RUHR-UNIVERSITÄT BOCHUM

Horst Görtz Institute for IT-Security
Chair of Embedded Security

ON THE SUITABILITY OF SHA-3 FINALISTS FOR LIGHTWEIGHT APPLICATIONS

Elif Bilge Kavun , Tolga Yalcin

Third SHA-3 Candidate Conference

hgi Lehrstuhl für Embedded Security

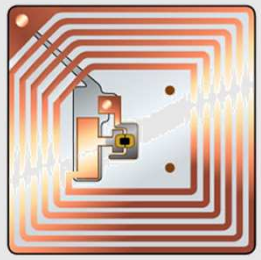
Overview

- SHA-3 Competition and Finalists
- Lightweight Applications
- Motivation
  - SHA-3 Finalists for Lightweight
- BLAKE
- Grøstl
- JH
- Keccak
- Skein
- Interface for Modules
- Results
- Comparison with Previous Works
- Remarks and Comments
- Conclusion and Future Directions

SHA-3 Competition and Finalists

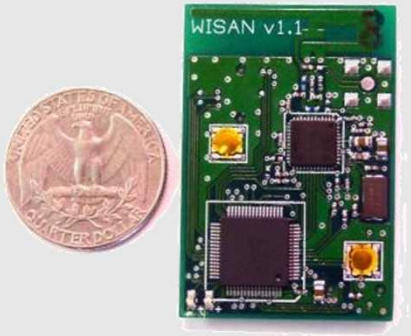
- NIST announced a public competition on November 2, 2007 to develop a new cryptographic hash algorithm
- The winning algorithm will be named "SHA-3"
  - Hash algorithms currently specified in FIPS 180-3, Secure Hash Standard, will be augmented
- Five SHA-3 finalists selected
  - BLAKE, Grøstl, JH, Keccak and Skein

Lightweight Applications



- Lightweight devices and applications:
  - RFID (Radio-Frequency IDentification) tags
    - For identification and tracking purposes using radio waves
  - Smart cards
    - Provide identification, authentication, data storage and application processing
  - Sensor nodes
    - Perform some processing, gathers sensory information and communicates with other connected nodes in a network
  - etc.

- Lightweight device requirements:
  - Low-power consumption
  - Low-energy consumption
  - Compactness



- Speed/throughput not so important
- With the increase in RFID usage, security and identification problems arose
- Several protocols using hash functions for the privacy of consumers

Motivation

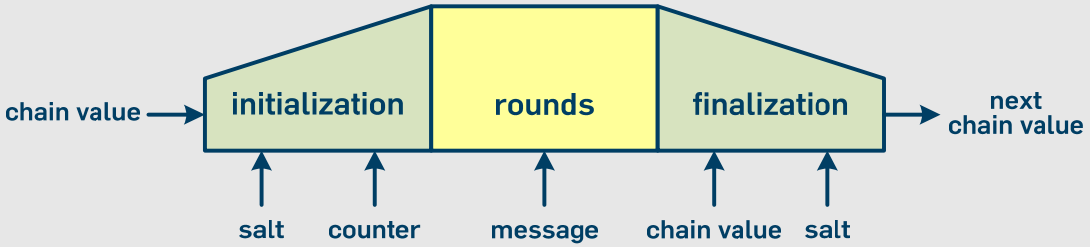
SHA-3 Finalists for Lightweight

- Need for lightweight cryptographic hash functions repeatedly expressed
  - For implementing cryptographic protocols on lightweight devices
- Few algorithms exist for providing satisfactory security and performance
- *Suitability of SHA-3 finalists for lightweight applications should be investigated!*
- The aim of this study: To present efficient lightweight (serial) implementations of SHA-3 finalists (256-bit message digest for all)
  - Serial implementations result in low-throughput, but acceptable for lightweight applications

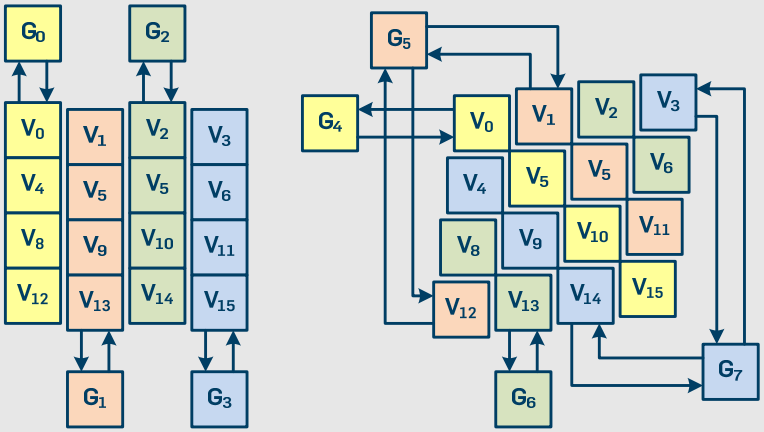
BLAKE

Algorithm

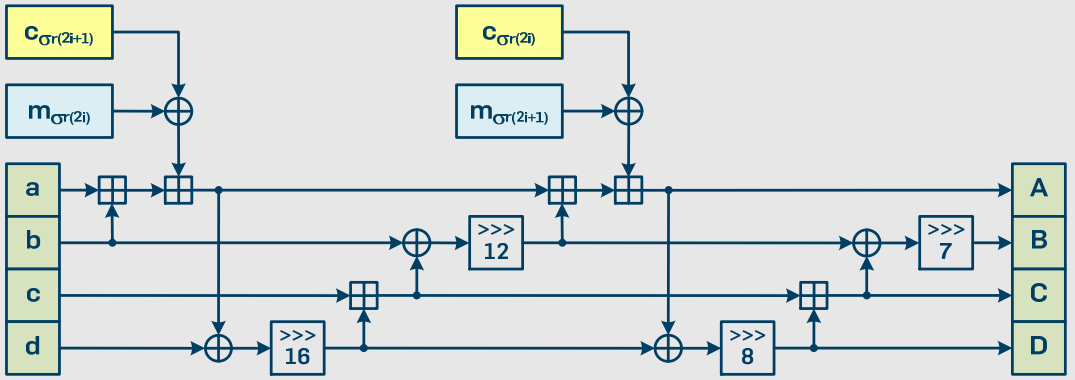
Algorithm	Word	Message	Block	Salt	Rounds	Digest
BLAKE-256	32-bit	$< 2^{64}$ - bit	512-bit	128-bit	14	256-bit



- Round function:



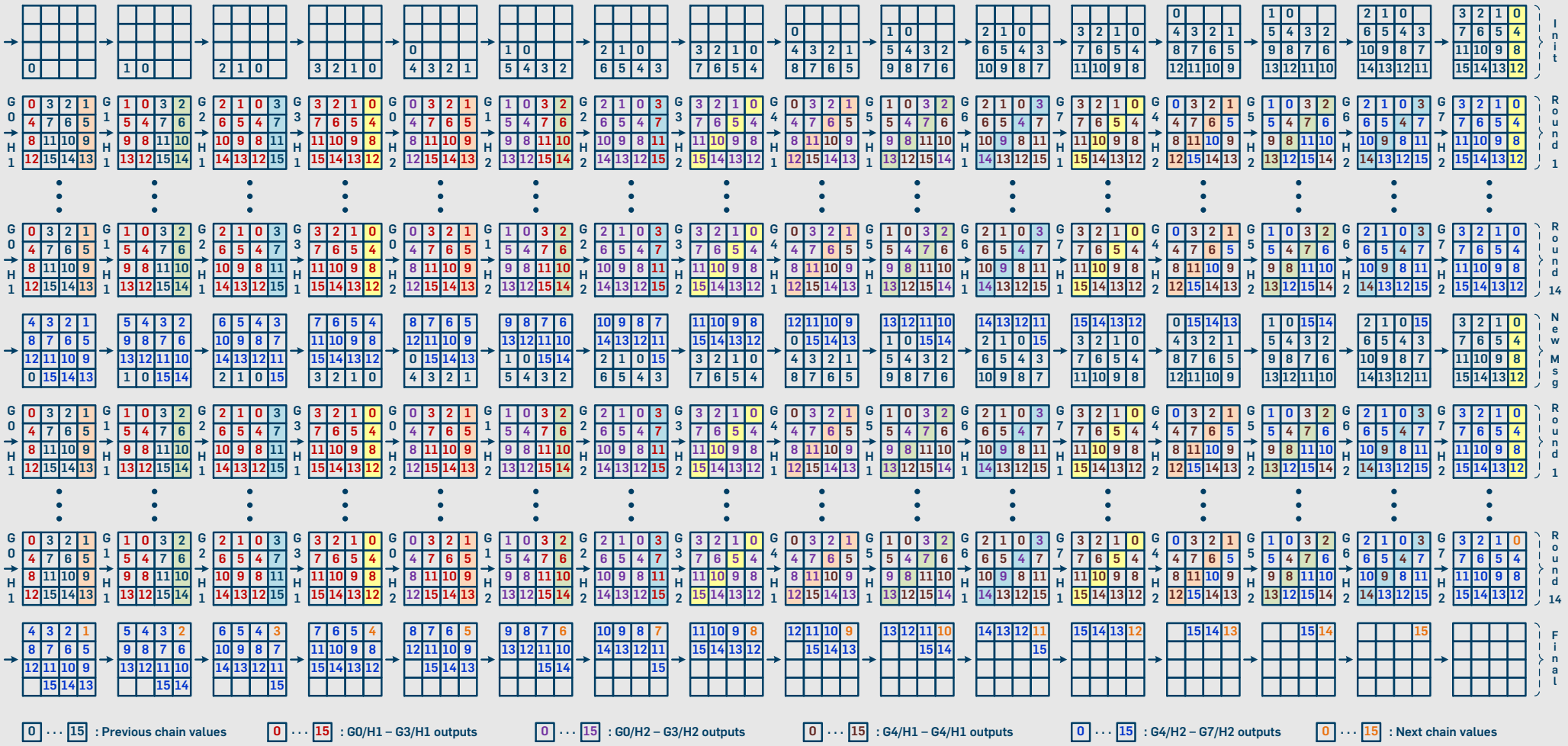
- G_i function:



SHA-3 Finalists

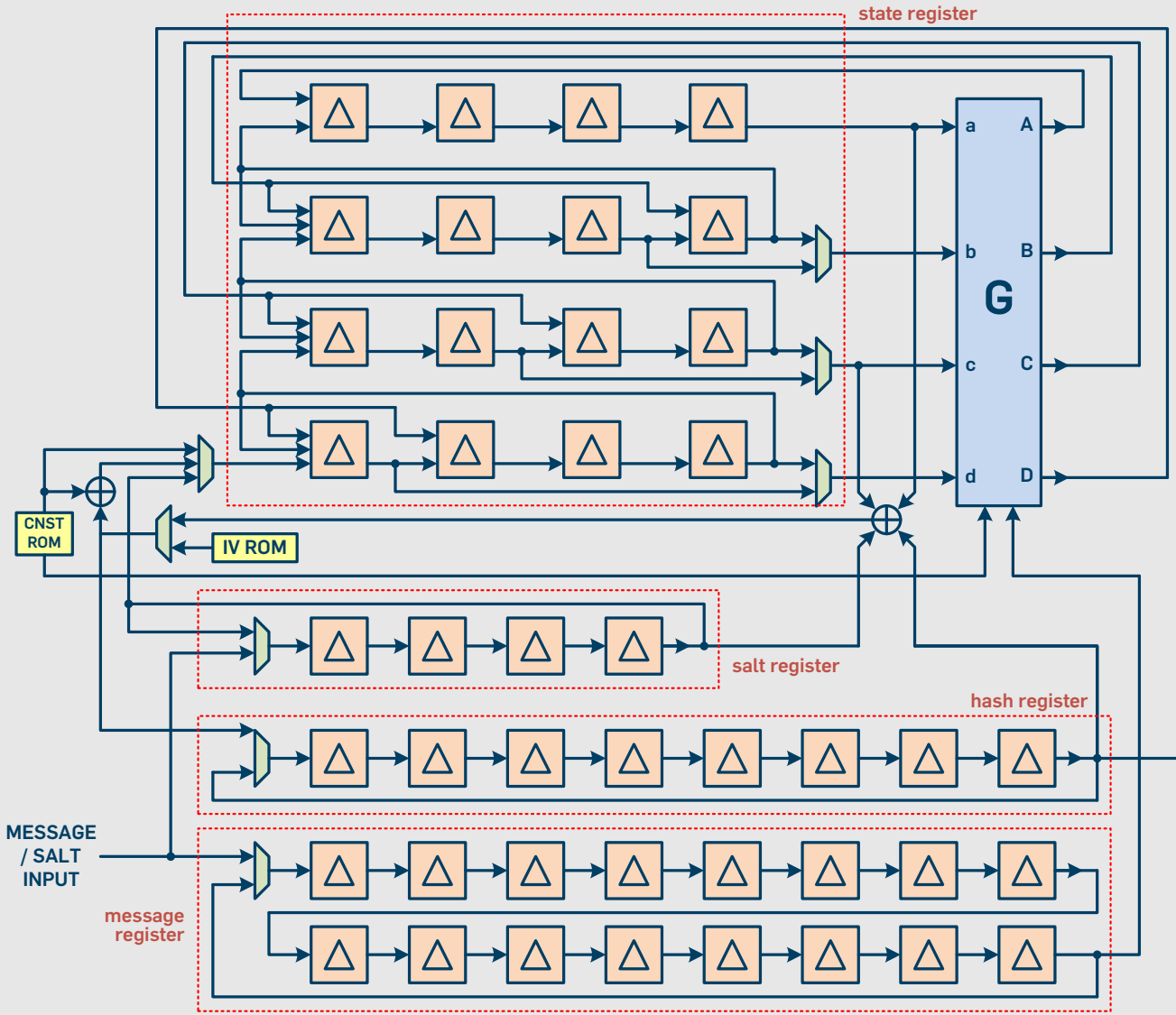
BLAKE

Serial Data Flow

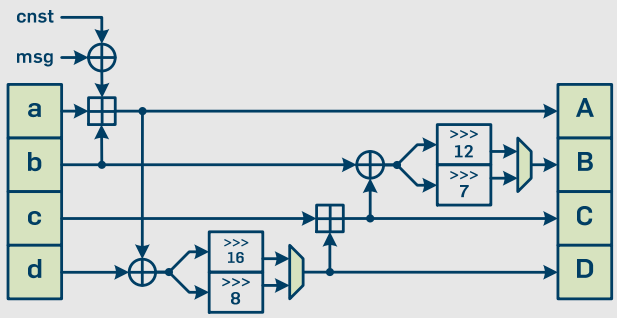


BLAKE

Block Diagram

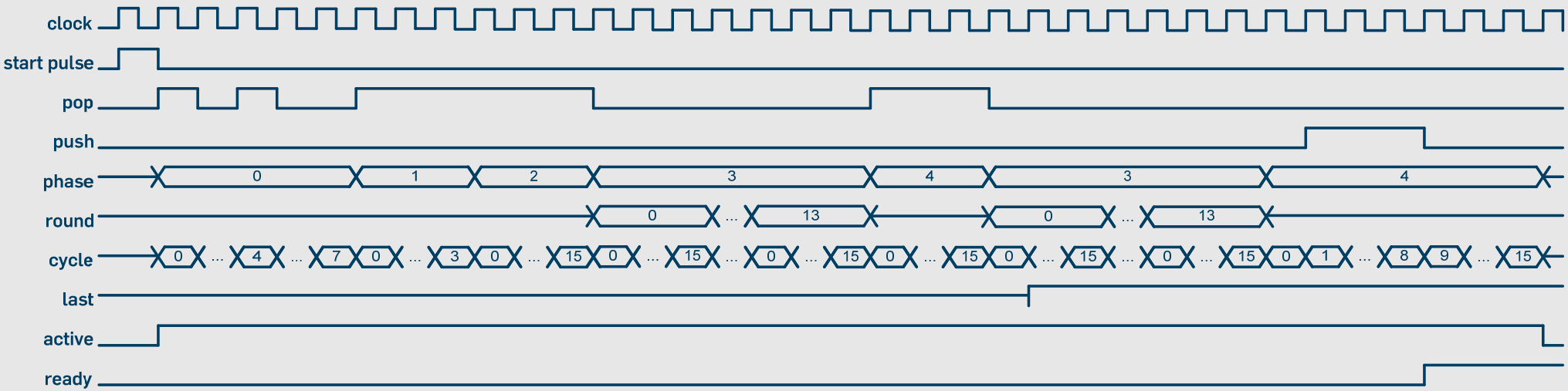


- ***G*** (Half G_i) module:



BLAKE

Timing Flow

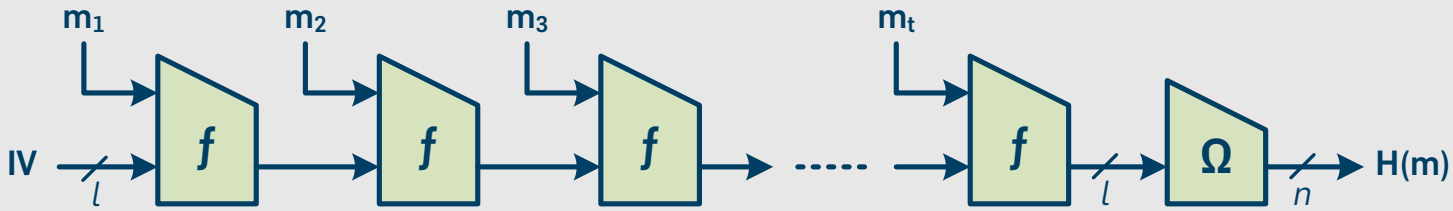


- Phase-0: Read SALT
- Phase-1: Read LENGTH
- Phase-2: Read first MESSAGE block
- Phase-3: Processing
- Phase-4: Read further MESSAGE blocks
- Phase-4: Write MESSAGE DIGEST

Grøstl

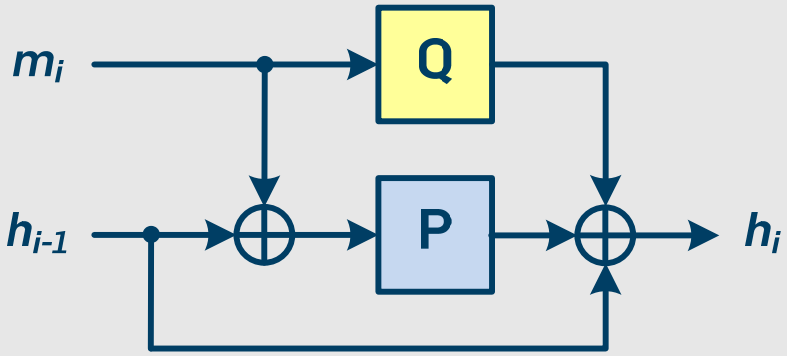
Algorithm (1)

Algorithm	Word	Message	Block	Rounds	Digest
Grøstl-256	32-bit	$< (2^{73} - 577)$ - bit	512-bit	10	256-bit



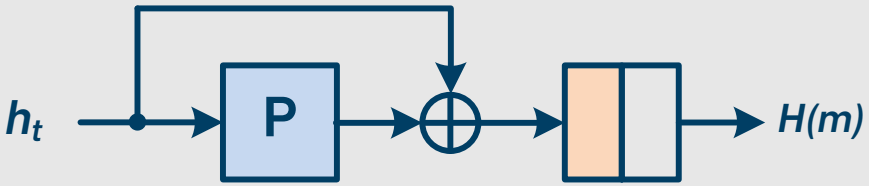
- Compression function:

$$f(h,m) = P(h \oplus m) \oplus Q(m) \oplus h$$



- Output transformation:

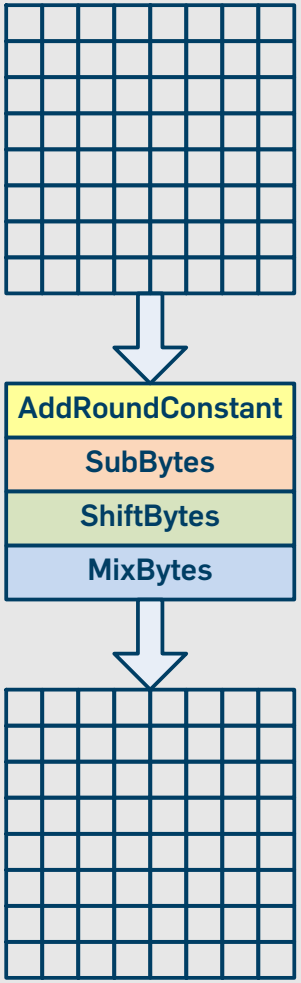
$$\Omega(x) = \text{trunc}_n(P(x) \oplus m)$$



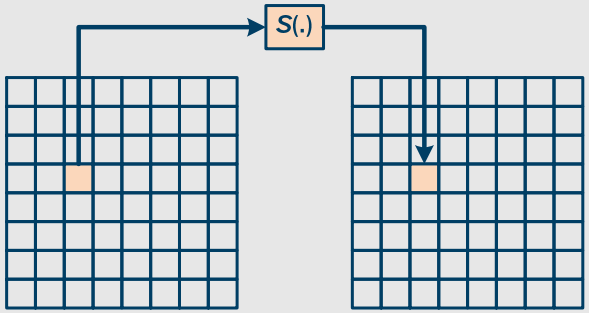
Grøstl

Algorithm (2)

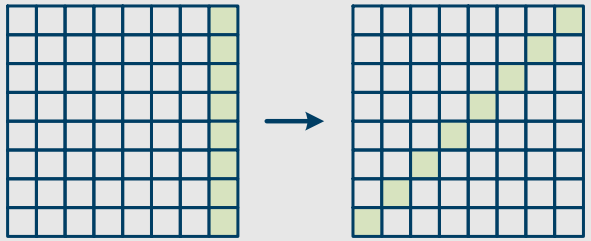
- P/Q Composition



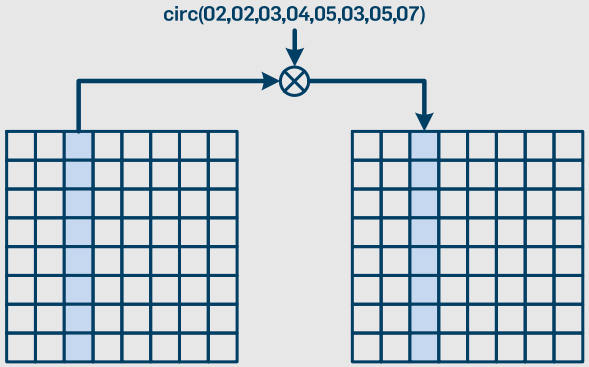
- SubBytes



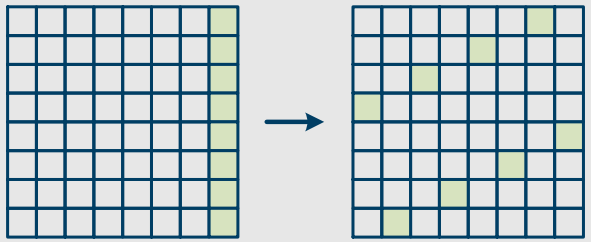
- ShiftBytes for P



- MixBytes



- ShiftBytes for Q

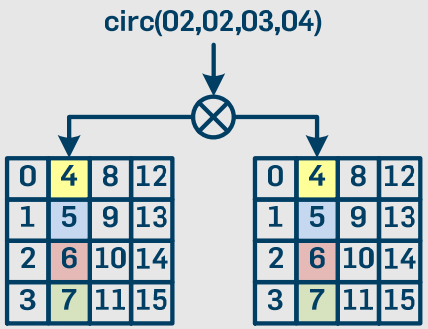


Grøstl

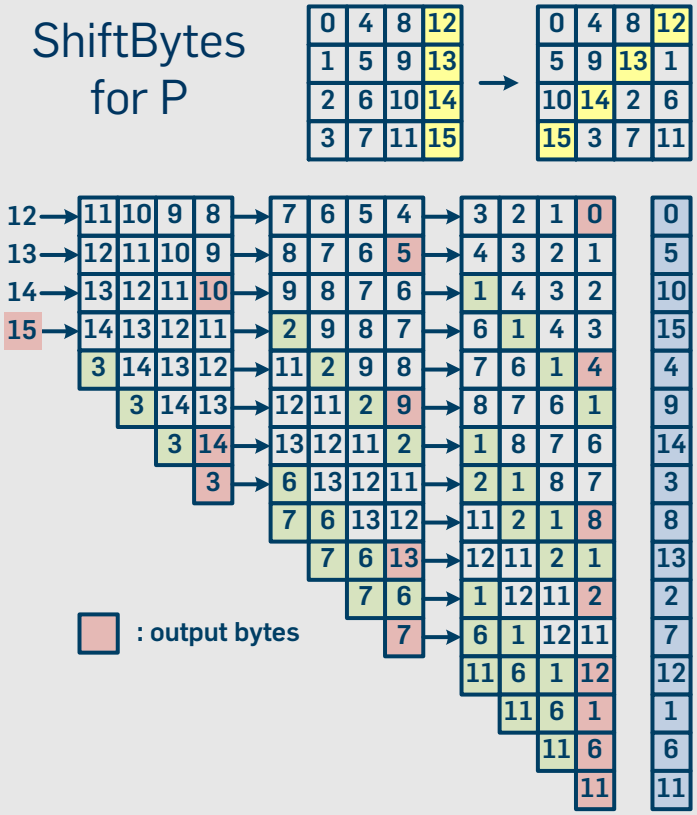
Serial Flow

4×4 toy version

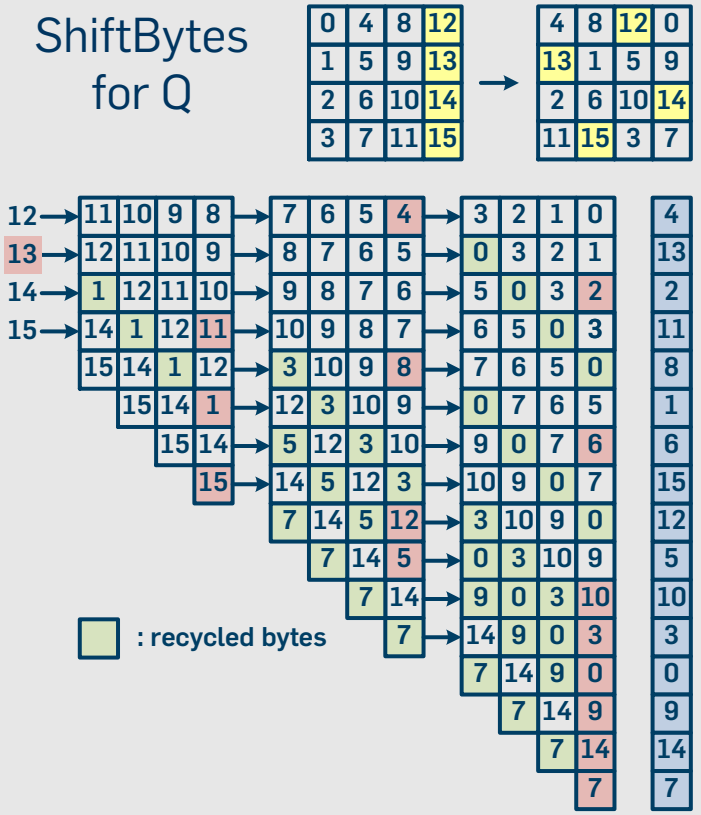
MixBytes

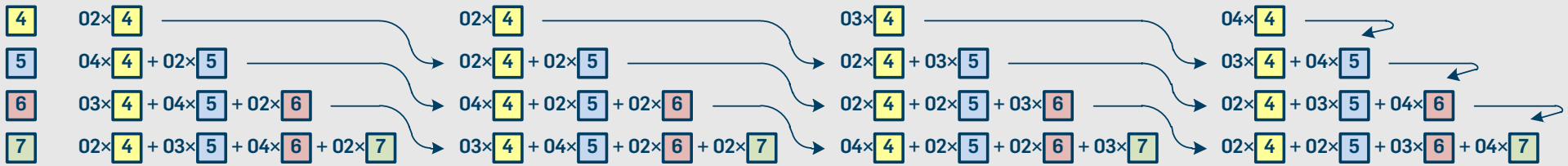


ShiftBytes for P



ShiftBytes for Q

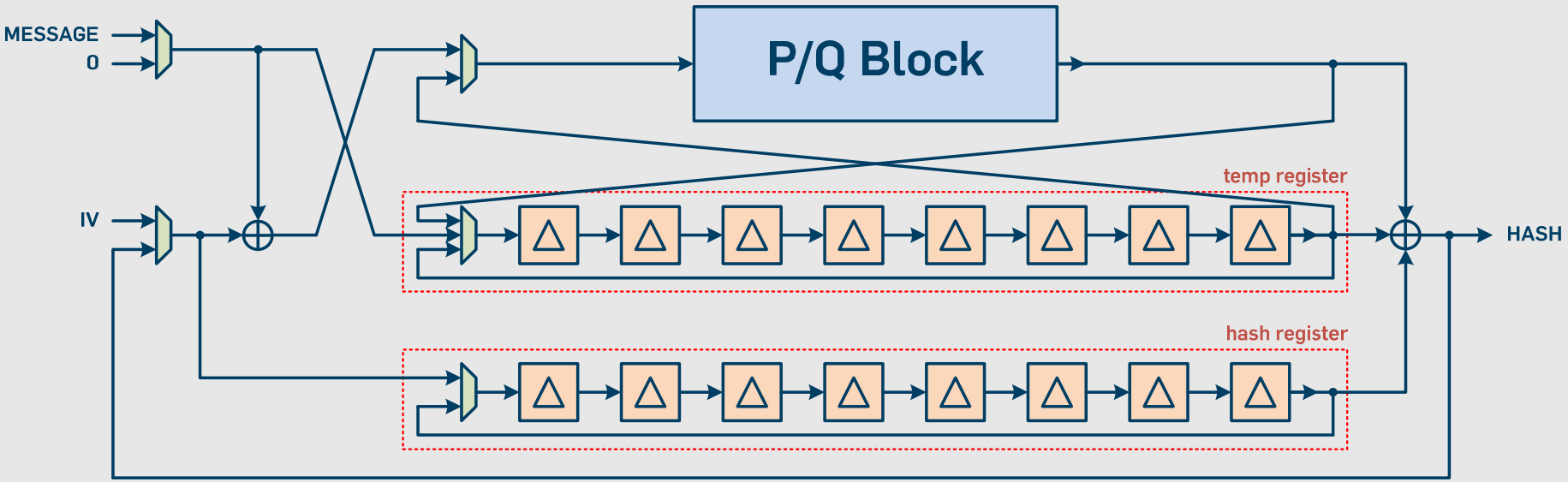




Grøstl

Block Diagram (1)

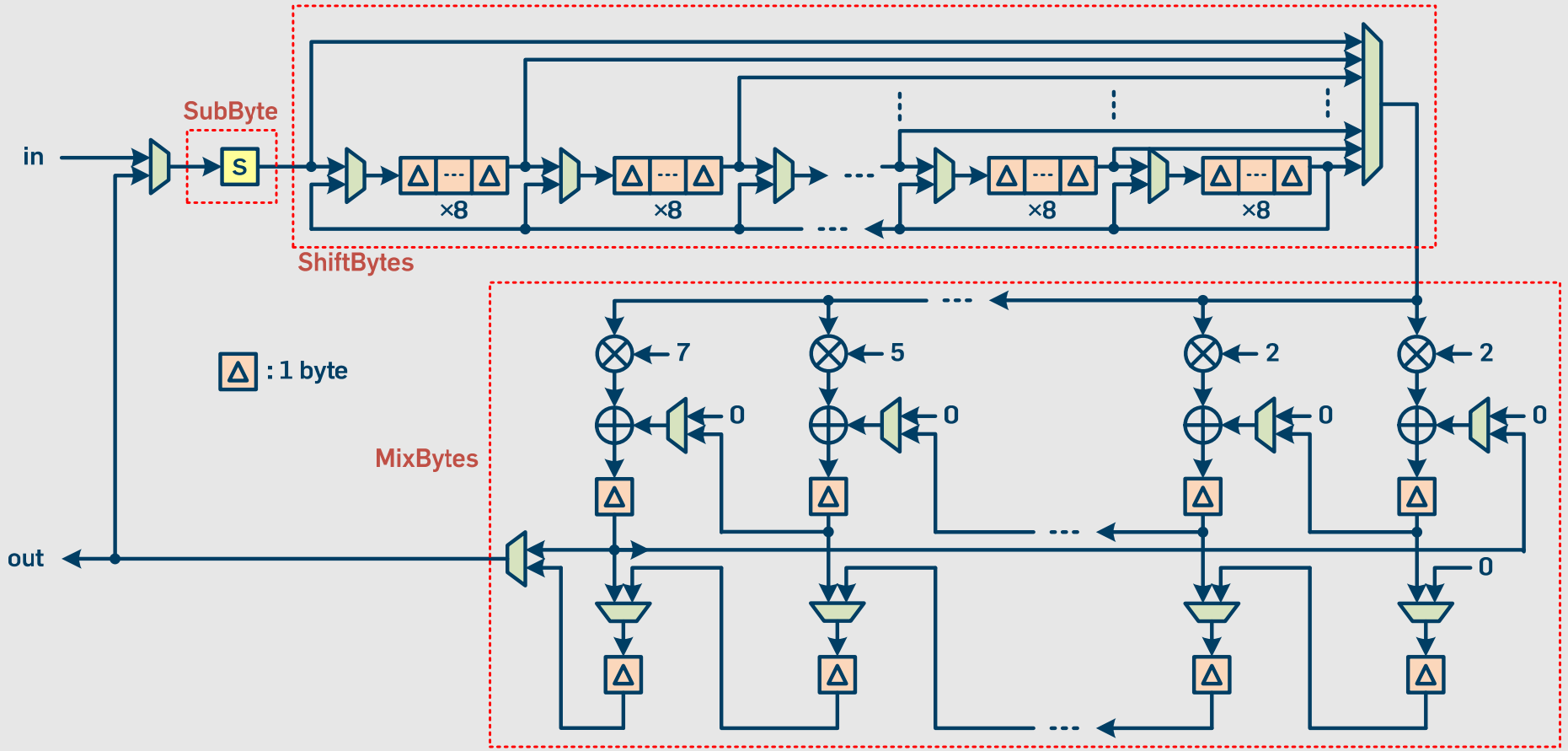
- Single block for both P and Q operations,
- Message stored in the temp register while being processed for P,
- Result of P stored in the temp register while message restored from temp being processed for Q
- Same block used for both f and Ω functions.



Grøstl

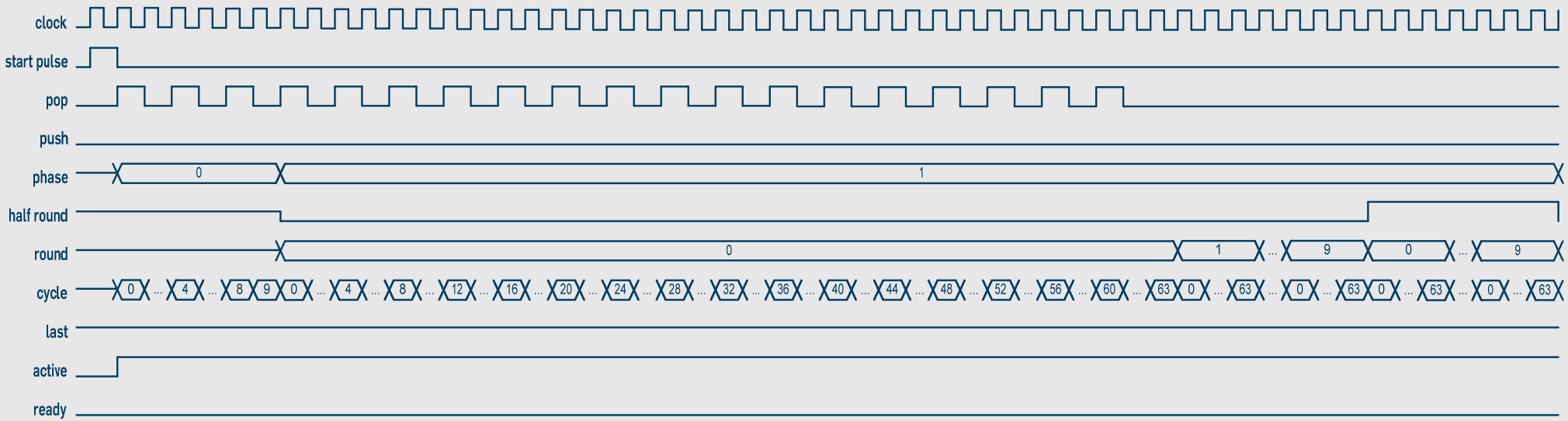
Block Diagram (2)

- P/Q Block



Grøstl

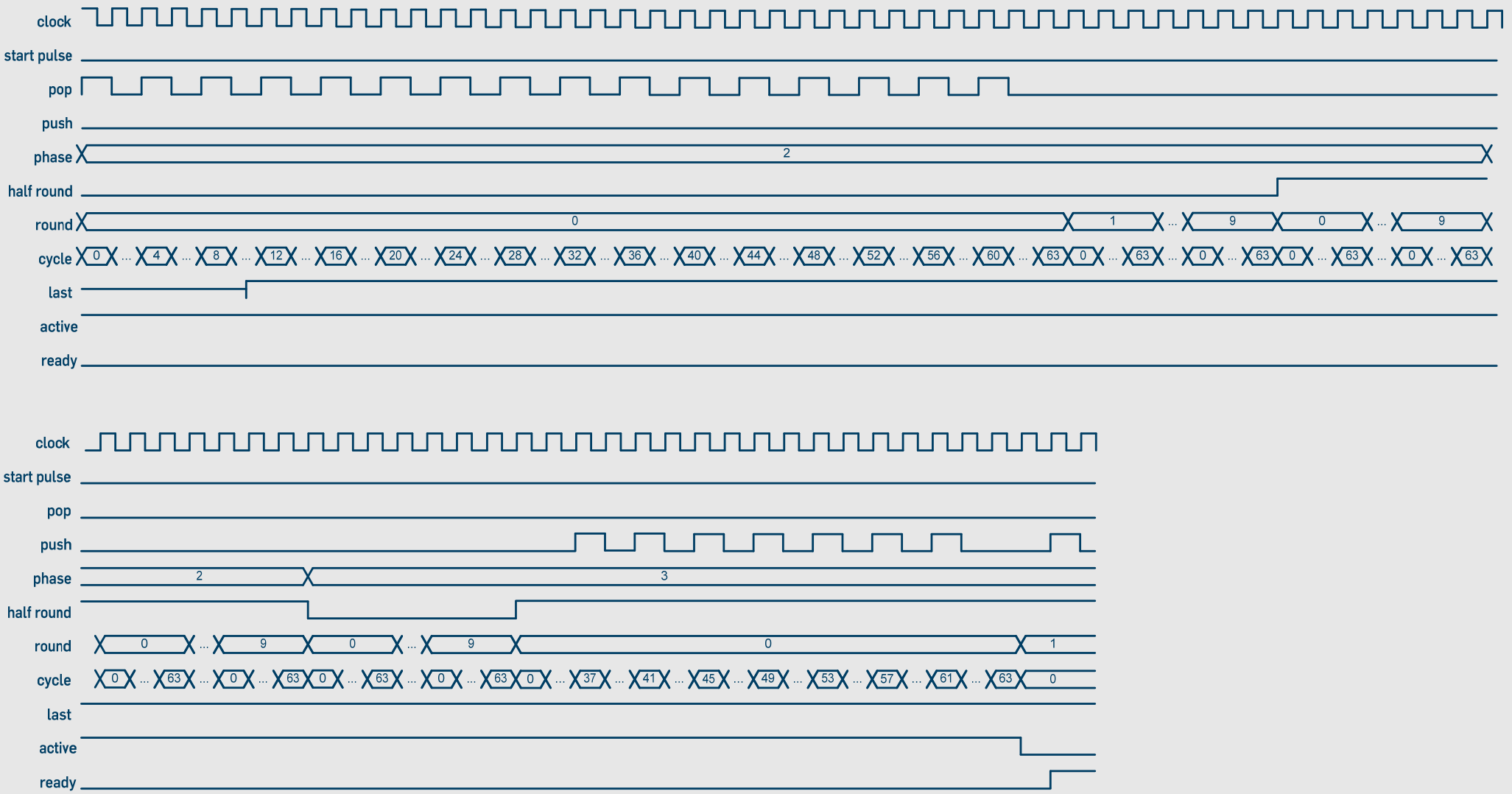
Timing Flow (1)



- Phase-0: Read LENGTH
- Phase-1: Read IV
- Phase-2: Read MESSAGE blocks and Process
- Phase-3: Write MESSAGE DIGEST

Grøstl

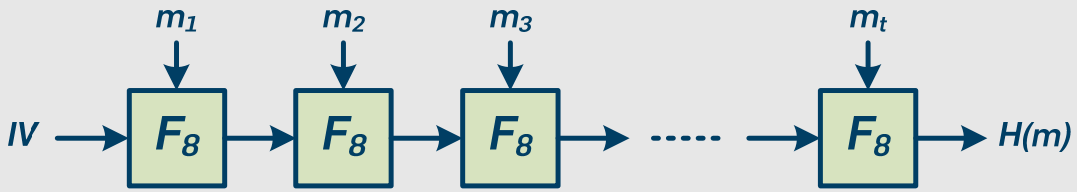
Timing Flow (2)



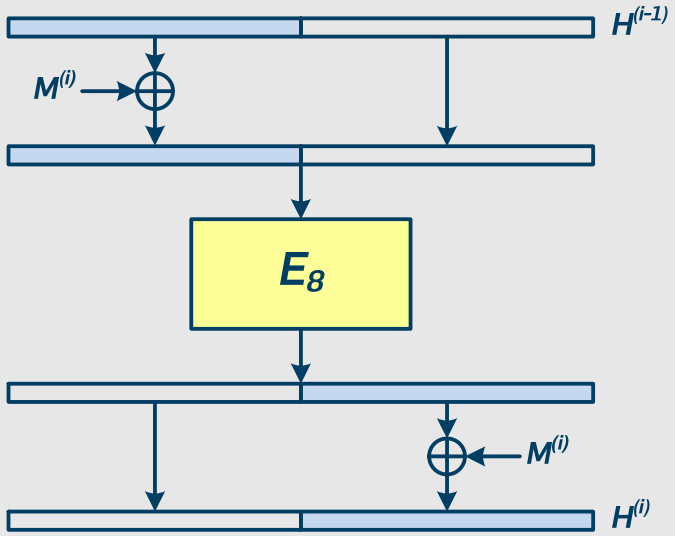
JH

Algorithm (1)

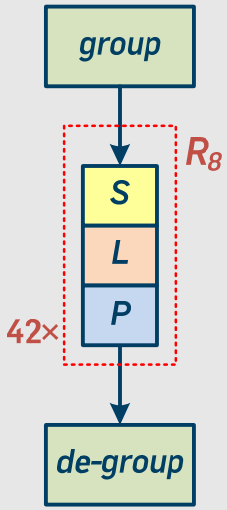
Algorithm	Word	Message	Block	Rounds	Digest
JH-256	32-bit	$< 2^{64}$ - bit	512-bit	42	256-bit



- Compression function F_8:



- Bijective function E_8:

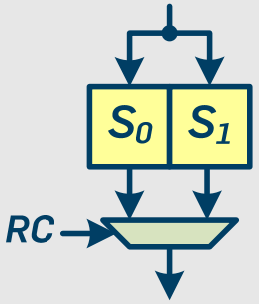


- Bit grouping
- 42 rounds of round function R_8
- Bit de-grouping

JH

Algorithm (2)

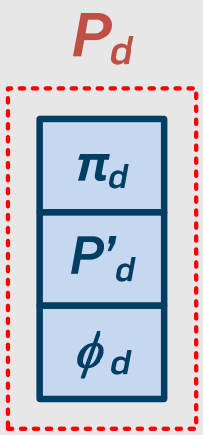
- 4-bit S-boxes selected via round constant



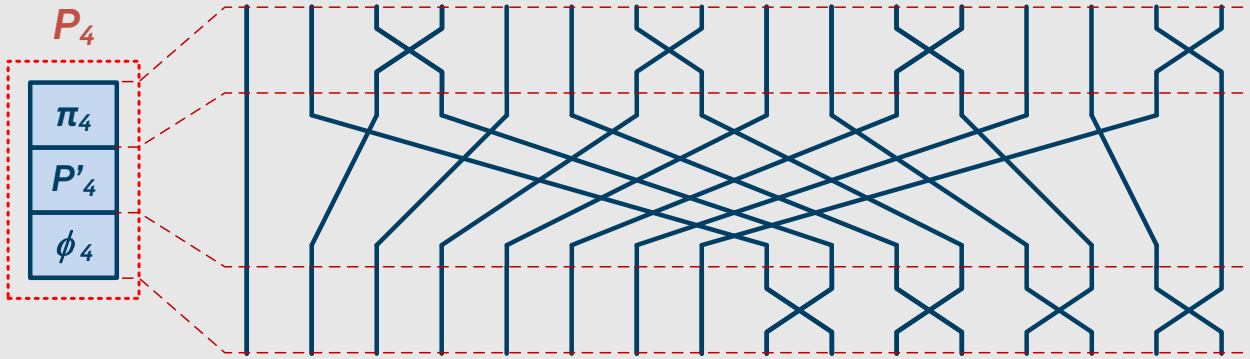
- Linear tranformation ***L*** applied on bytes

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ D_0 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

- Permutation layer P_d composed of three permutations: π_d, P'_d, ϕ_d

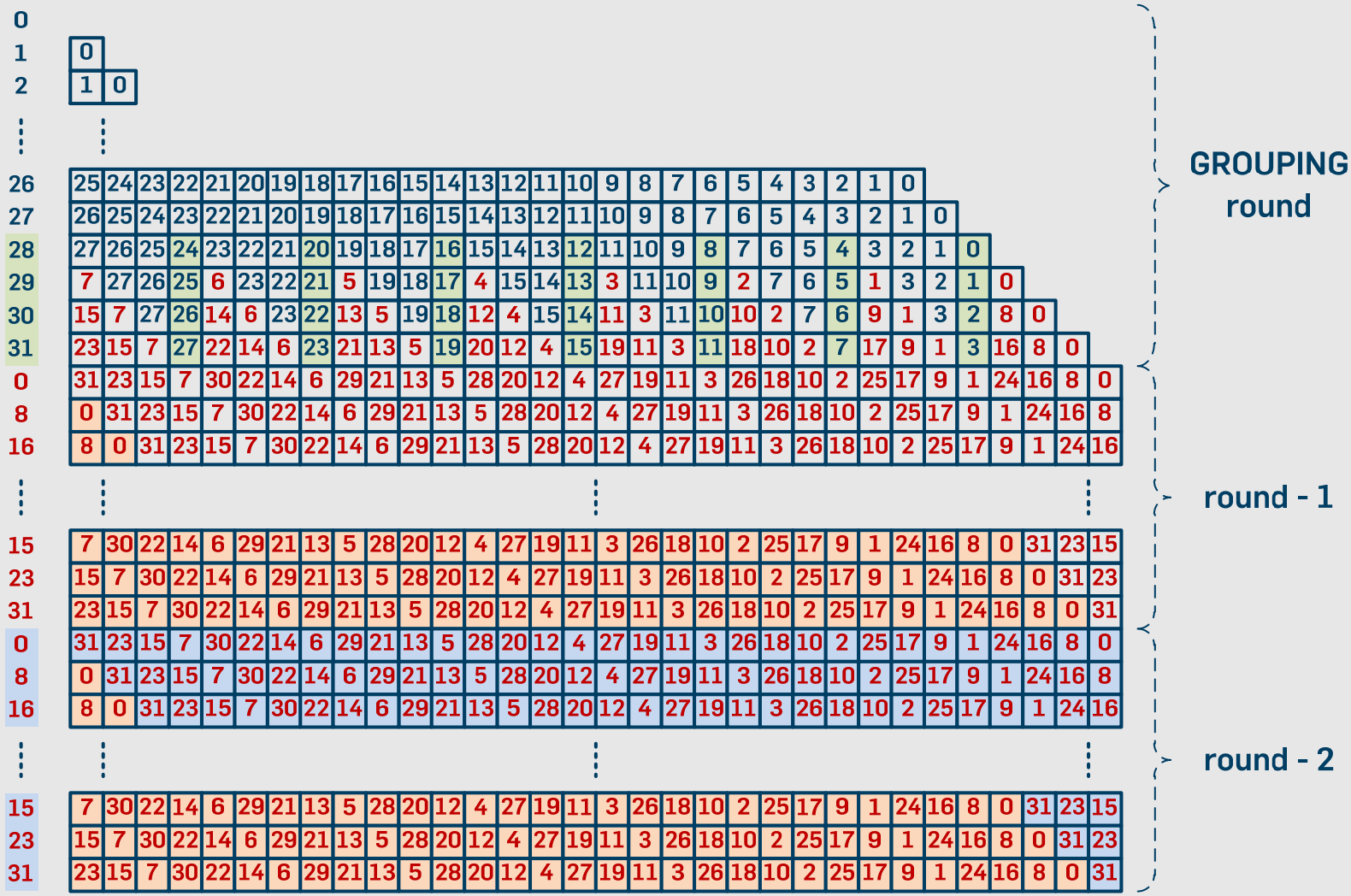


Example for d=4



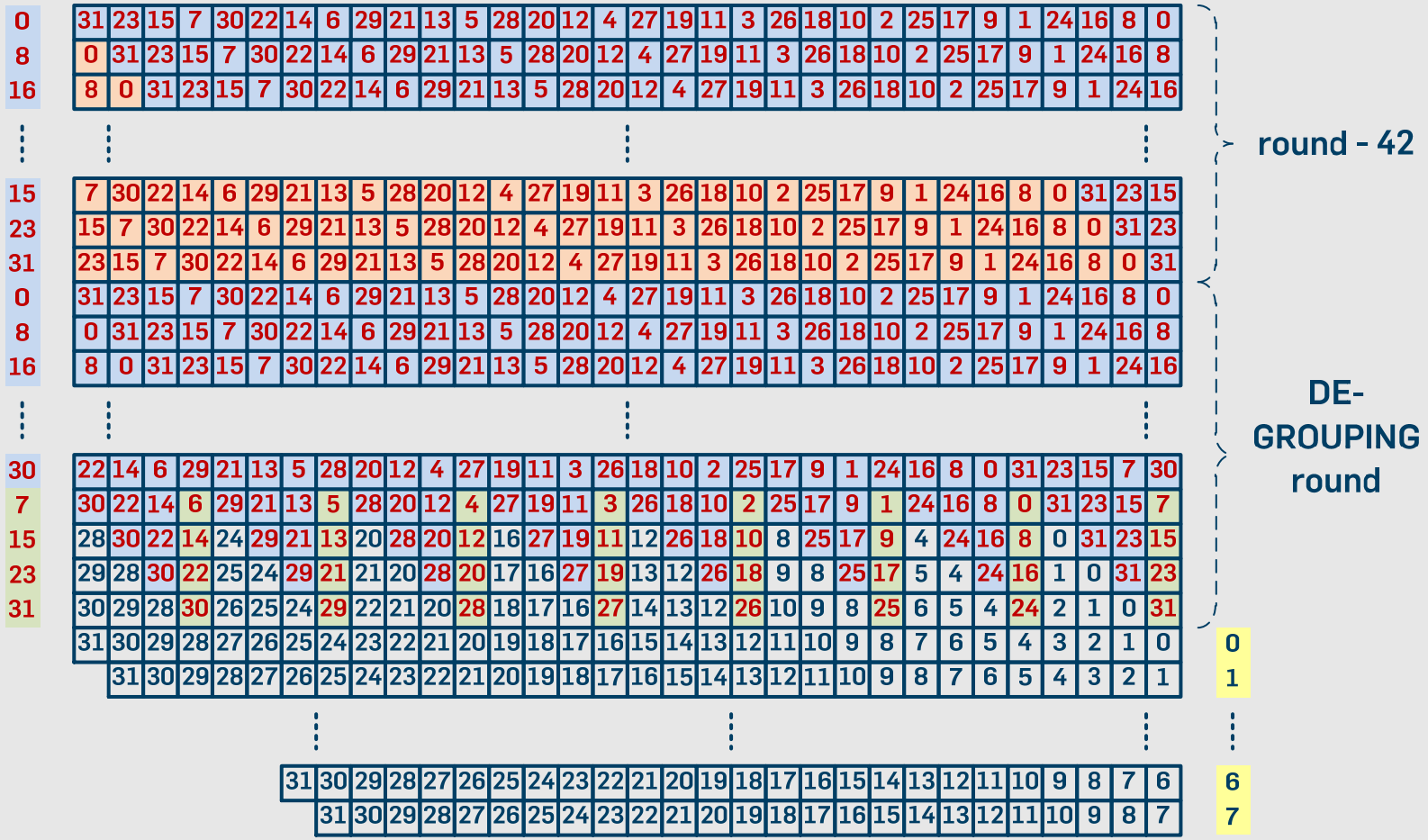
JH

Serial Flow (1)



JH

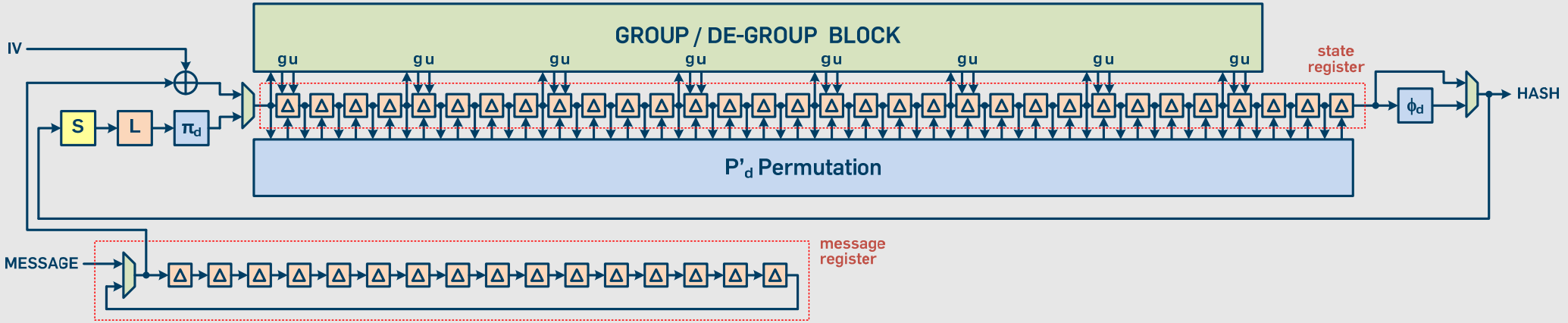
Serial Flow (2)



- Additional rounds for GROUPING and DE-GROUPING
- One extra quarter round required for output squeezing

JH

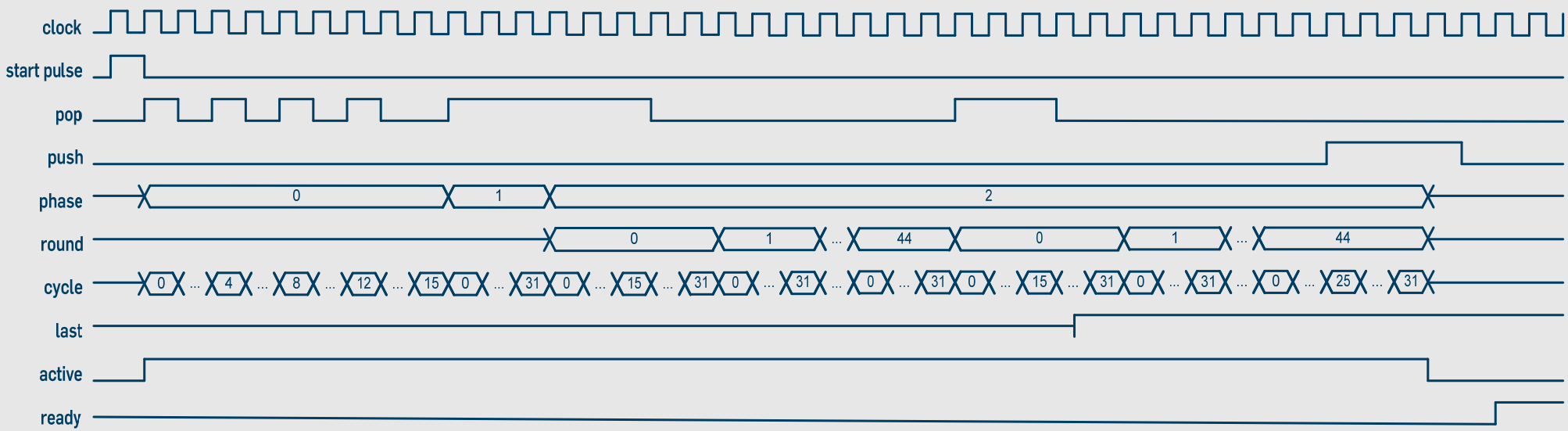
Block Diagram



- 32-bit datapath
- Message backed up in message register for post-processing
- Partial grouping (only at word level)
- S-box, linear transformation and π-section of permutation layer applied in 32-bits
- P‘-section of permutation modified w.r.t. new definition of group function and applied in parallel on the whole state
- ϕ-section of permutation also applied in 32-bits
- Partial de-grouping (adapted to partial grouping function)

JH

Timing Flow

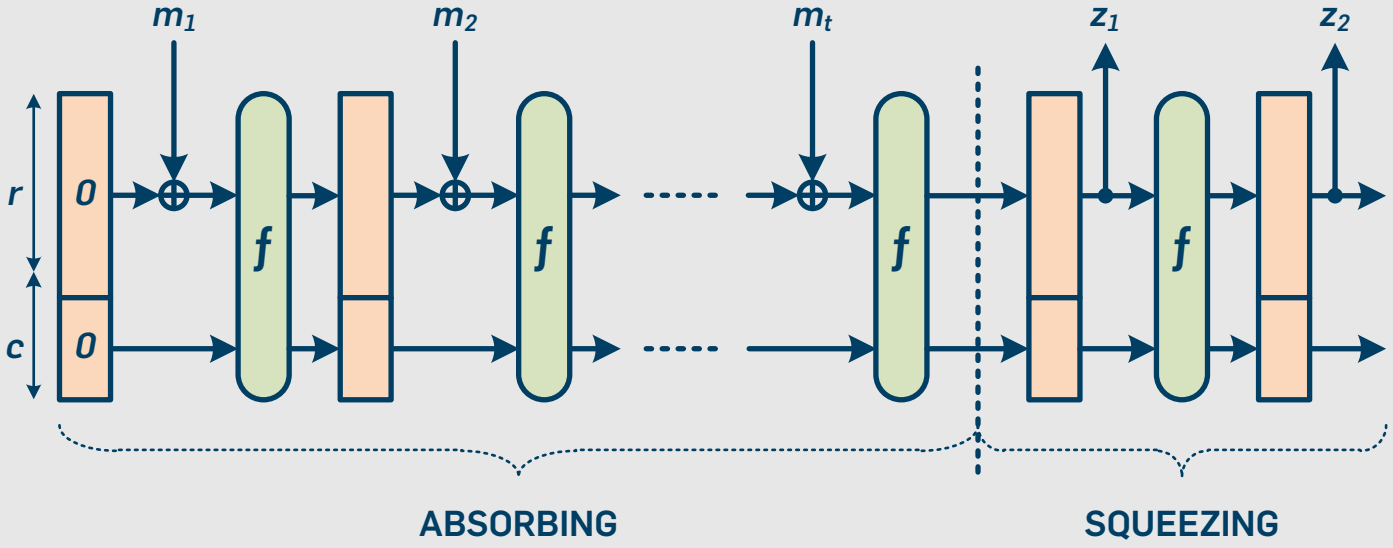


- Phase-0: Read LENGTH
- Phase-1: Read IV
- Phase-2: Read MESSAGE (Round-0) blocks and Process
- Phase-2: Write MESSAGE DIGEST (Round-44 of last message block)

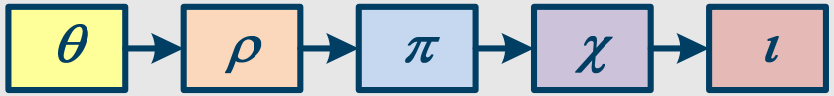
Keccak

Algorithm (1)

Algorithm	Word	Message	Block	Rounds	Digest
Keccak-256	64-bit	$< 2^{128}$ – bit	1088-bit	24	256-bit



- Permutation function f:

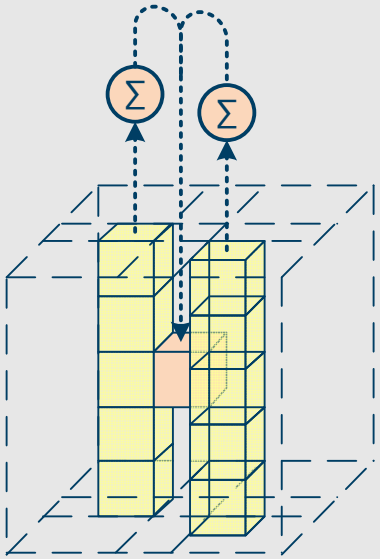


- State organized as a 5×5 matrix of 2^l-bits (l=64)
- r=1088, c=512

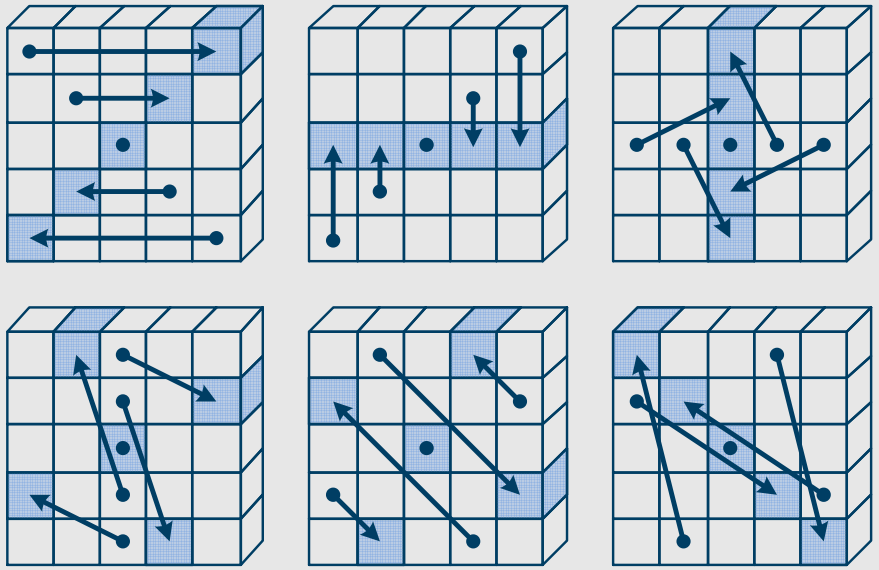
Keccak

Algorithm (2)

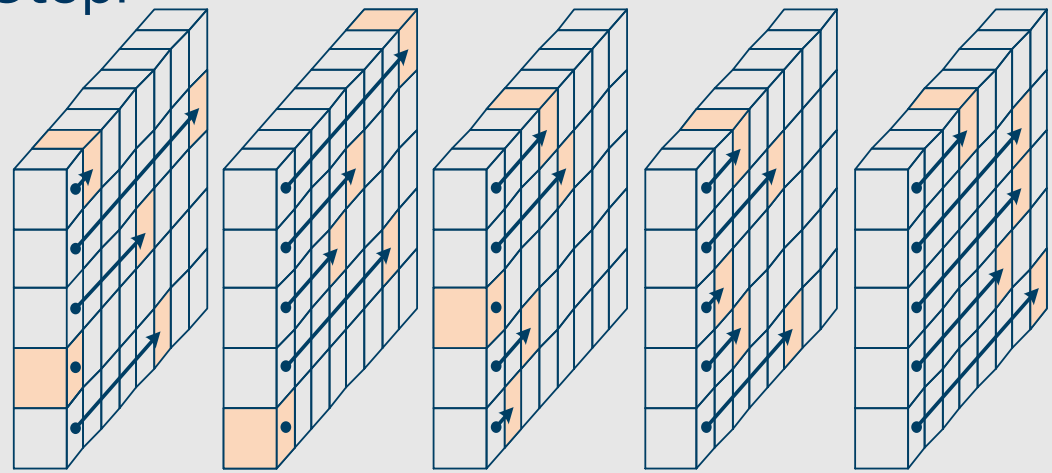
- θ Step:



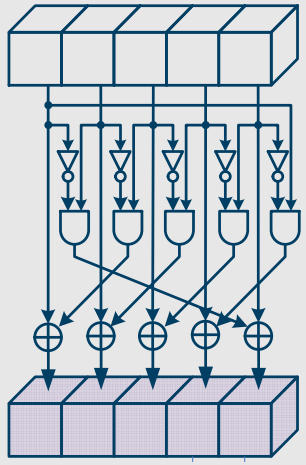
- π Step:



- ρ Step:



- χ Step:



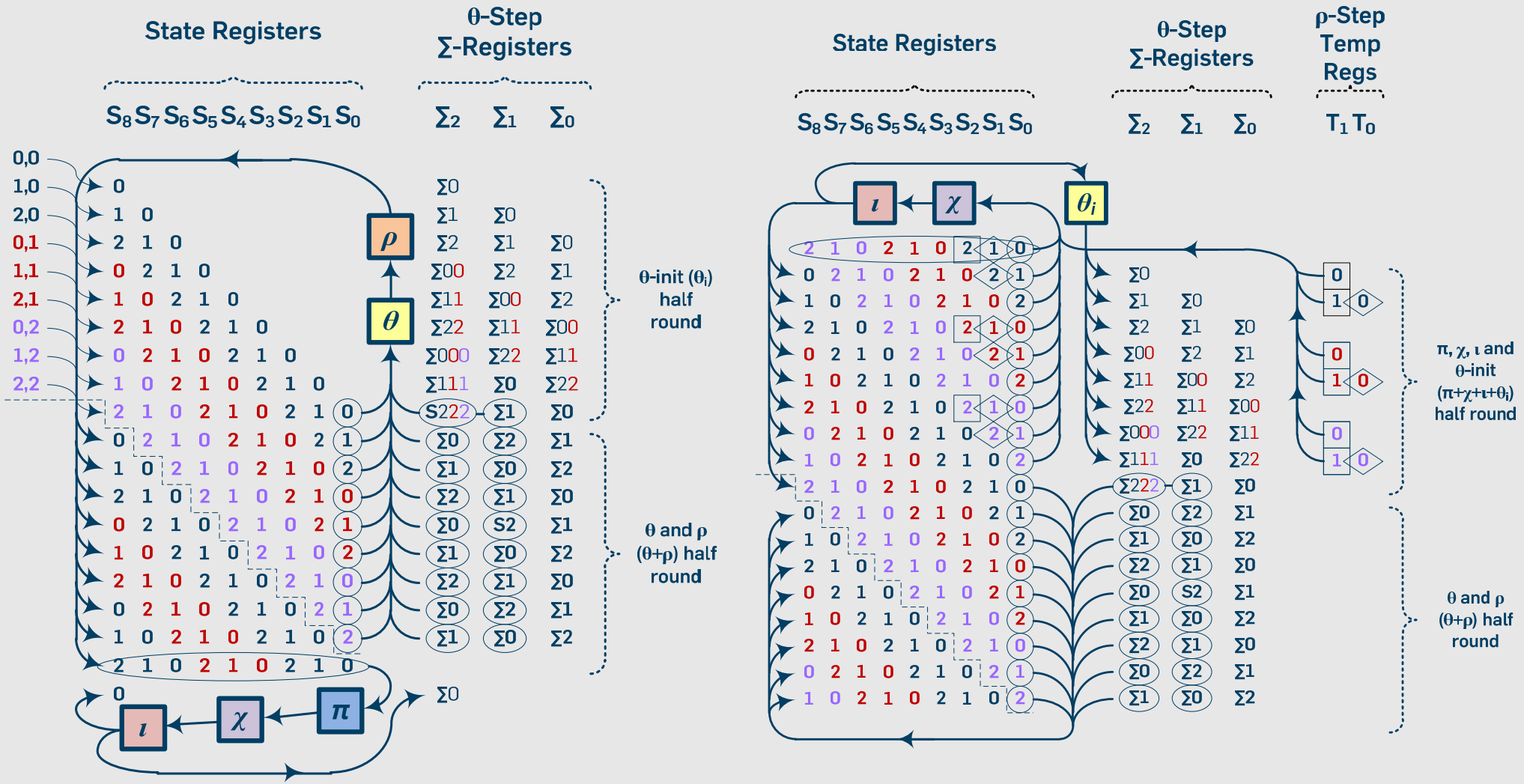
Keccak

Serial Flow

hgi Lehrstuhl für Embedded Security

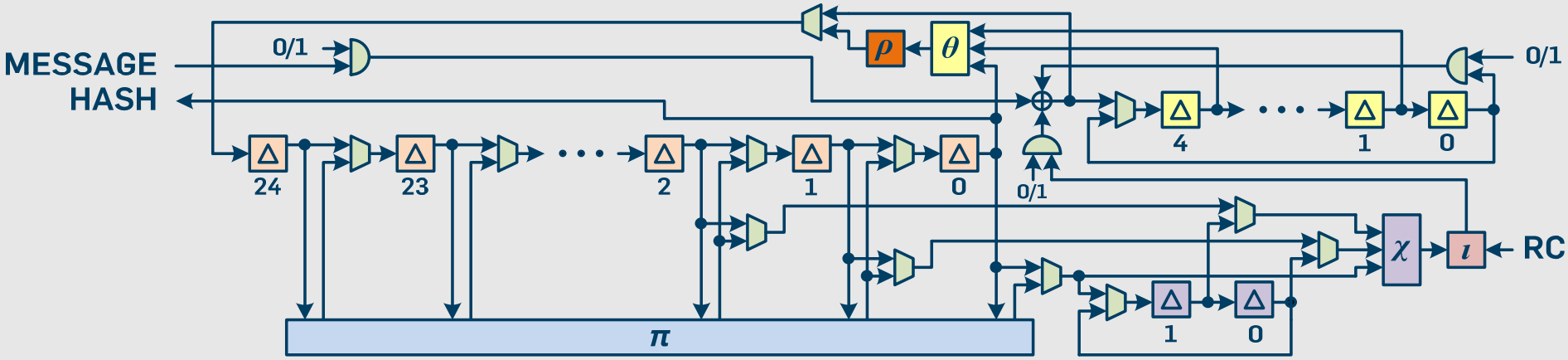
RUB

3×3 toy version



Keccak

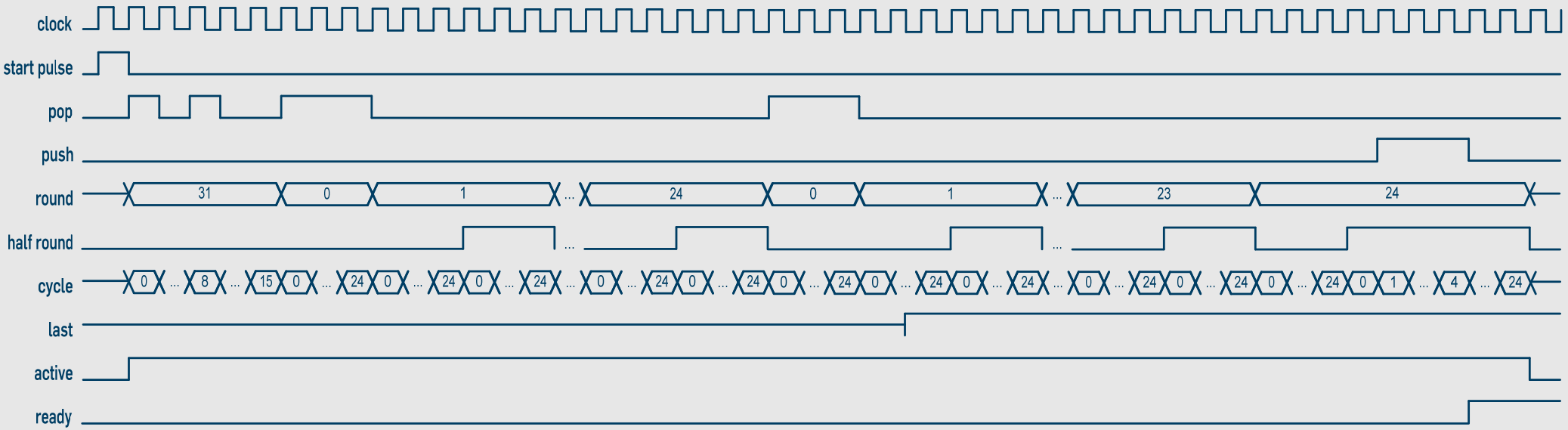
Block Diagram



- 64-bit datapath
- Each round completed in two half rounds:
  - First half round for θ and ρ steps,
  - Second half round for π, χ and ι as well as initialization of columns summations for the next θ step.

Keccak

Timing Flow



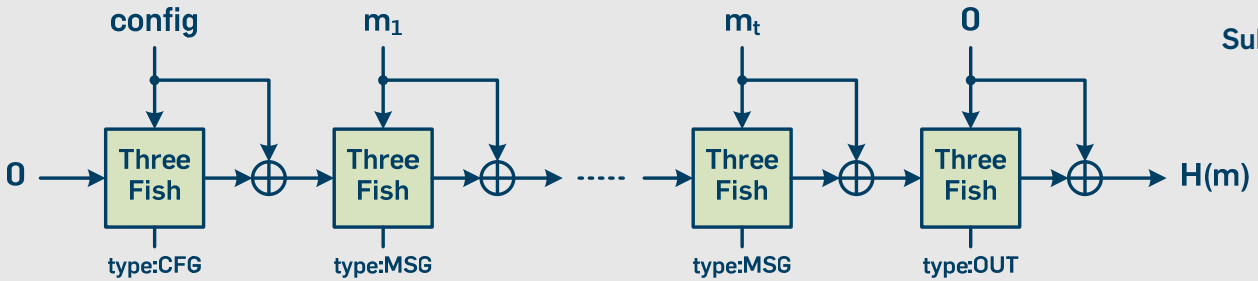
- Round-31: Read LENGTH
- Round-0: Read MESSAGE blocks
- Rounds-1 to 24: Process
- Round-24: Write MESSAGE DIGEST (last message block)

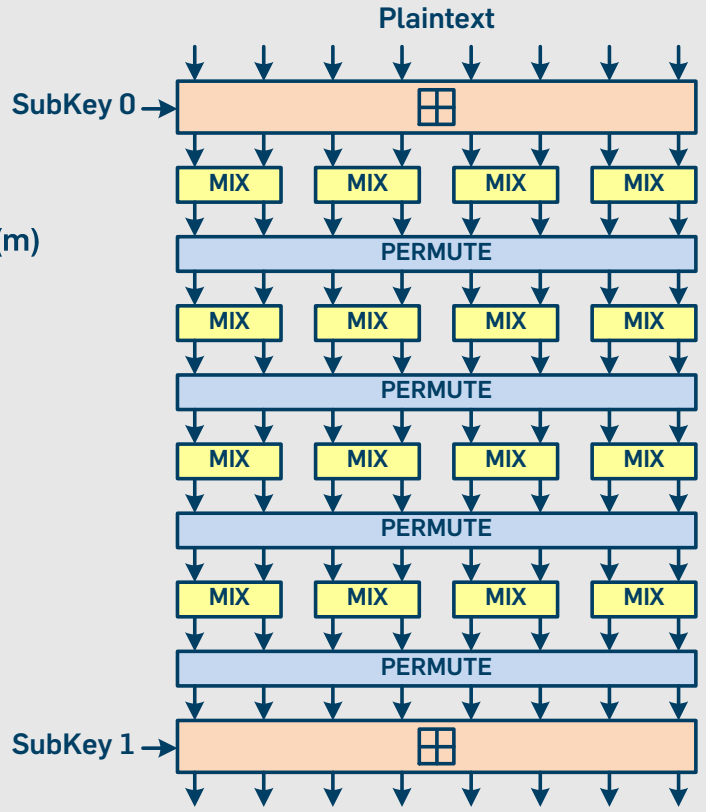
Skein

Algorithm

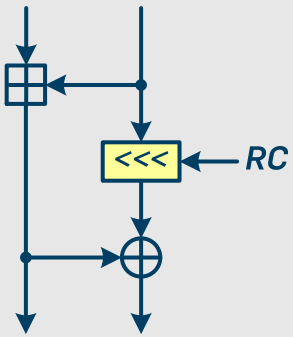
Algorithm	Word	Message	Block	Rounds	Digest
Skein-256	32-bit	$< 2^{64}$ - bit	512-bit	72	256-bit

- Based on the ThreeFish cipher



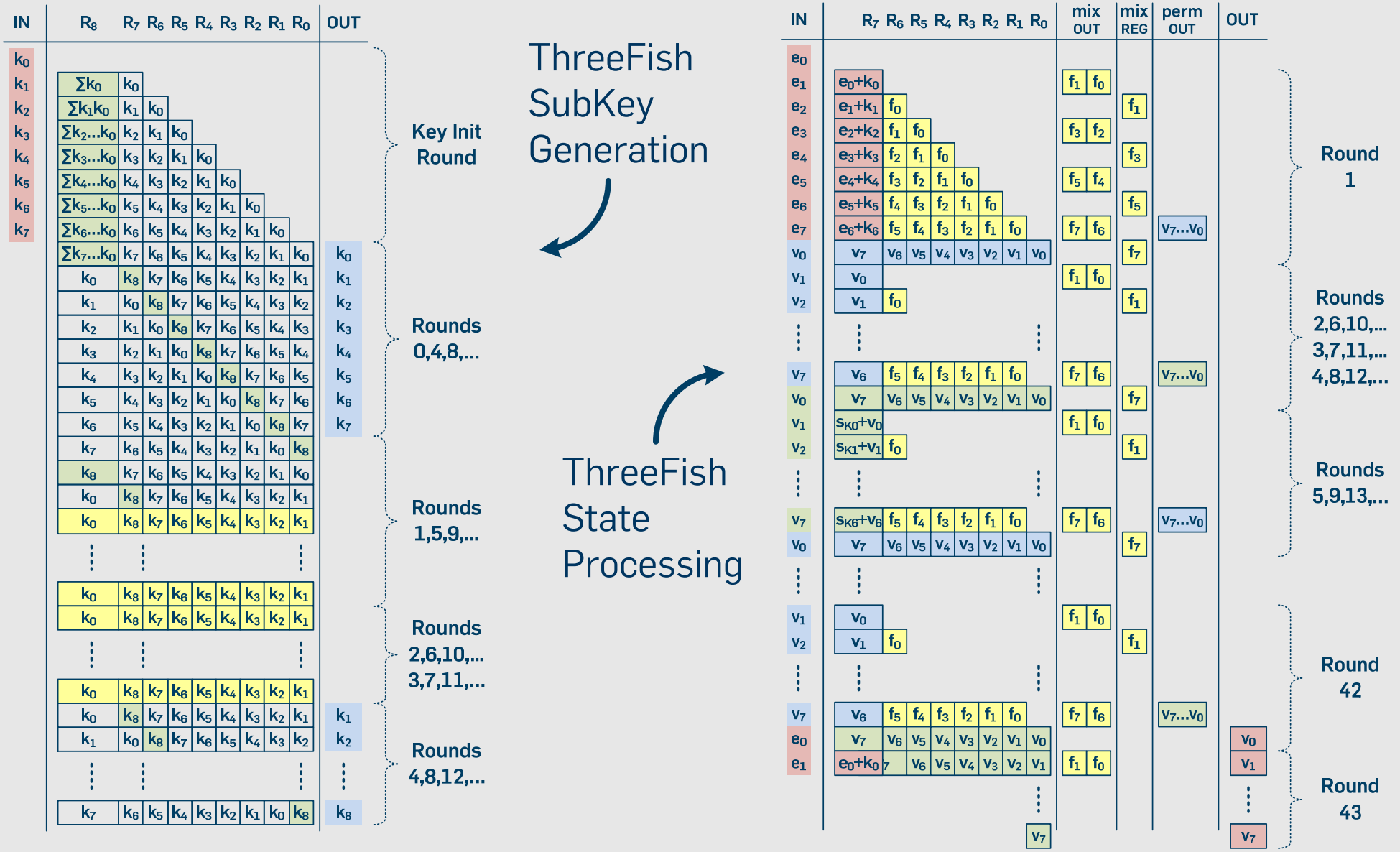


- 72 rounds with SubKey addition every 4 rounds
- Add-Rotate-XOR (ARX) based MIX function (rotation controlled via round constant
- Word (64-bits) permutation same for every round



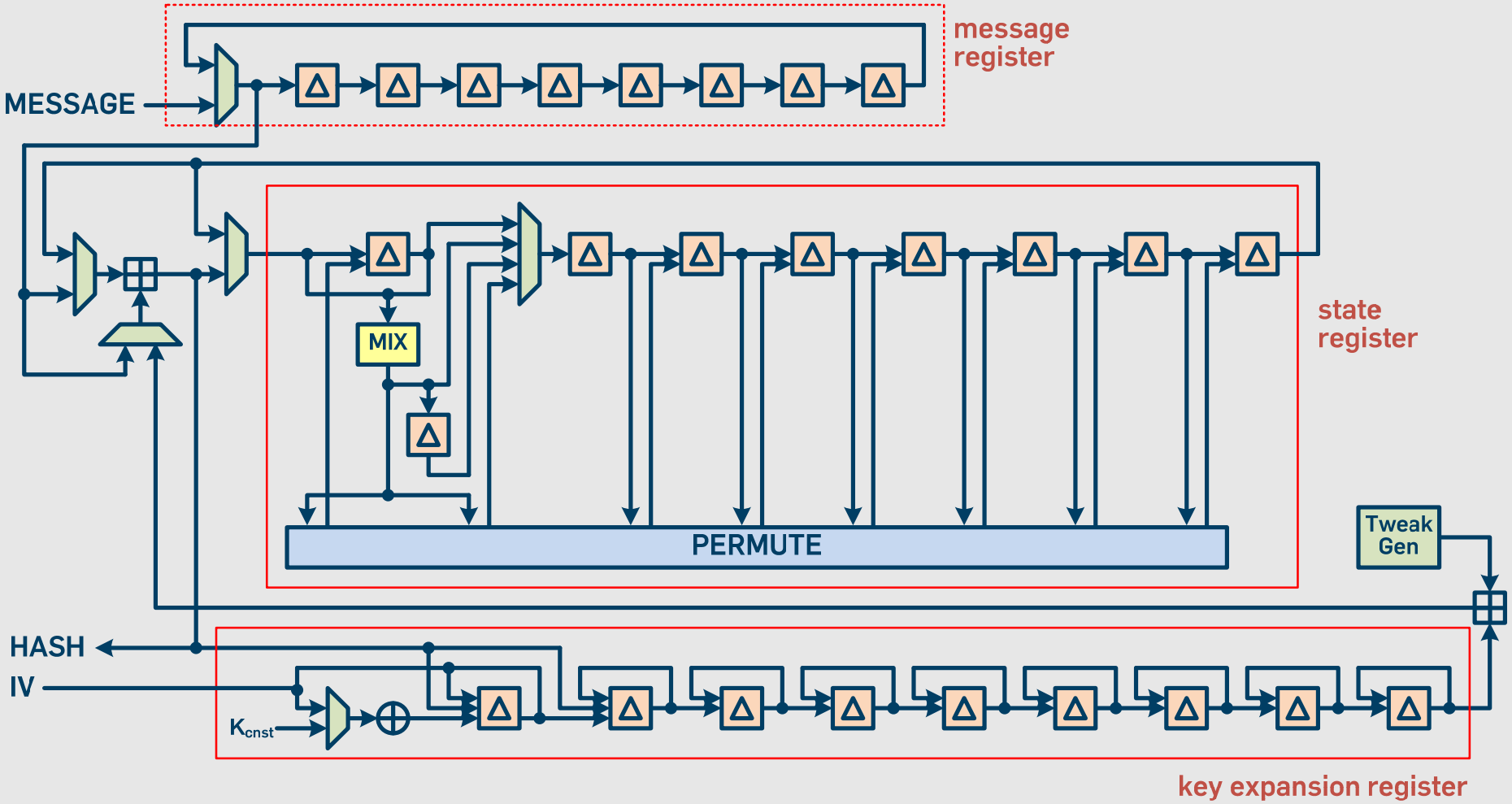
Skein

Serial Flow



Skein

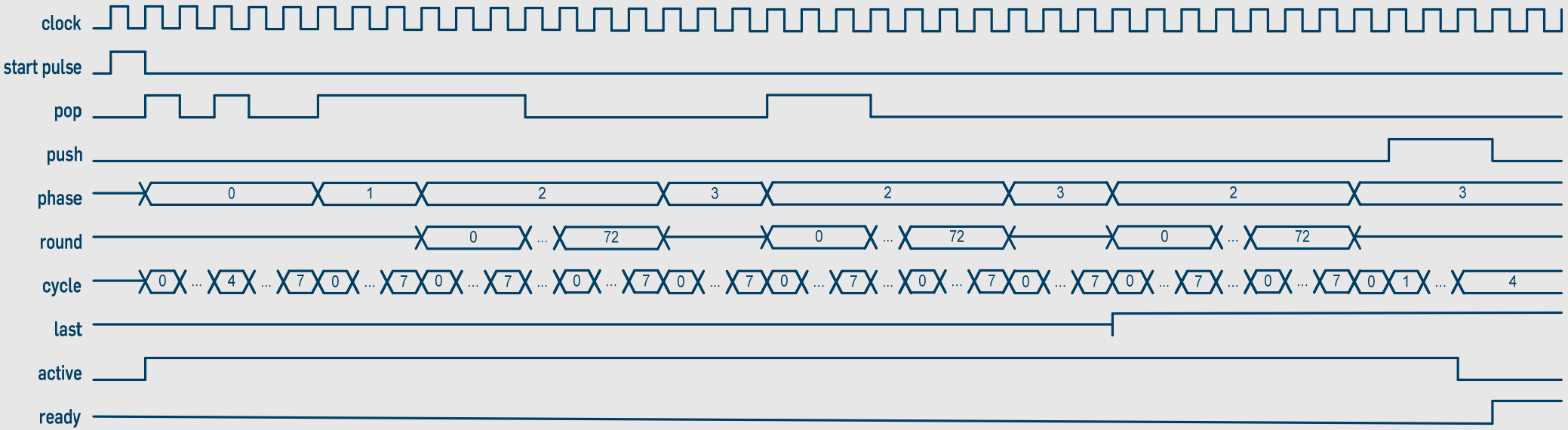
Block Diagram



- 64-bit datapath
- 512-bit IV value directly read from RAM (additional ThreeFish run not necessary)

Skein

Timing Flow



- Phase-0: Read LENGTH
- Phase-1: Read IV
- Phase-2: Read MESSAGE blocks and Process
- Phase-3: Update HASH
- Phase-3: Write MESSAGE DIGEST (last message block)

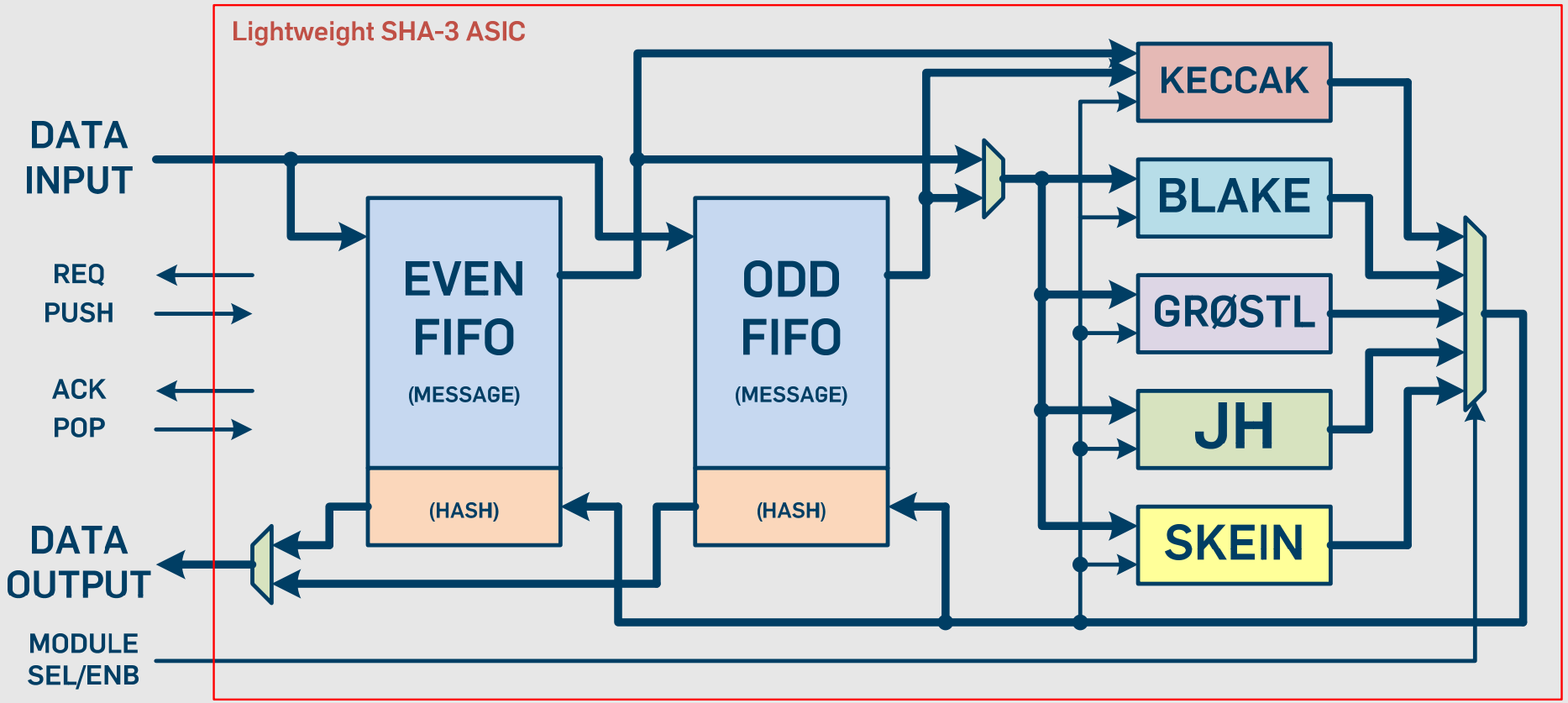
Interface for Modules

Structure

- 32-bit I/O interface
- FIFO based interface to external controller
  - Simple REQuest/ACKnowledge signaling (REQ when FIFO almost empty, ACK when result ready)
  - Even/Odd FIFOs for message read/write via external controller
  - Internal interface with modules 64-bit for KECCAK, 32-bit for all others
  - 2016 byte memory for MESSAGE/DATA
  - 32 byte for HASH result (message DIGEST)
- Only selected module active, others shut down via clock gating

Interface for Modules

Block Diagram



Results

Hash Function	Tech	Area (GE)	Message Block Size (bits)	Cycles per Block	Throughput @ 100 KHz (Kbps)	Tput / Area (bps per GE)
BLAKE	90nm	11.3K	512	240	213.33	18.88
Grøstl	90nm	9.25K	512	1280	40.00	4.32
JH	90nm	13.6K	512	1440	35.55	2.61
Keccak	90nm	15.2K	1088	1200	90.67	5.96
Skein	90nm	15.5K	512	592	86.49	5.58

Comparison with Previous Works

BLAKE

Reference	Technology	Area (KGE)	Frequency (MHz)	Throughput (Mbps @ 1MHz)	Clock Cycles
[1]	180nm	13.58	215	0.63	816
[1]	180nm	8.6[(a)]	100	0.63	N.A.
Our Work	90nm	11.3	N.A.	2.13	240

a) This compact core uses an external memory to hold the message block and does not provide salted hashing.

Comparison with Previous Works

Grøstl

Reference	Technology	Area (kGE)	Frequency (MHz)	Throughput (Mbps @ 1MHz)	Clock Cycles
[2]	350nm	14.622	56	2.61	N.A.
Our Work	90nm	9.2	N.A.	0.40	1280

Comparison with Previous Works

JH

Reference	Technology	Area (kGE)	Frequency (MHz)	Throughput (Mbps @ 1MHz)	Clock Cycles
[3]	180nm	58.832	380.22	13.13	39
[4]	90nm	31.864	353	13.14	N.A.
Our Work	90nm	13.6	N.A.	0.36	1440

Comparison with Previous Works

Keccak

Reference	Technology	Area (kGE)	Frequency (MHz)	Throughput (Mbps @ 1MHz)	Clock Cycles
[5]	130nm	9.3[(a)]	200	0.20	5160
Our Work	90nm	15.2	N.A.	0.91	1200

a) This value includes the area of the RAM. With external RAM, the coprocessor uses 5kGE (as reported in the Keccak main document). Including the area of the RAM yields 9.3kGE.

Comparison with Previous Works

Skein

Reference	Technology	Area (kGE)	Frequency (MHz)	Throughput (Mbps @ 1MHz)	Clock Cycles
[2]	350nm	12.890[(a)]	80	0.25	N.A.
[4]	90nm	22.562[(b)]	50	26.94	10
Our Work	90nm	15.5	N.A.	0.86	592

a) Skein-256-256.

b) Skein-512-256.

Remarks and Comments

- The first comprehensive study on the suitability of SHA-3 finalists for lightweight applications
  - Better results than most of the previous works in terms of area and throughput
- **Best gate count: Grøstl** (first) and **BLAKE** (second)
- **Best Throughput: BLAKE** (first) and **Keccak** (second)
- **Best Tput/Area: BLAKE** (first) and **Keccak** (second)
  - ✓ Note that, except for Keccak, all hash functions have half the block size defined for SHA-256 w.r.t. SHA-512. Such a normalization (i.e. Keccak-800-256) would make Keccak best in gate count, worst in throughput!

Conclusion and Future Directions

- SHA-3 candidates are implemented in “lightweight„ form
- These candidates can be used in lightweight devices, depending on the application and its needs
- **Options/extensions** can be added into the **standard** for **lightweight** version(s) of the selected **hash algorithm**
- Chip realization in progress for:
  - Comprehensive power analysis
  - Testing against side-channel attacks

References

1) L. Henzen, J.-P. Aumasson, W. Meier, R. C.-W. Phan, VLSI Characterization of the Cryptographic Hash Function BLAKE, IEEE Transactions on VLSI Systems, Volume: 19, Issue:10, Pages: 1746-1754, Oct. 2011.
2) S. Tillich, M. Feldhofer, W. Issovits, T. Kern, H. Kureck, M. Mühlberghuber, G. Neubauer, A. Reiter, A. Köfler, and M. Mayrhofer, Compact Hardware Implementations of the SHA-3 Candidates ARIRANG, BLAKE, Grøstl, and Skein, Cryptology ePrint Archive: Report 2009/349, 2009.
3) S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J.-M. Schmidt, and A. Szekely, High-speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein, Cryptology ePrint Archive, Report 2009/510, 2009.
4) M. Knezevic, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, U. Kocabas, J. Fan, T. Katashita, T. Sugawara, K. Sakiyama, I. Verbauwhede, K. Ohta, N. Homma, T. Aoki, Fair and Consistent Hardware Evaluation of Fourteen Round Two SHA-3 Candidates, IEEE Transactions on VLSI Systems, PP(99), pp. 1-13, 2011.
5) G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, Keccak Sponge Function Family," Submission to NIST, 2008.