Symphony: Threshold Evaluation of Symmetric Primitives

A protocol family for threshold AES, SHA2, SHA3 and G-/C-/H-/KMAC evaluation in the three-party, honest majority setting

Erik Pohle

erik.pohle@cs.au.dk

January 28th, 2026

Aarhus University, Denmark

MPTS 2026
NIST Workshop on Multi-Party Threshold Schemes

joint work with
Hiraku Morita (University of Southern Denmark),
Peter Scholl (Aarhus University),
Daniel Tschudi (Concordium, Eastern Switzerland University of Applied Sciences)

AARHUS UNIVERSITY

Overview

Planned Submission

Threshold evaluation of

- AES *(secret-shared key)*
- GMAC and CMAC *(secret-shared key)*
- SHA2 and SHA3 *(secret-shared input)*
- HMAC and KMAC *(secret-shared key)*

Setting

- 3 parties, 1 tolerated corruption
- active security with abort

Techniques

- replicated secret sharing [FLNW17]
- generation of random one-hot vector correlations
- oblivious table lookup
- distributed inner-product checks [BBC+19,MPS+25]

Applications

- threshold symmetric-key (authenticated) encryption
- distributed authentication protocols, private set intersection, secure database joins
- virtual TPM

Outline

- Background
  - Replicated Secret Sharing
  - Oblivious Table Lookup
  - Achieving Active Security
- Module Overview
- Detail Modules
- Experimental Results

Background

Replicated Secret Sharing (1)

secret $x \in \mathrm{GF}(2^k)$

Secret Sharing

- additive sharing, $\langle x \rangle_i = x_i$ where $x_1 + x_2 + x_3 = x$
- replicated sharing $[\![x]\!]_1 = (x_1, x_2)$, $[\![x]\!]_2 = (x_2, x_3)$ and $[\![x]\!]_3 = (x_3, x_1)$.

Linear Computation

- Linear operation L: $[\![L(x)]\!]_i = (L(\langle x \rangle_i), L(\langle x \rangle_{i+1}))$
- Squaring: $[\![x^2]\!] = (\langle x \rangle_i^2, \langle x \rangle_{i+1}^2)$
- Bit decomposition: $[\![x]\!] = ([\![x_0]\!], \ldots, [\![x_{k-1}]\!])$, $x_j \in \{0, 1\}$

$\rightarrow$ no communication

Replicated Secret Sharing (2)

Multiplication $[\![x]\!] \cdot [\![y]\!]$

Given $\langle 0 \rangle$ from preprocessing *(can be derived from correlated PRF keys)*

1. Local multiplication $\langle xy \rangle_i = x_i y_i + (x_i + x_{i+1})(y_i + y_{i+1})$
   $\rightarrow$ obtain additive sharing of xy
2. Send $\langle xy \rangle_i + \langle 0 \rangle_i$ to P_{i-1}, set $[\![xy]\!]_i = (\langle xy \rangle_i + \langle 0 \rangle_i, \langle xy \rangle_{i+1} + \langle 0 \rangle_{i+1})$

Table Lookup LUT $[\![T[x]]\!]$

Table T,given *random one-hot vector* correlation from preprocessing $[\![r]\!], [\![\boldsymbol{e}^{(r)}]\!]$

$$[\![\boldsymbol{e}^{(r)}]\!] = (\underbrace{[\![0]\!], \ldots, [\![0]\!], [\![1]\!]}_{r}, [\![0]\!], \ldots)$$

1. Reveal value $c = r + x$
2. Local computation $[\![T[x]]\!] = \sum_i T[c - i] \cdot [\![e_i^{(r)}]\!]$

Active Security

Linear Operations

- each share of the corrupted party is held by an honest party
- consistency is checked for all revealed values
- before accepting result: parties exchange views (pairwise)
- compress transcript using hash function

Multiplications

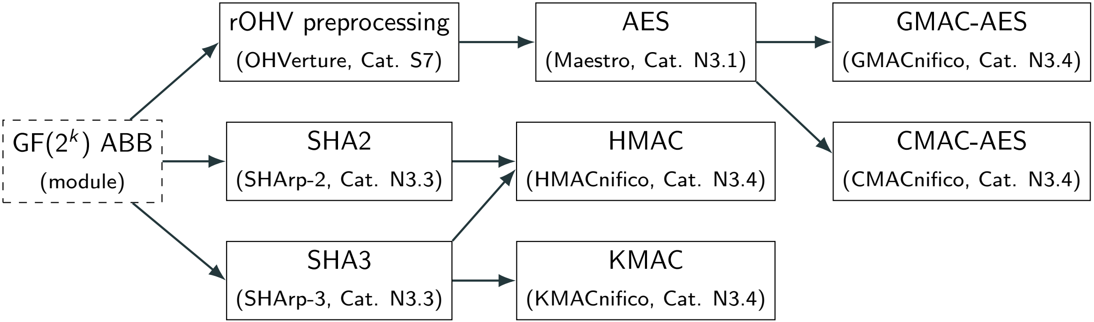
- recursive inner-product check (adapted to our setting from [BBC+19,BGIN20])
- for L inner-product triples $\sum_{i=1}^{m} [\![x_i]\!] \cdot [\![y_i]\!] = [\![z]\!]$; let $N = L \cdot m$

Cost

- $\mathcal{O}(\log N) \cdot (\log |\mathbb{F}|)$ bits of communication (we use $\mathbb{F} = \mathrm{GF}(2^{64})$)
- in $\log N$ rounds of communication

Symphony – Overview

Symphony Modules



GF(2^k) **Arithmetic Black Box (ABB)**

- Input(x, P_i) $\rightarrow [\![x]\!]$: party P_i inputs $x \in$ GF(2^k) into the ABB

- Linop($f, [\![x_1]\!], \dots$) $\rightarrow [\![f(x_1, \dots)]\!]$, for any GF(2)-linear function f
- LocalMul($[\![x]\!], [\![y]\!]$) $\rightarrow \langle x \cdot y \rangle$

- Reshare($\langle x \rangle$) $\rightarrow [\![x]\!]$
- BitMul($[\![x]\!], [\![y]\!]$) $\rightarrow [\![x \cdot y]\!]$ where $x \in$ GF(2) and $y \in$ GF(2^k).
- Reconst($[\![x]\!]$) $\rightarrow x$: unverified opening of the secret-shared $[\![x]\!]$.

- VerifyRecon(): verifies all opened values (via Reconst) so far.
- VerifyMul(): verifies all multiplications (via LocalMul and BitMul) so far.
- Output($[\![x]\!]$) $\rightarrow x/\perp$: verified output

(for $k \in \{1, 4, 8, 64, 128\}$)

OHVerture: Preprocessing for random one-hot vectors

Output

$([\![r]\!], [\![\mathbf{e}^{(r)}]\!])$ or $([\![r]\!], \langle \mathbf{e}^{(r)} \rangle)$
where $\mathbf{e}^{(r)} = (\underbrace{0, \ldots, 0}_{r}, 1, 0, \ldots)$ and r random

Usefulness as Gadget

enable richer MPC gadgets such as table lookup protocols

- secure sampling for common noise distributions [MRS25,FFG+25]
- fast online phase when compiling Boolean circuits into LUT gates [BHS+23]
- ...

- $\mathsf{rOHVrss}(N = 2^k, L) \to ([\![r_j]\!], [\![\mathbf{e}^{(r_j)}]\!])_{j=1}^{L}$ outputs shares of L correlations, where $r_j \leftarrow \$ \mathrm{GF}(2^k)$
- $\mathsf{rOHVadd}(N = 2^k, L) \to ([\![r_j]\!], \langle \mathbf{e}^{(r_j)} \rangle)_{j=1}^{L}$ outputs shares of L correlations, where $r_j \leftarrow \$ \mathrm{GF}(2^k)$

Maestro: AES enciphering & deciphering

Output

distributed key (schedule) generation, enciphering, deciphering for AES-128 and AES-256

- $\mathrm{DistKeyGen}() \to [\![\mathrm{ks}]\!]$: outputs the key schedule ks of a fresh random AES-128/-256 key as shares to the parties
- $\mathrm{InputKey}(k, P_i) \to [\![k]\!]$: inputs the key by party P_i
- $\mathrm{ComputeKS}([\![k]\!]) \to [\![\mathrm{ks}]\!]$: computes the key schedule
- $\mathrm{InputBlock}(m, P_i) \to [\![m]\!]$: inputs a 128-bit block by party P_i
- $\mathrm{OutputBlock}([\![m]\!]) \to m$

- $\mathrm{Encipher}([\![\mathrm{ks}]\!], [\![m]\!]) \to [\![\mathrm{AES}(\mathrm{ks}, m)]\!]$: computes AES enciphering

  with performance mode
  - “online phase throughput”
  - “total throughput”
  - “low latency”

Output

hashing of a secret-shared message for SHA-256, SHA-512, SHA3-256 and SHA3-512

- InputMessage$(m, P_i) \rightarrow [\![m]\!]$: inputs a message by party P_i.
- Hash$([\![m]\!]) \rightarrow [\![\mathcal{H}(m)]\!]$: computes the threshold hash.
- OutputDigest$([\![d]\!]) \rightarrow d$, where $[\![d]\!]$ is obtained from any of the hash computations above.

MACnifico: Threshold GMAC-AES, CMAC-AES, HMAC and KMAC

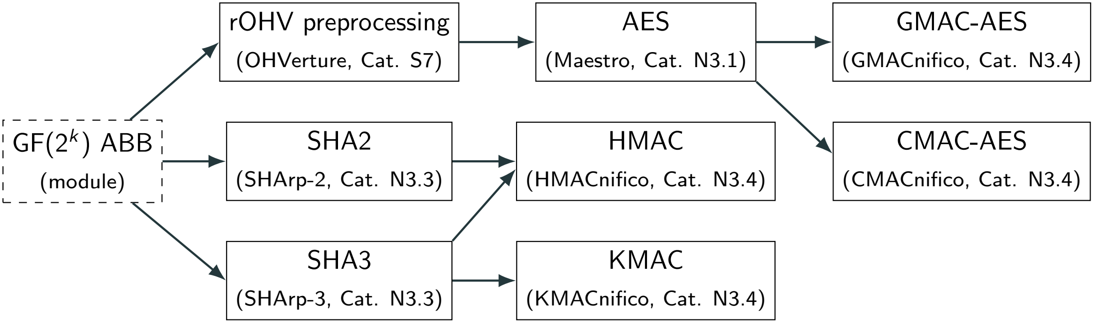
Output

MAC tag computation using secret-shared key:

- GMAC and CMAC (from AES-128 and AES-256)
- HMAC (from SHA-256, SHA-512, SHA3-256 and SHA3-512)
- KMAC-128 and KMAC-256

- InputKey$(k, P_i) \to [\![k]\!]$: inputs the MAC key by party P_i
- InputMessage$(m, P_i) \to [\![m]\!]$: inputs the message by party P_i
- TagGen$([\![k]\!], [\![m]\!]) \to [\![\tau]\!]$: computes the threshold MAC tag generation on the given key and message
- OutputTag$([\![\tau]\!]) \to \tau$, where $[\![\tau]\!]$ is obtained from any thresholdized TagGen computation.

Symphony Modules



Selected Technical Details

rOHV preprocessing (1)

Length 16 using $GF(2^4)$

1. generate random $[\![\cdot]\!]$-shared bits r_0, r_1, r_2, r_3
2. compute partial products $r_i r_j$, $r_i r_j r_k$ and $r_0 r_1 r_2 r_3$
3. each element in $\boldsymbol{e}^{(r)}$ is a linear combination of partial products

Mult. check requires checking 2 $GF(2^{64})$ multiplications

Cost

- 11 GF(2) multiplications $\rightarrow$ 11 bits of communication
- in 2 communication rounds

rOHV preprocessing (2)

Length 256 using GF(2^8)

1. obtain 2 length-16 random one-hot vectors: $[\![u]\!], [\![\boldsymbol{e}^{(u)}]\!]$ and $[\![v]\!], [\![\boldsymbol{e}^{(v)}]\!]$
2. *local* tensor product $\langle \boldsymbol{e}^{(r)} \rangle \leftarrow [\![\boldsymbol{e}^{(u)}]\!] \otimes [\![\boldsymbol{e}^{(v)}]\!]$ where $r = u \| v$
3. output $([\![r]\!], \langle \boldsymbol{e}^{(r)} \rangle)$

Mult. check requires checking 4 GF(2^{64}) multiplications **+ AES S-box check for LUT**

Cost

- 22 GF(2) multiplications $\rightarrow$ 22 bits of communication
- in 2 communication rounds

Advanced Encryption Standard (AES)

Block cipher composed of 10 rounds (AES-128)
Let $\mathrm{GF}(2^8) = \mathbb{F}_2[X]/X^8 + X^4 + X^3 + X + 1$.

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

only non-linear computation

1. SubBytes: $\text{S-box}(s_i) = f(s_i^{-1})$
2. ShiftRows: $$\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \to \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{pmatrix}$$
3. MixColumns: $$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot S$$
4. AddRoundKey: $S + RK$

MAESTRO (1)

compute inversion $x^{-1} \in GF(2^8)$

Performance Mode: High Online Phase Throughput

1. use isomorphism between $GF(2^8)$ and $GF(2^4)^2$ (linear operation)
2. compute $v^{-1} \in GF(2^4)$ as table lookup (consume length 16 OHV from OHVerture)

Mult. check requires checking 1 $GF(2^{64})$ multiplication

Cost

- 3 $GF(2^4)$ mult. + 1 $GF(2^4)$ inversion $\rightarrow$ 16 bits of communication
- in 2 communication rounds

MAESTRO (2)

compute inversion $x^{-1} \in GF(2^8)$

Performance Mode: High Total Throughput

1. use isomorphism between $GF(2^8)$ and $GF(2^4)^2$ (linear operation)
2. compute $v^{-1} \in GF(2^4)$ as $v^2 \cdot v^4 \cdot v^8$

Mult. check requires checking 3 $GF(2^{64})$ multiplications

Cost

- 5 $GF(2^4)$ mult. $\rightarrow$ 20 bits of communication
- in 3 communication rounds

MAESTRO (3)

compute inversion $x^{-1} \in \mathrm{GF}(2^8)$

Performance Mode: Low Latency

1. compute $x^{-1} \in \mathrm{GF}(2^8)$ as table lookup (consuming length-256 OHV from OHVerture)

Mult. check requires checking 2 $\mathrm{GF}(2^{64})$ multiplications

Cost

- 16 bits of communication
- in 1 communication round

SHArp

SHA-2

compute as Boolean circuits

Cost (compression function of SHA-256)

- $\approx$ 2.8KB of communication
- in $\approx$ 1600 communication rounds

SHA-3

compute as Boolean circuits

Cost (24-round permutation Keccak-f)

- $\approx$ 4.8KB of communication
- in $\approx$ 24 communication rounds

MACnifico

GMAC

- implement $GF(2^{128})$ for GMAC
- mult. check requires 2 $GF(2^{64})$ inner-product checks

$\rightarrow$ remaining constructions make use of previously introduced modules

Preliminary Experimental Results

Benchmark from [MPS+25] (1)

Batched enciphering of 100000 AES blocks ($\approx$ 1.6 MB)

	Preprocessing		Online Phase		Throughput (blocks/s)	
Protocol	Time (s)	Data (MB)	Time (s)	Data (MB)	Online	Total
online tp.	0.23	22	2.24	$\approx$ 32	44624	40533
total tp.	–	–	2.34	$\approx$ 40	42799	42799
low latency	0.84	44	3.65	$\approx$ 32	27373	22280

run on 3 separate machines (16-core Intel Core i9-9900 3.10GHz, 128GB RAM) over a 9.47 Gbit/s LAN network

Benchmark from [MPS+25] (1)

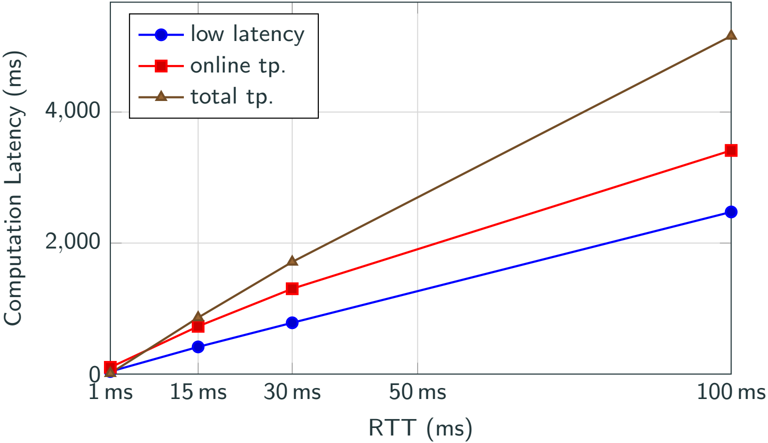
Batched enciphering of 100000 AES blocks ($\approx$ 1.6 MB)

	Preprocessing		Online Phase		Throughput (blocks/s)	
Protocol	Time (s)	Data (MB)	Time (s)	Data (MB)	Online	Total
online tp.	0.23	22	2.24	$\approx$ 32	$\approx$ 714 KB/s	$\approx$ 649 KB/s
total tp.	–	–	2.34	$\approx$ 40	$\approx$ 685 KB/s	$\approx$ 685 KB/s
low latency	0.84	44	3.65	$\approx$ 32	$\approx$ 438 KB/s	$\approx$ 356 KB/s

run on 3 separate machines (16-core Intel Core i9-9900 3.10GHz, 128GB RAM) over a 9.47 Gbit/s LAN network

Benchmark from [MPS+25] (2)

Computation latency of a single block enciphering



(same setup as previous slide; network delay and bandwidth altered with `tc`)

Summary

Planned Submission

Threshold evaluation of

- AES *(secret-shared key)*
- GMAC and CMAC *(secret-shared key)*
- SHA2 and SHA3 *(secret-shared input)*
- HMAC and KMAC *(secret-shared key)*

Setting

- 3 parties, 1 tolerated corruption
- active security with abort

Techniques

- replicated secret sharing [FLNW17]
- generation of random one-hot vector correlations
- oblivious table lookup
- distributed inner-product checks [BBC+19,MPS+25]

Applications

- threshold symmetric-key (authenticated) encryption
- distributed authentication protocols, private set intersection, secure database joins
- virtual TPM

References (1)

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO 2019*. `https://ia.cr/2019/188`

[BGIN20] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero-knowledge proofs. In *ASIACRYPT 2020*. `https://ia.cr/2020/1451`

[BHS+23] Andreas Brüggemann, Robin Hundt, Thomas Schneider, Ajith Suresh, and Hossein Yalame. FLUTE: Fast and Secure Lookup Table Evaluations. In *IEEE Security & Privacy 2023*. `https://ia.cr/2023/499`

[FFG+25] Olive Franzese, Congyu Fang, Radhika Garg, Somesh Jha, Nicolas Papernot, Xiao Wang, and Adam Dziedzic. Secure Noise Sampling for Differentially Private Collaborative Learning. In *ACM CCS 2025*. `https://ia.cr/2025/1025`

Some images from Flaticon.com

References (2)

[FLNW17] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *EUROCRYPT 2017*. https://ia.cr/2016/944

[MRS25] Fredrik Meisingseth, Christian Rechberger, and Fabian Schmid. Accelerating Multiparty Noise Generation Using Lookups. IACR ePrint 2025/805. https://ia.cr/2025/805

[MPS+25] Hiraku Morita, Erik Pohle, Kunihiko Sadakane, Peter Scholl, Kazunari Tozawa, and Daniel Tschudi. MAESTRO: Multi-party AES using Lookup Tables. In *Usenix Security 2025*. https://ia.cr/2024/1317

Some images from Flaticon.com