



Multi And Threshold Signatures for Starknet

(Warming up for Lisbon Hackathon)

Renaud Dubois

Ledger
Innovation Team

October 19, 2022

Summary



(Classical) Signatures



Multi-Signatures



Threshold Signatures

Summary

- 1 Signatures, MultiSigs and ThresholdSigs
 - Basic Concepts
 - Under the hood
- 2 Use cases
 - Multi factor authentication
 - Voting system
- 3 Ledger/Starknet Musig2, call for Lisbon



SCAN ME

Signatures

A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents.



Signatures

Definition ((Classical) Digital Signature)

A signature scheme is a tuple of function:

- *Setup*: returns domain parameters $E(F_p), G, H$
- *KeyGen*($E(F_p), G, H, seed$): returns $(pvk, pubk) = (x, Q)$
- *Sign*($x, message$): returns *Sig*
- *Verify*(*Sig*, Q): returns true/false

Formulaes for elliptic computations. Formulaes

Signatures

Properties

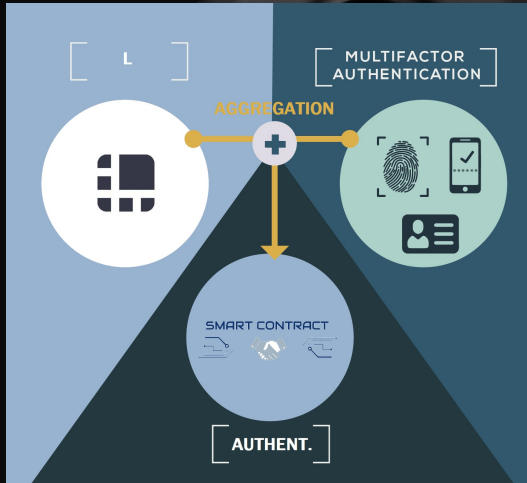
- Unforgeable
- Non repudiation
- *Not reusable*

Most commonly used signature scheme is ECDSA (Bitcoin, Ethereum)

- implemented in Starknet/[Cairo](#) (P256, NTT/Stark friendly Starknet Curve)
- available in your favorite sdk [Ledger](#)

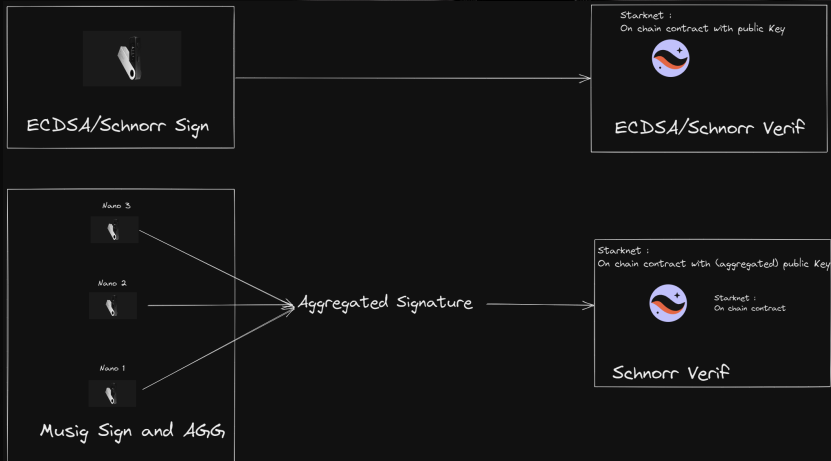
Signatures

How to aggregate several authenticators into one authentication to a smart contract ?



Multi-signatures

A multi-signature is a digital signature allowing users to *aggregate* their keys in an aggregated public key. The signatures are also aggregated. Verifier API is unchanged.



Multi-signatures

Definition (Multi-Signatures)

A multisig scheme is a tuple of function:

- $(Setup, Keygen, Verify, Sign)$
- $KeyAgg(Q_1, \dots, Q_n)$ returns X
- $SignAgg(Sig_1, \dots, Sig_n)$ returns Sig

Multi-signatures

Advantages (over naive concatenation/trusted aggregator)

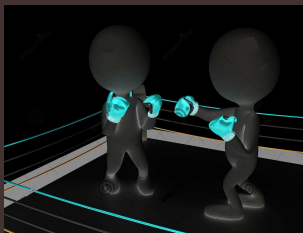
- only one signature over channel (bandwidth consumption)
- no need for a trusted aggregator (no remote private key, own your crypto !)
- no risk of contract failure (don't trust, no don't)
- verifier doesn't need to know the underlying group of users

Example: [Musig2](#), compatible with [BIP340](#)

Multi-signatures

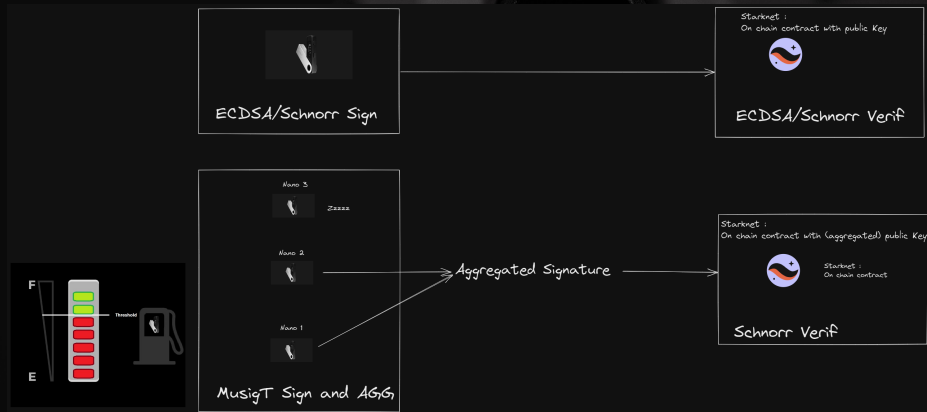
drawback

- requires Schorr (few implementations in crypto Libs, FIDO not in TLS)
- increased computational complexities for signers
- requires (off chain) communications between signers
- computation in 2 rounds (Sig1, Sig2)



Threshold-signatures

A (k, n) threshold signature (TS-Sig) is a digital signature allowing a subset (threshold) of k users from n to *aggregate* a signature .



Threshold-signatures

Definition (Threshold Signatures)

A multisig scheme is a tuple of function:

- $(Setup, Verify, Sign)$
- $DistributedKeygen,$
- $KeyAgg(Q_1, \dots, Q_n)$ returns X
- $SignAgg(Sig_1, \dots, Sig_n)$ returns Sig

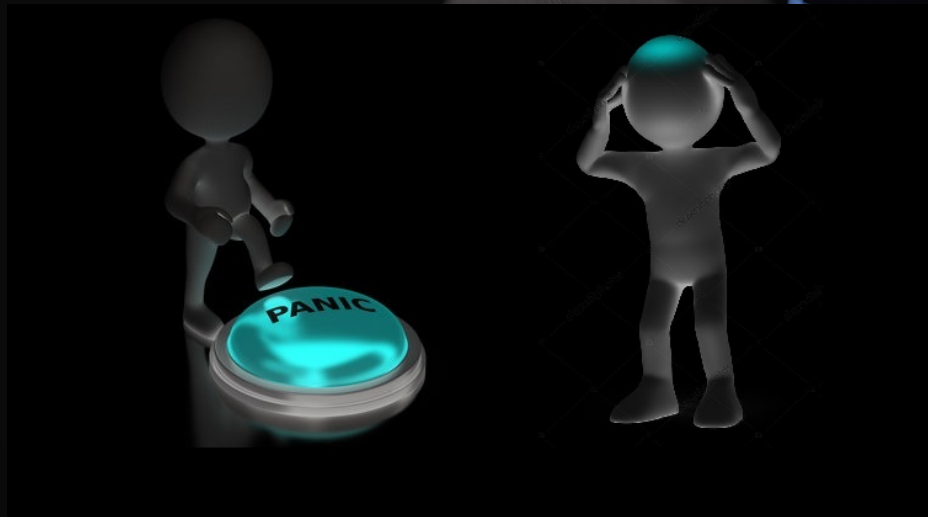
Threshold-signatures

Advantages (over naive concatenation/trusted aggregator)

- All of multisig ($k = n$ is equivalent)
- More flexibility in access policy


Example: FROST.

Disclaimer



EC-Schnorr and ECDSA

SetUP() : Pick a curve with parameters (p, a, b, Gx, Gy, q) (weierstrass equations and formulaes).

Operation		ECDSA
KeyGen	$Q = xG$	$Q = xG$
Nonce*	k	k
Ephemeral	$R = kG$	$R = kG$
Hash	$e = H(m R)$	$e = H(m)$
Sign	$s = k - xe$ $Sig = (R, s)$	$s = k^{-1}(e + xr)$ $Sig = (r, s)$
Verif	$R' = sG + eQ$ Accept if $R'=R$	$r' = (es^{-1}G + rs^{-1}Q)_x$ Accept if $r'=r$

(* nonce generation may use [RFC6979](#) for misuse resistance)

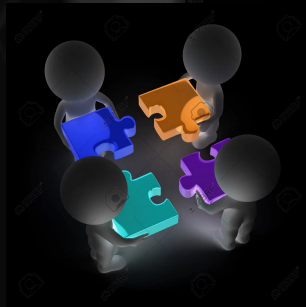
Musig2: using Schnorr additive properties

Schnorr's part is linear in (k, x) and **linearity** is cool:

$$s(k, x_1) + \text{Sig}(k, x_2) = s(k, x_1 + x_2), \quad \forall x = x_1 + x_2$$

$$s(k_1, x) + \text{Sig}(k_2, x) = s(k, x), \quad \forall k = k_1 + k_2$$

(while ECDSA has degree two monomial in (k, x))



Linearity allow homomorphic additions. Idea: split X into $X = \sum a_i X_i$, k into $k = \sum k_i$.

Musig2: using Schnorr additive properties

Operation	Schnorr	Insec_Musig
KeyGen	$X = xG$	$X_i = x_i G$
KeyAgg	-	$X = \sum_{i=0}^{n-1} a_i X_i$
Nonce*	k	k_i
Ephemeral	$R = kG$	$R_i = k_i G$
Aggregate R	-	$R = (\sum_{i=0}^{n-1} a_i \cdot k_i) \cdot G = k \cdot G$
Hash	$e = H(m R)$	$e = H(m R)$
Sign	$s = k - xe$	$s_i = k_i - a_i x_i e$
Aggregate s	-	$s = \sum s_i = k - xe$

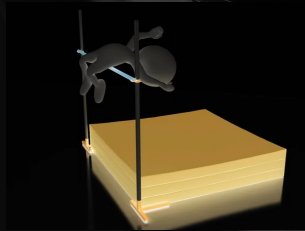
Musig2: using Schnorr additive properties

Musig2 uses a vectorial nonce of length μ , injected in previous Insec_Musig scheme.

Operation	Schnorr	Musig2
KeyGen	$X = xG$	$X_i = x_i G$
KeyAgg	-	$X = \sum_{i=0}^{n-1} a_i X_i$
Nonce*	k	$\vec{k}_i = (k_{i1}, \dots, k_{i\mu})$
Ephemeral	$R = kG$	$\vec{R}_i = \vec{k}_i G$
Hash Nonce	-	$b = H(X R_0 \dots R_\mu m)$
Aggregate R	-	$R = \sum_{j=1}^{\mu} b^{j-1} (\sum_{i=0}^{n-1} a_i \cdot k_i) \cdot G = k \cdot G$
Hash	$e = H(m R)$	$e = H(m R)$
Sign	$s = k - xe$	$s_i = (\sum_{j=1}^{\mu} k_{ij} b^{j-1}) - a_i x_i e$
Aggregate s	-	$s = \sum s_i = k - xe$

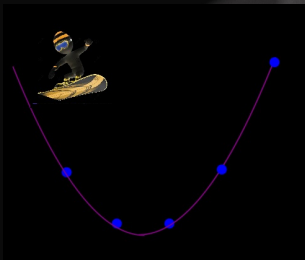
Musig2: Thresholdisation Principle

Thresholdisation use the principle of [Shamir's secret sharing scheme](#) , which is in fact a reed solomon erasure code.
Goal: Given enough shares, it is possible to reconstruct the initial value.



Musig2: Thresholdisation Principle

Lagrange interpolation enables to switch from points to polynomial coefficients using the following formulae:



$$l_j(x) = \prod_{m \neq j} \frac{x - x_m}{x_j - x_m}.$$

$$L(x) = \sum_{j=0}^k P(x_j) l_j(x).$$

The transformation L from $(P_0 \dots P_k)$ to $(a_0 \dots a_k)$ is a **linear** transformation in x .

Sidenote: This is closely related to the principle of FRI used in starks.

Musig2: Thresholdisation Principle

Key ideas:

- interpret aggregated secret key as a polynomial P of degree k ,
- each share (user secret key) is a point of the polynomial,
- blind the computation in the curve domain to perform the aggregation only handling public elements,
- replace ' $\sum_{i=0}^n$ ' in previous scheme by Lagrange polynomials,
- some more steps are necessary (commitments) to avoid cheating.

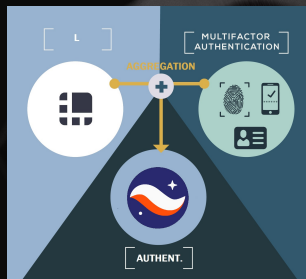
Read [FROST](#) for full description.

Use cases



Multi factor authentication to Starknet Contract

Implement enhanced policy access to assets.



Access Policy

- Low amount: Host (hot wallet) only
- High amount: Host (smartphone) + HW wallet (Nano)

Use WebAuthn (FIDO) into Starknet ?

Voting system

Reduce risk and complexity of a contract implementing a voting system. A vote is adopted only though a valid TS-Sig.



Gnosis advanced

- implement a threshold voting system, with $k = \frac{n}{2}$

Ledger/Starknet Musig2

Specificities of Musig2 Starknet (ease Prover/Verifier computations)

- Addition of EC-Schnorr to Cairo contracts
- Uses Pedersen Hash as core hash function
- Uses Starknet curve as elliptic domain
- Implement x-only verification

The implementation is a generic one (takes hash and elliptic domain as SetUp parameter). The aim is to overlap

- with B340 if selecting P256k1 and SHA256 instead (TBD).
- with RFC Eddsa if selecting Ed25519.

Ledger/Starknet Musig2

Current state

- Schnorr verification available in Cairo,
- High-level simulation in Sagemath of full protocol (Sign/Verify for a pool of users)
- Musig2 implementation on top of a virtualization layer (only integrating bolos for now)
- currently working on thresholdization

C Library (Nano Signer)



SCAN ME

Cairo Code (Contract Verifier)



SCAN ME

Ledger/Starknet Musig2

Next steps

- Thresholdisation,
- Compatibility with Ed25519 (FIDO compliance)



C Library (Nano Signer)



SCAN ME

Cairo Code (Contract Verifier)



SCAN ME

Starknet Hackathon



Join the team for:

- Front end integration (wallet, current development over Argent) over verifier (Cairo) or signer (C)
- Integration of a different accelerator/library in the virtualization layer (C)
- Contribute to the threshold version (Sagemath/C)



Questions ?



C Library



SCAN ME

R. Dubois (LIT)

Slides



SCAN ME

Multi And Threshold Signatures for Starknet

Cairo&Sage



SCAN ME

October 19, 2022