

Locally Verifiable Aggregated Multi-signature Schemes

Ahmet Ramazan Ağırtaş¹ Neslihan Yaman Gökce² Oğuz Yayla²

¹AYWARE, Ankara, Turkey

²Middle East Technical University, Ankara, Turkey

NuTMiC - September 23, 2026

Outline

- 1 Introduction
- 2 Preliminaries
- 3 LVAMS: Locally Verifiable Aggregated Multi-Signatures
- 4 Comparison
- 5 Conclusion

Outline

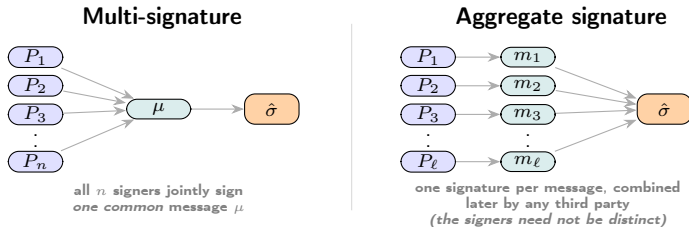
- 1 Introduction
- 2 Preliminaries
- 3 LVAMS: Locally Verifiable Aggregated Multi-Signatures
- 4 Comparison
- 5 Conclusion

Digital Signatures at Scale

Digital signatures are crucial for ensuring the integrity and authenticity of electronic data, facilitating trust and verification in a wide range of applications.

- blockchain transactions, jointly controlled accounts, joint custody of digital assets
- certificate transparency logs storing millions of certificates
- distributed ledger technologies

In all of these, the number of signatures grows with the size of the system, so transmitting and storing them quickly becomes expensive. Two primitives compress many signatures into a single short one:



The Verification Bottleneck

The problem

Standard verification of aggregate signatures and multi-signatures requires processing *all* underlying messages and *all* signers, respectively.

$$\text{Verification cost} = \mathcal{O}(\ell) \text{ or } \mathcal{O}(n)$$

But often a verifier only cares whether *one specific* message or signer is included within the signature.

- Certificate transparency logs: does my certificate appear in the compressed log?
- Blockchain: did a specific signer co-sign this transaction?
- Medical IoT (IoMT): is this one measurement authenticated?

LVAS: Locally Verifiable Aggregate Signatures

Goyal–Vaikuntanathan (CRYPTO 2022)[1]

A short opening aux , generated by a `LOCALOPEN` algorithm, lets a verifier check that one designated **message** is inside an aggregate signature *without* processing the complete message set.

- $\text{LOCALOPEN}(\hat{\sigma}, vk, \{m_i\}_{i \in [\ell]}, j \in [\ell]) \rightarrow aux_j$
- $\text{LOCALAGGVERIFY}(\hat{\sigma}, vk, m, aux) \rightarrow 0/1$

- Local verification cost = $\mathcal{O}(1)$, independent of ℓ , which is the number of aggregated messages.
- Two single-signer constructions were given (RSA- and BDHI-based).

Ağırtaş et al. [2]

A short opening aux , generated by a LOCALOPEN algorithm, lets a verifier check that one designated **signer** has contributed a multi-signature *without* processing the verification key set.

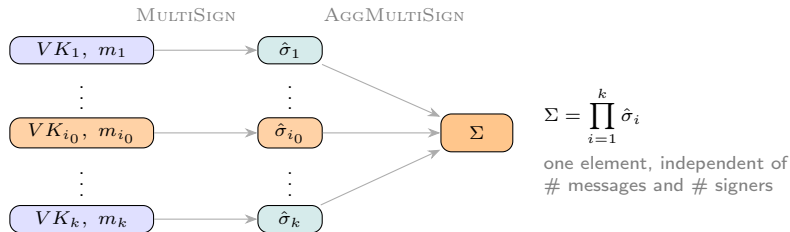
- $\text{LOCALOPEN}(j_1, j_2, \dots, j_k, m, VK) \rightarrow \text{aux}$.
 - $\text{LOCALAGGVERIFY}(\text{aux}, \{vk_{j_i}\}_{i=1}^k, m, \hat{\sigma}, VK) \rightarrow 0/1$.
-
- Local verification cost = $\mathcal{O}(1)$, independent of n , which is the number of signers contributed to the multi-signature $\hat{\sigma}$.
 - A BDHI-based construction was given.

Motivation and Contribution

The previous two schemes LVAS and LVMS keep one thing fixed:

- LVAS: one signer, one message.
- LVMS: one message, many signers.

In practice: different signer groups sign different messages, and everything gets stored together.



The question: did signer $vk_0 \in VK_{i_0}$ sign message m_{i_0} ?

Contribution. Local verification of a designated message-signer pair, after *both* aggregation layers, with the signer set allowed to differ from one message to the next.

Outline

- 1 Introduction
- 2 Preliminaries
- 3 LVAMS: Locally Verifiable Aggregated Multi-Signatures
- 4 Comparison
- 5 Conclusion

Bilinear Pairings

Bilinear map

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T, \quad e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}.$$

Computational co-Diffie–Hellman Problem

Given g_1^α , g_1^β , and g_2^β , the co-CDH problem assumes that computing $g_1^{\alpha\beta}$ is computationally hard.

Computational ψ -co-Diffie–Hellman Problem

The ψ -co-CDH problem is a stronger variant of co-CDH. The adversary has access to an oracle

$$\mathcal{O}^\psi(g_2^x) = g_1^x,$$

which maps elements from $\mathbb{G}_2$ to $\mathbb{G}_1$. Even with this oracle, computing $g_1^{\alpha\beta}$ from g_1^α , g_1^β , and g_2^β is assumed to be computationally hard.

$\mathcal{MSP}$

- A BLS-based multi-signature scheme with key aggregation, where all signers sign one common message.
- Unforgeable under the co-CDH assumption in the random oracle model.

$\mathcal{AMSP}$

- An aggregate multi-signature scheme that combines several $\mathcal{MSP}$ multi-signatures on distinct messages, possibly with different signer sets, into a single group element.
- Unforgeable under the ψ -co-CDH assumption in the random oracle model.

Roles

Three roles: signers (SETUP, SIGN), an aggregator (AGGREGATE, LOCALOPEN), verifiers (VERIFY, AGGVERIFY, LOCALAGGVERIFY).

$$H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1, \quad H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q, \quad VK = \{vk_1, \dots, vk_n\}.$$

- 1 $\text{SETUP}(1^\lambda) \rightarrow (vk_i, sk_i): \quad sk_i \xleftarrow{\$} \mathbb{Z}_q, \quad vk_i = g_2^{sk_i}, \quad \Pi = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$
- 2 $\text{SIGN}(sk_i, m, VK) \rightarrow \sigma_i: \quad \sigma_i = H_0(m)^{a_i \cdot sk_i}, \quad \text{where } a_i = H_1(vk_i, VK)$
- 3 $\text{VERIFY}(\sigma_i, m, vk_i, VK) \rightarrow 0/1: \quad e(\sigma_i, g_2) \stackrel{?}{=} e(H_0(m), vk_i^{a_i})$
- 4 $\text{AGGREGATE}(\{\sigma_i\}_{i=1}^n, m, VK) \rightarrow \hat{\sigma} / \perp: \quad \hat{\sigma} = \prod_{i=1}^n \sigma_i$
- 5 $\text{AGGVERIFY}(\hat{\sigma}, m, VK) \rightarrow 0/1: \quad e(\hat{\sigma}, g_2) \stackrel{?}{=} e\left(H_0(m), \prod_{i=1}^n vk_i^{a_i}\right)$

Target signers $\{j_1, \dots, j_k\} \subset [n]$

- 6 **LOCALOPEN** $(j_1, \dots, j_k, m, VK) \rightarrow \text{aux}$: It computes the product over the signers *not* being verified:

$$\text{aux} = e\left(H_0(m), \prod_{\substack{i=1 \\ i \neq j_1, \dots, j_k}}^n vk_i^{a_i}\right), \quad \text{where } a_i = H_1(vk_i, VK)$$

- 7 **LOCALAGGVERIFY** $(\text{aux}, \{vk_{j_i}\}_{i=1}^k, m, \hat{\sigma}, VK) \rightarrow 0/1$:

$$e(\hat{\sigma}, g_2) \stackrel{?}{=} \text{aux} \cdot e\left(H_0(m), \prod_{i=1}^k vk_{j_i}^{a_{j_i}}\right), \quad \text{where } a_i = H_1(vk_i, VK)$$

Why it works. AGGVERIFY's product splits in two:

$$\underbrace{\prod_{i=1}^n vk_i^{a_i}}_{\text{AGGVERIFY}} = \underbrace{\prod_{i \in [n] \setminus \{j_1, \dots, j_k\}} vk_i^{a_i}}_{\text{precomputed into aux}} \cdot \underbrace{\prod_{i=1}^k vk_{j_i}^{a_{j_i}}}_{\text{the verifier computes}}$$

Trust Model: What the Opening Generator Is (and Is Not)

The opening generator

A public service (aggregator, storage server, transparency log) that runs the deterministic public algorithms `AGGREGATE` and `LOCALOPEN`.

- It holds **no secret key** of any signer, so it is a delegated computation service, not a trusted third party in the classical sense.
- Its computation is **publicly auditable**: any party may recompute the opening in $\mathcal{O}(n)$.
- **Graceful degradation**: a verifier who does not trust it simply runs full `AGGVERIFY`, obtaining security with *no* trust assumption.

Local verification is an optimization over aggregate verification, not a replacement for it.

Outline

- 1 Introduction
- 2 Preliminaries
- 3 LVAMS: Locally Verifiable Aggregated Multi-Signatures
- 4 Comparison
- 5 Conclusion

LVAMS introduces a local verification mechanism that enables each individual message–signature pair inside the aggregated multi-signature to be verified independently.

- We have k messages and up to ℓ parties jointly generate a short multi-signature $\hat{\sigma}_j$ on each message m_j .
- **Layer 1:** a subset of signers jointly produces a multi-signature $\hat{\sigma}_j$ on message m_j .
- **Layer 2:** multi-signatures on distinct messages, possibly with different signer sets, are combined:
 $\Sigma = \prod_j \hat{\sigma}_j$.

LVAMS: Two-Layer Construction

- 1 $\text{SETUP}(1^\lambda) \rightarrow pp : pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \ell)$
- 2 $\text{KEYGEN}(pp) \rightarrow (vk_i, sk_i) : sk_i \xleftarrow{\$} \mathbb{Z}_q, \quad vk_i = g_2^{sk_i}$
- 3 $\text{SIGN}(pp, sk_i, m, VK) \rightarrow \sigma_i : \sigma_i \leftarrow H_0(VK, m)^{a_i \cdot sk_i}, \text{ where } a_i \leftarrow H_1(vk_i, VK)$
- 4 $\text{VERIFY}(pp, \sigma_i, m, vk_i, VK) \rightarrow 0/1 : e(\sigma_i, g_2) \stackrel{?}{=} e(H_0(VK, m), vk_i^{a_i}), \text{ where } a_i \leftarrow H_1(vk_i, VK)$
- 5 $\text{MULTISIGN}(pp, \{\sigma_i\}_{i=1}^n, VK, m) \rightarrow \hat{\sigma}/\perp : \text{ checks } n \leq \ell \text{ and verifies individual signatures } \sigma_i$
computes the multi-signature as follows: $\hat{\sigma} \leftarrow \prod_{i=1}^n \sigma_i$
- 6 $\text{MULTIVERIFY}(\hat{\sigma}, m, VK) \rightarrow 0/1 : \text{ checks } e(\hat{\sigma}, g_2) \stackrel{?}{=} e(H_0(VK, m), \prod_{i=1}^n vk_i^{a_i}) \text{ where } a_i \leftarrow H_1(vk_i, VK) \text{ and } n = |VK|.$
- 7 $\text{AGGMULTISIGN}(pp, \{\hat{\sigma}_i, VK_i, m_i\}_{i=1}^k) \rightarrow \Sigma/\perp : \text{ computes final aggregated multi-signature } \Sigma \leftarrow \prod_{i=1}^k \hat{\sigma}_i$
- 8 $\text{AGGMULTIVERIFY}(pp, \Sigma, \{VK_i, m_i\}_{i=1}^k) \rightarrow 0/1 : \text{ verifies the final aggregated multi-signature } e(\Sigma, g_2) \stackrel{?}{=} \prod_{i=1}^k e(H_0(VK_i, m_i), \prod_{vk \in VK_i} vk^{H_1(vk, VK_i)})$

Local verification of one message-signer pair

- 8 $\text{LOCALOPEN}(pp, \{VK_i, m_i\}_{i=1}^k, i_0 \in [k], vk_0 \in VK_{i_0}) \rightarrow \text{aux}$: generates a specific auxiliary value for the particular message m_{i_0} and a signer.

$$\text{aux} = e\left(H_0(VK_{i_0}, m_{i_0}), \prod_{\substack{vk_j \in VK_{i_0}, \\ vk_j \neq vk_0}} vk_j^{a_{i_0,j}}\right) \cdot \prod_{\substack{i=1, \\ i \neq i_0}}^k e\left(H_0(VK_i, m_i), \prod_{vk_j \in VK_i} vk_j^{a_{i,j}}\right),$$

where $a_{i,j} = H_1(vk_j, VK_i)$, and i_0 is the index of the message to be verified.

- 9 $\text{LOCALAGGVERIFY}(pp, \Sigma, \text{aux}, VK_{i_0}, m_{i_0}, vk_0) \rightarrow 0/1$: checks if Σ contains the signature of the signer, to whom the verification key vk_0 corresponds, on the message m_{i_0}

$$e(\Sigma, g_2) \stackrel{?}{=} \text{aux} \cdot e\left(H_0(VK_{i_0}, m_{i_0}), vk_0^{H_1(vk_0, VK_{i_0})}\right)$$

Remark: We also presented a generalized version for LOCALOPEN and LOCALAGGVERIFY where multiple message-signer pairs can be locally verified at once.

LVAMS: Four Forgery Types and Local Consistency

Security is defined via four forgery types, arising from the two aggregation layers:

	Forgery against	Security follows from	
I	VERIFY	BLS	inherited
II	MULTIVERIFY	$\mathcal{MSP}$, co-CDH	inherited
III	AGGMULTIVERIFY	$\mathcal{AMSP}$, ψ -co-CDH	inherited
IV	LOCALAGGVERIFY	Type III	this work

Lemma (Local consistency)

With an honest opening, LOCALAGGVERIFY checks *exactly the same equation* as AGGMULTIVERIFY. Hence every Type IV forgery already is a Type III forgery.

$\Rightarrow$ LVAMS is unforgeable under co-CDH / ψ -co-CDH, with honestly generated openings.

Outline

- 1 Introduction
- 2 Preliminaries
- 3 LVAMS: Locally Verifiable Aggregated Multi-Signatures
- 4 Comparison**
- 5 Conclusion

LVAMS: Efficiency Gains

- **SETUP, KEYGEN, VERIFY, MULTISIGN, MULTIVERIFY:** identical cost to LVMS.
- **LOCALAGGVERIFY:** constant cost, independent of the total number of messages and signers
- **Remark:** When all messages have the same signer set, **AGGMULTIVERIFY** and **LOCALOPEN** require only 2 pairings instead of $k + 1$.

LVAMS = aggregation efficiency of $\mathcal{AMSP}$ + local verifiability of LVMS

We take the following restrictions into account for a meaningful comparison:

- In LVMS, we set the number of signers locally verified at once to 1.
- We consider k instantiation of LVMS so that it covers k different messages, just like LVAMS.
- We consider the generalized LOCALAGGVERIFY for LVAMS.

Table: Comparison for local verification of t message-signer pairs

	$\text{Exp}_{\mathbb{G}_2}$	$\text{Mul}_{\mathbb{G}_T}$	Pairings
LVMS	t	t	$2t$
LVAMS	t	t	$t + 1$

Outline

- 1 Introduction
- 2 Preliminaries
- 3 LVAMS: Locally Verifiable Aggregated Multi-Signatures
- 4 Comparison
- 5 Conclusion

- 1 **LVAS**: Locally verifiable aggregate signatures
- 2 **LVMS**: Locally verifiable multi-signatures
- 3 **AMSP**: Aggregated multi-signature protocol

LVAMS has the following properties:

- has two-layer aggregation
- facilitates dynamic signer sets across different multi-signatures
- allows constant-cost local verification of any message–signer pair



R. Goyal and V. Vaikuntanathan, “Locally verifiable signature and key aggregation,” in *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*. Springer, 2022, pp. 761–791.



A. R. Ağırtaş, N. Yaman Gökce, and O. Yayla, “Locally verifiable signature schemes: A study of aggregate and multi-signatures,” in *International Conference on Information Technology and Communications Security*. Springer, 2024, pp. 36–56.



D. Boneh, M. Drijvers, and G. Neven, “Compact multi-signatures for smaller blockchains,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2018, pp. 435–464.