Korea Institute of Information Security & Cryptology

The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

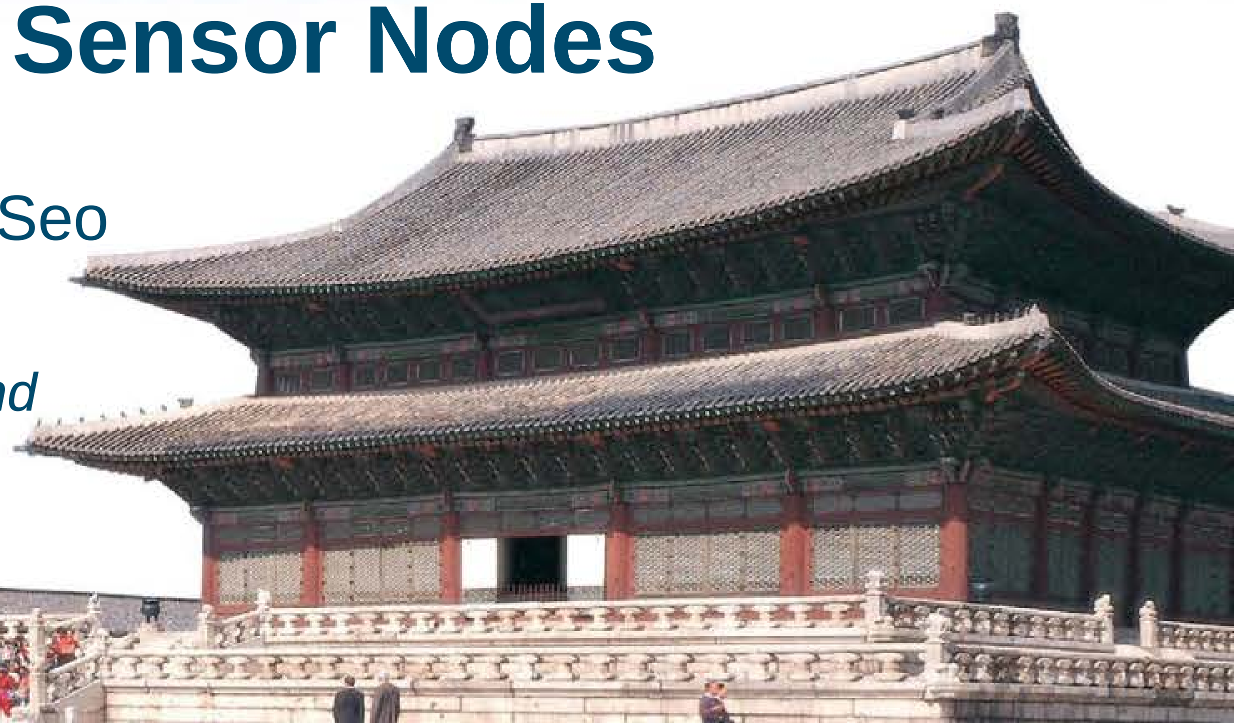
Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Efficient Implementation of SHA-3 Hash Function on 8-bit AVR-based Sensor Nodes

YoungBeom Kim, Hojin Choi, Seog Chung Seo

Cryptography Optimization & Application Lab,

Department of Information Security, Cryptology, and Mathematics, Kookmin University



Contents

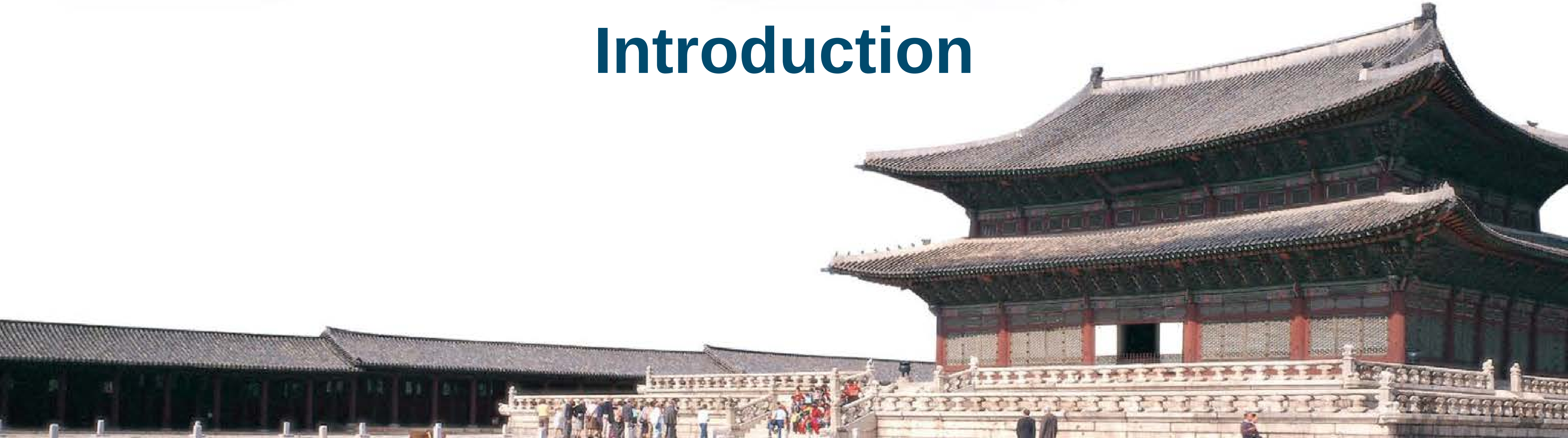
The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Introduction



Some Context

- Hash Function provides data integrity
- Fatal reverse attack has been filed against the existing SHA-2 Family
- The importance and demand of SHA-3 is increasing
- No single implementation method is more efficient than all others on ever possible platforms
- Existing efficient designs are usually hardware or specific architectures (Parallel system) oriented
- SHA-3 is a core algorithm used in MAC, digest, digital signature, DRBG, PQC, and so on.
- General software optimization method for various platforms is an important issue
- As 5G industry increases, a efficient implementation method of SHA-3 in embedded devices is important.

Overview of SHA-3

- Keccak algorithm selected to be next-generation hash function in SHA-3 competition held by NIST
- SHA-3 based on Sponge structure
  - Absorbing Process : Compressing message and updating internal state by f-function
  - Squeezing Process : Computing digest

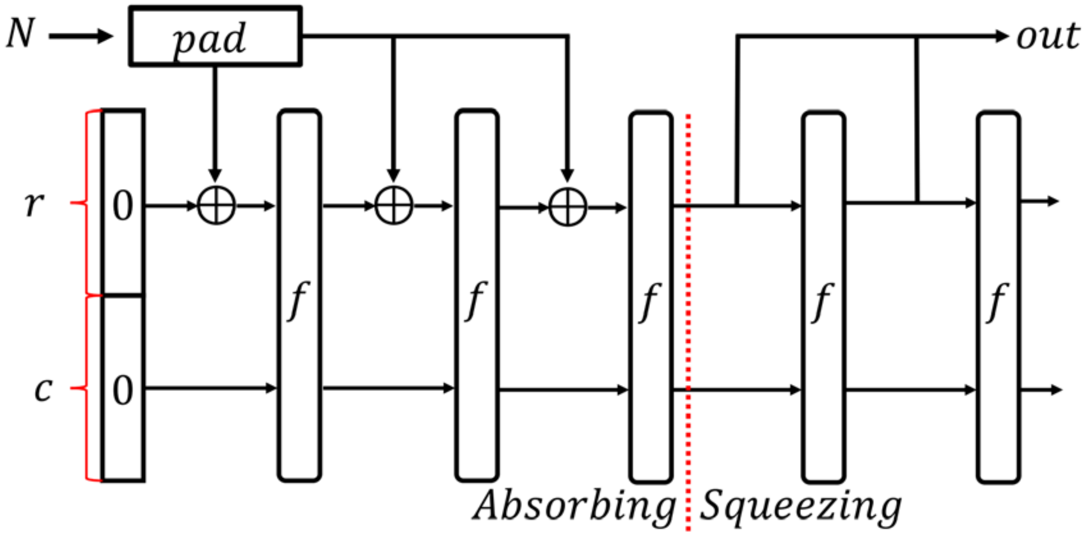


Fig. 1: Overview of Sponge structure

The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) – December 4 (Fri), 2020 | Virtual Conference

Korea Institute of Information Security & Cryptology

Overview of SHA-3

- *state* of f-function is a three-dimensional $x \times y \times z$ matrix
  - Row x and Column y are both fixed to five
  - Consisting of 25 lanes

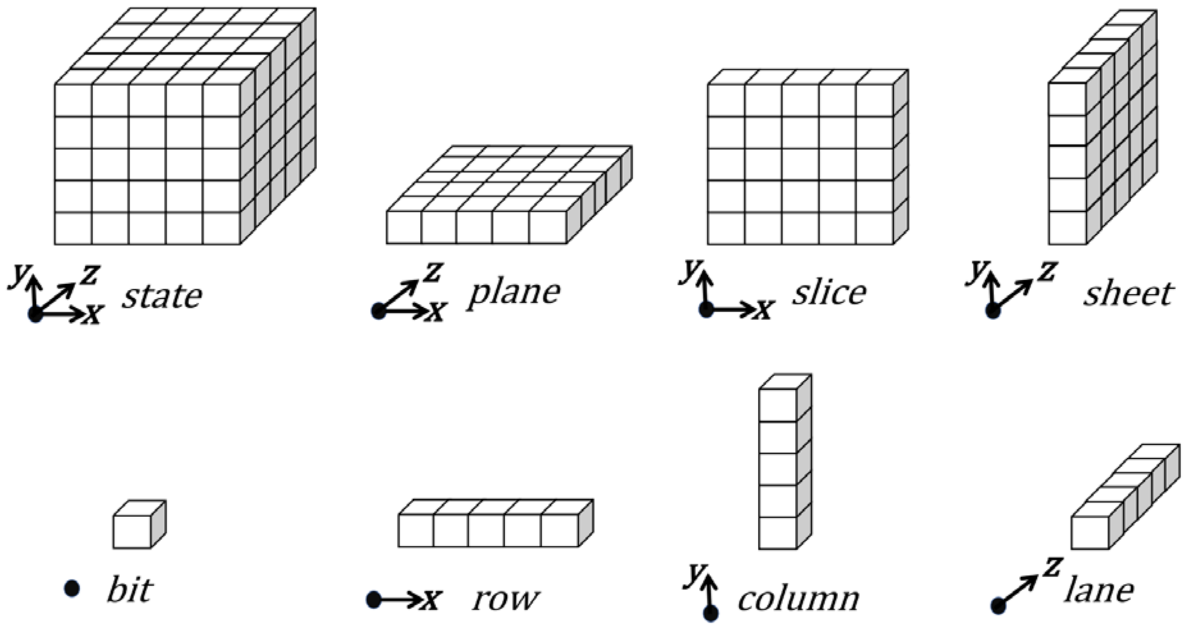


Fig. 2: State of SHA-3

Overview of SHA-3

- θ process
  - XOR each bit in $state$ with parties of two columns
  - XORing sum of columns $((x-1), z)$ and $((x+1), (z-1))$

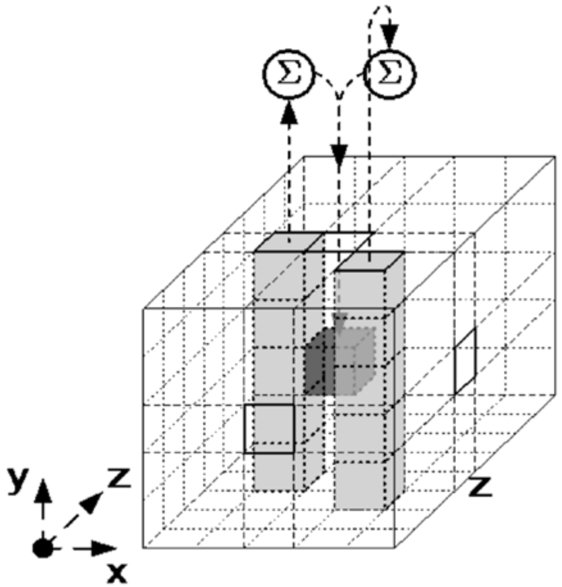


Fig. 3: Overview of θ process

Require: $state\ A$
Ensure: $state\ A'$
For all pairs(x, z) such that $0 \leq x < 5$ and $0 \leq z < w$
  $C[x, z] = A[x, 0, z] \oplus A[x, 1, z] \oplus A[x, 2, z] \oplus A[x, 3, z] \oplus A[x, 4, z]$;
For all pairs(x, z) such that $0 \leq x < 5$ and $0 \leq z < w$
  //This step is initial θ
  $D[x, z] = C[(x - 1) \bmod 5, z] \oplus C[(x + 1) \bmod 5, (z - 1) \bmod w]$;
For all triples(x, y, z) such that $0 \leq x, y < 5$ and $0 \leq z < w$
  $A'[x, y, z] = A[x, y, z] \oplus D[x, z]$;
return A'

Alg. 1: Algorithm of θ process

Overview of SHA-3

- π process
  - Rearranging the positions of the lanes
  - Not changing value of lanes

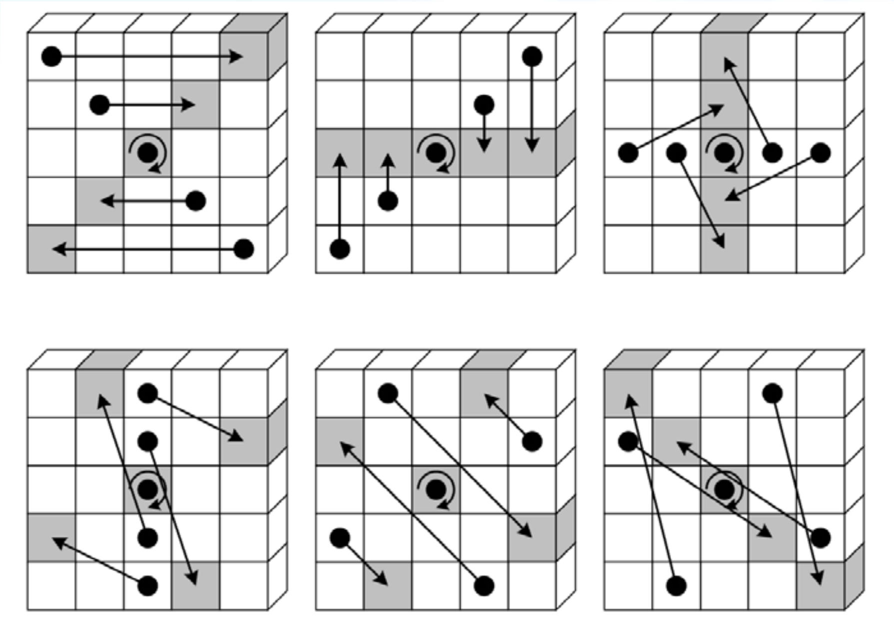


Fig. 4: Overview of π process

Require: *state A*
Ensure: *state A′*
For all triples (x, y, z) such that $0 \le x, y < 5$ and $0 \le z < w$
    $A'[x,y,z] = A[(x + 3y) \bmod 5, x, z]$.
return A'

Alg. 2: Algorithm of π process

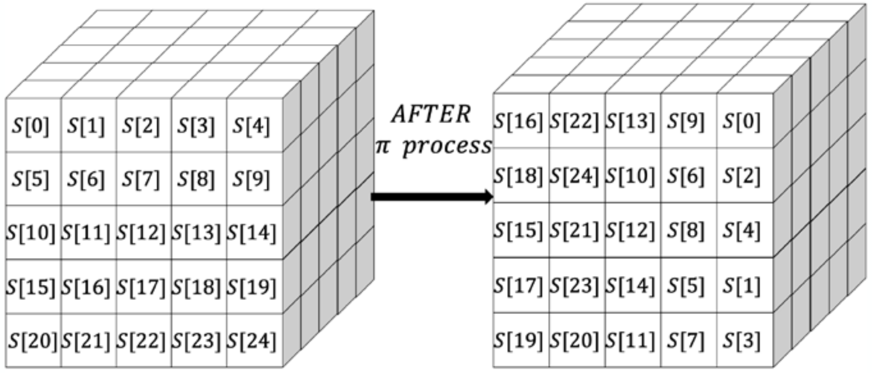


Fig. 5: Detail Structure of π process

Overview of SHA-3

- ρ process
  - Right-rotating the bits of each lane as much as offset
  - Not changing position of lanes
  - Implemented in combination with π process in standard implementation method

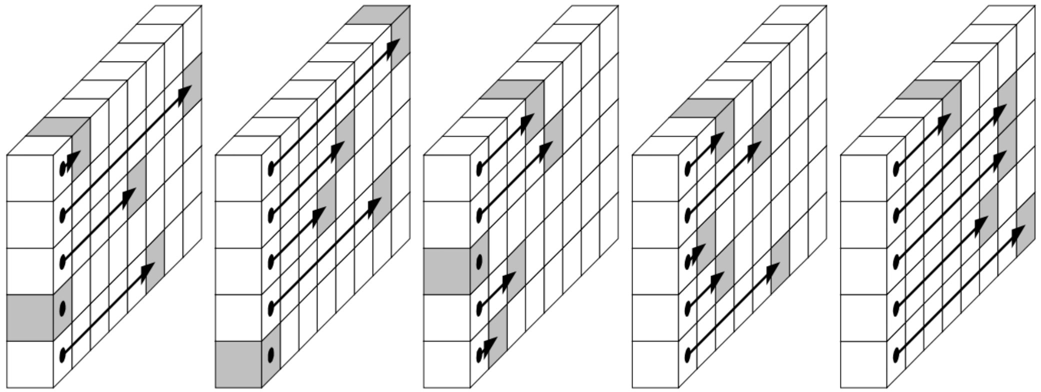


Fig. 5: Overview of ρ process

```
Require: state A
Ensure: state A'
 1: For all z such that 0 ≤ z < w
        Let A'[0, 0, z] = A[0, 0, z].
 2: Let (x, y) = (1, 0).
 3: For t from 0 to 23:
        a. for all z such that 0 ≤ z < w
             A'[x, y, z] = A[x, y, (z-((t + 1)(t + 2)/2) mod w];
        b. Let (x,y) = (y, (2x + 3y) mod 5).
 4: return A'
```

Alg. 3: Algorithm of ρ process

Overview of SHA-3

- χ process
  - XORing each bit with a nonlinear function of two other bits in its row
  - Operating in row form
- ι process
  - XORing Round-constant and S[12] of $state$
  - Operating for single lane

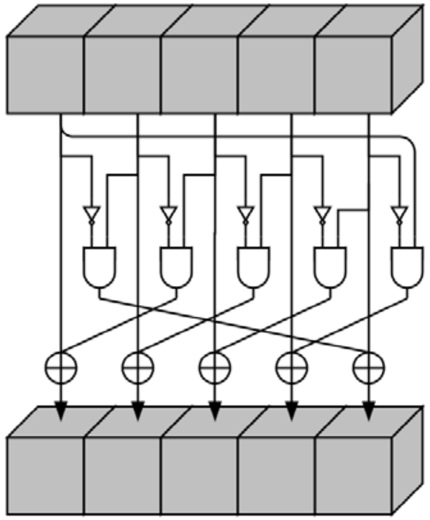


Fig. 6: Overview of χ process

Require: *state* A
Ensure: *state* A'
For all triples (x, y, z) such that $0 \leq x, y < 5$ and $0 \leq z < w$
    $A'[x,y,z] = A[x,y,z] \oplus ((A[(x+1) \bmod 5, y, z) \oplus 1] \cdot A[(x+2) \bmod 5, y, z])$.
return A'

Alg. 4: Algorithm of χ process

Standard Method

- The standard implementation method of SHA-3 follows as: $\theta \rightarrow \pi\sim\rho \rightarrow \chi\sim\iota$
- Combing π process and ρ process into $\pi\sim\rho$ process
- Accessing 7 times to $State$ during f-function
  - When b = 1600, $State$ is 200 bytes and f-function comprise 24 round
  - Requiring 168 memory access to $State$ during f-function
- Memory access cause higher overhead than arithmetic and logical operations in low-end-processor

Standard Method	Initial θ	θ process	$\pi\sim\rho$ process	$\chi\sim\iota$ process	Total Access
Load	O	O	O	O	7 times
Store	X	O	O	O	

Table. 1: Number of memory access to $State$ in Standard Method

The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Korea Institute of Information Security & Cryptology

Memory Optimization

Memory Optimization

- Proposed implementation method of SHA-3 follows as: $\theta \sim \rho\ (\pi) \rightarrow \chi \sim \iota$
- Implementing π process implicitly in $\theta \sim \rho$ process
- Combing θ process and ρ process into $\theta\ \sim \rho$ process
- Accessing 5 times to $State$ during f-function
  - Requiring 120 memory access to $State$ during f-function
  - Proposed Method : 120 < Standard Method 168
- Reducing memory access twice compared to the standard implementation method

Proposed Method	Initial θ	$\theta \sim \rho$ process	π process	$\chi \sim \iota$ process	Total Access
Load	O	O	X (Implicitly)	O	5 times
Store	X	O	X (Implicitly)	O	

Table. 2: Number of memory access to $State$ in Proposed Method

The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) – December 4 (Fri), 2020 | Virtual Conference

Korea Institute of Information Security & Cryptology

Memory Optimization

- $\boldsymbol{\theta}$ and $\boldsymbol{\rho}$ process execute independent operation for lane

- Appling $\boldsymbol{\rho}$ process before storing in $\boldsymbol{\theta}$ process

- Appling $\boldsymbol{\pi}$ process implicitly when updating state (store)
  - $\boldsymbol{\pi}$ process is a rearrange process for each lane
  - $\boldsymbol{\pi}$ process can be executed implicitly

- Memory address translation operation occurs only once
  - $\boldsymbol{\theta}$ and $\boldsymbol{\pi}\sim\boldsymbol{\rho}$ process require twice translation in standard method
  - Standard Method : $\boldsymbol{\theta} \rightarrow \boldsymbol{\pi}\sim\boldsymbol{\rho}$; twice
  - Proposed Method : $\boldsymbol{\theta}\sim\boldsymbol{\rho}$ $(\boldsymbol{\pi})$; once

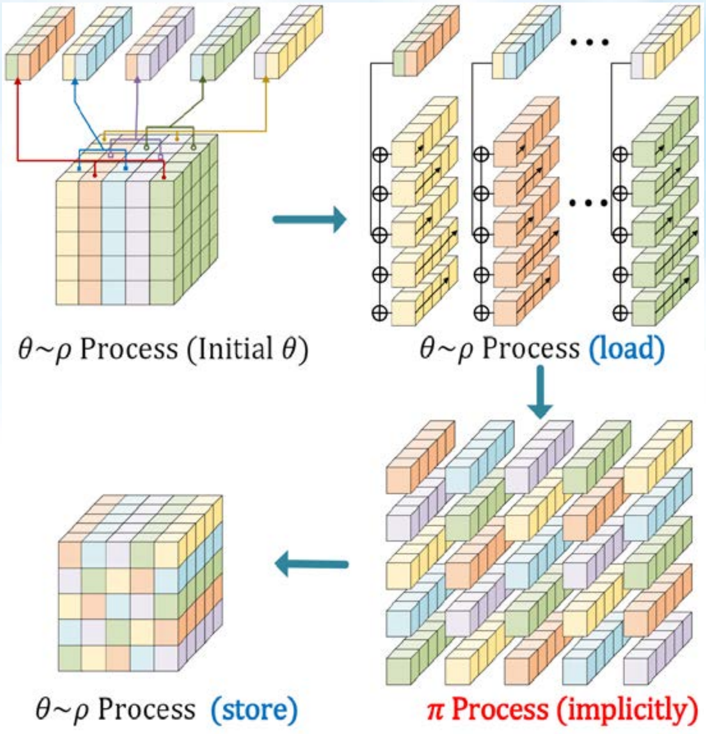


Fig. 7: Overview of Proposed Method

Korea Institute of Information Security & Cryptology

The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Chaining optimization methodology

Target Platforms

- 8-bit AVR MCUs
  - ATmega 128
  - Popularly used in WSNs (Wireless Sensor Networks)

- Spec of ATmega 128
  - Flash Memory : 128 KB
  - SRAM : 4KB
  - EEPROM : 4KB
  - 32 8-bit general-purpose registers

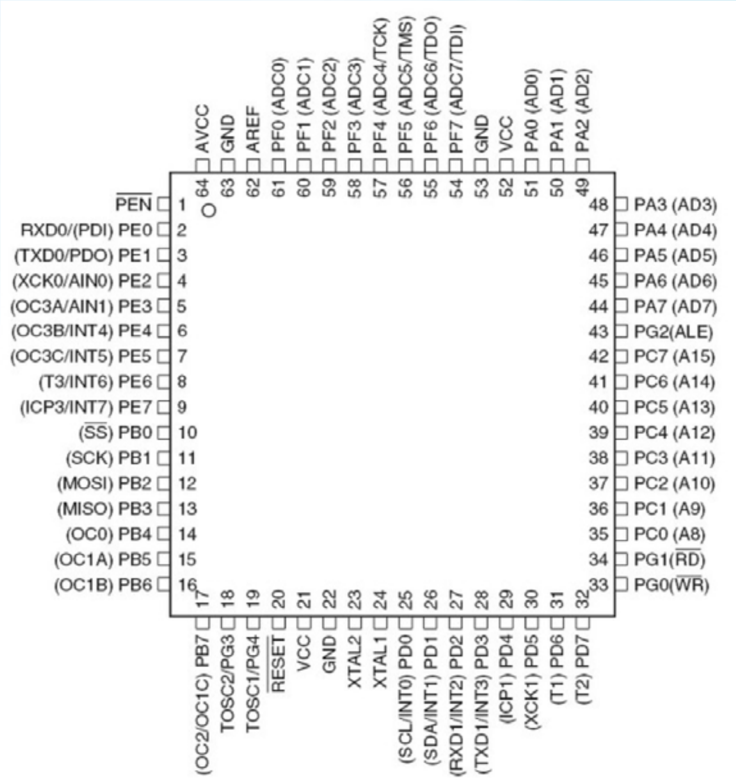


Fig. 8: ATmega 128

Register Scheduling

- The generally used parameter is b = 1600, where the state is 200 bytes
- R8-R15 and R16-R23 hold two lanes
- R2-R5 are used to translate the memory address
  - Initial $\boldsymbol{\theta}$ and lanes of $State$

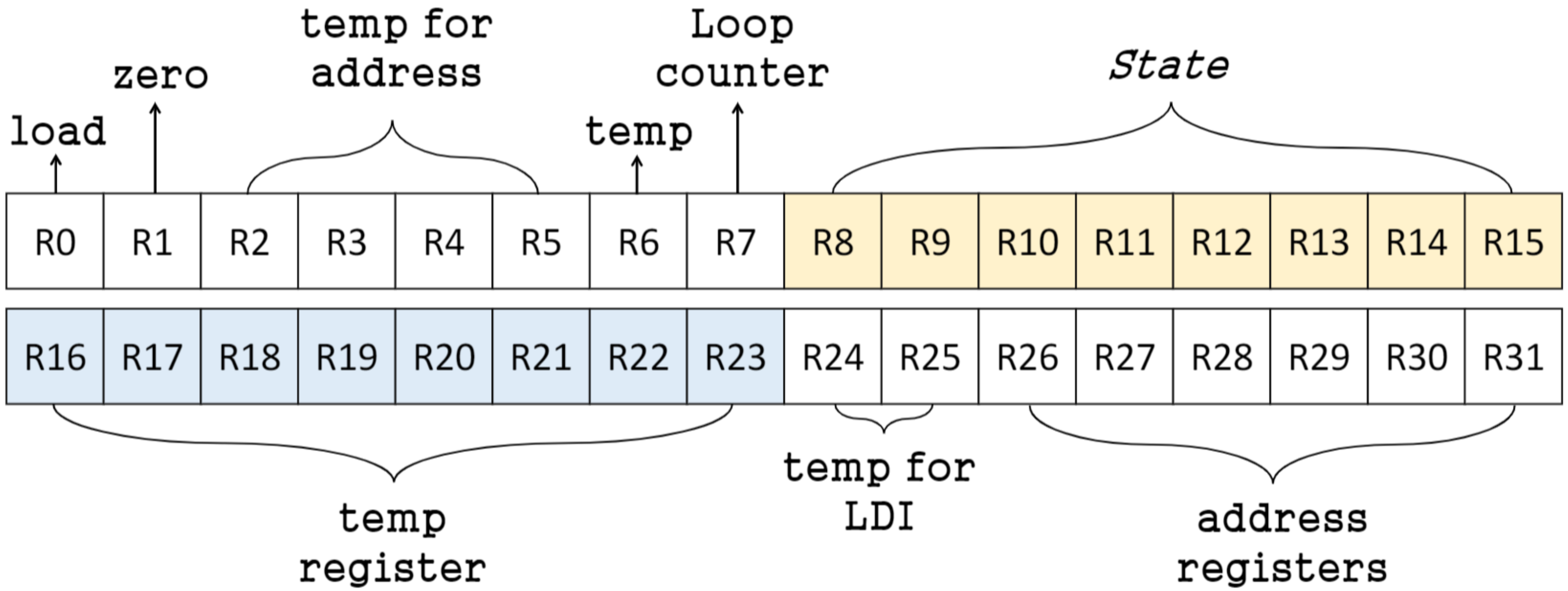


Fig. 9: Register Scheduling for Proposed Method in 8-bit AVR MCUs

Chaining optimization methodology

- To apply $\boldsymbol{\pi}$ process implicitly, we propose a *Chaining optimization methodology* in 8-bit AVR
  - Data Load to register ($\boldsymbol{\theta}$ process) → Memory translation in register ($\boldsymbol{\pi}$ process) → Data Store to Memory ($\boldsymbol{\rho}$ process)
  - $\boldsymbol{\theta}$~$\boldsymbol{\rho}$ ($\boldsymbol{\pi}$) process uses R8-R15, R16-R23 alternately → we call it "Chain Implementation"
- $State$ (200 bytes) cannot be held in the register → operating lane unit
  - Here, memory address translation (cost α) is occurred in each process
  - Combining $\boldsymbol{\theta}$~$\boldsymbol{\rho}$ process, memory address translation cost reduced to two times (4 → 2)

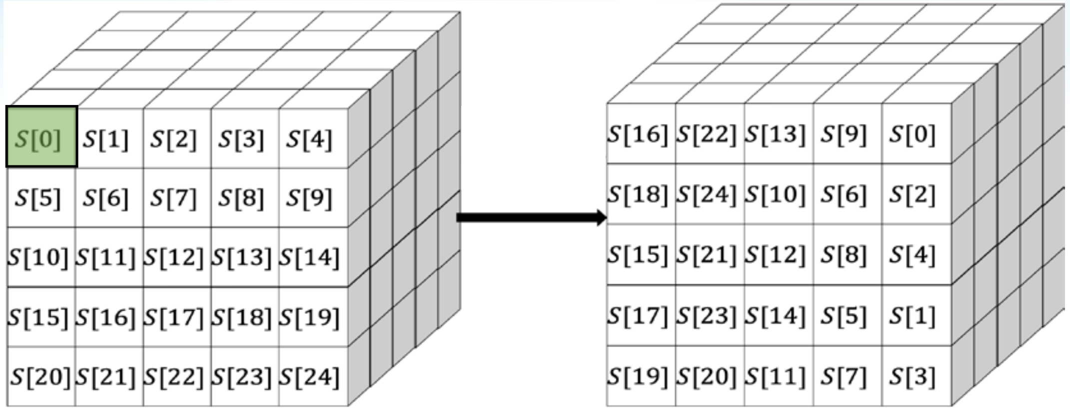
Standard Method	Initial θ	θ process	π~ρ process
Load	O + α	O + α	O + α
Store	X	O	O + α

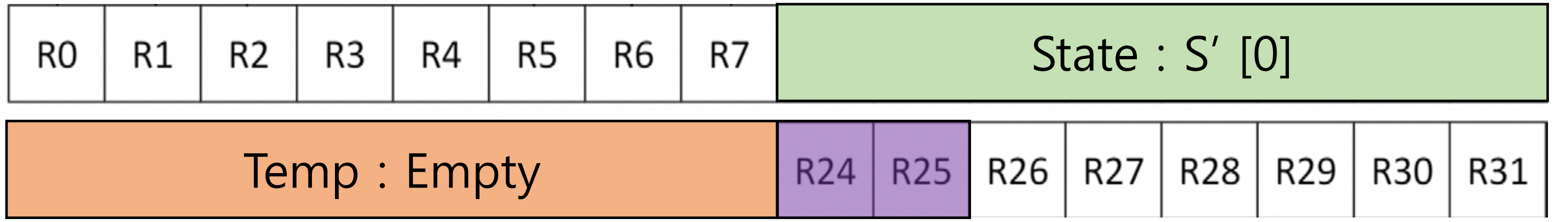
→

Proposed Method	Initial θ	θ~ρ process	π process
Load	O + α	O + α	X (Implicitly)
Store	X	O	X (Implicitly)

The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) – December 4 (Fri), 2020 | Virtual Conference
Korea Institute of Information Security & Cryptology

Chaining optimization methodology





$S[4] \leftarrow \bar{S}[0]$ computation

```
load_state
    //R8-R15 : S'[0]←(S[0]⊕D[0])
LDI R24, 32 // S[4]
LDI R25, 32 // D[4]
load_temp
    //R16-R23 : S'[4]←(S[4]⊕D[4])
rotate_store_s //S[4]←S̄[0]
```

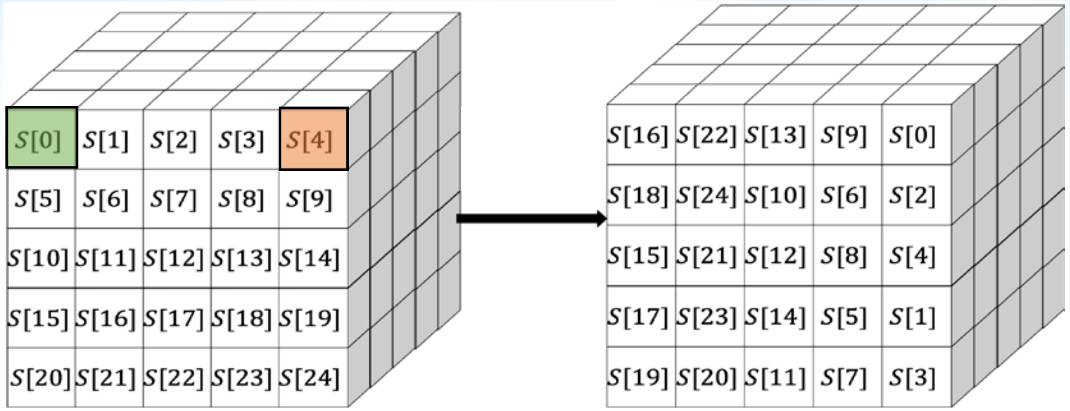
$S[14] \leftarrow \bar{S}[4]$ computation

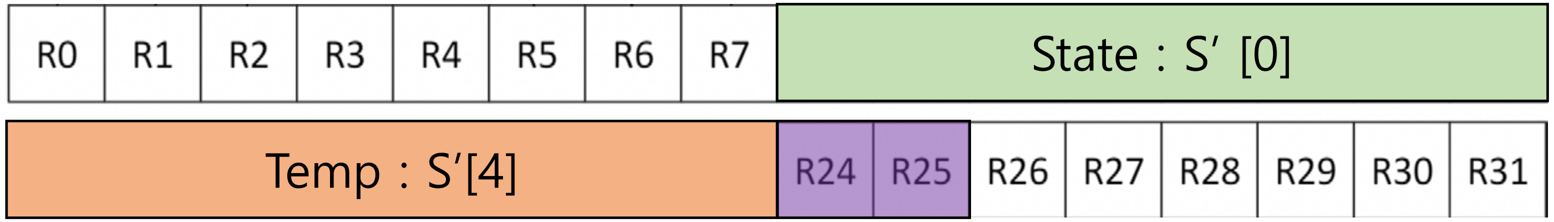
```
LDI R24, 112 // S[14]
LDI R25, 32 // D[4]
load_state
    //R8-R15 : S'[14]←(S[14]⊕D[4])
rotate_store_t // S[14]←S̄[4]
```

Fig. 11: Proposed Implementation

The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) – December 4 (Fri), 2020 | Virtual Conference
Korea Institute of Information Security & Cryptology

Chaining optimization methodology





$S[4] \leftarrow \bar{S}[0]$ computation

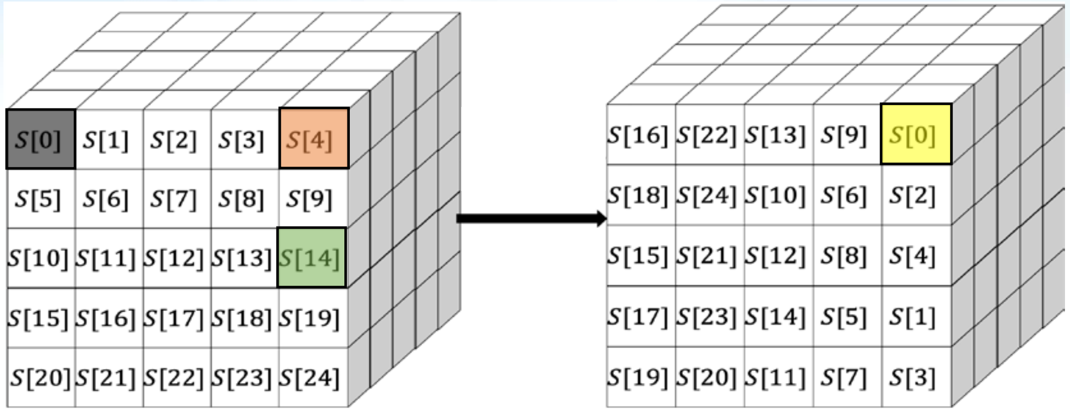
1: load_state
//R8-R15 : $S'[0] \leftarrow (S[0] \oplus D[0])$
2: LDI $R24, 32$ // $S[4]$
3: LDI $R25, 32$ // $D[4]$
4: load_temp
//R16-R23 : $S'[4] \leftarrow (S[4] \oplus D[4])$
5: rotate_store_s //$S[4] \leftarrow S[0]$

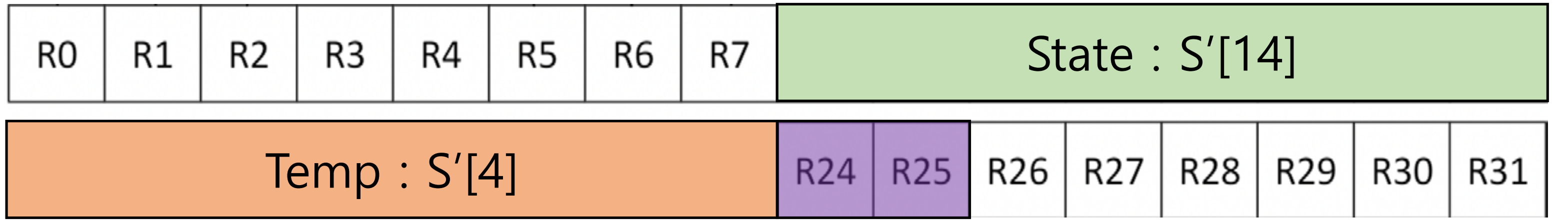
$S[14] \leftarrow \bar{S}[4]$ computation

6: LDI $R24, 112$ // $S[14]$
7: LDI $R25, 32$ // $D[4]$
8: load_state
//R8-R15 : $S'[14] \leftarrow (S[14] \oplus D[4])$
9: rotate_store_t // $S[14] \leftarrow \bar{S}[4]$

Fig. 11: Proposed Implementation

Chaining optimization methodology





$S[4] \leftarrow \bar{S}[0]$ computation

```
load_state
    //R8-R15 : S'[0]←(S[0]⊕D[0])
LDI R24, 32 // S[4]
LDI R25, 32 // D[4]
load_temp
    //R16-R23 : S'[4]←(S[4]⊕D[4])
rotate_store_s  //S[4]←S̄[0]
```

$S[14] \leftarrow \bar{S}[4]$ computation

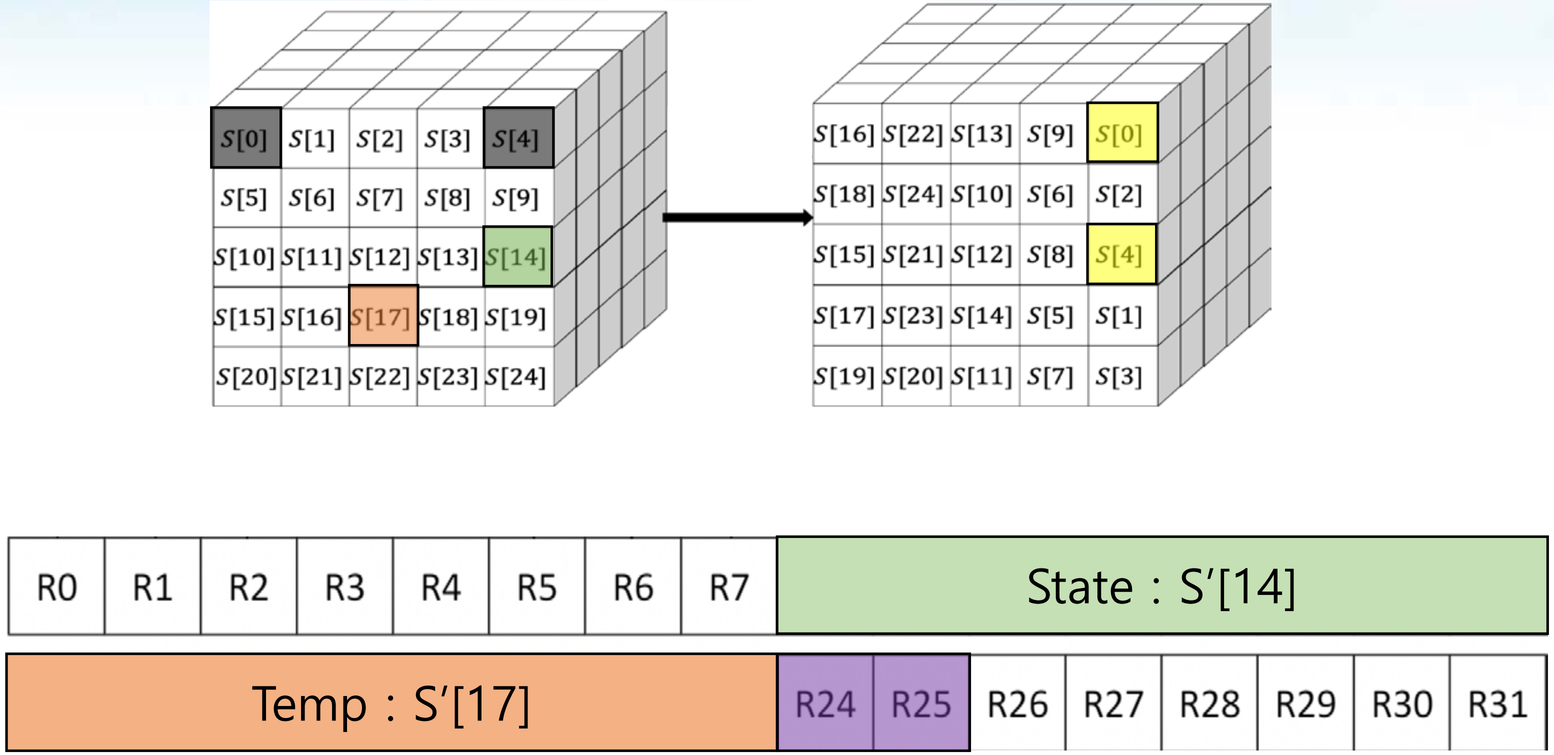
```
LDI R24, 112 // S[14]
LDI R25, 32 // D[4]
load_state
    //R8-R15 : S'[14]←(S[14]⊕D[4])
rotate_store_t  // S[14]←S̄[4]
```

Fig. 11: Proposed Implementation

The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) – December 4 (Fri), 2020 | Virtual Conference

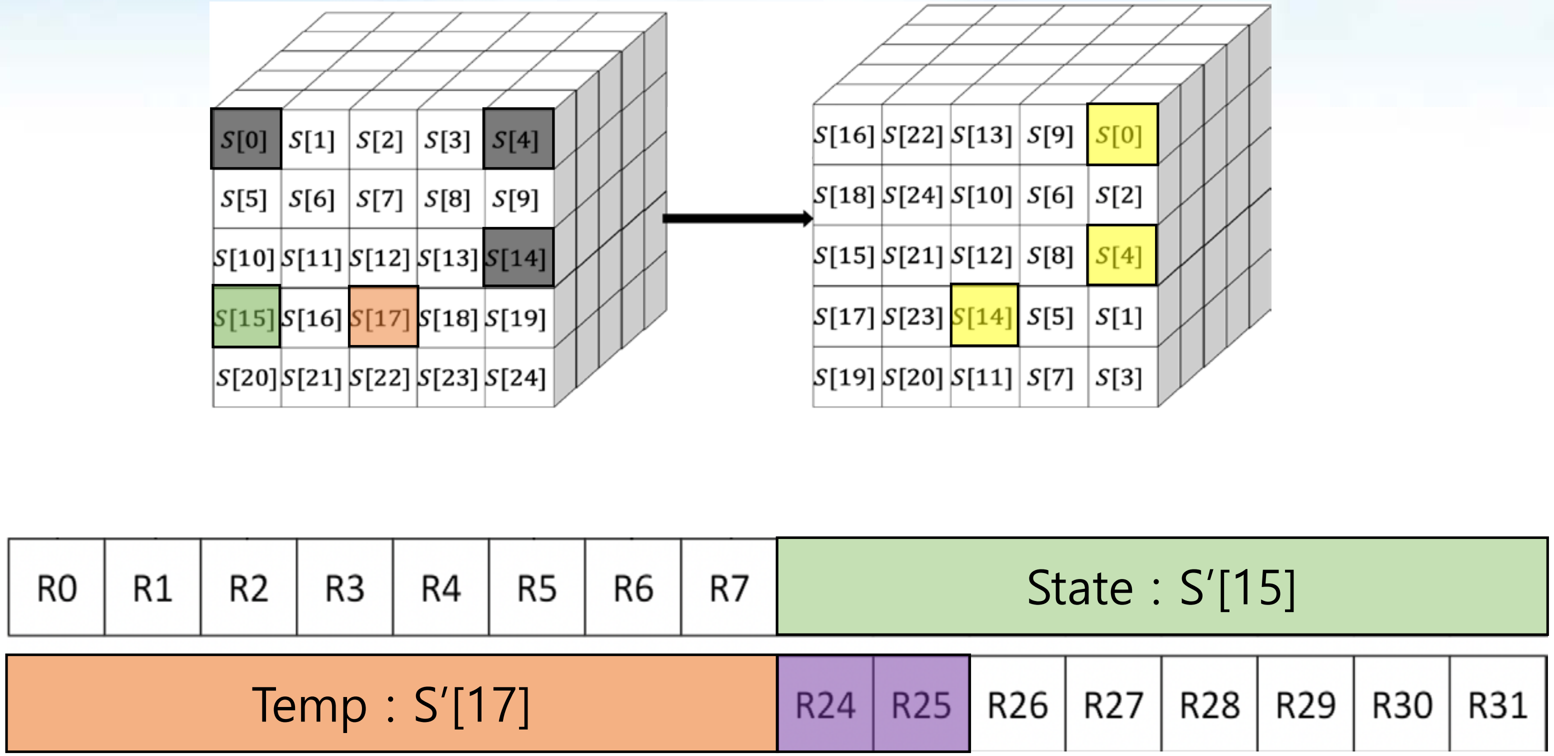



```
rotate_store_t  // S[14]←S̄[4]
    S[17]←S̄[14] computation
LDI R17, 136 // S[17]
LDI R17, 16 // D[2]
load_temp
    //R16-R23 : S′[17]←(S[17]⊕D[2])
rotate_store_s  // S[17]←S̄[14]

    S[15]← S̄[17] computation
LDI R24, 120 // S[15]
EOR R25, R25 // D[0]
load_state
    //R8-R15 : S′[15]←(S[15]⊕D[0])
rotate_store_t  // S[15]←S̄[17]
```

Fig. 11: Proposed Implementation

The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) – December 4 (Fri), 2020 | Virtual Conference

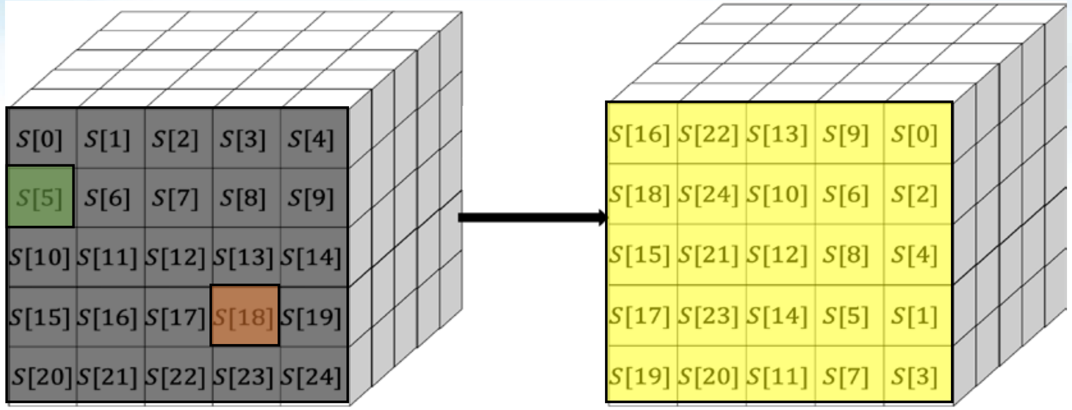


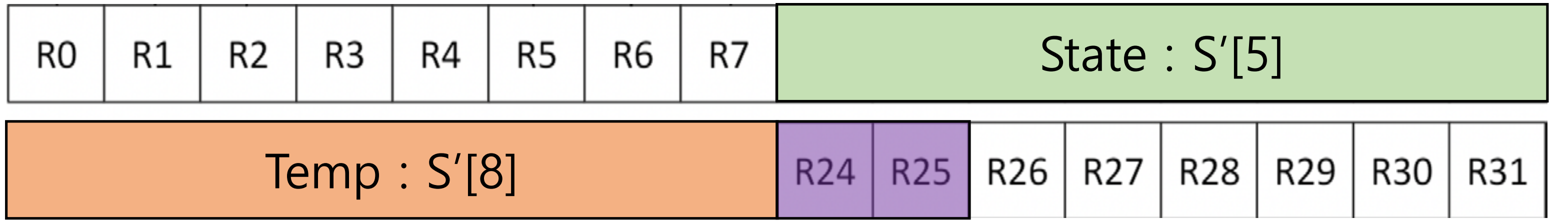

```
rotate_store_t // S[14]←S̄[4]
   S[17]←S̄[14] computation
LDI R17, 136 // S[17]
LDI R17, 16 // D[2]
load_temp
   //R16-R23 : S′[17]←(S[17]⊕D[2])
rotate_store_s // S[17]←S̄[14]

   S[15]← S̄[17] computation
LDI R24, 120 // S[15]
EOR R25, R25 // D[0]
load_state
   //R8-R15 : S′[15]←(S[15]⊕D[0])
rotate_store_t // S[15]←S̄[17]
```

Fig. 11: Proposed Implementation

Chaining optimization methodology





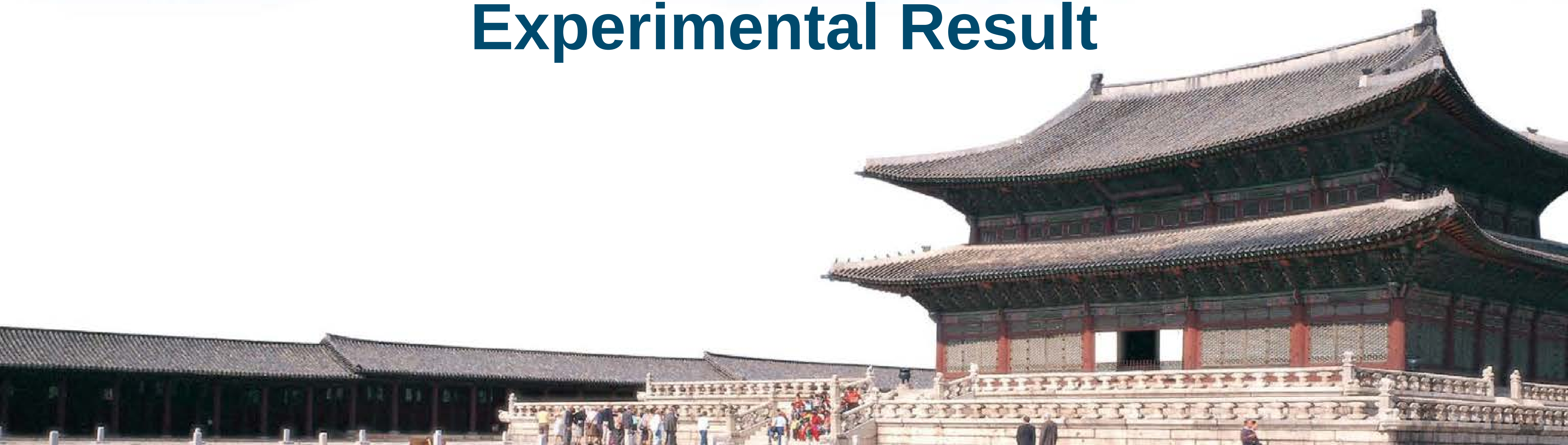
The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Experimental Result



Experimental Result

- 25.7% performance improvement over Balasch et al.'s implementation
- Our Work is the fastest implementation of SHA-3 in 8-bit AVR microcontroller
- Narrowing the difference in performance by about two times compared to the SHA-2 Family
  - Existing implementations have nearly three times the difference in performance

Reference	Algorithm	Language	Length of message byte		
			50 byte	100 byte	500 byte
This Work	SHA-3 (256-bit)	**Asm**	2667 (+25.1%)	1333 (+25.7%)	1073 (+25.0%)
Otte et al.	SHA-3 (256-bit)	C, Asm	12854	6427	1672
Balasch et al.	SHA-3 (256-bit)	Asm	3560 (-)	1795 (-)	1432 (-)
Balasch et al.	SHA-256	Asm	672	668	532
Balasch et al.	Blake (256-bit)	Asm	714	708	562
Balasch et al.	Photon (256-bit)	Asm	9723	7892	4788

Table. 3: Performance of SHA-3 by hash rate (*CPB*), when hashing a byte of various message in 8-bit AVR

The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference
Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Conclusion

Conclusion

- We introduced a new generic fast implementation method of SHA-3
- Proposed Method not requires a lookup table or additional operations
- Proposed Chaining optimization methodology of SHA-3 is the fastest implementation
- Our Work is efficiently applicable in PQC, DRBG, MAC, and so on
- Our Work is a generic method that can be a applied to various platforms

The 23rd Annual International Conference on Information Security and Cryptology

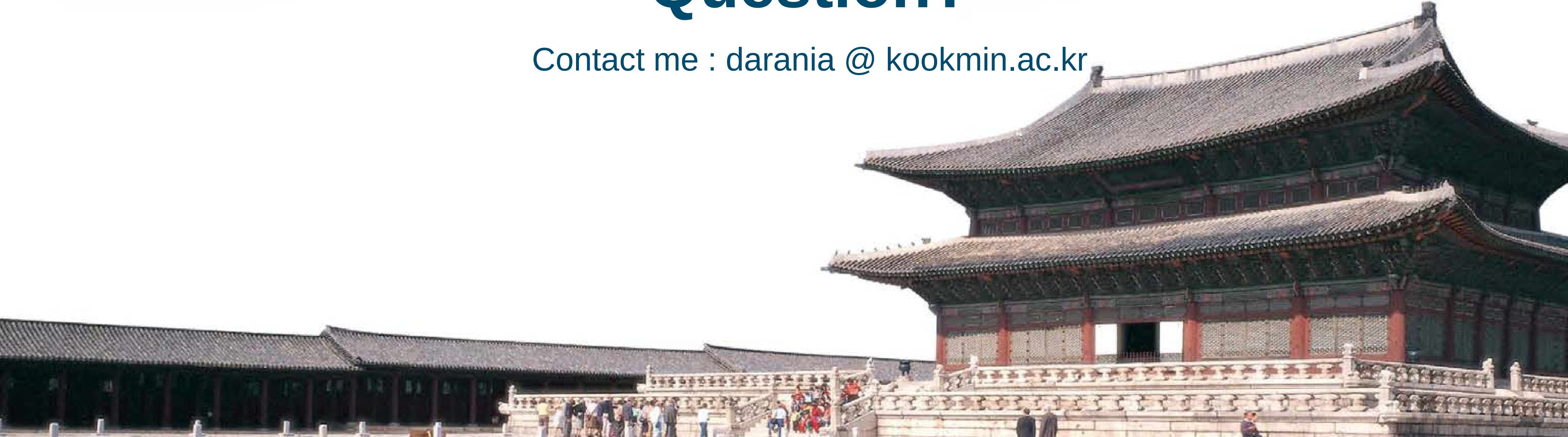
ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Question?

Contact me : darania @ kookmin.ac.kr



The 23rd Annual International Conference on Information Security and Cryptology

ICISC 2020

December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Korea Institute of Information Security & Cryptology

Thank You~

Cryptography Optimization & Application Lab