Faster Homomorphic Evaluation of SHA3

Tim Seuré, Elias Suvanto
University of Luxembourg
elias.suvanto@uni.lu

Full Keccak Permutation Evaluated in 68 s under FHE

Leveraging the free-XOR framework [NHY+25] and a SIMD-optimized data layout, we reduce state-of-the-art latency of SHA3 under FHE from 327 s (using a TFHE-based approach) to 68 s (using discrete CKKS).

Motivation & Applications

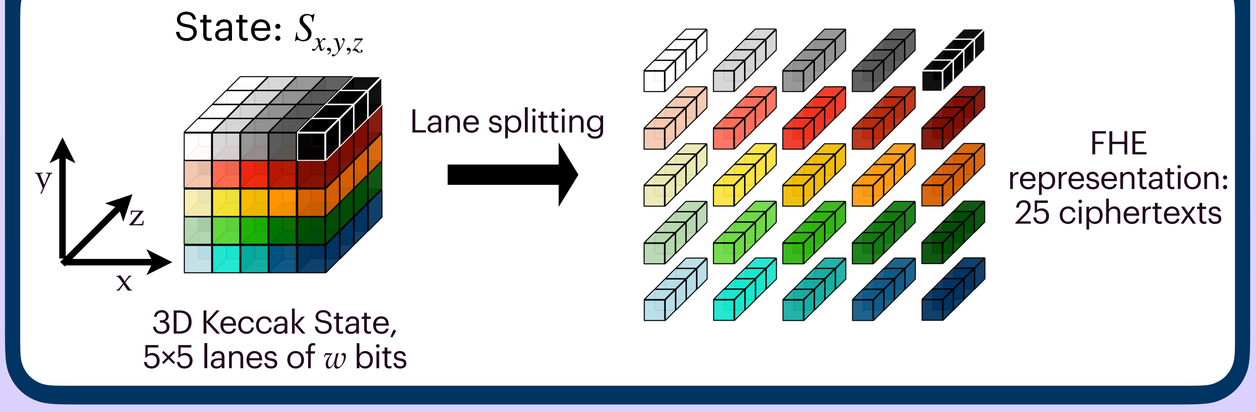
Hash functions are required in digital signature algorithms (DSA) like ML-DSA. Evaluating DSA under Fully Homomorphic Encryption (FHE) unlocks extremely communication-efficient cryptographic protocols:

- **One-Round Threshold Signatures** - decentralized consensus
- **One-Round Blind Signatures** - secure e-voting systems

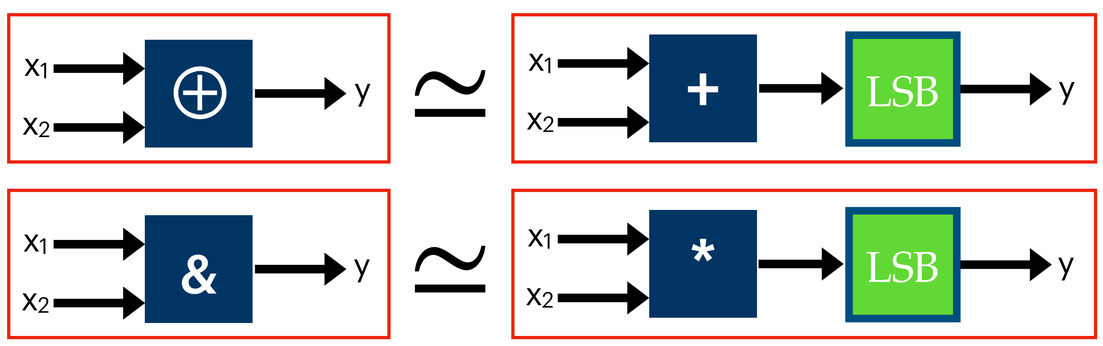
We focus on the NIST-standardized hash function SHA3.

Challenge: The Keccak permutation of SHA3 is a deep and dense Boolean circuit (24 rounds, more than 100,000 Boolean gates).

Lane Data Layout for 3D Keccak State



Arithmetization of Keccak Permutation



LSB — The Least Significant Bit (LSB) extraction gate can be applied lazily at the end of the circuit.

Below is the arithmetized Keccak round at index r, where each variable is a lane of size w and the operations are SIMD arithmetic.

Step	Arithmetized Keccak Round	
	$C_x := \sum_{y=0}^{4} S_{x,y}$	Addition (free)
θ	$D_x := C_{x-1} + \mathrm{Rot}(C_{x+1}, 1)$	Rotation (1 KS)
	$S_{x,y} := S_{x,y} + D_x$	
ρ, π	$B_{y,(2x+3y) \bmod 5} := \mathrm{Rot}(S_{x,y}, R_{x,y})$	
χ	$S_{x,y} := B_{x,y} + (1 - B_{x+1,y}) \times B_{x+2,y}$	Multiplication (1 KS + 1 level)
ι	$S_{0,0} := S_{0,0} + \mathrm{RC}_r$	

KS = key switching

CKKS multiplication significantly increases the integer size and error, requiring a refresh procedure. We chose to refresh once per round though parameters can be designed to refresh every 2 rounds.

XBOOT

XBOOT homomorphically removes redundant bits and refreshes the error.

XBOOT parameters

log N	log PQ	(H, h)	dnum
15	833	(256, 32)	4

log q_i					log P
CtS	EvalMod	Mult	StC	Base	
3 × 29	8 × 42	2 × 41	3 × 27	42	5 × 41

Packing for Bootstrapping

Optimizing latency: one dense bootstrap for 5 × 5 sparse lanes.

	0	1	…	24	…	32	33	…	56	…	32 · (w-1)	32·w - 31	…	32·w - 8	up to 32·w
$S_{0,0}$	X_0		…		0	X_1				…	X_{w-1}				0
$S_{0,1}$		Y_0	…				Y_1					Y_{w-1}			
…			…					…					…		
$S_{4,4}$				Z_0					Z_1					Z_{w-1}	
ct_{BTS}	X_0	Y_0	…	Z_0	0	X_1	Y_1	…	Z_1	…	X_{w-1}	Y_{w-1}	…	Z_{w-1}	0

When $32 \cdot w \cdot k = N/2$ with $k > 1$, each colored slot can be seen as a slice of k slots containing bits from k different Keccak states, improving throughput.

Experimental Results

Hardware Setup: MacBook M4 Pro (24 GB RAM), single-threaded execution.

Work	Scheme	Software	Failure Probability	Latency (s)	Time / msg (s)	Evaluation Keys	Ciphertext Memory (MB)
[BPR24]	TFHE	TFHE-rs	2^{-40}	327	327	**67 MB**	**8**
Ours	discrete CKKS	Lattigo	**< 2^{-128}**	**68**	**8.5**	4.8 GB	62

Code Repository & References

github.com/suvan-to/faster-sha3-under-fhe

[BPR24] N. Bon, D. Pointcheval, and M. Rivain, *Optimized Homomorphic Evaluation of Boolean Functions*, TCHES 2024, doi: 10.46586/tches.v2024.i3.302-341
[NHY+25] C. Niu et al., *XBOOT: Free-XOR Gates for CKKS with Applications to Transciphering*, TCHES 2025, 10.46586/tches.v2025.i4.118-144.