad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$

Aggregator uses cross-terms to translate σ_1, σ_2 to use **common randomness**

New Boneh-Boyen signature on m_1 under randomness $r_1 + r_2$!

$$g^{\alpha_1}(u_1^{m_1}h_1)^{r_1+r_2}$$

$$g^{r_1+r_2}$$

Slotted Aggregate Signatures

Solution: give out helper terms for each slot to "recode" signatures with common randomness

Suppose we have two signatures σ_1 and σ_2

Signature for Slot 1: $\sigma_1 = \left(g^{\alpha_1}\left(u_1^{m_1} h_1\right)^{r_1}, g^{r_1}, u_2^{r_1}, h_2^{r_1}\right)$ $\quad \mathrm{vk}_1 = e(g,g)^{\alpha_1}$

Signature for Slot 2: $\sigma_2 = \left(g^{\alpha_2}\left(u_2^{m_2} h_2\right)^{r_2}, g^{r_2}, u_1^{r_2}, h_1^{r_2}\right)$ $\quad \mathrm{vk}_2 = e(g,g)^{\alpha_2}$

Aggregator uses cross-terms to translate σ_1, σ_2 to use **common randomness**

$g^{\alpha_1}\left(u_1^{m_1} h_1\right)^{r_1+r_2} \quad g^{r_1+r_2}$ Boneh-Boyen signature on m_1 with randomness $r_1 + r_2$

Apply same procedure to σ_2 to obtain

Same randomness, so can aggregate as before!

$g^{\alpha_2}\left(u_2^{m_2} h_2\right)^{r_1+r_2} \quad g^{r_1+r_2}$ Boneh-Boyen signature on m_2 with randomness $r_1 + r_2$

Slotted Aggregate Signatures

Solution: give out helper terms for each slot to "recode" signatures with common randomness

In general:

Signature for Slot 1: $$\sigma_1 = \left(\underbrace{g^{\alpha_1}\left(u_1^{m_1} h_1\right)^{r_1}, g^{r_1}}_{\text{Boneh-Boyen signature}}, \underbrace{u_2^{r_1}, h_2^{r_1}, \dots, u_k^{r_1}, h_k^{r_1}}_{\text{cross-terms for "recoding"}} \right)$$

Given k signatures with randomness $r_1, \dots, r_k$

Cross-terms can be used to recode all of them to have common randomness $r_1 + \cdots + r_k$

Base signatures now contain $2k$ group elements

Aggregated signature just contains 2 group elements

Secure assuming (bilateral) CDH in a pairing group

This Work

This work: pairing-based aggregate signatures in the plain model (with falsifiable assumptions) that make black-box use of the group

Aggregate signatures are short: 2 group elements

Unforgeability relies on (bilateral) CDH in a pairing group (in the plain model)

Two relaxations:

- bounded aggregation (i.e., scheme allows one to aggregate up to k signatures and we allow public parameters to scale with k)
- base signatures are long (size linear in k)

Open Questions

Aggregate signatures from pairings in the plain model where **all** signatures consist of a constant number of group elements

This work: base signatures are long, aggregated signature is short

Direct construction that supports aggregating unbounded number of signatures

This work: can aggregate up to k signatures and parameters scale with k

Direct construction of aggregate signatures from post-quantum assumptions (without BARGs) – e.g., post-quantum version of BLS signatures?

Thanks!

`https://eprint.iacr.org/2025/1548.pdf`