Compact Implementation of CHAM Block Cipher on Low-End Microcontrollers

Hyeokdong Kwon, Hyunji Kim, Seung Ju Choi, Kyoungbae Jang, Jaehoon Park, Hyunjun Kim, and Hwajeong Seo

korlethean@gmail.com

Hansung University

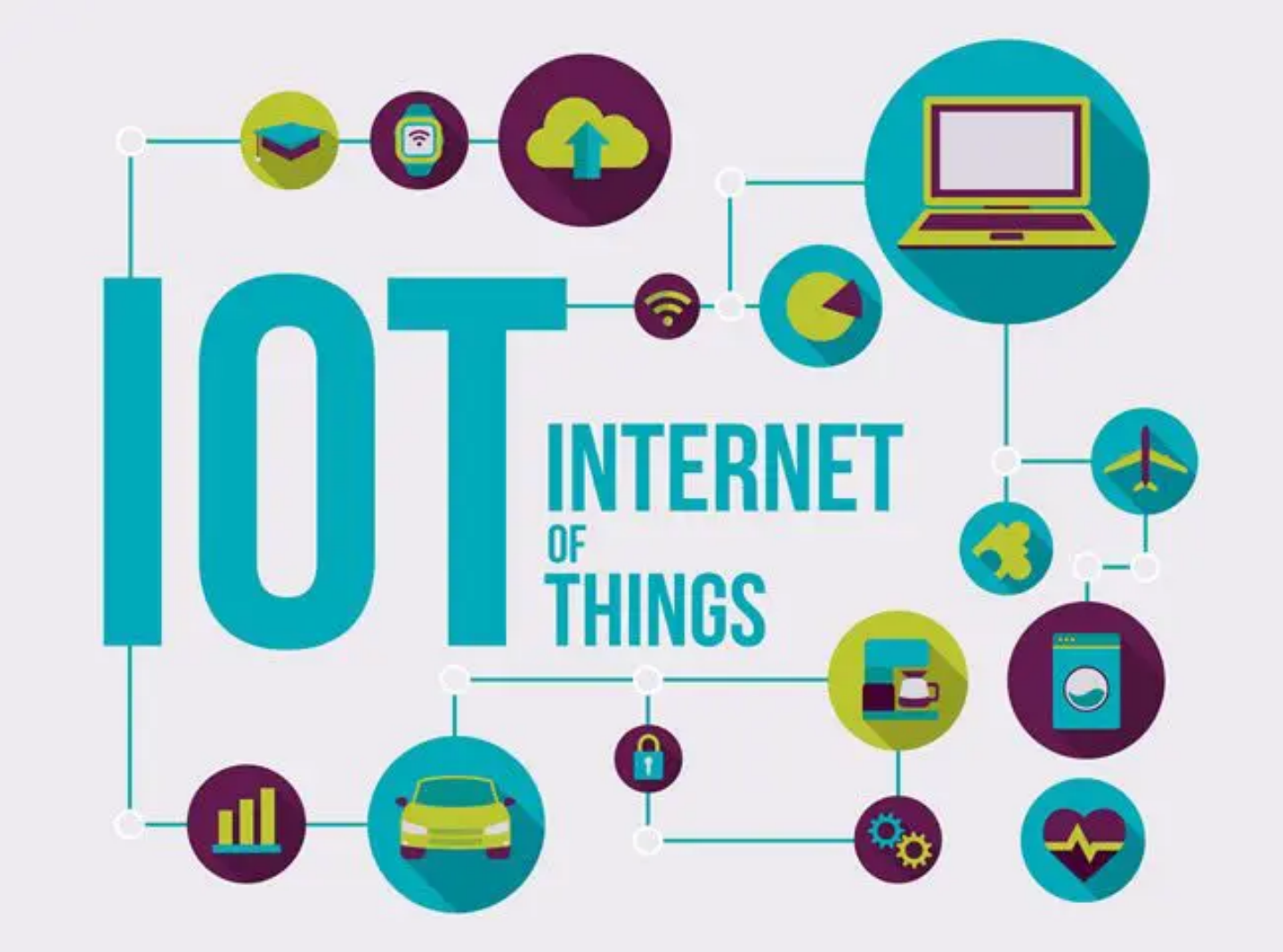
HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

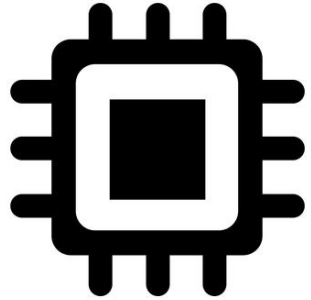
CryptoCraft LAB

Overview



Overview

- Lightweight Block Cipher



Less computing power

Little memory usage

Fast execution time

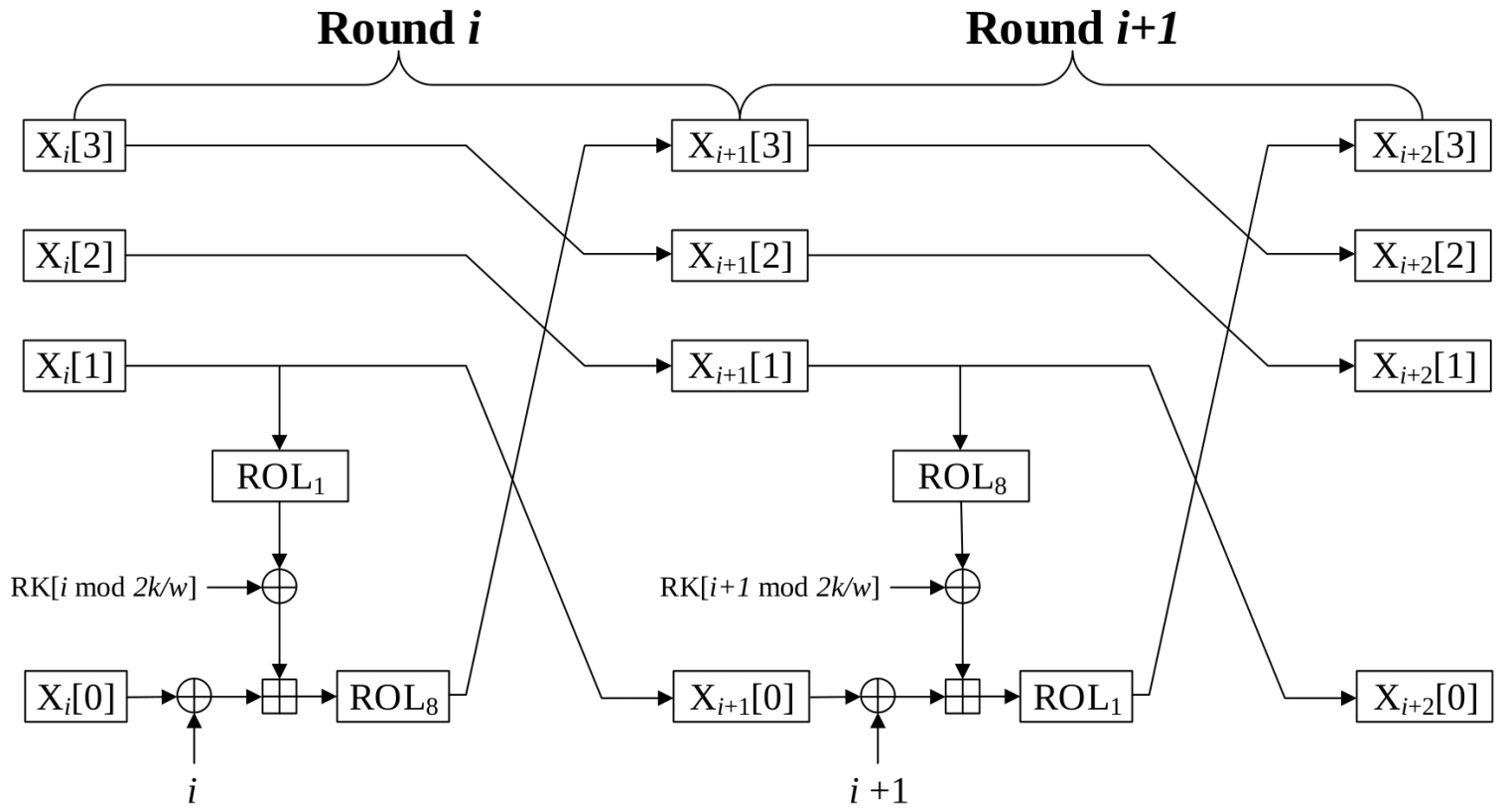
CHAM Block Cipher

- [ICISC'17]: CHAM proposed from NSR(National Security Research)
- [ICISC'19]: Revised CHAM proposed
  - For convenience, revised CHAM is referred to as CHAM

Cipher	n	k	rk	r	r(old)
CHAM-64/128	64	128	16	88	80
CHAM-128/128	128	128	32	112	80
CHAM-128/256	128	256	32	120	96

CHAM Block Cipher

- 4-branch Feistel architecture
- ARX based

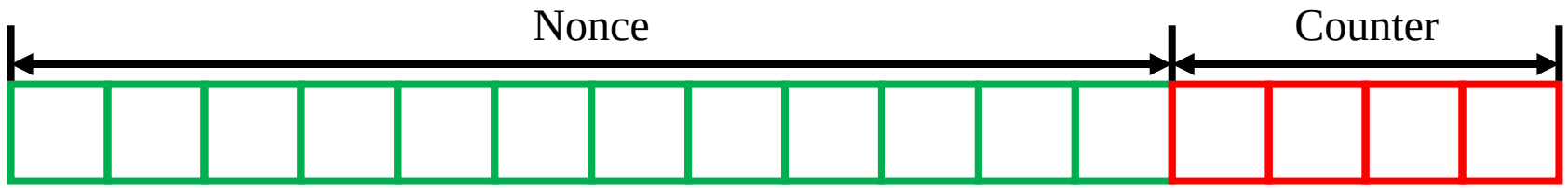


Proposed Implementation

- The optimized CHAM-CTR mode encryption
- On 8-bit AVR microcontrollers
- 2 kinds of implementations
  - **Pre-computation model**
  - **Parallel implementation**

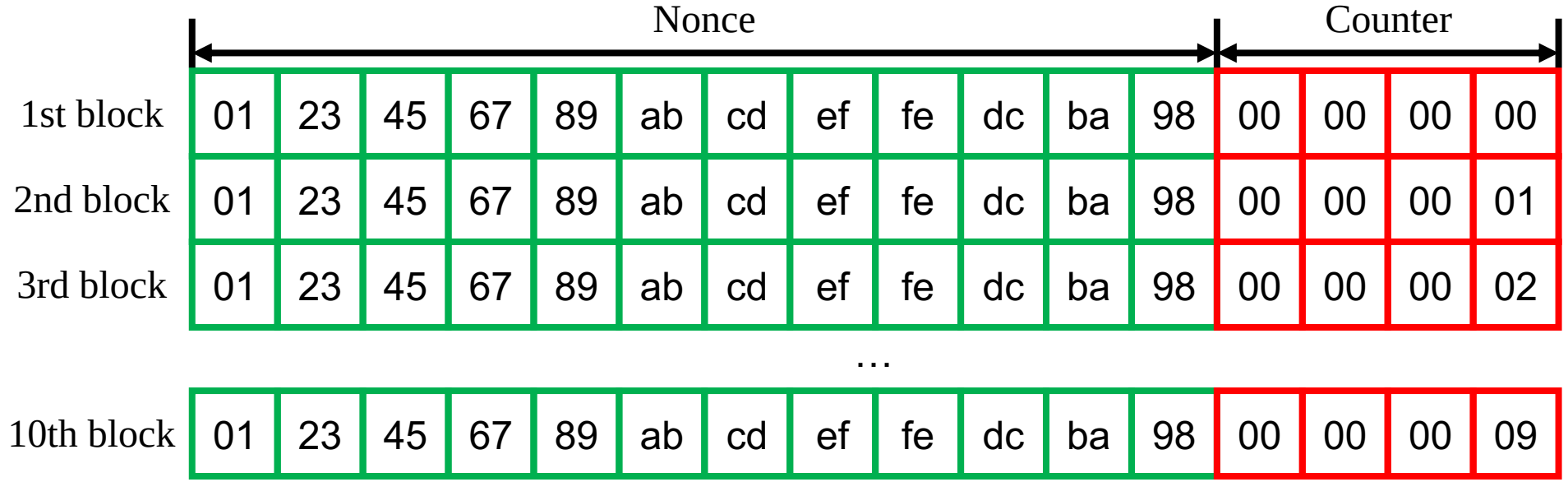
Proposed Implementation: Pre-computation

- The input values of CTR mode of operation
  - Nonce: Fixed value, 3/4 of total length
  - Counter: Variable, 1/4 of total length



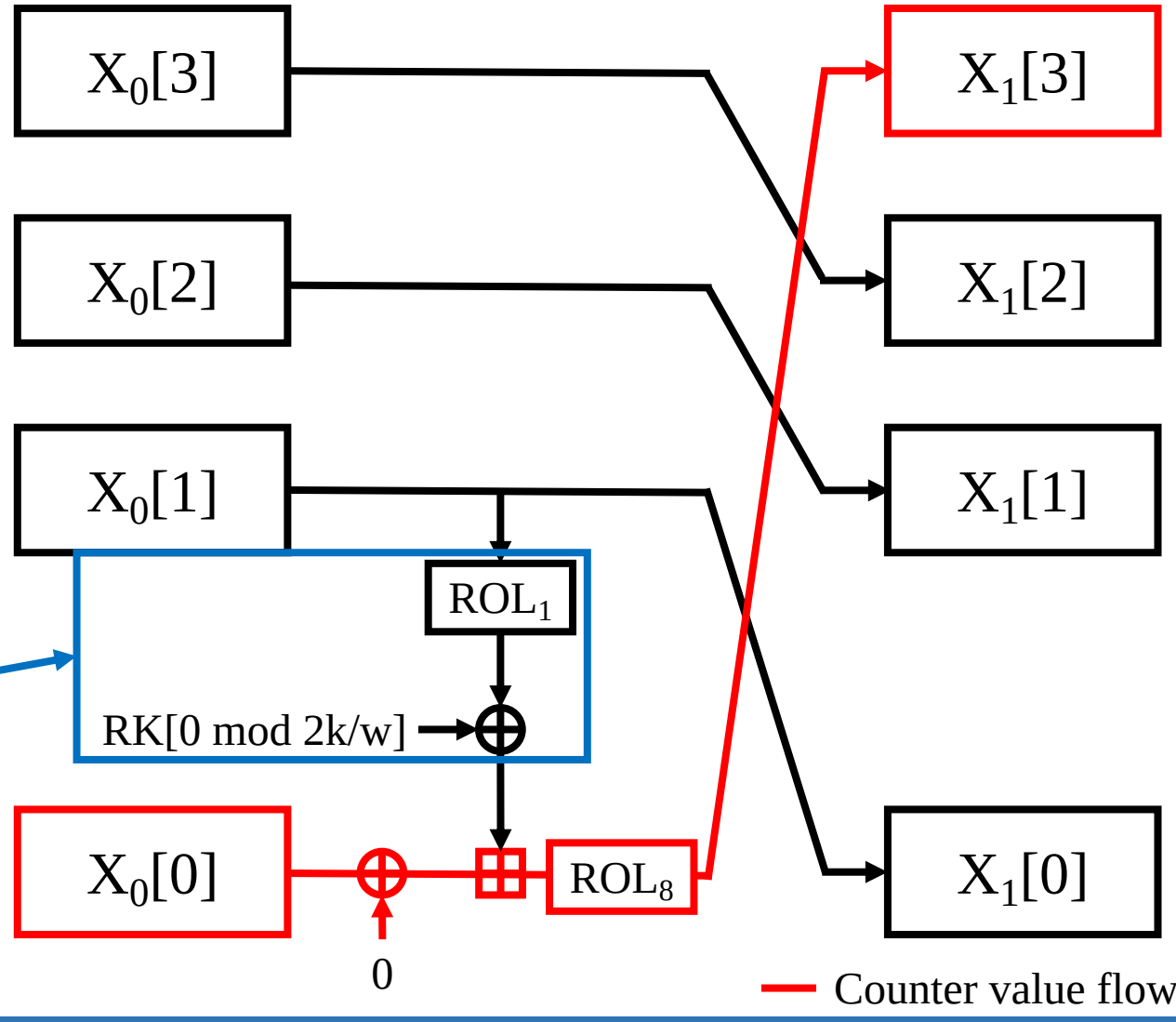
Proposed Implementation: Pre-computation

- The counter value is indicates number of block
- Each blocks has identical Nonce value
- **Calculation result of Nonce part can be pre-computation**



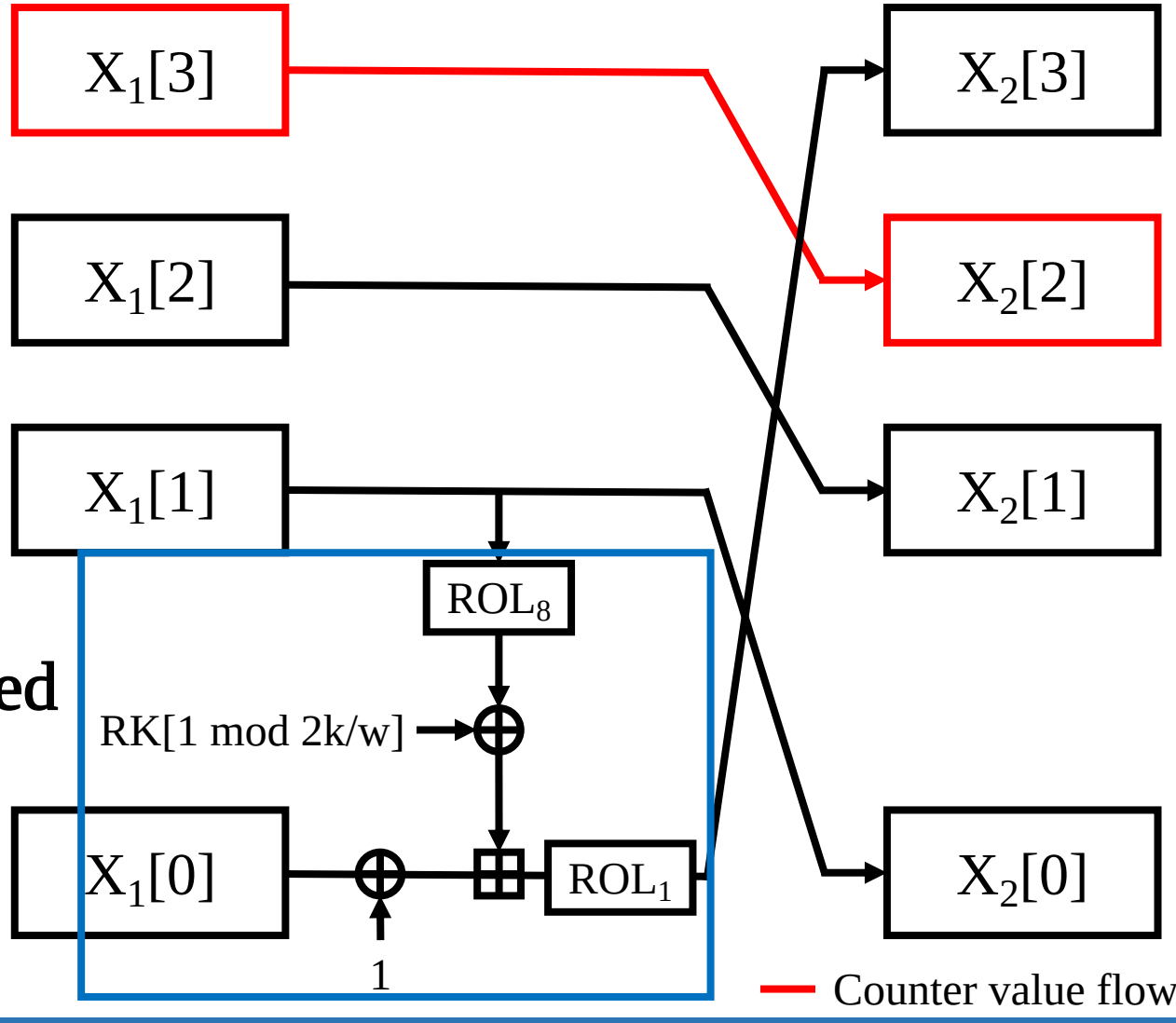
Proposed Implementation: Pre-computation

- **Round 0**
- Input values
  - $X_0[0]$: Counter
  - $X_0[1]$: Nonce
- **Calculate with Nonce, RK part can be pre-computation**
  - These values are fixed



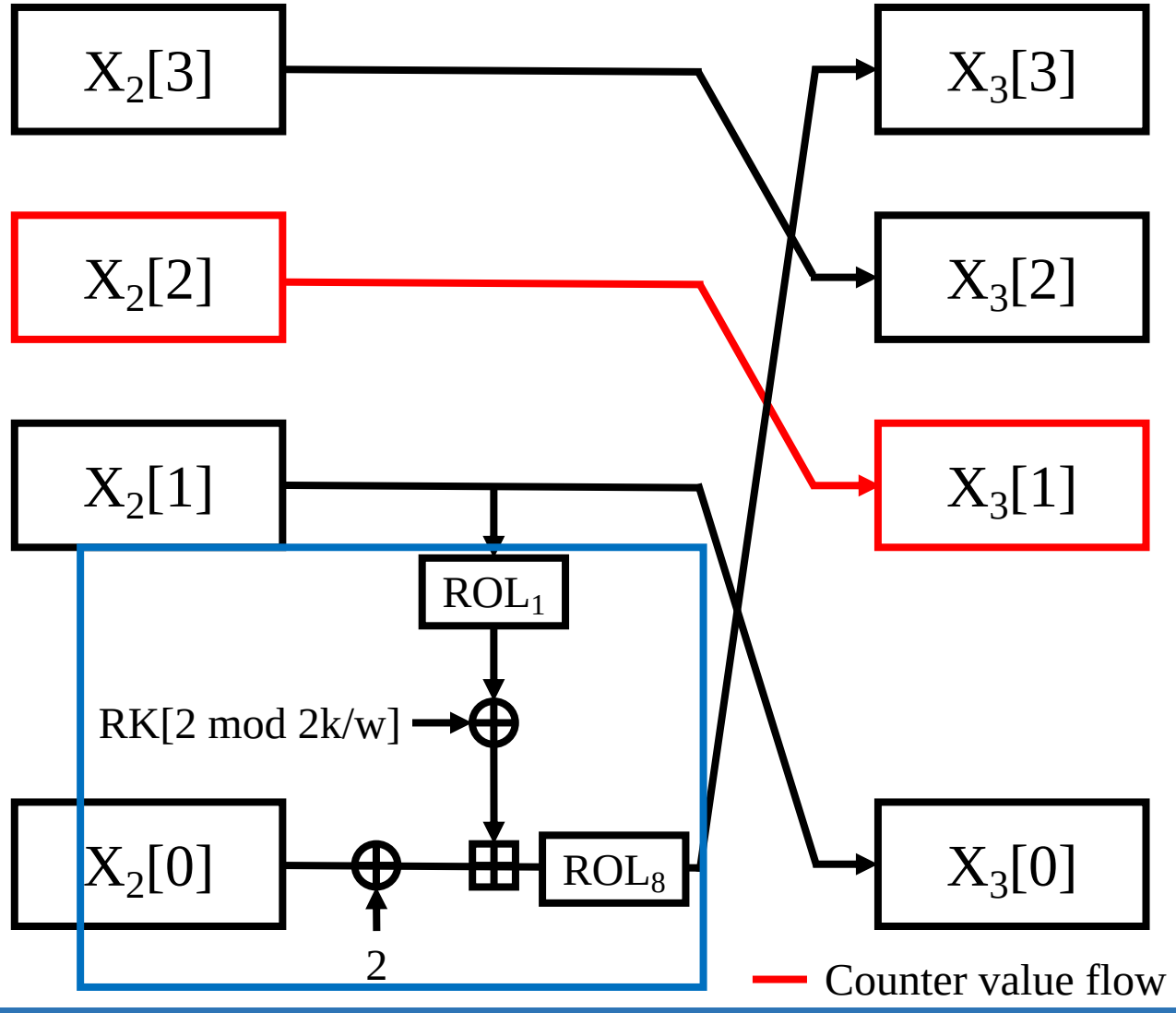
Proposed Implementation: Pre-computation

- **Round 1**
- Input values
  - $X_1[0]$: Nonce
  - $X_1[1]$: Nonce
- **Whole instructions are can be skipped**



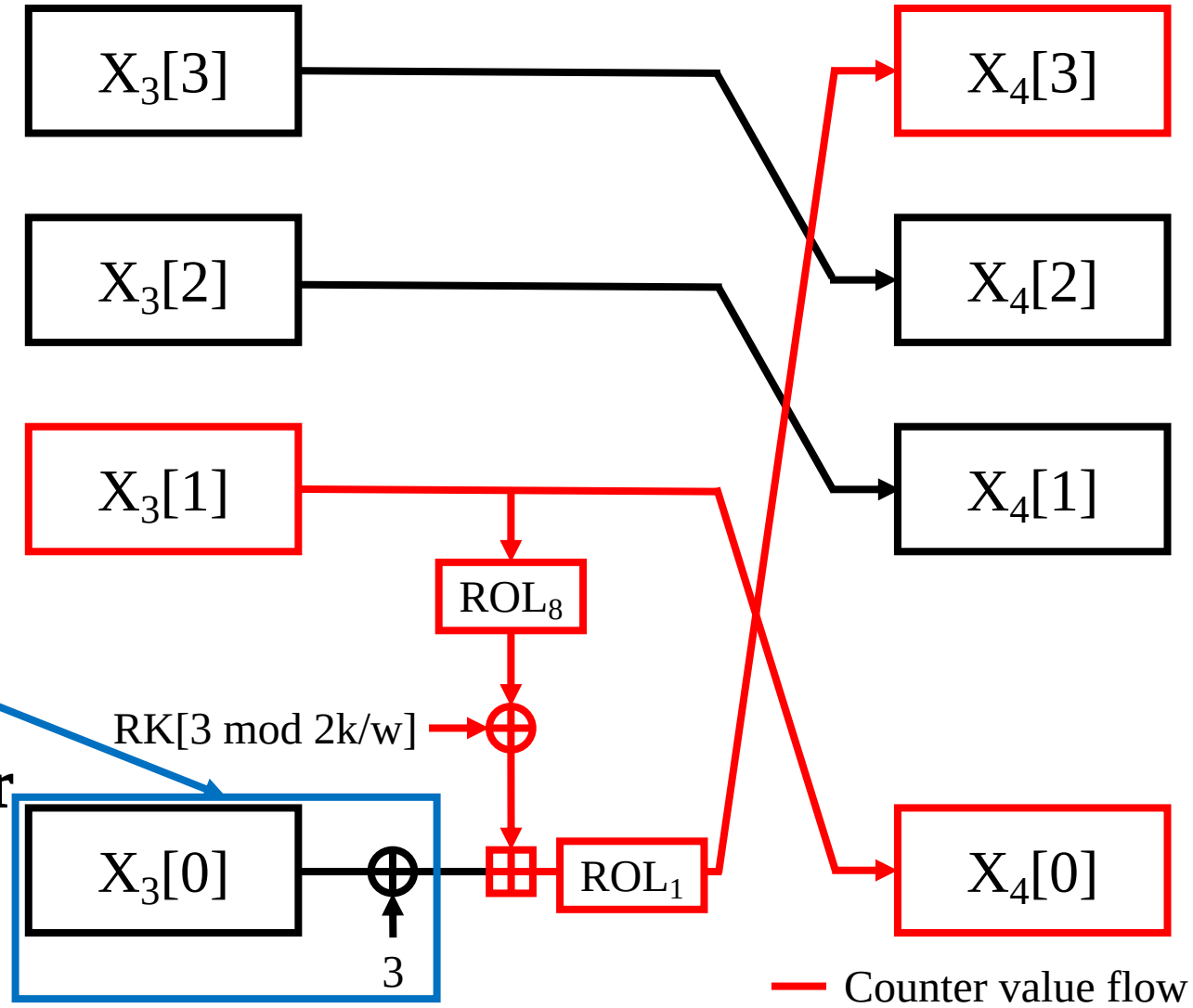
Proposed Implementation: Pre-computation

- **Round 2**
- Input values
  - $X_2[0]$: Nonce
  - $X_2[1]$: Nonce
- **Same as Round 1**



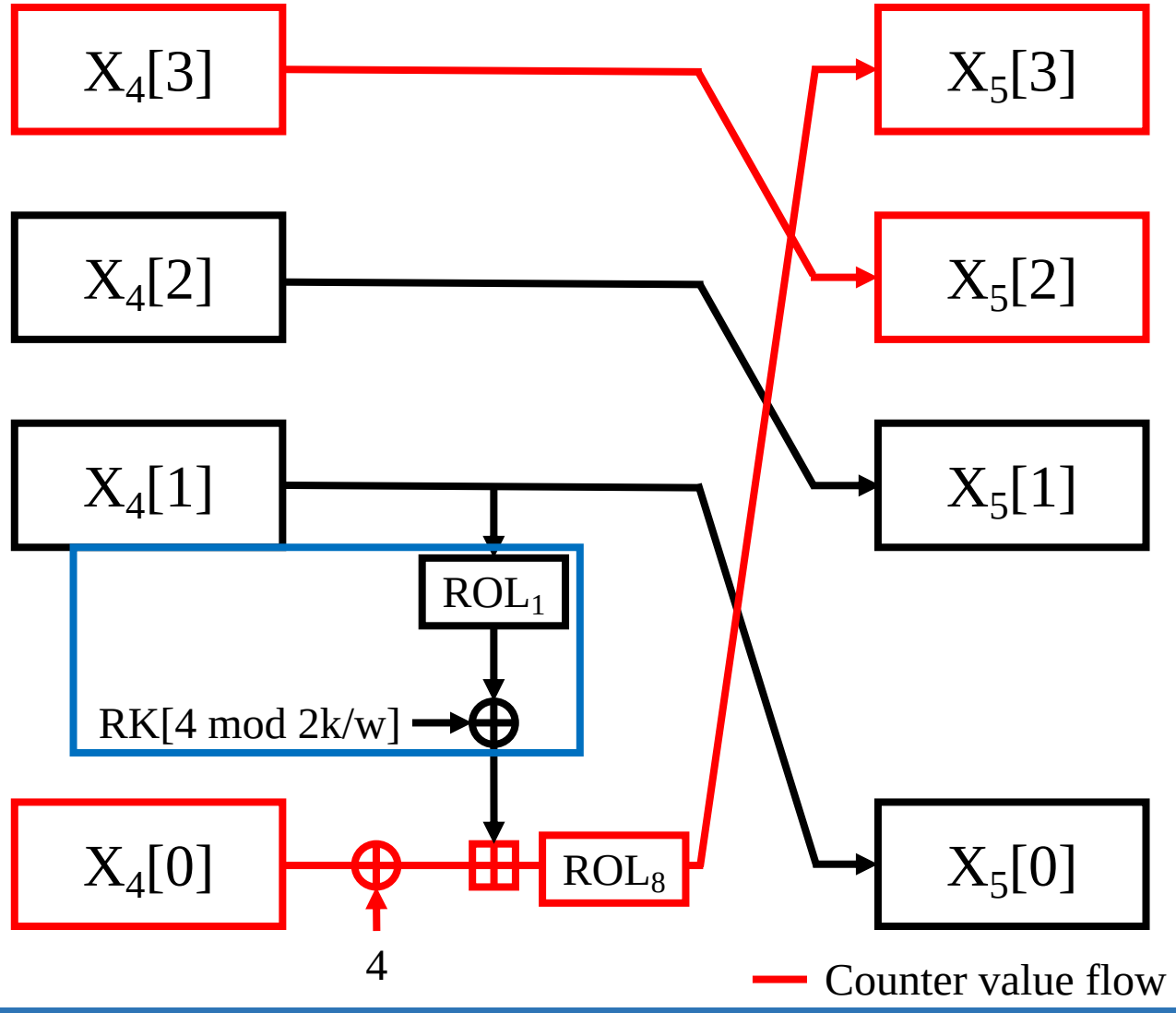
Proposed Implementation: Pre-computation

- **Round 3**
- Input values
  - $X_3[0]$: Nonce
  - $X_3[1]$: Counter
- **$X_3[0]$ ^ (Round Counter) part skip**
- **One more block affected by counter**



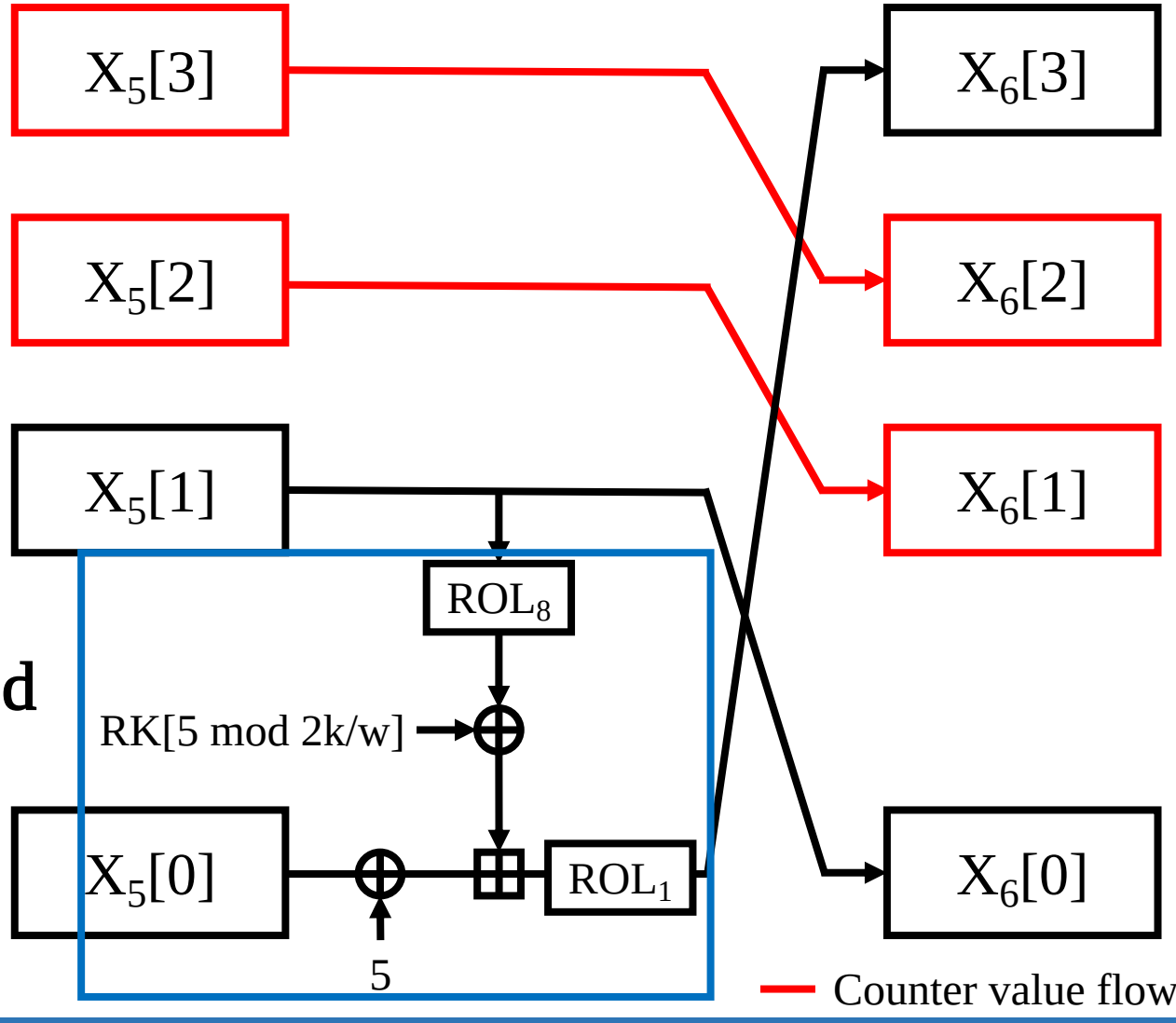
Proposed Implementation: Pre-computation

- **Round 4**
- Input values
  - $X_4[0]$: Counter
  - $X_4[1]$: Nonce
- **Same as Round 0**



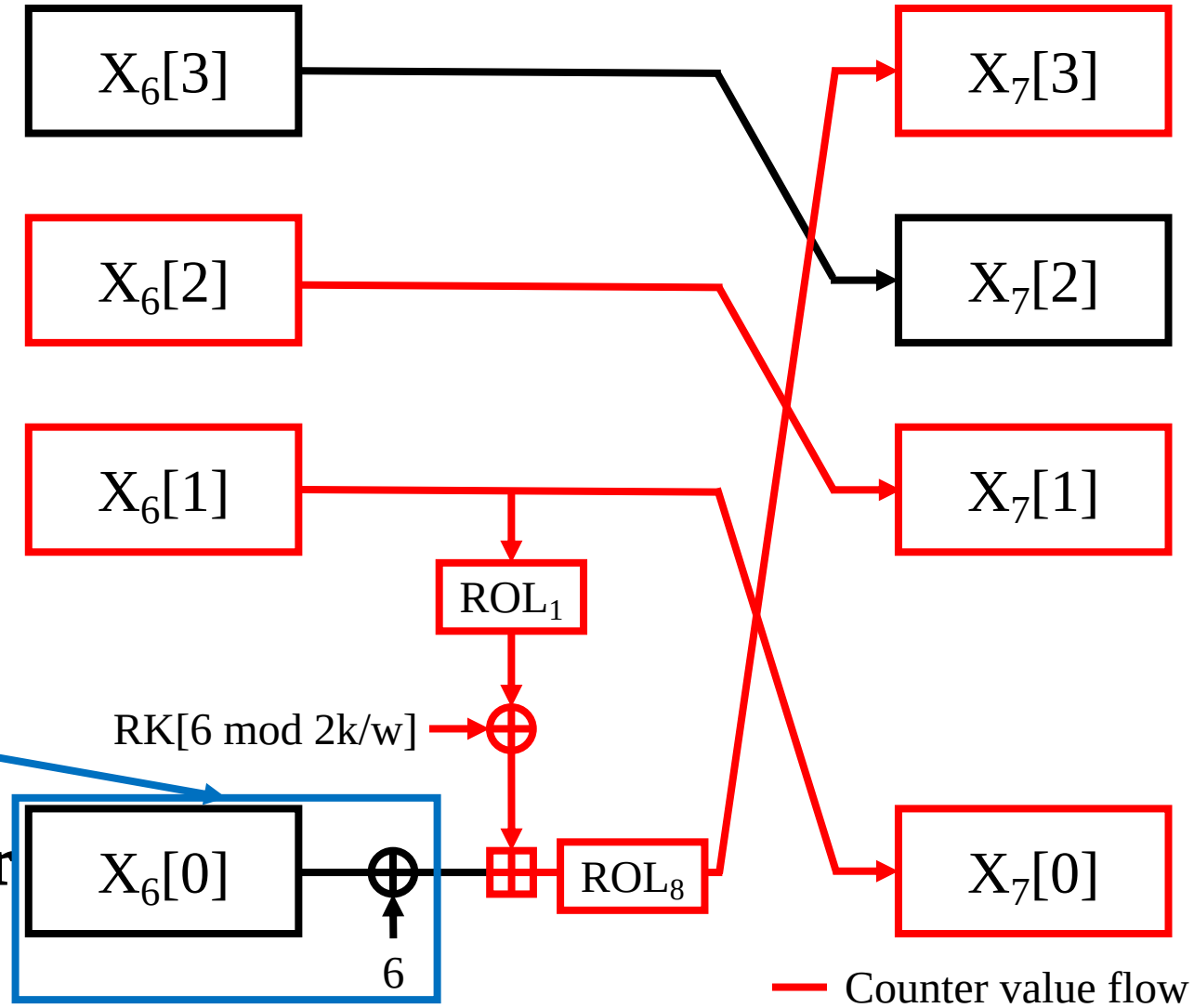
Proposed Implementation: Pre-computation

- **Round 5**
- Input values
  - $X_5[0]$: Nonce
  - $X_5[1]$: Nonce
- **All of instructions are can be skipped**
  - Same as Round 1, 2



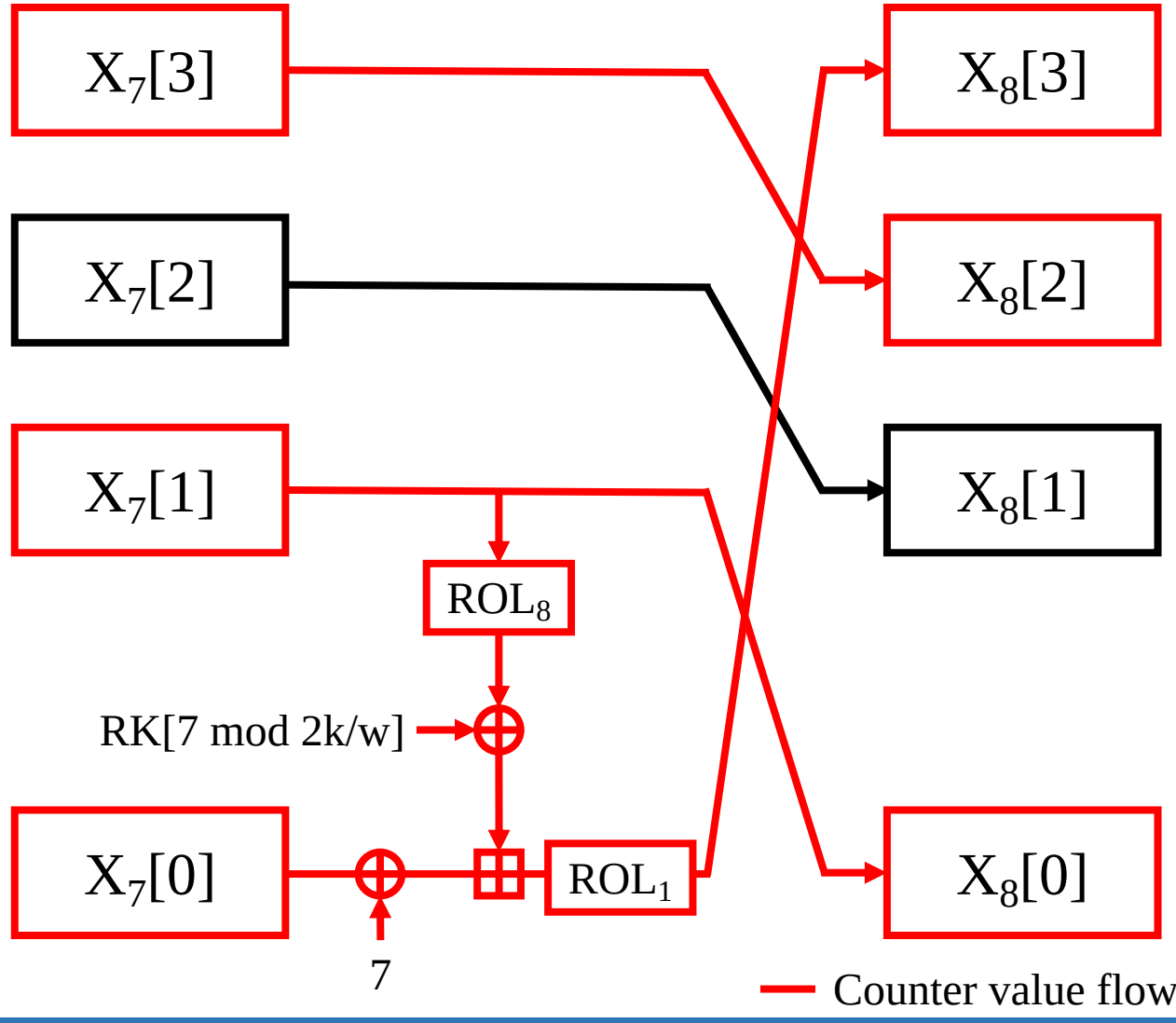
Proposed Implementation: Pre-computation

- **Round 6**
- Input values
  - $X_6[0]$: Nonce
  - $X_6[1]$: Counter
- **XOR with Round Counter part can be pre-computation**
- **One more block affected by counter**



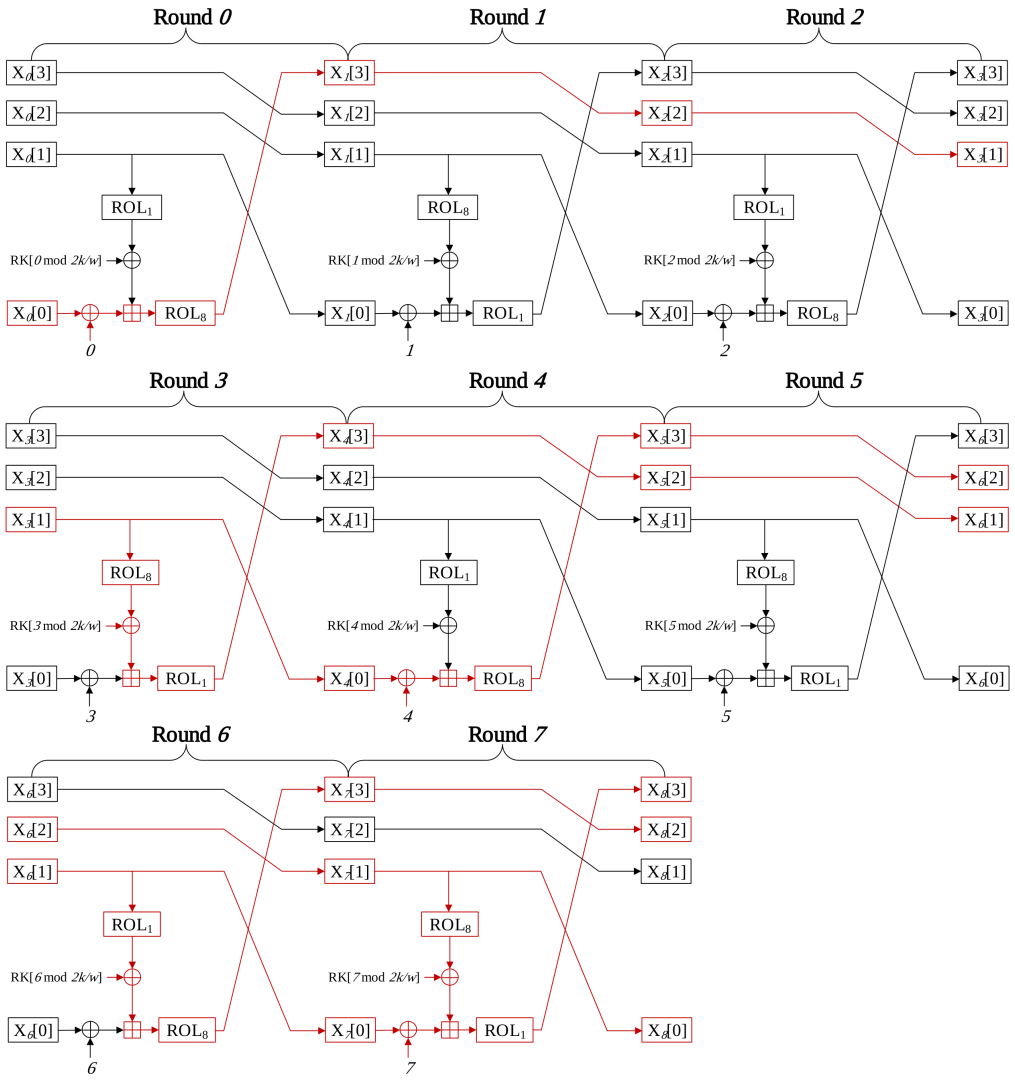
Proposed Implementation: Pre-computation

- **Round 7**
- Input values
  - $X_7[0]$: Counter
  - $X_7[1]$: Counter
- **Whole operations are must be implemented**



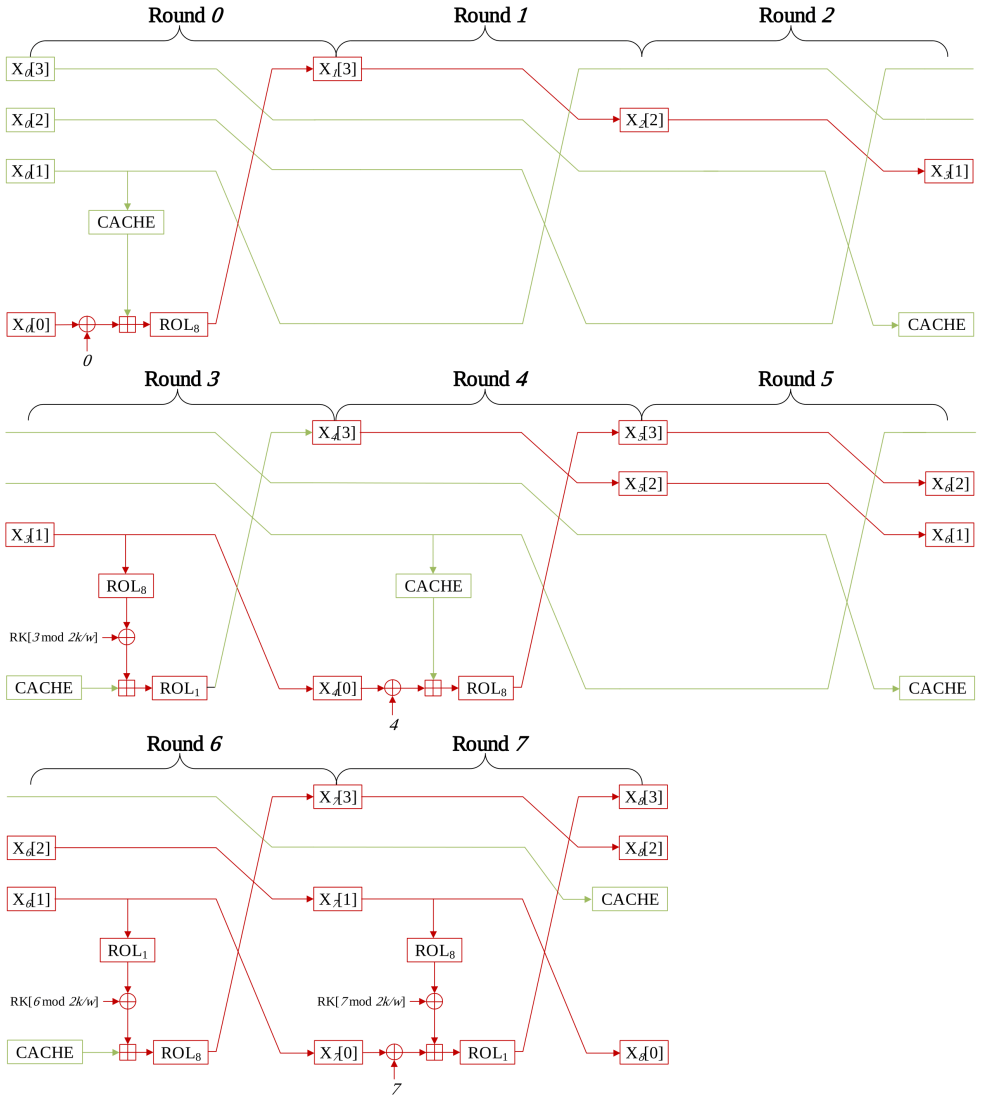
Proposed Implementation: Pre-computation

- Original CHAM structure
  - Red line: Counter value flow
- Pre-computation part can be skipped



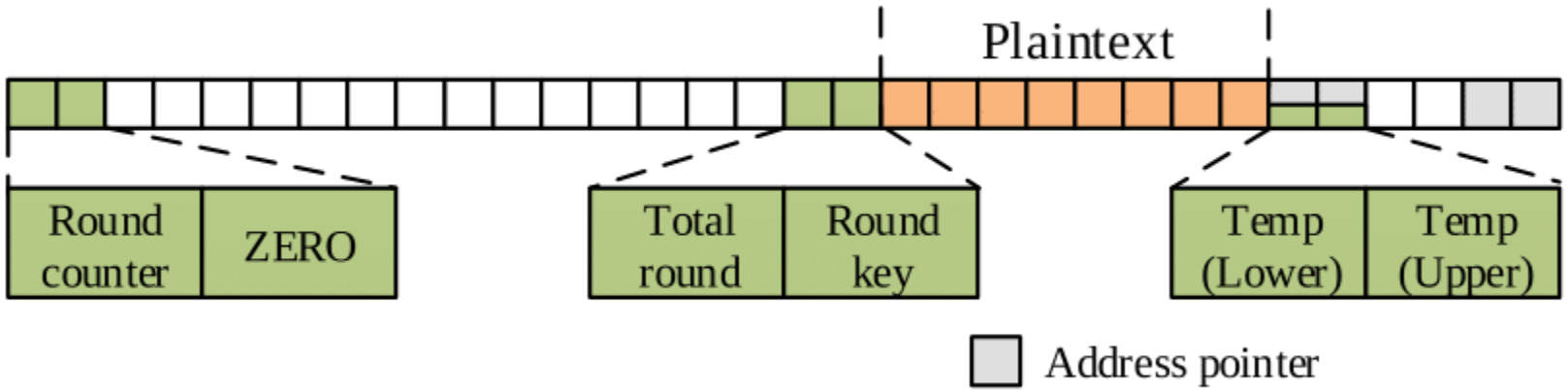
Proposed Implementation: Pre-computation

- **Optimized CHAM structure**
  - **Some instructions are removed**
    - 5 ROL(1-bit)
    - 3 ROL(word-wise)
    - 10 XOR
    - 3 ADD
  - **Result is load from look-up table**



Proposed Implementation: Parallel implementation

- CHAM-64/128 has unused registers
- Utilize these registers for Parallel implementation

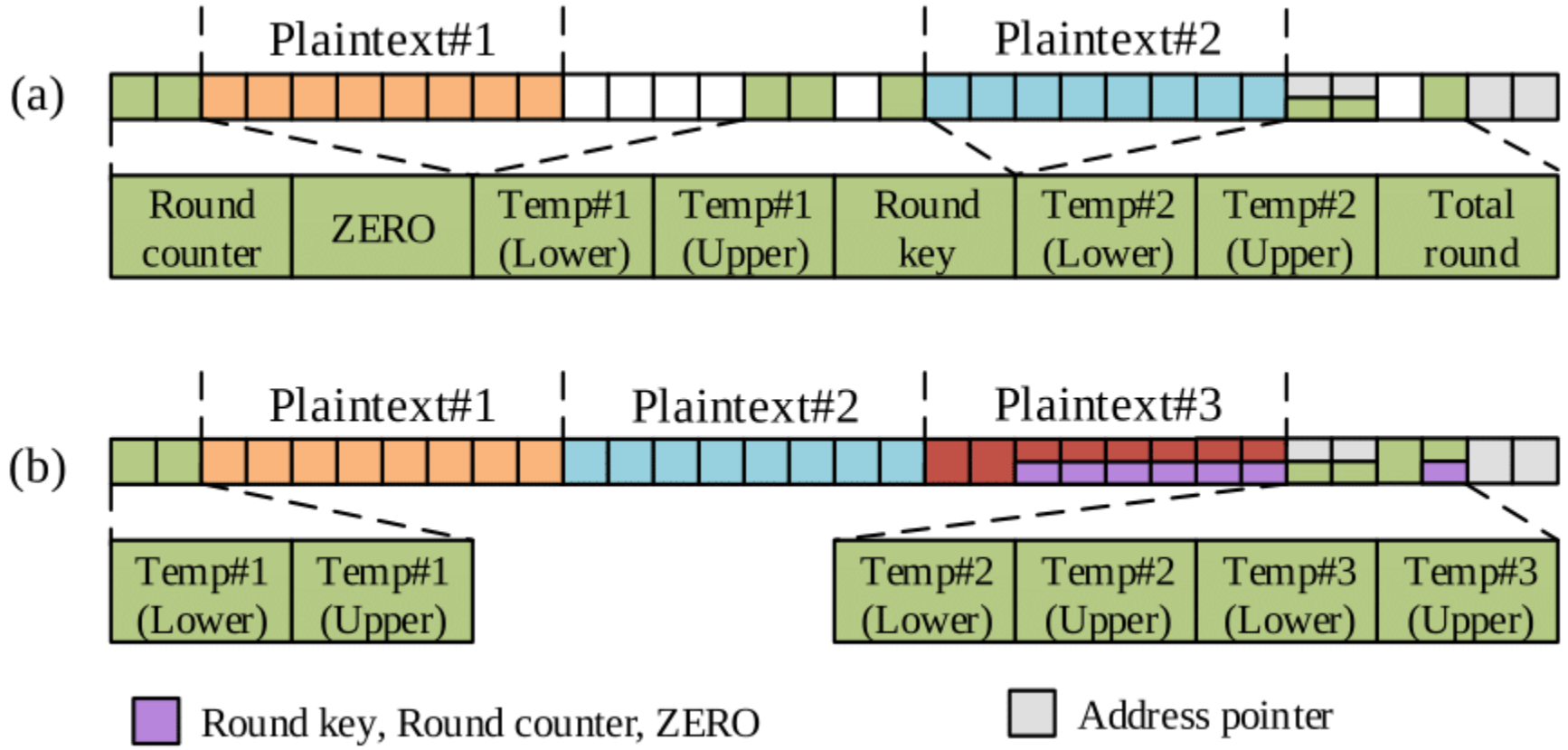


Proposed Implementation: Parallel implementation

Value	Description
Total round variable	Using CPI instruction, total round value is not maintained in the register
Plaintext block	Temporarily stored in STACK memory
Address pointer	Stored in STACK, since pointer is not used throughout computations
Round key	By accessing byte by byte, only single register is utilized
Round counter	Saved in STACK and only loaded when performing XOR operation
ZERO	In case of 3-parallel, some registers are temporarily initialized

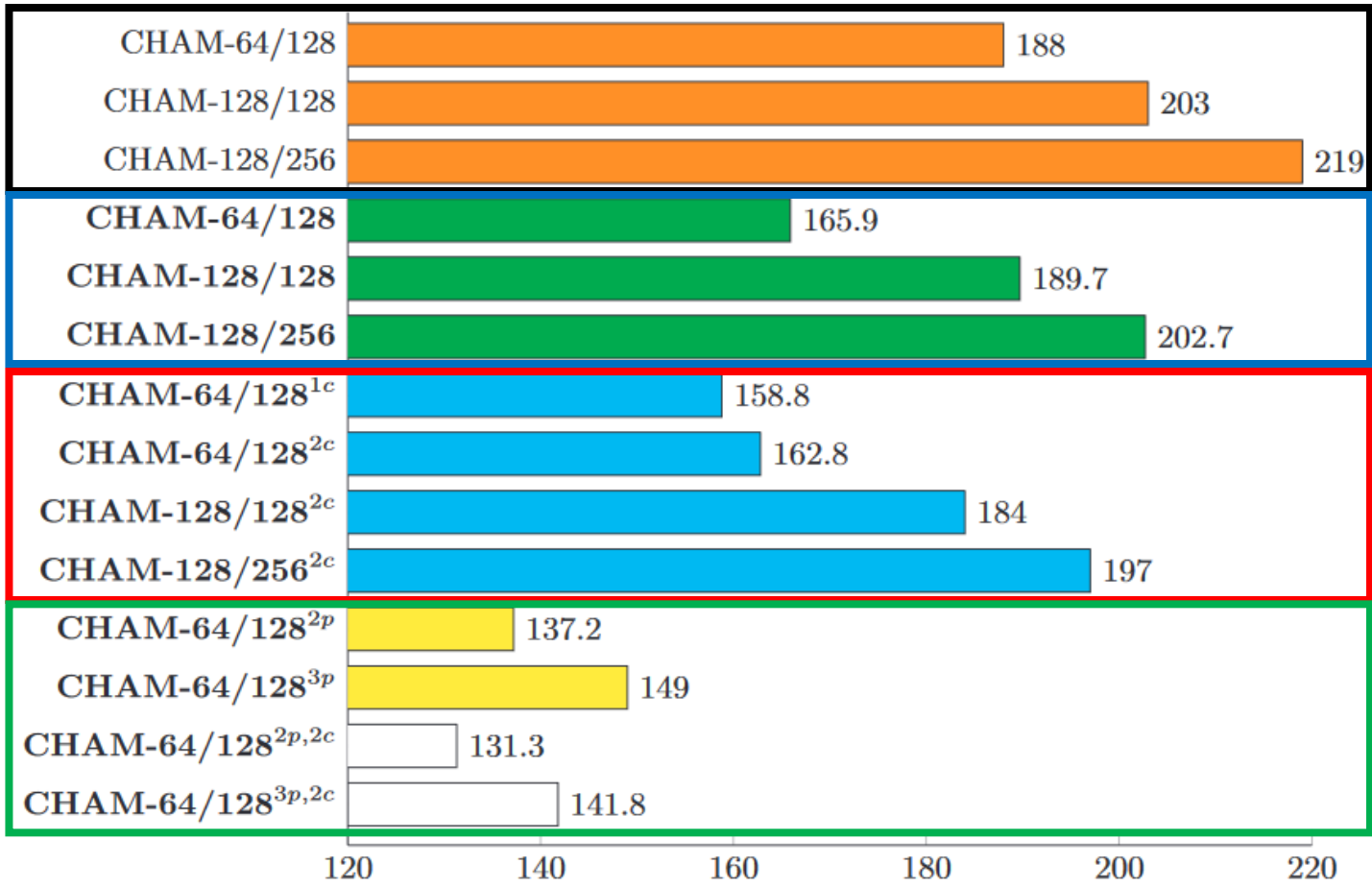
Proposed Implementation: Parallel implementation

- (a): 2-parallel implementation
- (b): 3-parallel implementation



Evaluation

- ICISC'19 implementation
- Basic optimal implementation
- Optimized CTR mode implementation
- Parallel implementation
  - Unit: clock cycles per byte



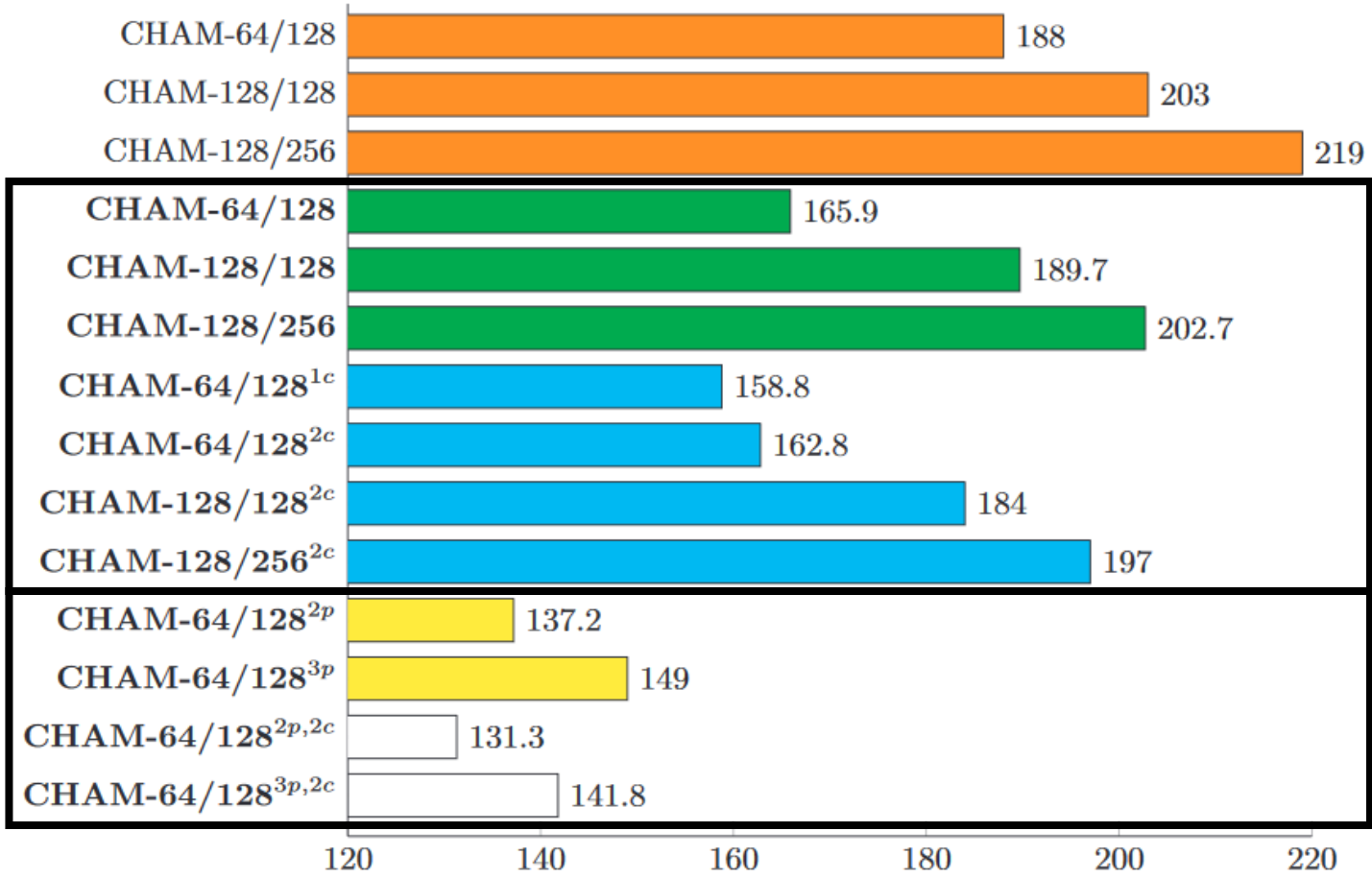
Evaluation

- The implementation improved performance 4.2%, 3.0%, 2.8%

- 2-parallel: 137.2

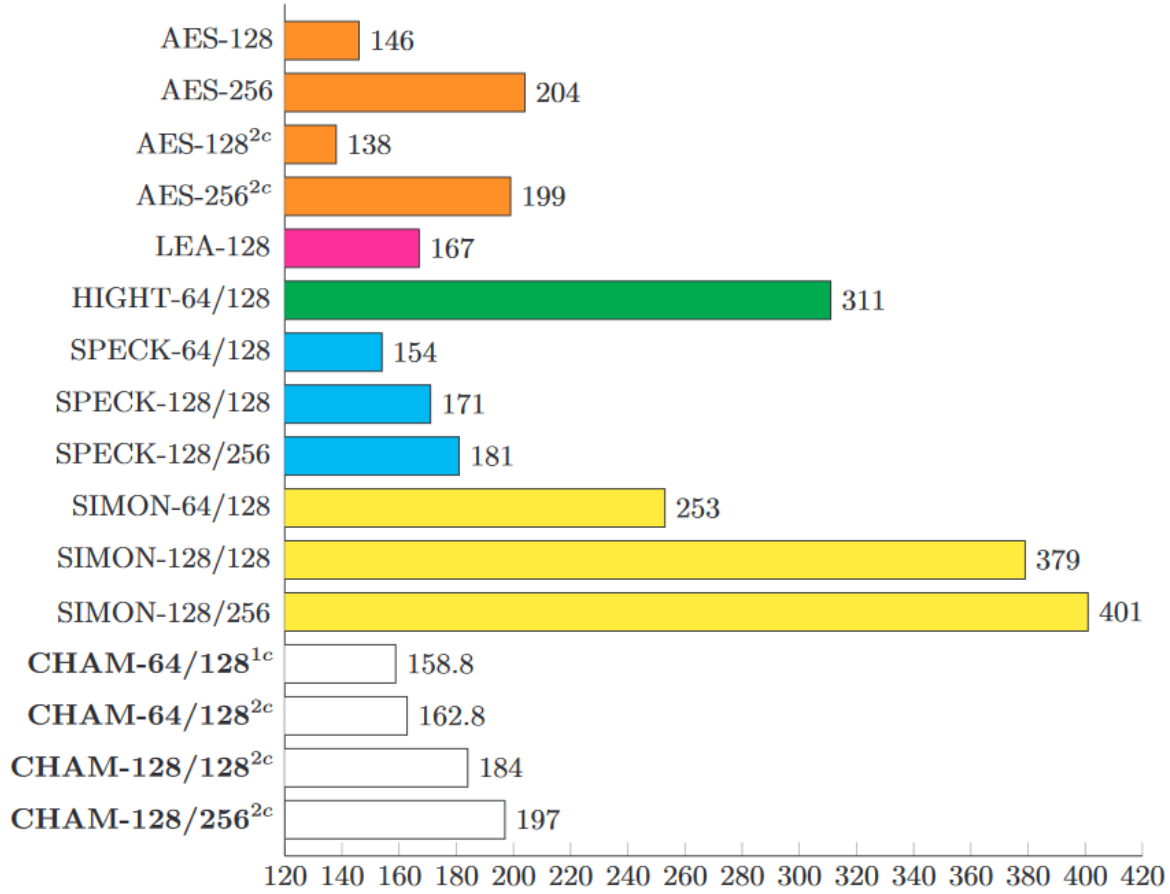
  3-parallel: 149.0

  - Unit: clock cycles per byte



Evaluation

- Compared with other block ciphers
- AES achieved the highest performance
  - SPN based
- **Proposed CHAM is second winner**
  - Among ARX based



Conclusion

- In this paper optimized CHAM-CTR proposed
  - Pre-computation model: 5 ROL(1-bit), 3 ROL(word-wise), 10 XOR, 3 ADD skipped
  - Parallel implementation: 2-Parallel, 3-Parallel CHAM-64/128
- **Proposed implementation achieved fast execution timing**
  - **It is practical IoT applications**
- Future works
  - Applying proposed techniques to other ARX based block ciphers
    - i.e. SPECK, SIMON
  - Variable key scenario implementation
  - Other microcontrollers will be investigated

Q & A