Silent Threshold Cryptography from Pairings

David Wu

UT Austin

January 26, 2026

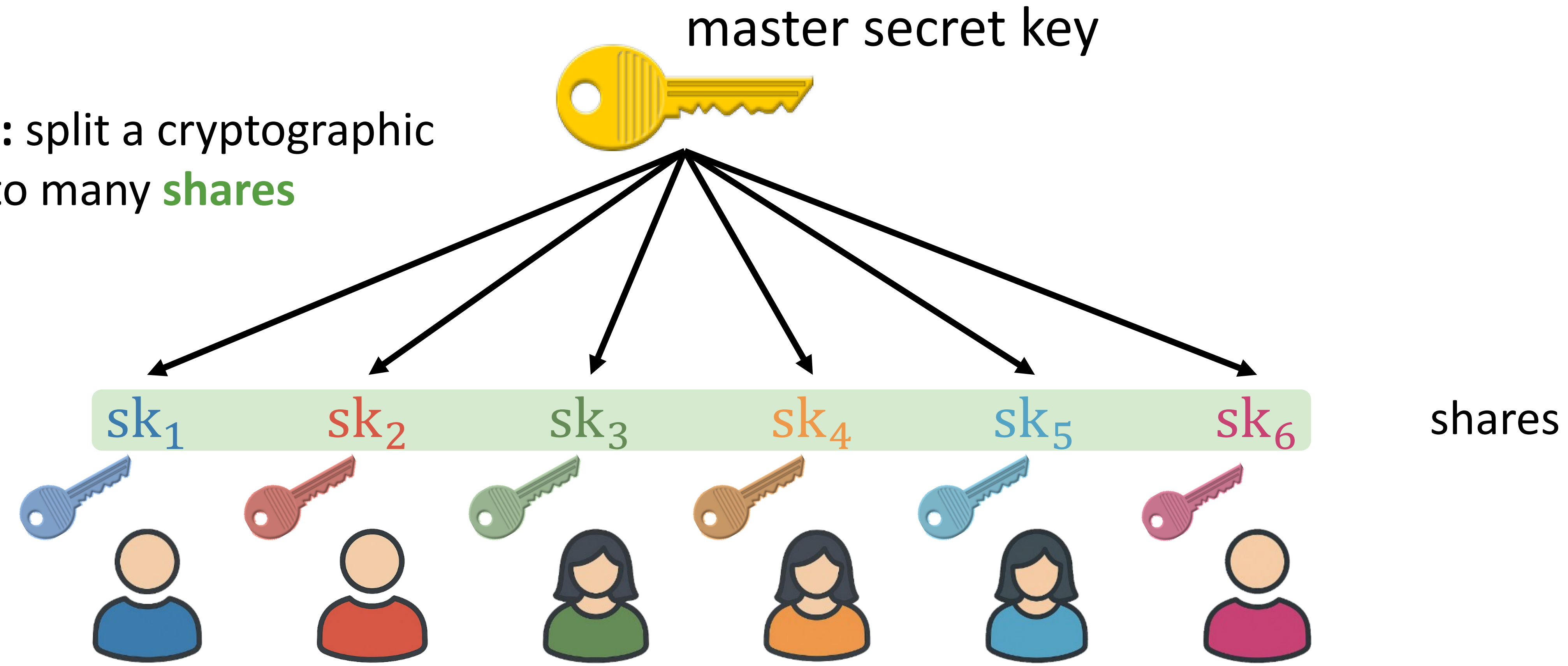
NIST Workshop on Multi-Party Threshold Schemes

joint work with Brent Waters

Threshold Cryptography

[Des87, Fra89, DF89]

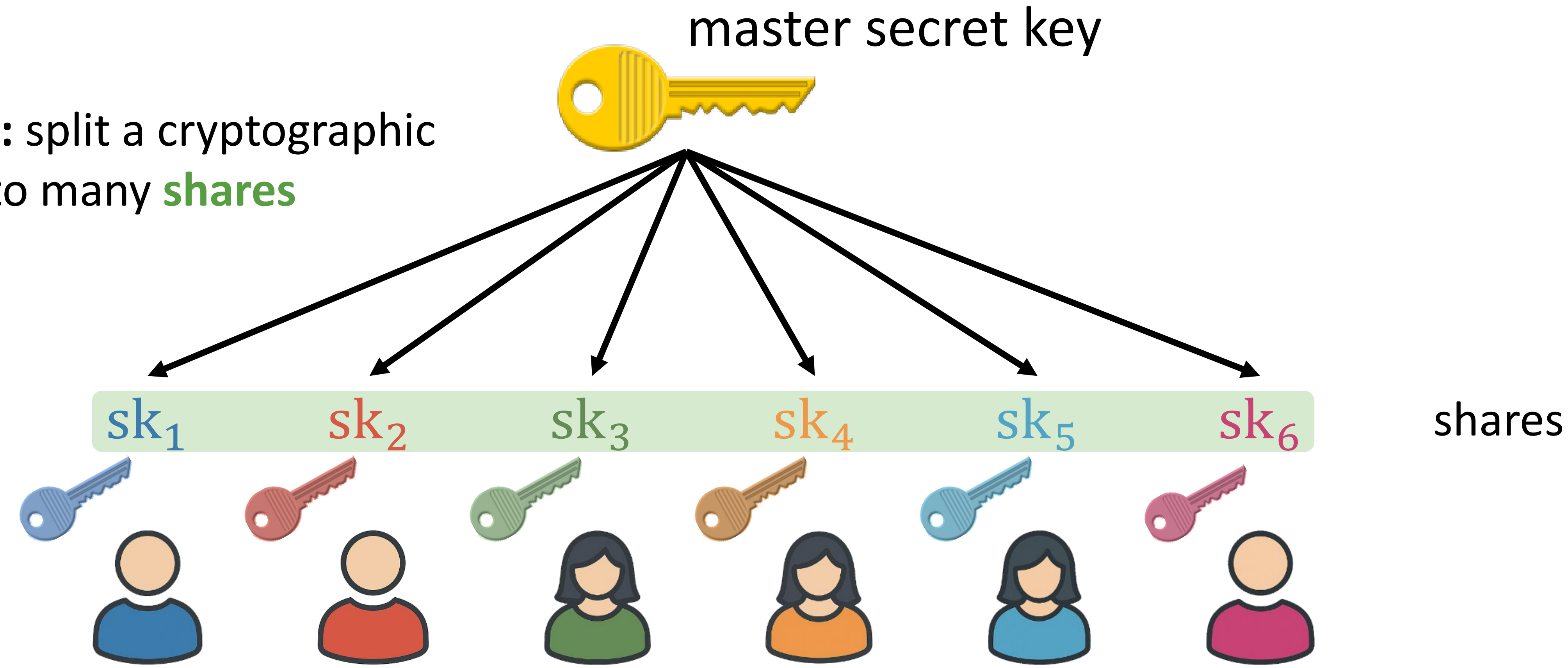
Typical setup: split a cryptographic key into many **shares**



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

Who Generates the Shares?

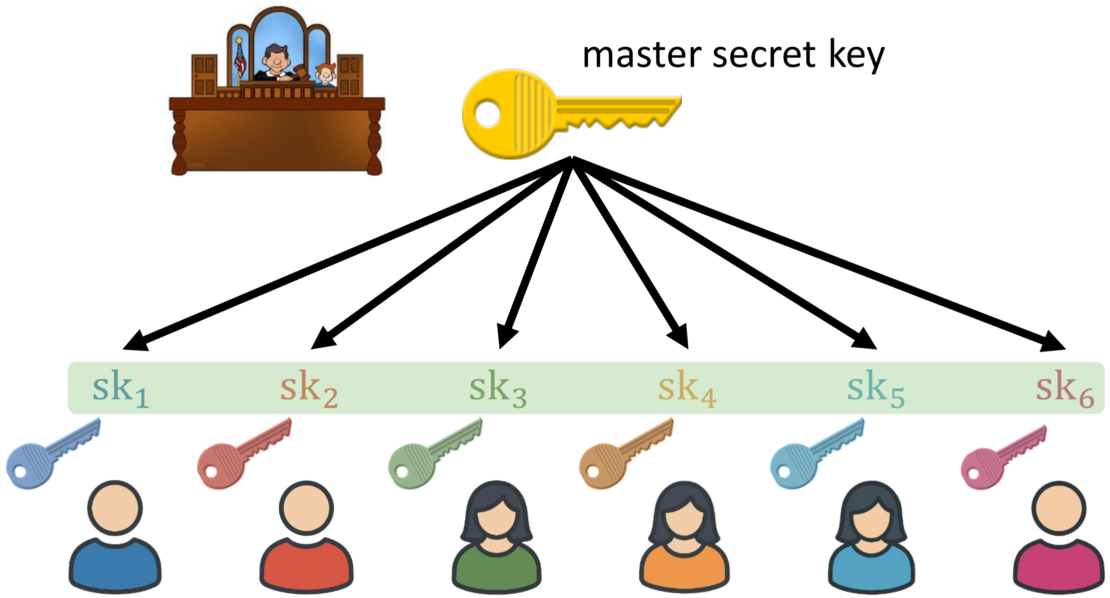
Typical setup: split a cryptographic key into many **shares**



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

Who Generates the Shares?

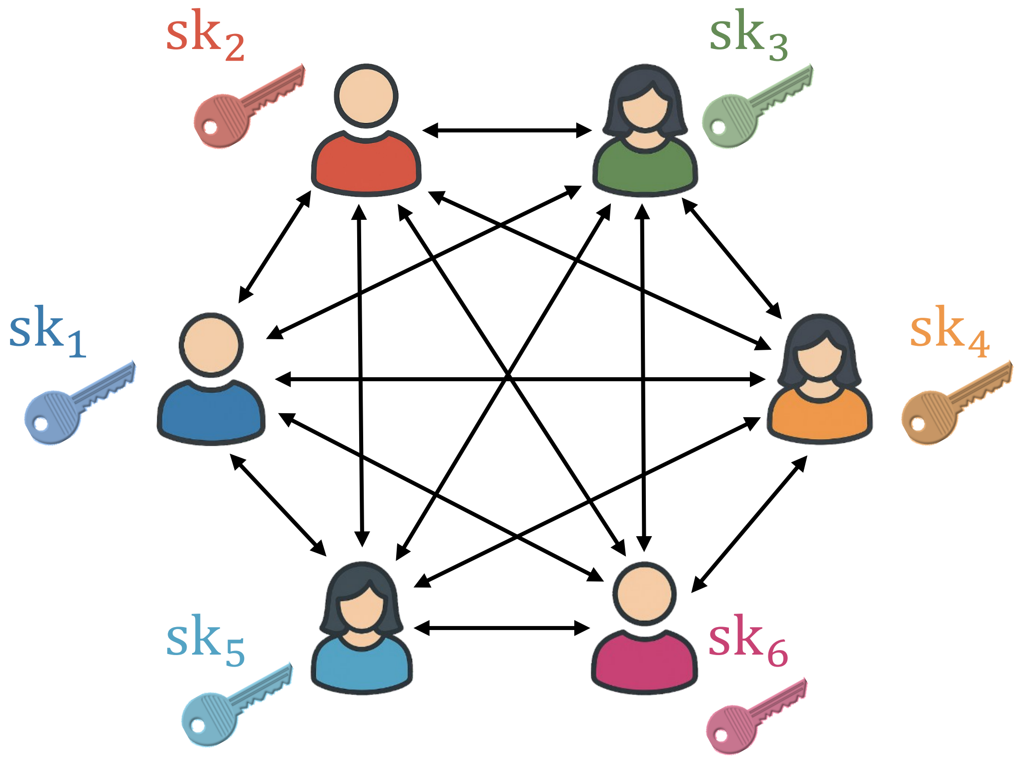
Option 1: Trusted dealer



Needs a trusted party

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

Option 2: Distributed key generation

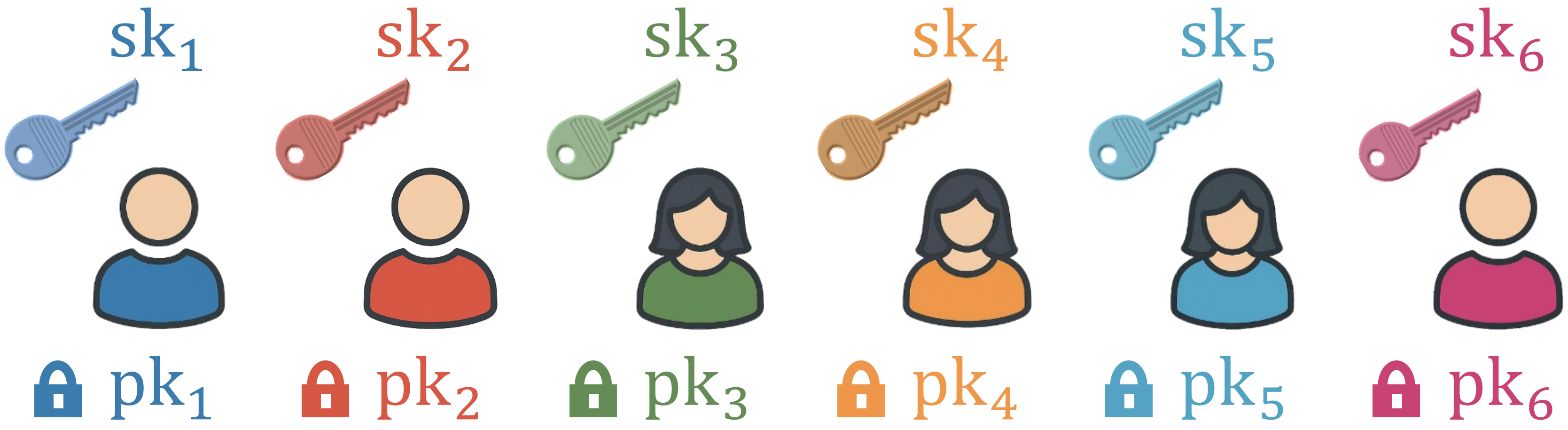


Requires parties to coordinate and interact

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

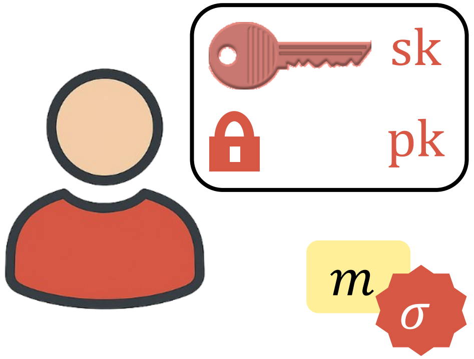
Can **deterministically** aggregate any set of public keys together with a policy

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)

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

Signing key sk can be used to sign messages

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

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

T: target threshold

mpk: aggregated verification key for quorum

ht: aggregation hint

mpk

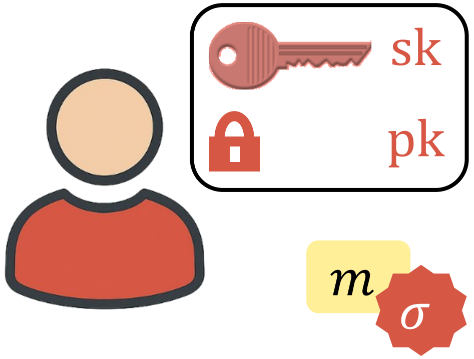
arbitrary collection of public keys

policy

aggregated public key

Example: Threshold Signatures with Silent Setup

[MRVWZ21, DCXNBR23, GJMSW24]



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

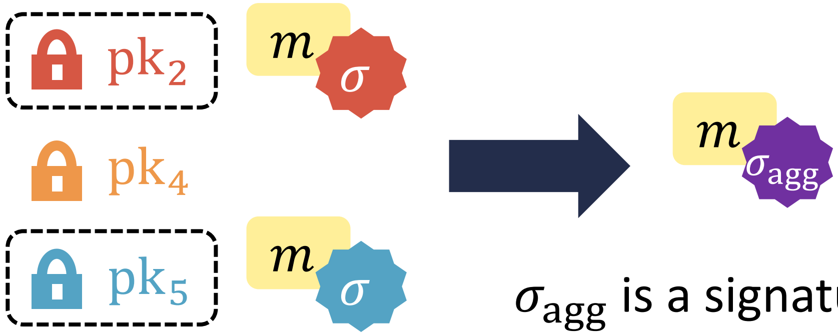
$\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$

Signing key sk can be used to sign messages

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

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

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



σ_{agg} is a signature on m under mpk

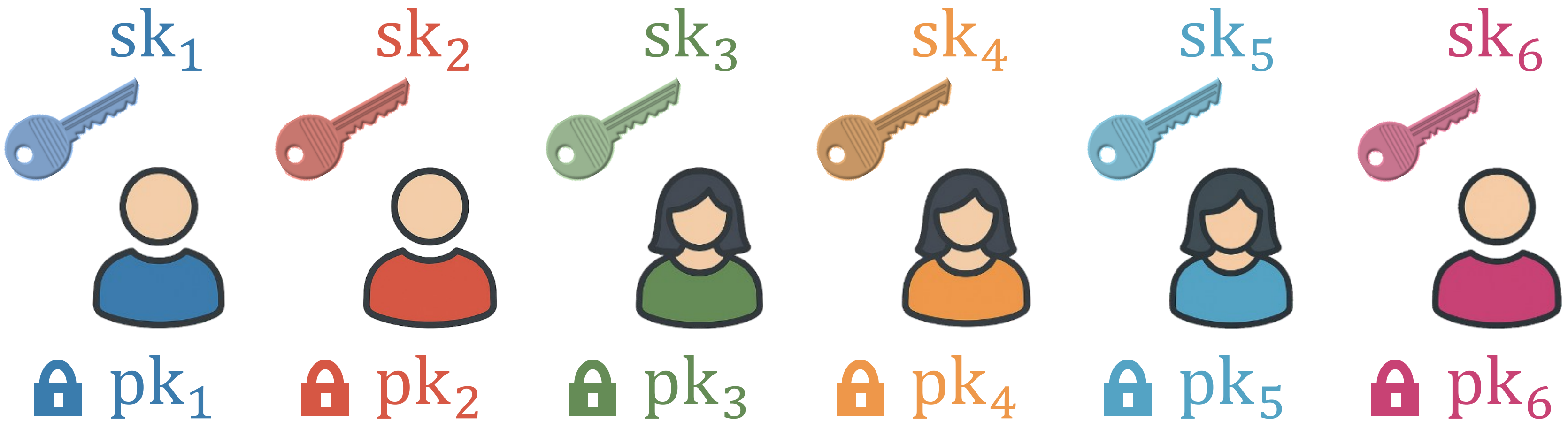
Efficiency: $|\sigma_{\mathsf{agg}}|$ and signature verification time are independent of number of users

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

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

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	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\mathrm{agg}}\rvert$
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	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$

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	$\lvert\text{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\text{agg}}\rvert$
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	$O_\lambda(N \log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

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	$\lvert \text{crs} \rvert$	$\lvert \sigma \rvert$	$\lvert \sigma_{\text{agg}} \rvert$
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	$O_\lambda(N \log N)$	$2\lvert \mathbb{G} \rvert$	$3\lvert \mathbb{G} \rvert$

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 $(\mathbb{G}, \mathbb{G}_T)$
- Group order p
- Generator g
- Pairing $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$

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

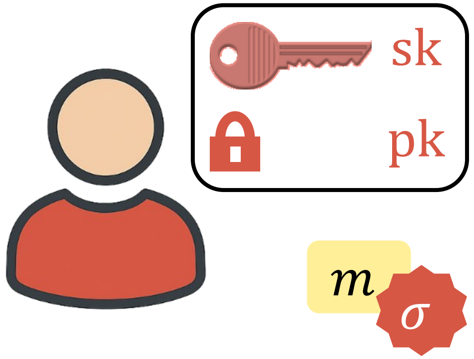
Verification: check that $e(g,g)^\alpha = \frac{e(g, g^\alpha(u^m h)^r)}{e(g^r, u^m h)}$

"decrypt in the target group via the pairing"

Construction Template

hash key

CRS: u, h



$\mathrm{sk} = \alpha$

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

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

Each user chooses their own Boneh-Boyen public key

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

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

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

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

$\sigma_{\text{agg}} = (\sigma_{\text{agg},1}, \sigma_{\text{agg},2})$ 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 aggregated verification key $\prod_{i \in S} e(g,g)^{\alpha_i}$ and check the signature

Our approach: Derive aggregated key $e(g,g)^{\sum_{i \in S} \alpha_i}$ from the pairing implicitly

Aggregating Public Keys

hash key   commitment key

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

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

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

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

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

Aggregating Public Keys

hash key — commitment key

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

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 $\alpha_1, \dots, \alpha_n$

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

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

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

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

set selector — target quantity — cross terms

Aggregating Public Keys

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

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

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

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

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

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

set selector — target quantity — cross terms

Still need to certify that S satisfies the policy

[see paper for details]

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