Silent Threshold Cryptography from Pairings

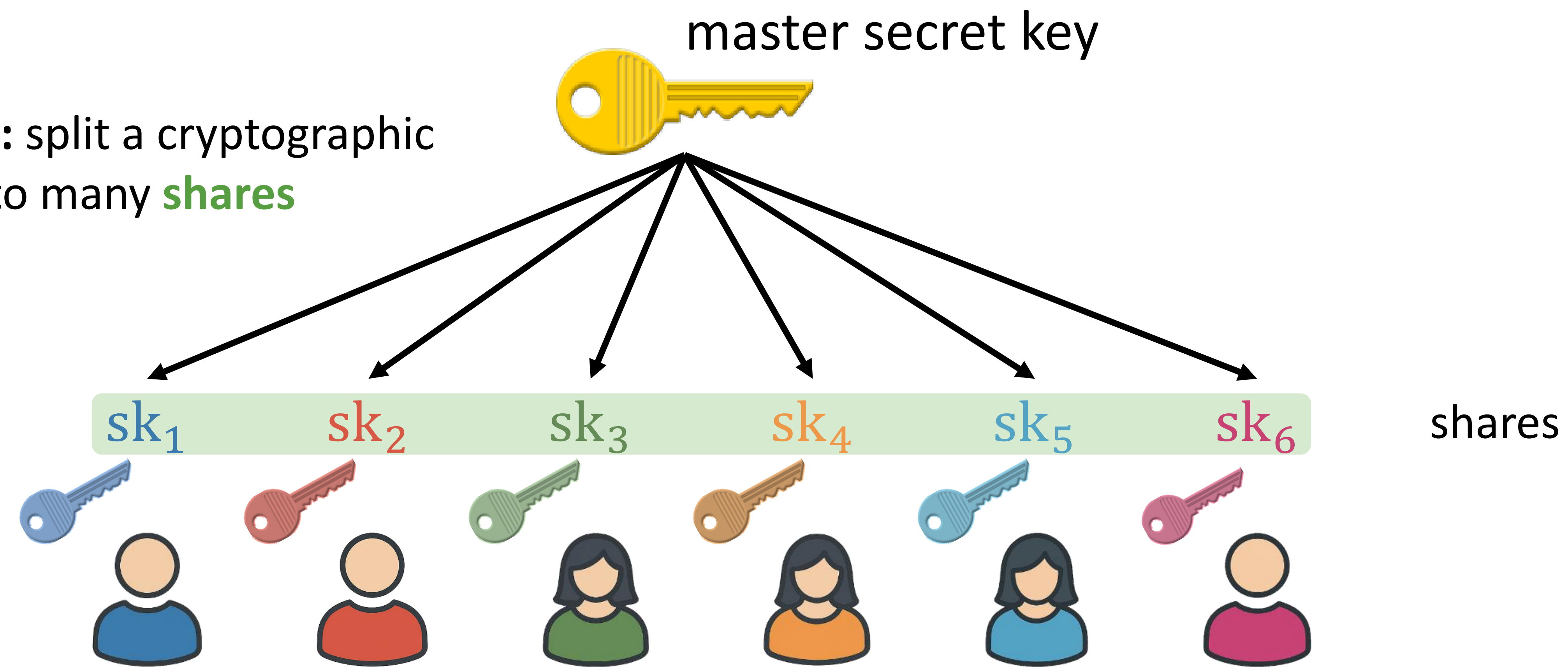
Brent Waters and **David Wu**

March 2026

Threshold Cryptography

[Des87, Fra89, DF89]

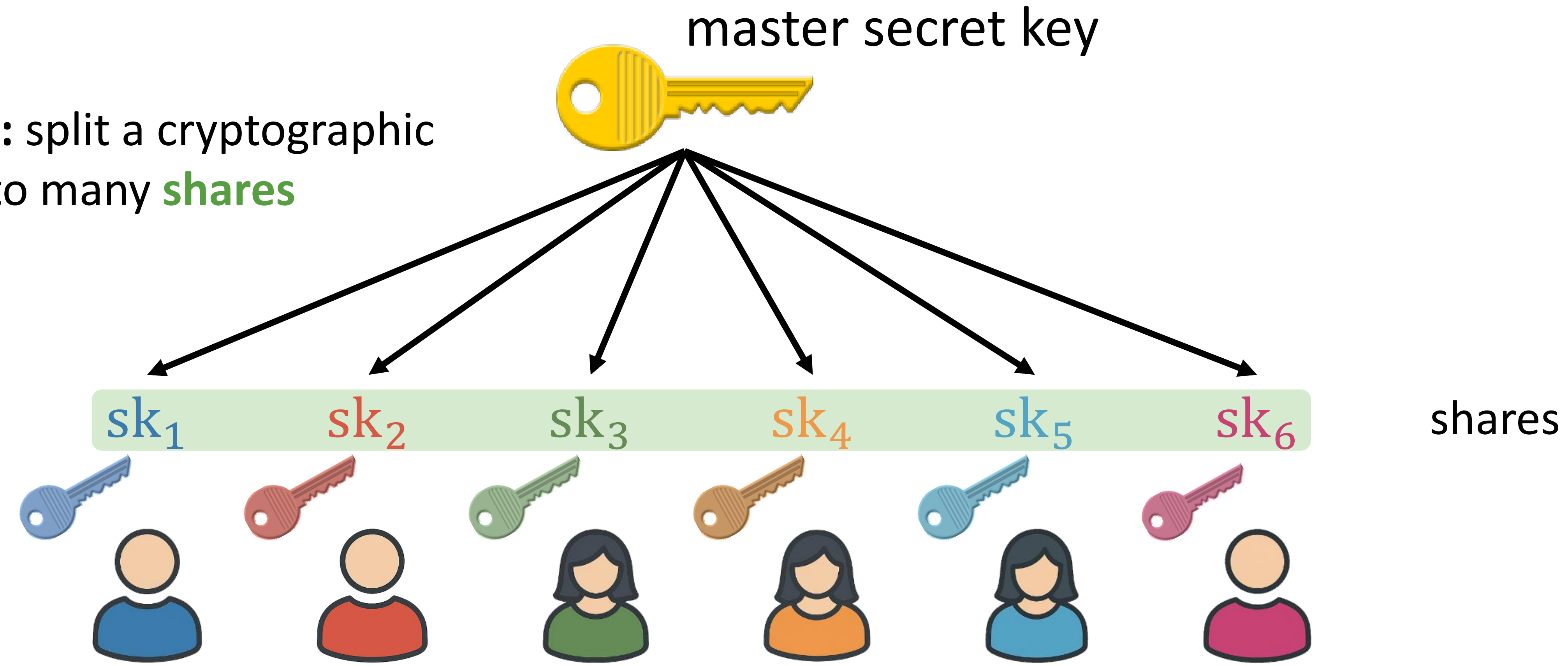
Typical setup: split a cryptographic key into many **shares**



Only an **authorized set** of parties can perform a target action (e.g., signing, decryption, etc.)

Who Generates the Shares?

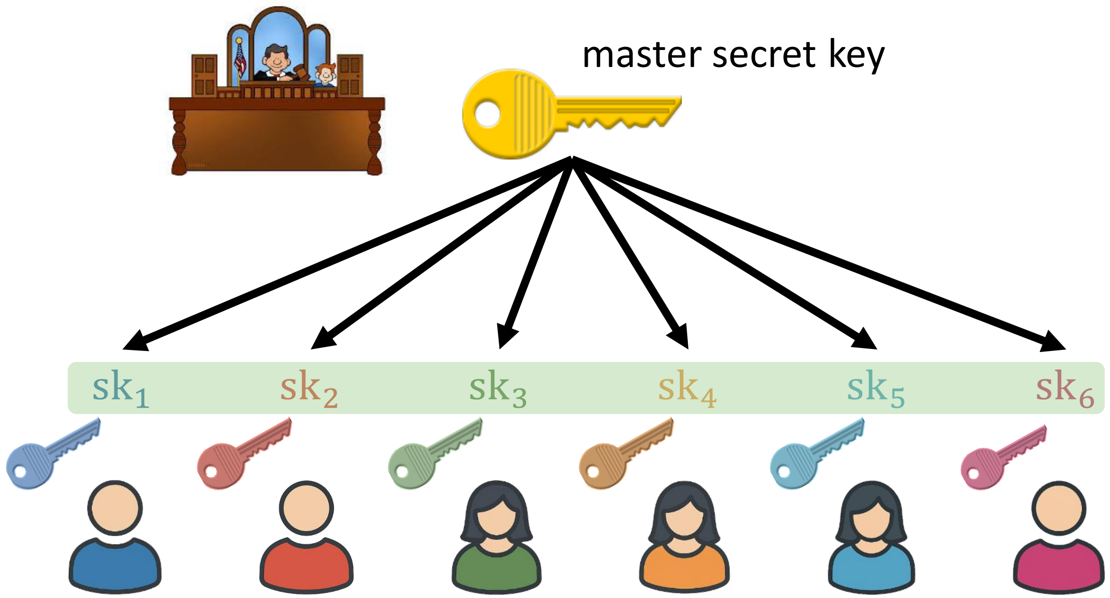
Typical setup: split a cryptographic key into many **shares**



Only an **authorized set** of parties can perform a target action (e.g., signing, decryption, etc.)

Who Generates the Shares?

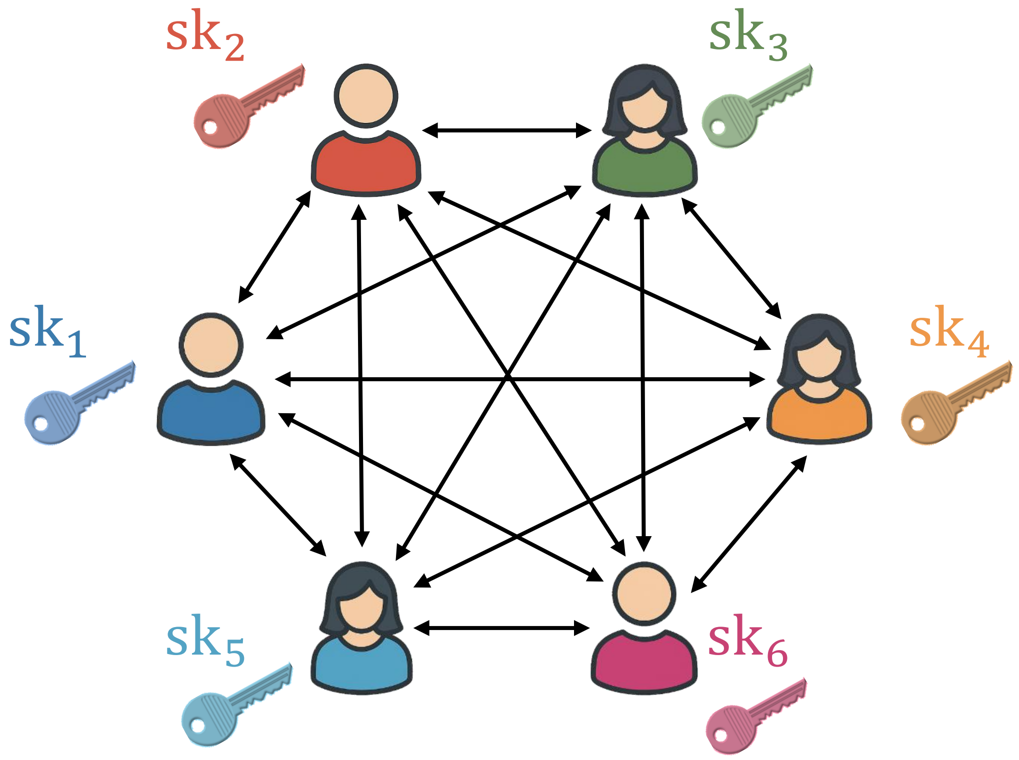
Option 1: Trusted dealer



Needs a trusted party

Redeal shares if policy changes (e.g., new user joins)

Option 2: Distributed key generation

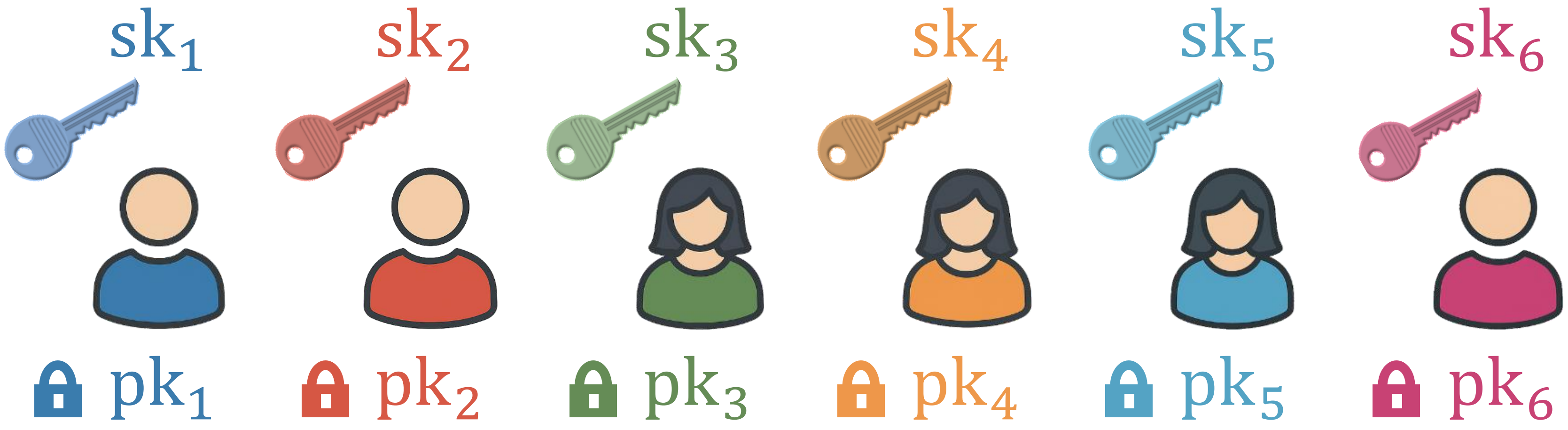


Requires parties to coordinate and interact

Rerun setup if policy changes (e.g., new user joins)

Silent Threshold Cryptography

[MRVWZ21, DCXNBR23, GJMSW24]



Users **independently** generate their own public key pk_i and secret key sk_i

---

arbitrary collection of public keys

policy

aggregated public key

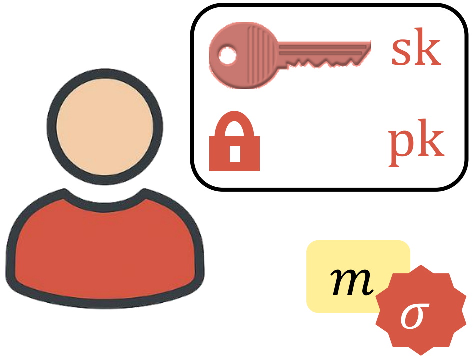
Can **deterministically** aggregate any set of public keys together with a policy

Master public key mpk serves as the key for the threshold cryptosystem for the chosen quorum

Individual secret key sk_i is each user's share

Example: Threshold Signatures with Silent Setup

[MRVWZ21, DCXNBR23, GJMSW24]



Users generate their own keys (relative to a common reference string)

$\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$

Signing key sk can be used to sign messages

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$ *(σ must verify relative to pk)*

$\mathsf{Aggregate}(\mathsf{crs}, \{\mathsf{pk}_i\}_{i \in S}, T) \to (\mathsf{mpk}, \mathsf{ht})$

T: target threshold

mpk: aggregated verification key for quorum

ht: aggregation hint

mpk

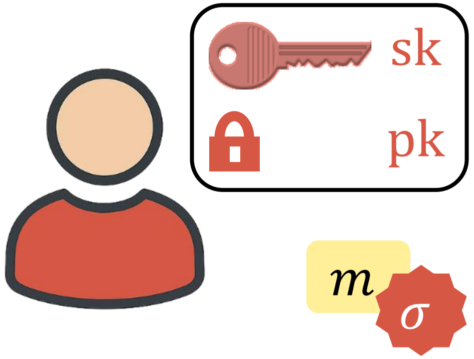
arbitrary collection of public keys

policy

aggregated public key

Example: Threshold Signatures with Silent Setup

[MRVWZ21, DCXNBR23, GJMSW24]



Users generate their own keys (relative to a common reference string)

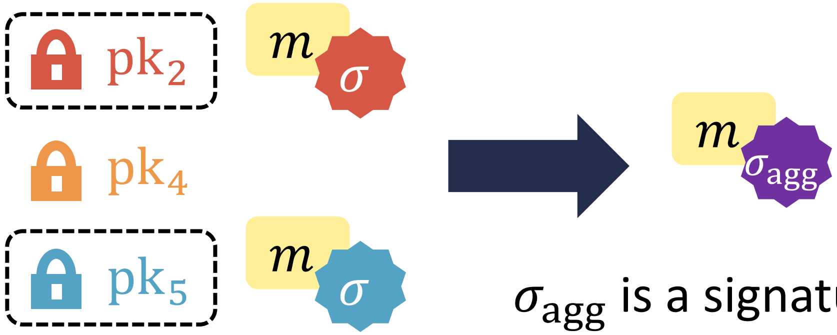
$\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$

Signing key sk can be used to sign messages

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$ *(σ must verify relative to pk)*

$\mathsf{Aggregate}(\mathsf{crs}, \{\mathsf{pk}_i\}_{i \in S}, T) \to (\mathsf{mpk}, \mathsf{ht})$

$\mathsf{AggSig}\left(\mathsf{ht}, \{\sigma_i\}_{i \in S'}\right) \to \sigma_{\mathsf{agg}}$



σ_{agg} is a signature on m under mpk

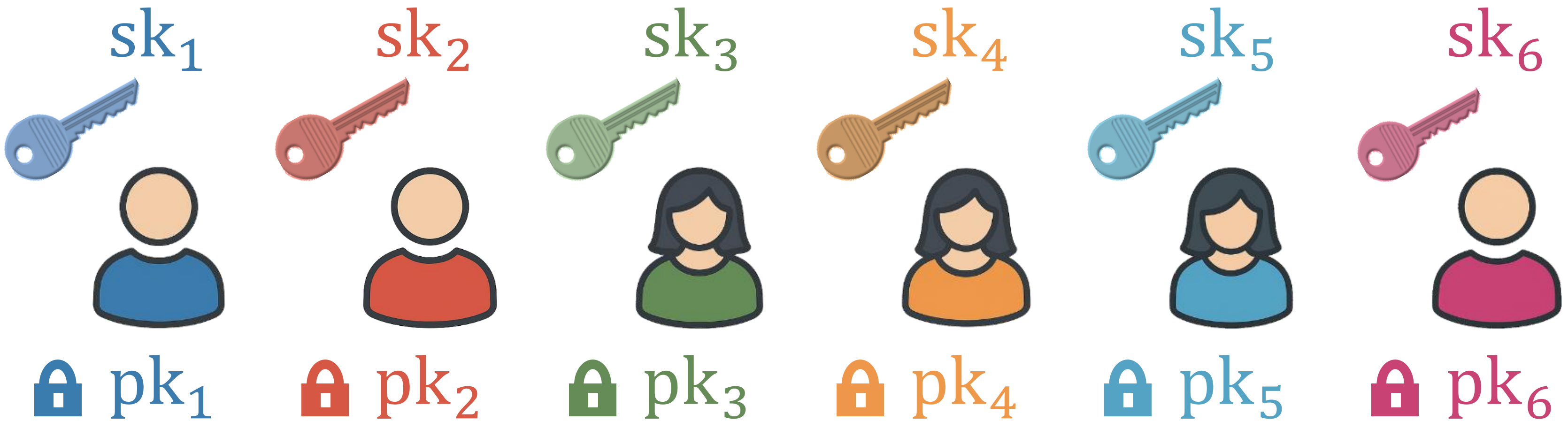
Efficiency: $|\sigma_{\mathsf{agg}}|$ and signature verification time are independent of number of users

Security: adversary with fewer than T signatures on m cannot forge signature with respect to mpk

$\mathsf{Verify}\left(\mathsf{mpk}, m, \sigma_{\mathsf{agg}}\right) = 1$

Silent Threshold Cryptography

[MRVWZ21, DCXNBR23, GJMSW24]



Users **independently** generate their own public key pk_i and secret key sk_i

---

"Threshold cryptography where users choose their own shares"

Well-suited for decentralized settings: no need for trusted dealer, users do not need to be aware of each other

Supports dynamic policies (i.e., shares are not tied to a policy); users do not need to be aware of policy

Does rely on common reference string (CRS), which requires a one-time setup (rather than per-policy setup)

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\text{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\text{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\text{poly}(\lambda) \cdot \lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$

Relatively few constructions (other than via generic tools like SNARKs or BARGs)

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\text{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\text{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\text{poly}(\lambda) \cdot \lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N \log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Relatively few constructions (other than via generic tools like SNARKs or BARGs)

Existing constructions either have long signatures (super-constant number of group elements) or only shown secure in the generic bilinear group model

> **In fact:** all constructions with short signatures rely on some kind of SNARK machinery (e.g., sum check, inner product arguments, etc.)

This work: a direct algebraic construction (no SNARK machinery)

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\mathrm{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\mathrm{poly}(\lambda)\cdot\lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N \log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Base signatures have two group elements, but final signature is as short as that using a pairing-based SNARK (e.g., [Gro16])

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert \mathsf{crs} \rvert$	$\lvert \sigma \rvert$	$\lvert \sigma_{\mathsf{agg}} \rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert \mathbb{G} \rvert$	$3\lvert \mathbb{G} \rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert \mathbb{G} \rvert$	$\mathrm{poly}(\lambda) \cdot \lvert \mathbb{G} \rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert \mathbb{G} \rvert$	$8\lvert \mathbb{G} \rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert \mathbb{G} \rvert$	$9\lvert \mathbb{G} \rvert + 5\lvert \mathbb{F} \rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert \mathbb{G} \rvert$	$3\lvert \mathbb{G} \rvert$
This work	threshold	q-type assumption	$O_\lambda(N \log N)$	$2\lvert \mathbb{G} \rvert$	$3\lvert \mathbb{G} \rvert$

Can support general policies beyond threshold policies (e.g., majority of majorities, monotone Boolean formulas)

Security in the **plain** model

Drawback: larger CRS (quadratic for general policies, quasi-linear for threshold policies)

Silent Threshold Encryption

Techniques directly generalize to setting of silent threshold public-key encryption

Scheme	Policy Family	Assumption	|crs|	|ct|
[RSY21, ADMSW24]	threshold	$i\mathcal{O}$ + OWF/SSB	None	$O_\lambda(1)$
[GKPW24]	threshold	generic bilinear group	$O_\lambda(N)$	$9\|\mathbb{G}\|$
[DJWW25]	S-space read-once TM	$i\mathcal{O}$ + SSB	$O_\lambda(1)$	$O_\lambda(2^S)$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$3\|\mathbb{G}\| + \|\mathbb{F}\|$
	threshold	q-type assumption	$O_\lambda(N \log N)$	$3\|\mathbb{G}\| + \|\mathbb{F}\|$

No generic SNARK-based solution in the case of encryption (except with *extractable* witness encryption)

For general policies, problem is challenging even with strong tools like witness encryption or obfuscation

This work: first construction for Boolean formulas and thresholds, but does need a large CRS

Starting Point: Boneh-Boyen Signatures

This talk: will focus just on signatures (same techniques work for encryption)

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 $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_\mathrm{T}$

Starting Point: Boneh-Boyen Signatures

This talk: will focus just on signatures (same techniques work for encryption)

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: Boneh-Boyen Signatures

This talk: will focus just on signatures (same techniques work for encryption)

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

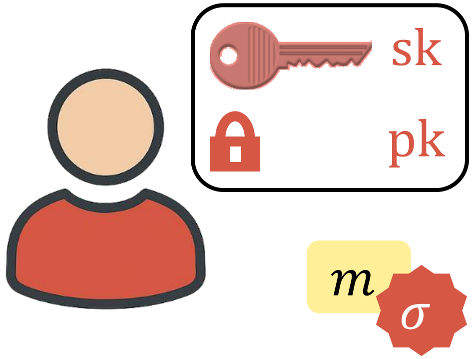
Verification: check that $e(g,g)^{\alpha} = \frac{e(g, g^{\alpha}(u^m h)^r)}{e(g^r, u^m h)}$

"decrypt in the target group via the pairing"

Construction Template

hash key

CRS: u, h



$\mathrm{sk} = \alpha$

$\mathrm{pk} = e(g, g)^{\alpha}$

Each user chooses their own Boneh-Boyen public key

$\sigma = (g^{\alpha}(u^m h)^r, g^r)$

Signature is plain Boneh-Boyen signature

$\mathrm{pk}_1 = e(g, g)^{\alpha_1}$

$\mathrm{pk}_2 = e(g, g)^{\alpha_2}$

$\vdots$

$\mathrm{pk}_N = e(g, g)^{\alpha_N}$

Need to design two mechanisms:

1. Aggregate the user public keys relative to a policy
2. Aggregate signatures for users in the set

Aggregating Signatures

hash key

CRS: u, h

Suppose one has signature from each party $i \in S$

We will rely on linear homomorphism: $\tilde{\sigma}_i = (\underbrace{g^{\alpha_i}(u^m h)^{r_i}}_{\tilde{\sigma}_{i,1}}, \underbrace{g^{r_i}}_{\tilde{\sigma}_{i,2}})$

$$\sigma_{\text{agg},1} = \prod_{i \in S} \tilde{\sigma}_{i,1} = g^{\sum_{i \in S} \alpha_i}(u^m h)^{\sum_{i \in S} r_i} = g^{\sum_{i \in S} \alpha_i}(u^m h)^{\tilde{r}}$$

$$\tilde{r} = \sum_{i \in S} r_i$$

$$\sigma_{\text{agg},2} = \prod_{i \in S} \tilde{\sigma}_{i,2} = g^{\sum_{i \in S} r_i} = g^{\tilde{r}}$$

$\sigma_{\text{agg}} = (\sigma_{\text{agg},1}, \sigma_{\text{agg},2})$ is a Boneh-Boyen signature on m with respect to $e(g,g)^{\sum_{i \in S} \alpha_i}$

Aggregating Signatures

hash key

CRS: u, h

Suppose one has signature from each party $i \in S$

We will rely on linear homomorphism: $\tilde{\sigma}_i = (\overbrace{g^{\alpha_i}(u^m h)^{r_i}}^{\tilde{\sigma}_{i,1}}, \overbrace{g^{r_i}}^{\tilde{\sigma}_{i,2}})$

Verifier does not know the set S!

Aggregate signature $(\sigma_{\text{agg},1}, \sigma_{\text{agg},2})$ verifies with respect to $\prod_{i \in S} e(g,g)^{\alpha_i}$

Inefficient approach: Aggregate signature includes description of S so verifier can check that S satisfies the policy, and if so, compute the set-specific verification key $\prod_{i \in S} e(g,g)^{\alpha_i}$ and check the signature

Our approach: Derive set-specific key $e(g,g)^{\sum_{i \in S} \alpha_i}$ from the pairing implicitly (from aggregated public key associated with the whole group)

Aggregating Public Keys

hash key　　　　commitment key

CRS: u, h　　　　$g^{c_1}, \dots, g^{c_N}$

Aggregated public key is a Pedersen vector commitment to the users' public keys

Sample $c_1, \dots, c_N \leftarrow \mathbb{Z}_p$ and publish $g^{c_1}, \dots, g^{c_N}$ in CRS (as the commitment key)

Aggregated public key is a commitment to $\alpha_1, \dots, \alpha_n$

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$

Aggregating Public Keys

hash key

commitment key

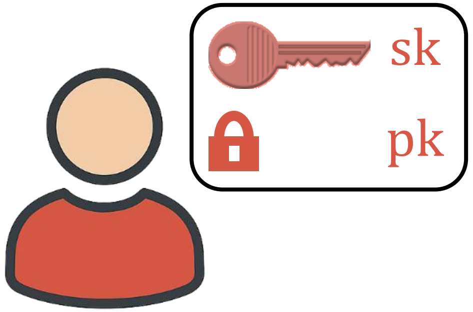
CRS: u, h $\quad g^{c_1}, \dots, g^{c_N}$

Aggregated public key is a Pedersen vector commitment to the users' public keys

Sample $c_1, \dots, c_N \leftarrow \mathbb{Z}_p$ and publish $g^{c_1}, \dots, g^{c_N}$ in CRS (as the commitment key)

Aggregated public key

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$



$\mathrm{sk} = \alpha$

$\mathrm{pk} = e(g, g)^{\alpha}$

Each user's public key will also contain $g^{\alpha c_i}$ for all $i \in [N]$

Aggregating Public Keys

hash key — commitment key

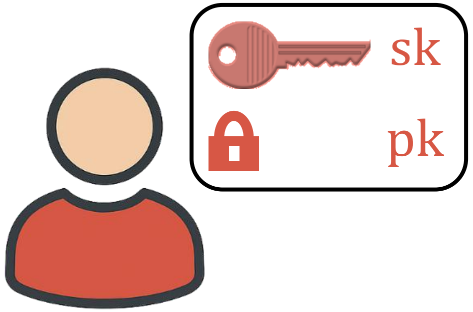
CRS: u, h — $g^{c_1}, \dots, g^{c_N}$

Aggregated public key is a Pedersen vector commitment to the users' public keys

Sample $c_1, \dots, c_N \leftarrow \mathbb{Z}_p$ and publish $g^{c_1}, \dots, g^{c_N}$ in CRS (as the commitment key)

Aggregated public key

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$



$\mathrm{sk} = \alpha$

$\mathrm{pk} = e(g,g)^{\alpha}, g^{\alpha c_1}, \dots, g^{\alpha c_N}$

Each user's public key will also contain $g^{\alpha c_i}$ for all $i \in [N]$

Aggregating Public Keys

hash key

commitment key

CRS: u, h $g^{c_1}, \dots, g^{c_N}$

$\mathrm{pk}_1 = e(g,g)^{\alpha_1}, \{g^{\alpha_1 c_i}\}_{i \in [N]}$

$\mathrm{pk}_2 = e(g,g)^{\alpha_2}, \{g^{\alpha_2 c_i}\}_{i \in [N]}$

$\vdots$

$\mathrm{pk}_N = e(g,g)^{\alpha_N}, \{g^{\alpha_N c_i}\}_{i \in [N]}$

Aggregated public key is a commitment to $\alpha_1, \dots, \alpha_N$

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$

Now need a mechanism to sub-select only the keys for the users $i \in S \subseteq [N]$ in the signing quorum

Verification key for signing quorum S: $\mathrm{vk}_S = e(g,g)^{\sum_{i \in S} \alpha_i}$

Goal: transform z (commitment to all keys) to vk_S (commitment to S)

Strategy: publish g^{1/c_i} terms in the CRS

Aggregating Public Keys

hash key

commitment key

CRS: u, h $g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$

$\mathrm{pk}_1 = e(g,g)^{\alpha_1}, \{g^{\alpha_1 c_i}\}_{i \in [N]}$

$\mathrm{pk}_2 = e(g,g)^{\alpha_2}, \{g^{\alpha_2 c_i}\}_{i \in [N]}$

$\vdots$

$\mathrm{pk}_N = e(g,g)^{\alpha_N}, \{g^{\alpha_N c_i}\}_{i \in [N]}$

Aggregated public key is a commitment to $\alpha_1, \ldots, \alpha_N$

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$

Now need a mechanism to sub-select only the keys for the users $i \in S \subseteq [N]$ in the signing quorum

Verification key for signing quorum S: $\mathrm{vk}_S = e(g,g)^{\sum_{i \in S} \alpha_i}$

Goal: transform z (commitment to all keys) to vk_S (commitment to S)

Strategy: publish g^{1/c_i} terms in the CRS

Aggregating Public Keys

	hash key	commitment key	cross terms
CRS:	u, h	$g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$	$\forall i \neq j$: g^{c_i/c_j}

$z = g^{\sum_{i \in [N]} \alpha_i c_i}$

$\mathrm{pk}_1 = e(g,g)^{\alpha_1}, \{g^{\alpha_1 c_i}\}_{i \in [N]}$

$\mathrm{pk}_2 = e(g,g)^{\alpha_2}, \{g^{\alpha_2 c_i}\}_{i \in [N]}$

$\vdots$

$\mathrm{pk}_N = e(g,g)^{\alpha_N}, \{g^{\alpha_N c_i}\}_{i \in [N]}$

To construct cross terms:

- CRS needs cross terms g^{c_i/c_j}
- User public keys need $g^{\alpha c_i/c_j}$

Goal: transform z (commitment to all keys) to vk_S (commitment to S)

Strategy: publish g^{1/c_i} terms in the CRS

$$e\left(z, g^{\sum_{i \in S} 1/c_i}\right) = e(g,g)^{\sum_{i \in S} \alpha_i} \cdot e(g,g)^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \alpha_i c_i / c_j}$$

set selector | target quantity | "cross terms"

Aggregate signature will contain:

- Set selector $g^{\sum_{i \in S} 1/c_i}$
- Cross terms $g^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \alpha_i c_i / c_j}$

Verifier can use these to compute the target key $e(g,g)^{\sum_{i \in S} \alpha_i}$

Aggregating Public Keys

	hash key	commitment key	cross terms
CRS:	u, h	$g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$	$\forall i \neq j$: g^{c_i/c_j}

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$

$\mathrm{pk}_1 = e(g,g)^{\alpha_1}, \{g^{\alpha_1 c_i}, g^{\alpha_1 c_i / c_j}\}_{i \neq j}$

$\mathrm{pk}_2 = e(g,g)^{\alpha_2}, \{g^{\alpha_2 c_i}, g^{\alpha_2 c_i / c_j}\}_{i \neq j}$

⋮

$\mathrm{pk}_N = e(g,g)^{\alpha_N}, \{g^{\alpha_N c_i}, g^{\alpha_N c_i / c_j}\}_{i \neq j}$

To construct cross terms:

- CRS needs cross terms g^{c_i/c_j}
- User public keys need $g^{\alpha c_i / c_j}$

Quadratic-size CRS + user keys

Goal: transform z (commitment to all keys) to vk_S (commitment to S)

Strategy: publish g^{1/c_i} terms in the CRS

$$e\left(z, g^{\sum_{i \in S} 1/c_i}\right) = e(g,g)^{\sum_{i \in S} \alpha_i} \cdot e(g,g)^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \alpha_i c_i / c_j}$$

set selector — target quantity — "cross terms"

Aggregate signature will contain:

- Set selector $g^{\sum_{i \in S} 1/c_i}$
- Cross terms $g^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \alpha_i c_i / c_j}$

Verifier can use these to compute the target key $e(g,g)^{\sum_{i \in S} \alpha_i}$

Aggregating Public Keys

	hash key	commitment key	cross terms
CRS:	u, h	$g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$	$\forall i \neq j\colon\ g^{c_i/c_j}$

$\text{pk}_1 = e(g,g)^{\alpha_1}, \{g^{\alpha_1 c_i}, g^{\alpha_1 c_i/c_j}\}_{i \neq j}$

$\text{pk}_2 = e(g,g)^{\alpha_2}, \{g^{\alpha_2 c_i}, g^{\alpha_2 c_i/c_j}\}_{i \neq j}$

⋮

$\text{pk}_N = e(g,g)^{\alpha_N}, \{g^{\alpha_N c_i}, g^{\alpha_N c_i/c_j}\}_{i \neq j}$

Remaining wrinkle: how do we know that S satisfies the policy?

Strategy: add an interpolation check

Goal: transform z (commitment to all keys) to vk_S (commitment to S)

Strategy: publish g^{1/c_i} terms in the CRS

$$e\left(z, g^{\sum_{i \in S} 1/c_i}\right) = e(g,g)^{\sum_{i \in S} \alpha_i} \cdot e(g,g)^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \alpha_i c_i / c_j}$$

set selector — target quantity — "cross terms"

Aggregate signature will contain:

- Set selector $g^{\sum_{i \in S} 1/c_i}$
- Cross terms $g^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \alpha_i c_i / c_j}$

Verifier can use these to compute the target key $e(g,g)^{\sum_{i \in S} \alpha_i}$

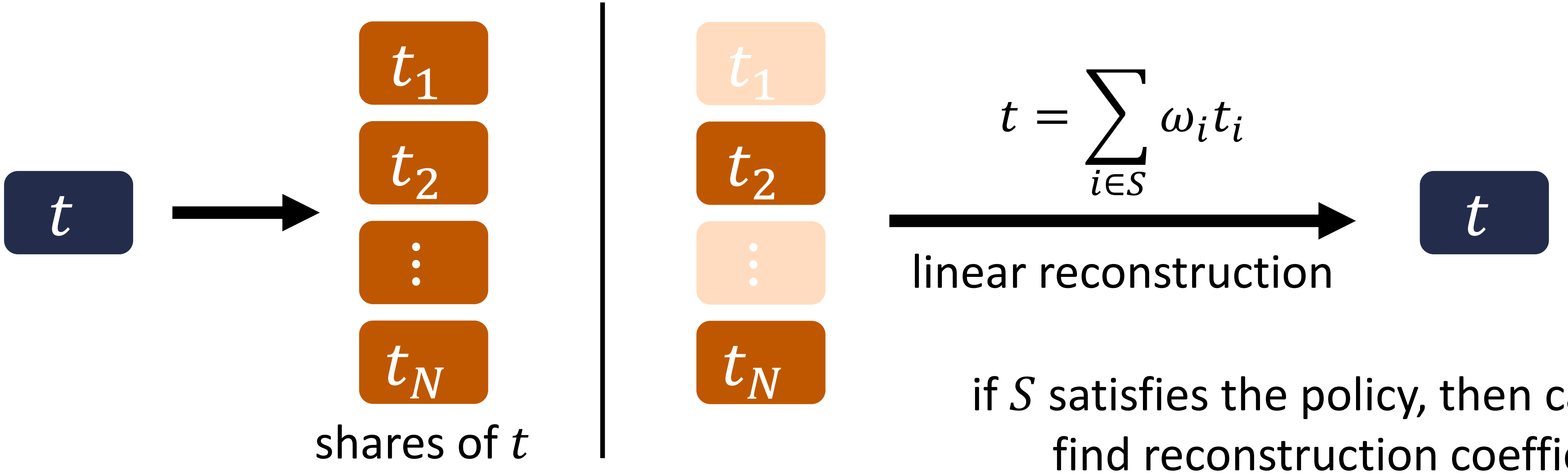
Policy Certification

hash key | commitment key | cross terms

CRS: u, h | $g^{c_1}, \dots, g^{c_N}, g^{1/c_1}, \dots, g^{1/c_N}$ | $\forall i \neq j$: g^{c_i/c_j}

For simplicity: suppose that the policy P is **known in advance**

We will assume that P has a linear secret sharing scheme



if S satisfies the policy, then can efficiently find reconstruction coefficients ω_i

Policy Certification

	hash key	commitment key	cross terms
CRS:	u, h	$g^{c_1}, \dots, g^{c_N}, g^{1/c_1}, \dots, g^{1/c_N}$	$\forall i \neq j$: g^{c_i/c_j}

Suppose one has signature from each party $i \in S$

We will rely on linear homomorphism: $\tilde{\sigma}_i = (\underbrace{g^{\alpha_i}(u^m h)^{r_i}}_{\tilde{\sigma}_{i,1}}, \underbrace{g^{r_i}}_{\tilde{\sigma}_{i,2}})$

$$\sigma_{\text{agg},1} = \prod_{i \in S} \tilde{\sigma}_{i,1} = g^{\sum_{i \in S} \alpha_i}(u^m h)^{\sum_{i \in S} r_i} = g^{\sum_{i \in S} \alpha_i}(u^m h)^{\tilde{r}}$$

$$\sigma_{\text{agg},2} = \prod_{i \in S} \tilde{\sigma}_{i,2} = g^{\sum_{i \in S} r_i} = g^{\tilde{r}}$$

$\sigma_{\text{agg}} = (\sigma_{\text{agg},1}, \sigma_{\text{agg},2})$ is a Boneh-Boyen signature on m with respect to $e(g,g)^{\sum_{i \in S} \alpha_i}$

Policy Certification

	hash key	commitment key	cross terms
CRS:	u, h	$g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$	$\forall i \neq j\colon\ g^{c_i/c_j}$

Suppose one has signature from each party $i \in S$

ω_i are interpolation coefficients

We will rely on linear homomorphism: $$\tilde{\sigma}_i = (\underbrace{g^{\alpha_i}(u^m h)^{r_i}}_{\tilde{\sigma}_{i,1}}, \underbrace{g^{r_i}}_{\tilde{\sigma}_{i,2}})$$

New target key:
$\mathrm{vk}_S = e(g,g)^{\sum_{i\in S}\omega_i\alpha_i}$

$$\sigma_{\mathrm{agg},1} = \prod_{i\in S} \tilde{\sigma}_{i,1}^{\omega_i} = g^{\sum_{i\in S}\omega_i\alpha_i}(u^m h)^{\sum_{i\in S}\omega_i r_i} = g^{\sum_{i\in S}\omega_i\alpha_i}(u^m h)^{\tilde{r}}$$

$$\sigma_{\mathrm{agg},2} = \prod_{i\in S} \tilde{\sigma}_{i,2}^{\omega_i} = g^{\sum_{i\in S}\omega_i r_i} = g^{\tilde{r}}$$

$\sigma_{\mathrm{agg}} = (\sigma_{\mathrm{agg},1}, \sigma_{\mathrm{agg},2})$ is a Boneh-Boyen signature on m with respect to $e(g,g)^{\sum_{i\in S}\omega_i\alpha_i}$

Policy Certification

	hash key	commitment key	cross terms
CRS:	u, h	$g^{c_1}, \dots, g^{c_N}, g^{1/c_1}, \dots, g^{1/c_N}$	$\forall i \neq j\colon\ g^{c_i/c_j}$

Aggregated public key is a commitment to $\alpha_1, \dots, \alpha_n$

$$z = g^{\sum_{i \in [N]} \alpha_i c_i}$$

Suppose the target verification key is $e(g,g)^{\sum_{i \in S} \omega_i \alpha_i}$. Then:

$$e\left(z, g^{\sum_{i \in S} \omega_i / c_i}\right) = e(g,g)^{\sum_{i \in S} \omega_i \alpha_i} \cdot e(g,g)^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \omega_j \alpha_i c_i / c_j}$$

set selector — target quantity — "cross terms"

Policy Certification

	hash key	commitment key	cross terms	policy check
CRS:	u, h	$g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$	$\forall i \neq j$: g^{c_i/c_j}	$g^{\sum_{i \in [N]} c_i t_i}, g^{t_1}, \ldots, g^{t_N}$

Sample target $t \leftarrow \mathbb{Z}_p$ and let $t_1, \ldots, t_N$ be a linear secret sharing of t according to the policy P

Publish $g^{t_1}, \ldots, g^{t_N}$ and the commitment to the shares $g^{\sum_{i \in [N]} c_i t_i}$ in the CRS

Aggregated public key is now a commitment to $\alpha_1 + t_i, \ldots, \alpha_N + t_N$

$$z = g^{\sum_{i \in [N]} c_i (\alpha_i + t_i)}$$

Then selecting for a set S yields

Add additional cross terms $g^{t_k c_i / c_j}$ to CRS

$$e\left(z, g^{\sum_{i \in S} \omega_i / c_i}\right) = e(g,g)^{\sum_{i \in S} \omega_i \alpha_i} \cdot e(g,g)^{\sum_{i \in S} \omega_i t_i} \cdot e(g,g)^{\sum_{i \in [N]} \sum_{j \in S \setminus \{i\}} \omega_j (\alpha_i + t_i) c_i / c_j}$$

set selector — target quantity — $e(g,g)^t$ — cross terms

Ensures that the aggregator can only select S that interpolates to $e(g,g)^t$

Policy Certification

	hash key	commitment key	cross terms	policy check
CRS:	u, h	$g^{c_1}, \dots, g^{c_N}, g^{1/c_1}, \dots, g^{1/c_N}$	$\forall i \neq j$: g^{c_i/c_j}	$g^{\sum_{i\in[N]} c_i t_i}, g^{t_1}, \dots, g^{t_N}$ $\forall i \neq j$: $g^{t_1 c_i/c_j}, \dots, g^{t_N c_i/c_j}$

Sample target $t \leftarrow \mathbb{Z}_p$ and let $t_1, \dots, t_N$ be a linear secret sharing of t according to the policy P

Publish $g^{t_1}, \dots, g^{t_N}$ and the commitment to the shares $g^{\sum_{i\in[N]} c_i t_i}$ in the CRS

Aggregated public key is now a commitment to $\alpha_1 + t_i, \dots, \alpha_N + t_N$

$$z = g^{\sum_{i\in[N]} c_i(\alpha_i + t_i)}$$

Then selecting for a set S yields

Add additional cross terms $g^{t_k c_i/c_j}$ to CRS

$$e\left(z, g^{\sum_{i\in S} \omega_i/c_i}\right) = e(g,g)^{\sum_{i\in S} \omega_i \alpha_i} \cdot e(g,g)^{\sum_{i\in S} \omega_i t_i} \cdot e(g,g)^{\sum_{i\in[N]} \sum_{j\in S\setminus\{i\}} \omega_j (\alpha_i + t_i) c_i/c_j}$$

set selector — target quantity — $e(g,g)^t$ — cross terms

Ensures that the aggregator can only select S that interpolates to $e(g,g)^t$

Putting the Pieces Together

CRS:

- hash key: u, h
- commitment key: $g^{c_1}, \ldots, g^{c_N}, g^{1/c_1}, \ldots, g^{1/c_N}$
- cross terms: $\forall i \neq j\colon\ g^{c_i/c_j}$
- policy check: $g^{\sum_{i\in[N]} c_i t_i}, g^{t_1}, \ldots, g^{t_N}$; $\forall i \neq j\colon\ g^{t_1 c_i/c_j}, \ldots, g^{t_N c_i/c_j}$

sk

pk

$\mathrm{sk} = \alpha$

$\mathrm{pk} = e(g,g)^{\alpha}, \forall i \neq j\colon\ g^{\alpha c_i}, g^{\alpha c_i/c_j}$

m σ

$\sigma = (g^{\alpha}(u^m h)^r, g^r)$

pk_1

pk_2

$\vdots$

pk_N

Aggregated public key:

$$z = g^{\sum_{i\in[N]} c_i(\alpha_i + t_i)}$$

Given signatures

$$\forall i \in S\colon\ \tilde{\sigma}_i = (g^{\alpha_i}(u^m h)^{r_i}, g^{r_i}) = (\tilde{\sigma}_{i,1}, \tilde{\sigma}_{i,2})$$

Aggregated signature:

$$\sigma_{\mathrm{agg},1} = \prod_{i\in S} \tilde{\sigma}_{i,1}^{\omega_i} \qquad \sigma_{\mathrm{agg},2} = \prod_{i\in S} \tilde{\sigma}_{i,2}^{\omega_i}$$

$$\sigma_{\mathrm{agg},3} = g^{\sum_{i\in S} \omega_i / c_i}$$ selector

$$\sigma_{\mathrm{agg},4} = g^{\sum_{i\in[N]} \sum_{j\in S\setminus\{i\}} \omega_j(\alpha_i + t_i) c_i / c_j}$$ cross-term

Verification:

$$\frac{e(g, \sigma_{\mathrm{agg},1})}{e(\sigma_{\mathrm{agg},2}, u^m h)} \cdot e(g,g)^t \cdot e(g, \sigma_{\mathrm{agg},4}) = e(z, \sigma_{\mathrm{agg},3})$$

Putting the Pieces Together

CRS:

hash key	commitment key	cross terms	policy check
u, h	$g^{c_1}, \dots, g^{c_N}, g^{1/c_1}, \dots, g^{1/c_N}$	$\forall i \neq j$: g^{c_i/c_j}	$g^{\sum_{i\in[N]} c_i t_i}, g^{t_1}, \dots, g^{t_N}$ $\forall i \neq j$: $g^{t_1 c_i/c_j}, \dots, g^{t_N c_i/c_j}$

sk

pk

$\mathrm{sk} = \alpha$

$\mathrm{pk} = e(g,g)^{\alpha}, \forall i \neq j: \; g^{\alpha c_i}, g^{\alpha c_i / c_j}$

m σ

$\sigma = (g^{\alpha}(u^m h)^r, g^r)$

pk_1

pk_2

⋮

pk_N

Since $\sigma_{\mathrm{agg},1}$ and $\sigma_{\mathrm{agg},4}$ both paired with g, we can collapse them into a single group element: resulting signature has 3 group elements

Given signatures

$$\forall i \in S: \; \tilde{\sigma}_i = (g^{\alpha_i}(u^m h)^{r_i}, g^{r_i}) = (\tilde{\sigma}_{i,1}, \tilde{\sigma}_{i,2})$$

Aggregated signature:

$$\sigma_{\mathrm{agg},1} = \prod_{i\in S} \tilde{\sigma}_{i,1}^{\omega_i} \qquad \sigma_{\mathrm{agg},2} = \prod_{i\in S} \tilde{\sigma}_{i,2}^{\omega_i}$$

$$\sigma_{\mathrm{agg},3} = g^{\sum_{i\in S} \omega_i / c_i}$$ selector

$$\sigma_{\mathrm{agg},4} = g^{\sum_{i\in[N]} \sum_{j\in S\setminus\{i\}} \omega_j (\alpha_i + t_i) c_i / c_j}$$ cross-term

Verification:

$$\frac{e(g, \sigma_{\mathrm{agg},1})}{e(\sigma_{\mathrm{agg},2}, u^m h)} \cdot e(g,g)^t \cdot e(g, \sigma_{\mathrm{agg},4}) = e(z, \sigma_{\mathrm{agg},3})$$

Extensions

	hash key	commitment key	cross terms	policy check
CRS:	u, h	$g^{c_1}, \dots, g^{c_N}, g^{1/c_1}, \dots, g^{1/c_N}$	$\forall i \neq j\colon\ g^{c_i/c_j}$	$g^{\sum_{i \in [N]} c_i t_i}, g^{t_1}, \dots, g^{t_N}$ $\forall i \neq j\colon\ g^{t_1 c_i/c_j}, \dots, g^{t_N c_i/c_j}$

Extensions:

Can prove security from a q-type assumption on pairing groups (plain model)

Can support dynamic policies by having the public-key aggregation generate the shares $t_1, \dots, t_n$ at the cost of a larger CRS (cubic in number of parties)

Can take the $c_i = c^i$ to be powers to reduce to a linear-size CRS for fixed policy (quadratic for dynamic policy)

Same techniques also gives a silent threshold encryption scheme

(Recall that Boneh-Boyen is actually an identity-based encryption scheme)

Summary

Scheme	Policy Family	Assumption	$\lvert\text{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\text{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\text{poly}(\lambda) \cdot \lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N \log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Take-away: algebraic framework yields schemes for **general policies** with **shorter aggregate signatures** and **security in the plain model**

Cost: **larger CRS** (for dynamic threshold policies, difference is quasilinear vs. strictly linear)

Open Problems

Pairing-based schemes with transparent setup (and comparable signature/ciphertext size)

Silent threshold cryptography from post-quantum cryptographic assumptions

Thanks!

https://eprint.iacr.org/2025/1547.pdf