# Ring-LWE on 8-bit AVR Embedded Processor

Hwajeong Seo, **Hyeokdong Kwon,**
Yongbeen Kwon, Kyungho Kim,
Seungju Choi, Hyunjun Kim,
and Kyoungbae Jang

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

# Contents



CryptoCraft LAB

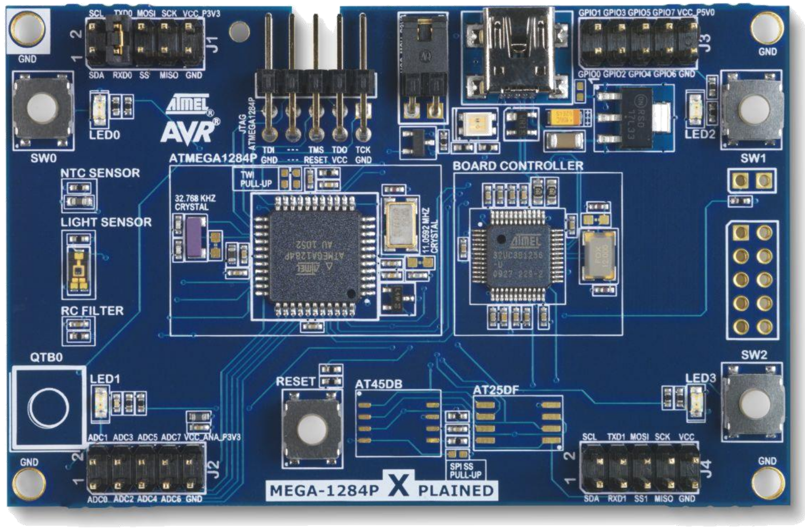
# Lattice-based Cryptography

- **RSA and ECC:**
  **Integer Factorization** and **Elliptic Curve Discrete Logarithm Problem**
  - **Hard problems** can be solved by **Shor's algorithm**

- **Lattice-based Cryptography:**
  Hard for quantum computers
  - **Ring-LWE Encryption** schemes: **Proposed [EUROCRYPT'10]**
  - **Generic Algorithm** for **other Lattice-based Cryptography**

# Implementation Platform

- **8-bit XMEGA128 Microcontroller**
  - **Internet of Things**
  - Operating Frequency: **32 MHz**
  - 128KB Flash, 8KB RAM, 32 registers
  - Core Instruction: **8-bit mul/add** (2/1 cycles)
  - AES/DES Crypto Engine (for **PRNG**)



# Previous Implementations on 8-bit AVR processor

- **8-bit AVR processor:**
  Boorghany et al. [ACM TEC'15]: **NTRU, Ring-LWE**
  → Poppelmann et al. [Latincrypt'15]: **BLISS**
  → Liu et al. [CHES'15]: **Ring-LWE**
  → Liu et al. [ACM TEC'17]: **Ring-LWE**, **BLISS**
  → Seo et al.[ICISC'17]: **Ring-LWE**
  → This work [WISA'19]: **Ring-LWE**

# Ring-LWE Scheme

**Key generation:** $Gen(\tilde{a})$

two error polynomials $r_1, r_2 \in R_q$ (from discrete Gaussian distribution)

$$\widetilde{r_1} = NTT(r_1), \qquad \widetilde{r_2} = NTT(r_2), \qquad \tilde{p} = \widetilde{r_1} - \tilde{a} \cdot \widetilde{r_2} \in R_q$$

Public key $(\tilde{a}, \tilde{p})$, Private key $(\widetilde{r_2})$

**Encryption:** $Enc(\tilde{a}, \tilde{p}, M)$

three error polynomials $e_1, e_2, e_3 \in R_q$, message $M$

$$(\widetilde{C_1}, \widetilde{C_2}) = (\tilde{a} \cdot \widetilde{e_1} + \widetilde{e_2}, \tilde{p} \cdot \widetilde{e_1} + NTT(e_3 + M))$$

**Decryption:** $Dec(\widetilde{C_1}, \widetilde{C_2}, \widetilde{r_2})$

Inverse NTT (INTT)

$$M = INTT(\widetilde{r_2} \cdot \widetilde{C_1} + \widetilde{C_2})$$

## NTT (Number Theoretic Transform):

Discrete Fourier transform: *n* degree polynomial multiplication $O(n^2) \rightarrow O(nlog\, n)$

$$\tilde{a} = NTT(a), \qquad \tilde{a}[i] = \sum_{j=0}^{n-1} a[j]w^{ij} \ mod\ q \quad (i = 0,1, \dots, n-1)$$

$$b = INTT(\tilde{a}), \qquad b[i] = n^{-1} \sum_{j=0}^{n-1} \tilde{a}[j]w^{-ij} \ mod\ q \ (\mathrm{i} = 0,1, \dots, \mathrm{n}-1)$$

$$INTT\big(NTT(a)\big) = a$$

| *n* | *q* | *a* | *w* |
|---|---|---|---|
| Power of 2 | Prime with $q \equiv 1 \ (mod\ 2n)$ | $a = (a[0], \dots, a[n-1]) \in Z_q^n$ | primitive *n*-th root of unity in $Z_q$: $w^n \equiv 1 \ (mod\ q)$ |

# Number Theoretic Transform

**Algorithm 1:** Iterative Number Theoretic Transform

**Require:** Polynomial $a(x)$, $n$-th root of unity $\omega$

**Ensure:** Polynomial $a(x) = \mathrm{NTT}(a)$

$a \leftarrow \mathrm{BitReverse}(a)$
**for** $i$ **from** 2 **by** $2i$ **to** $n$ **do**
  $\omega_i \leftarrow \omega_n^{n/i}$, $\omega \leftarrow 1$
  **for** $j$ **from** 0 **by** 1 **to** $i/2 - 1$ **do**
    **for** $k$ **from** 0 **by** $i$ **to** $n - 1$ **do**

Nested loop

      ① $U \leftarrow a[k+j]$, ② $V \leftarrow \omega \cdot a[k+j+i/2]$
      ③ $a[k+j] \leftarrow U + V$, ④ $a[k+j+i/2] \leftarrow U - V$
    **end for**
    $\omega \leftarrow \omega \cdot \omega_i$
  **end for**
**end for**
**return** $a$

# Number Theoretic Transform

**Algorithm 1:** Iterative Number Theoretic Transform

**Require:** Polynomial $a(x)$, $n$-th root of unity $\omega$

**Ensure:** Polynomial $a(x) = \mathrm{NTT}(a)$

```
1:  a ← BitReverse(a)
2:  for i from 2 by 2i to n do
3:      ω_i ← ω_n^{n/i}, ω ← 1
4:      for j from 0 by 1 to i/2 − 1 do
5:          for k from 0 by i to n − 1 do
6:              ① U ← a[k + j],        ② V ← ω · a[k + j + i/2]
7:              ③ a[k + j] ← U + V,    ④ a[k + j + i/2] ← U − V
8:          end for
9:          ω ← ω · ω_i
10:     end for
11: end for
12: return a
```

Overheads: modular multiplication

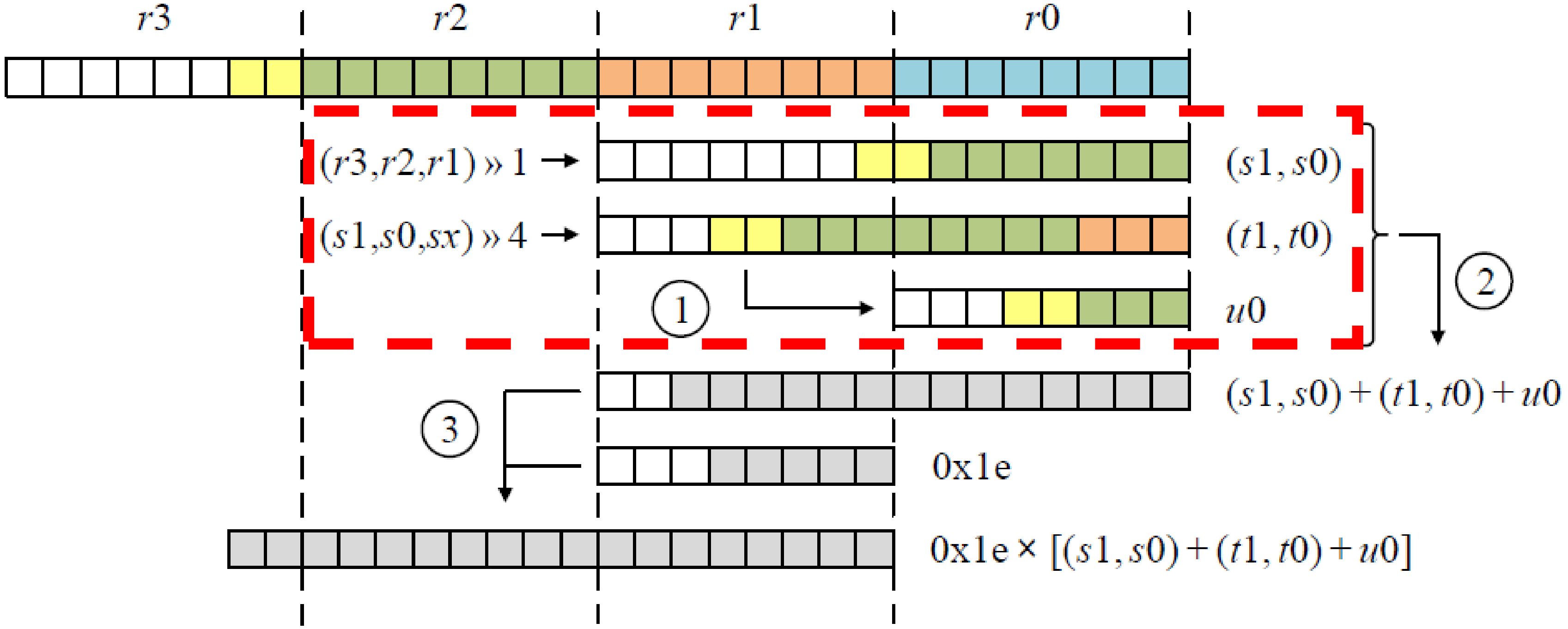
# Previous Modular Reduction

- **Approximation based reduction [ACM TEC'15]**
  - $\lfloor z/q \rfloor \cong \sum_{i=1}^{l} \left( z \gg (w - p_i) \right)$
  - $z \bmod q \ \cong z - q \times \lfloor z/q \rfloor$
  - $\lfloor z/7681 \rfloor \cong (z \gg 13) + (z \gg 17) + (z \gg 21)$

# Previous Modular Reduction

- **Optimized Implementation of approximation based reduction (8-bit) [CHES'15]**

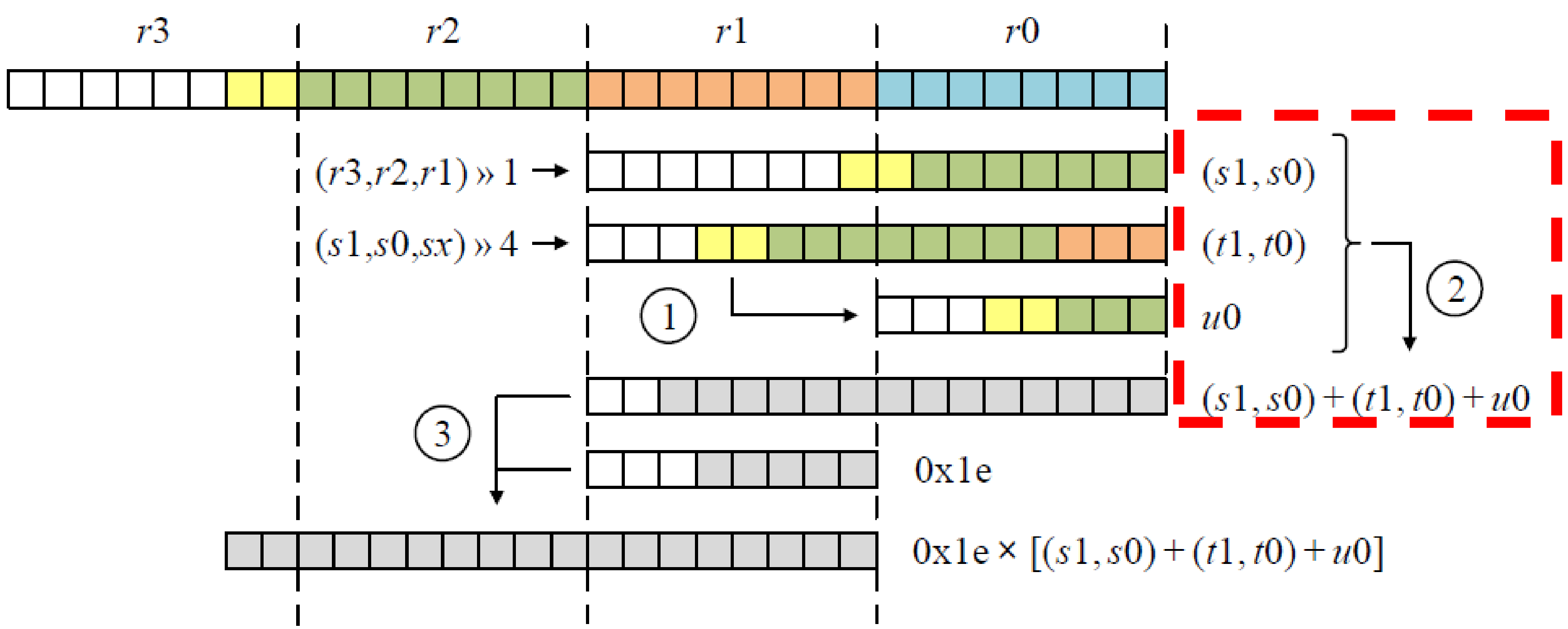
SAMS2 method, ①: shifting; ②: addition; ③: multiplication



# Previous Modular Reduction

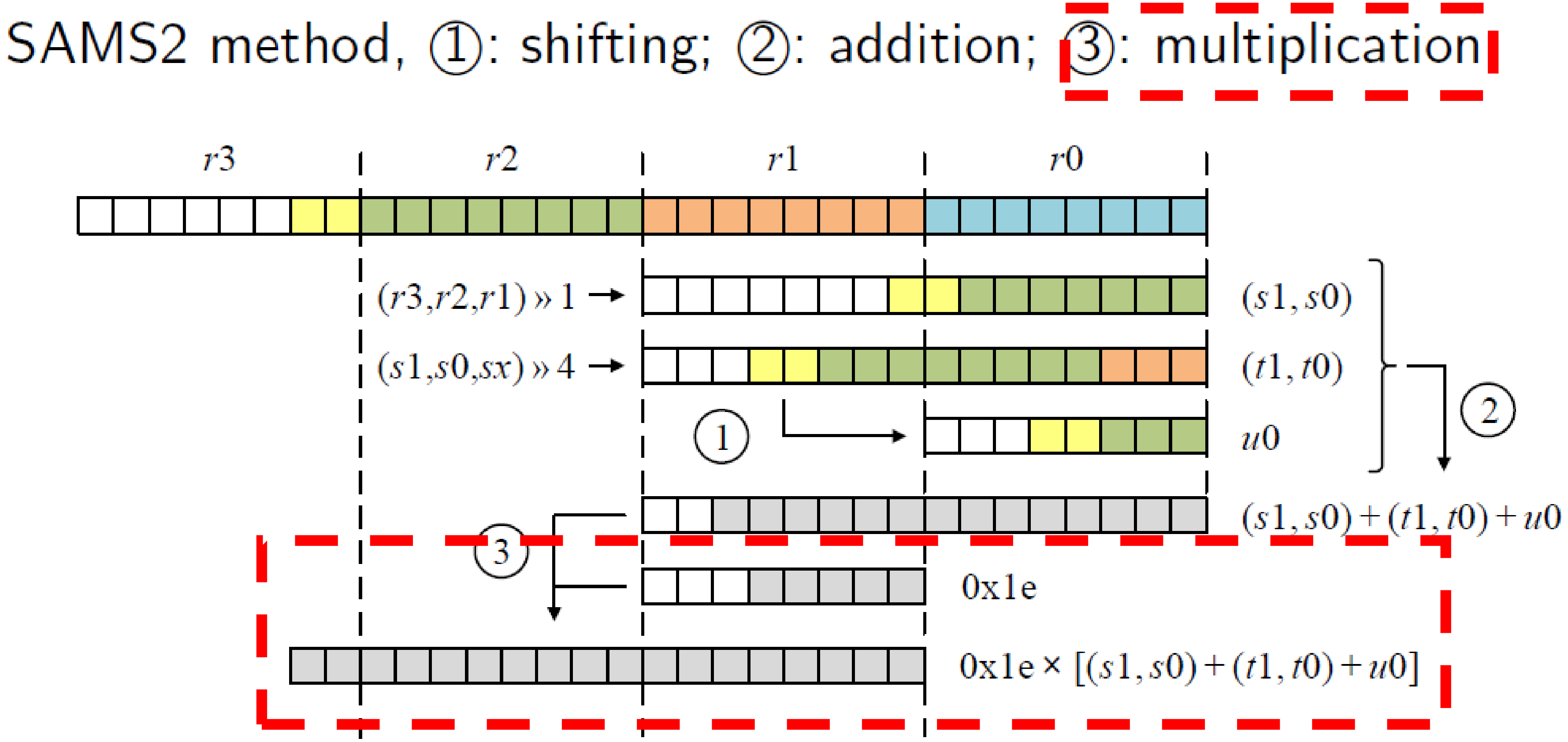
- **Optimized Implementation of approximation based reduction (8-bit) [CHES'15]**

SAMS2 method, ①: shifting; ②: addition; ③: multiplication



# Previous Modular Reduction

- **Optimized Implementation of approximation based reduction (8-bit) [CHES'15]**



# Previous Modular Reduction

- **Incomplete modular arithmetic [CHES'15]**
  - Complete: $s = a + b \; mod \; q$
  - Incomplete: $s = a + b \; mod \; 2^m$ where $m = [\log_2 q]$

# Previous Modular Reduction

- **Tiny Montgomery reduction [ACM TEC'17]**

**Precondition:** 13-bit modulus $q = 7681$, Montgomery radix $R = 2^{13}$, (incomplete) coefficients $a, b \in [0, 2^{13} - 1]$ , pre-computed constant $q' = -q^{-1} \bmod 2^{13} = 7679$

**function** $z = (a \cdot b \cdot R^{-1} \mod q)$
  $t \leftarrow a \cdot b$
  $s \leftarrow t \cdot q' \bmod R$
  $z \leftarrow (t + s \cdot q)/R$
  **if** $z \geq q$ **then** $z \leftarrow z - q$ **end if**
  **if** $z \geq q$ **then** $z \leftarrow z - q$ **end if**
  **return** $z$
**end function**

**Main Montgomery multiplication parts**

# Previous Modular Reduction

- **Tiny Montgomery reduction [ACM TEC'17]**

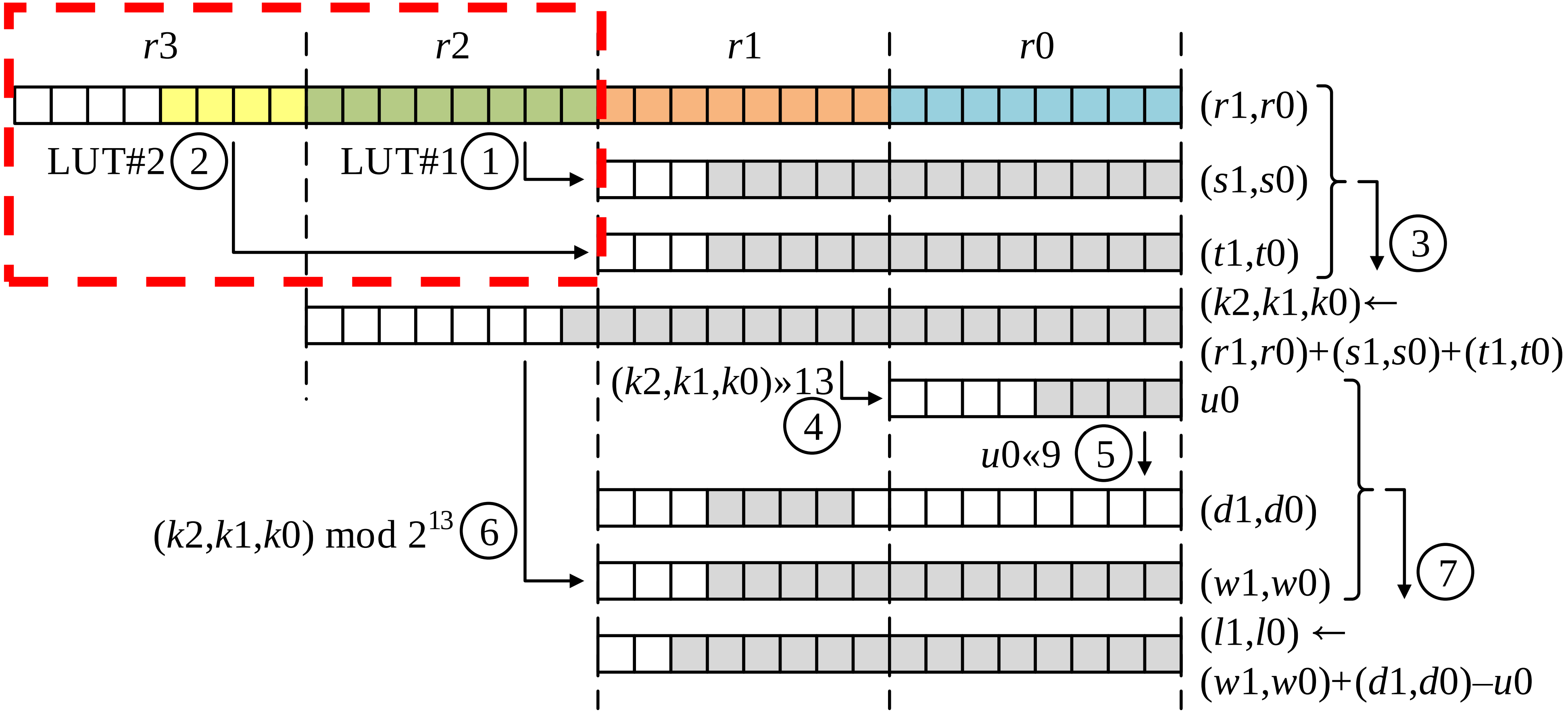
**Precondition:** 13-bit modulus $q = 7681$, Montgomery radix $R = 2^{13}$, (incomplete) coefficients $a, b \in [0, 2^{13} - 1]$ , pre-computed constant $q' = -q^{-1} \bmod 2^{13} = 7679$

**function** $z = (a \cdot b \cdot R^{-1} \mod q)$
  $t \leftarrow a \cdot b$
  $s \leftarrow t \cdot q' \bmod R$
  $z \leftarrow (t + s \cdot q)/R$
  **if** $z \geq q$ **then** $z \leftarrow z - q$ **end if**
  **if** $z \geq q$ **then** $z \leftarrow z - q$ **end if**
  **return** $z$
**end function**

Final subtraction in masked way

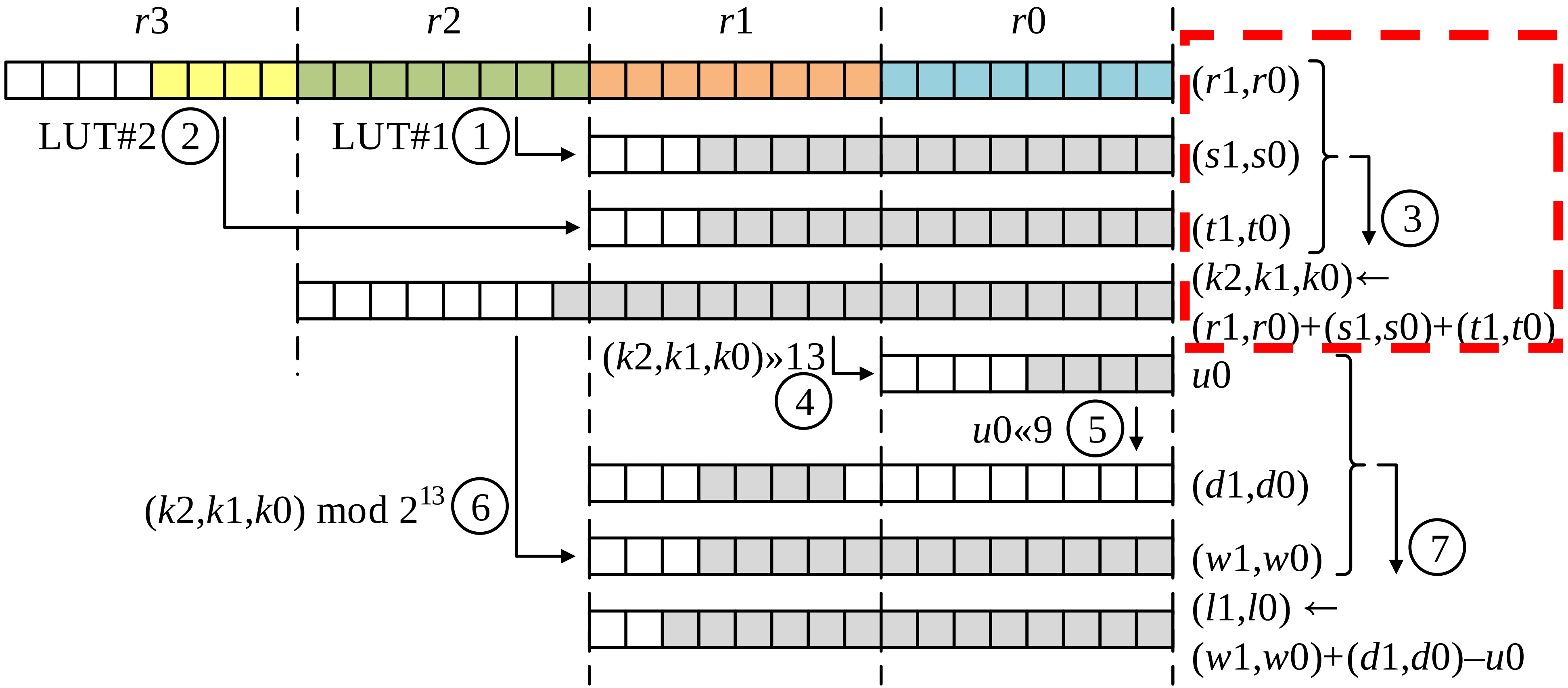
# Previous Modular Reduction

- LUT based Implementation: (1), (2) LUT access



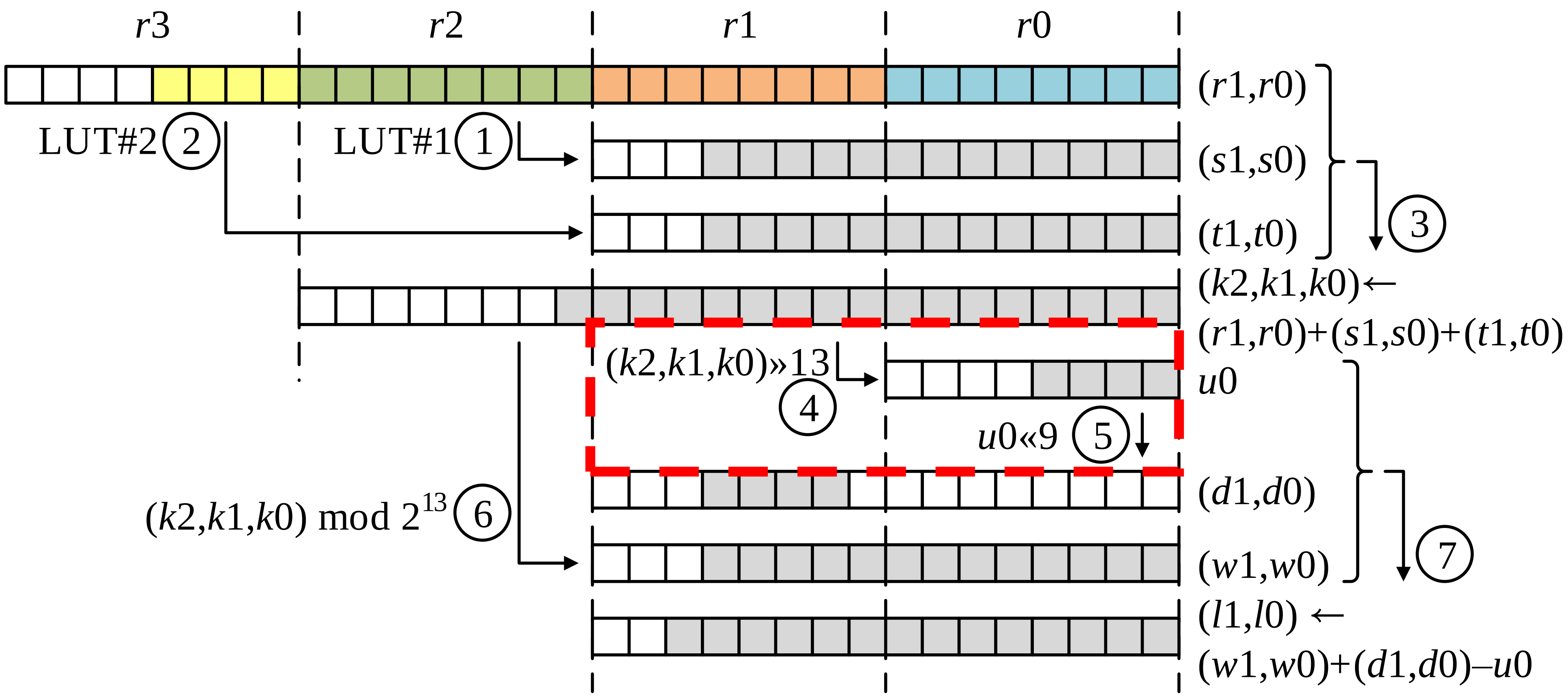
# Previous Modular Reduction

- LUT based Implementation: (3) addition



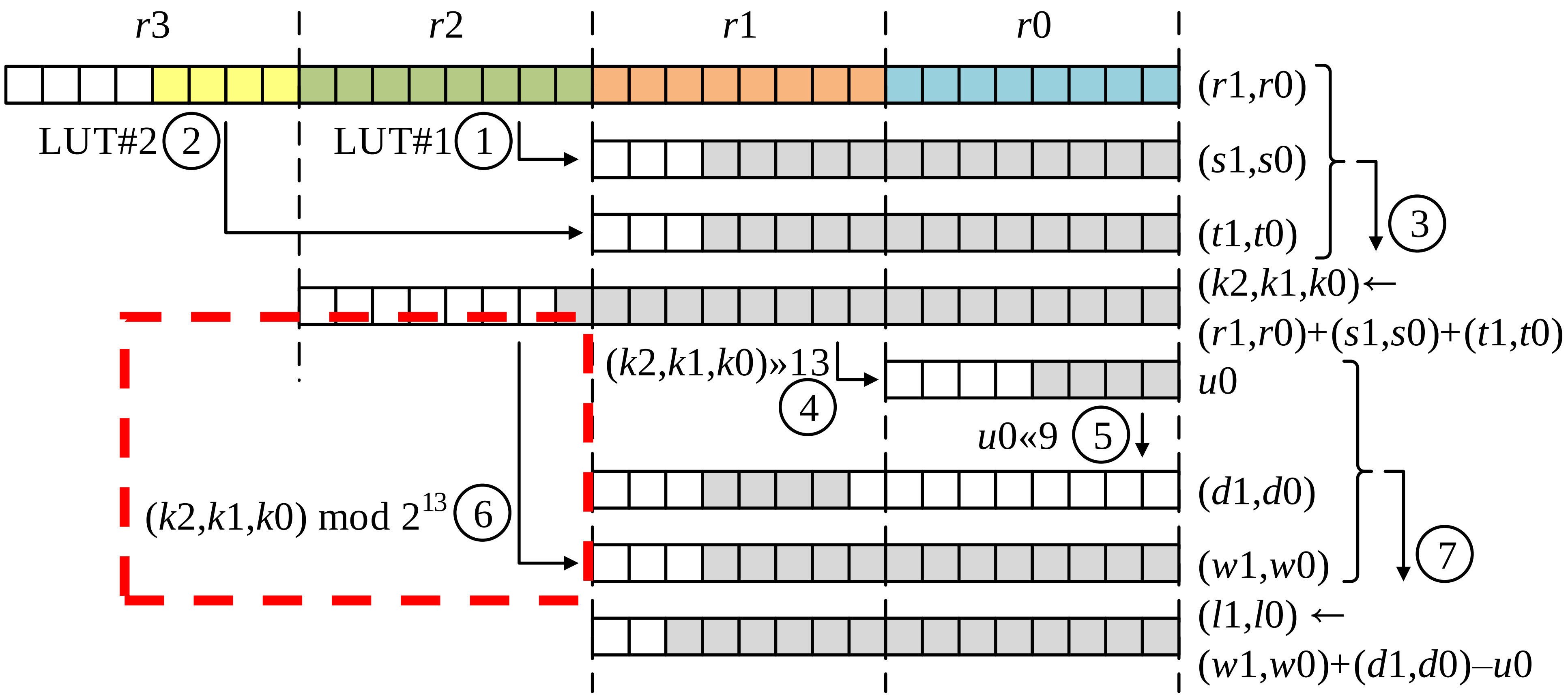
# Previous Modular Reduction

- LUT based Implementation: (4), (5) shifting



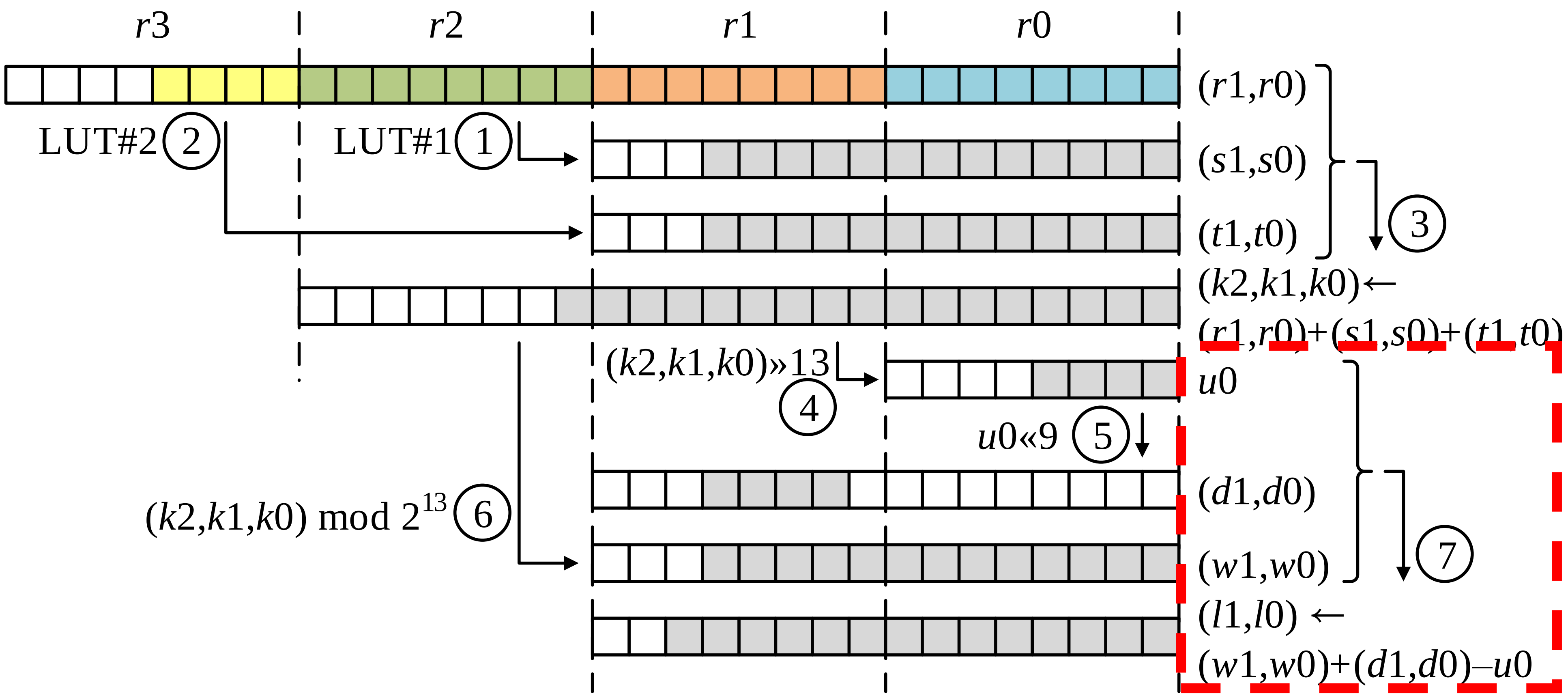
# Previous Modular Reduction

- LUT based Implementation: (6) modulo



# Previous Modular Reduction

- LUT based Implementation: (7) addition and subtraction



# Motivation & Contribution

- **Motivation**
  - Few **secure 8-bit AVR** implementations
  - **High security** → basic requirement for IoT

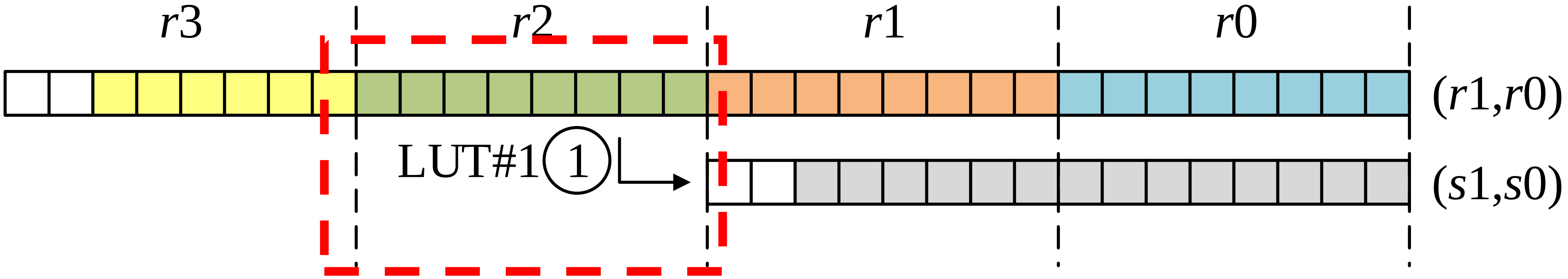
- **Contributions**
  **Secure & efficient Ring-LWE** implementation
  - **LUT** based multiplication
  - **Constant time** modular reduction

# Optimization Techniques for Modular Reduction

- Previous work: LUT#1 → LUT#2 → addition
- Proposed work: LUT#1 → addition → LUT#2
  - LUT#2 performs reduction on more bits at once than previous work.

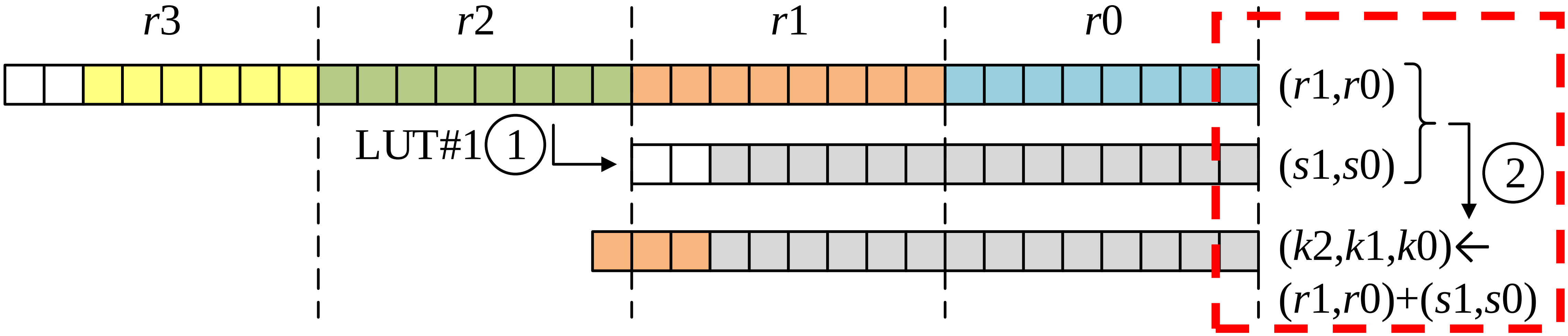
# Optimization Techniques for Modular Reduction

- Proposed method: (1) LUT access



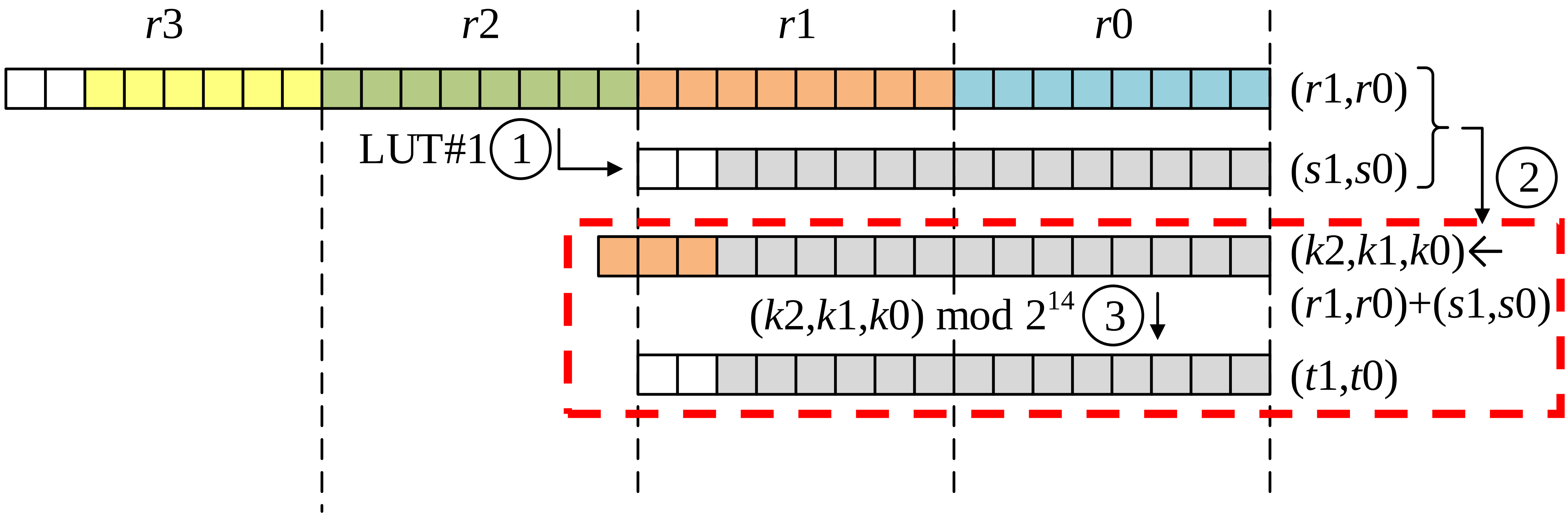
# Optimization Techniques for Modular Reduction

- Proposed method: (2) addition



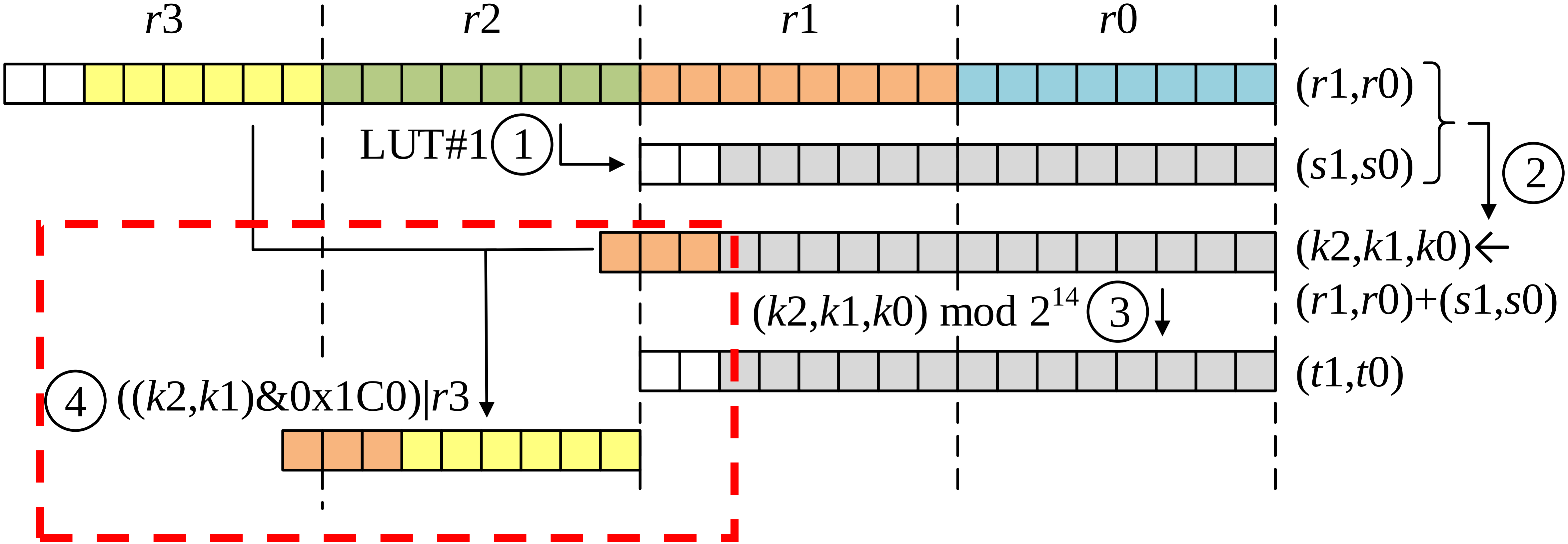
# Optimization Techniques for Modular Reduction

- Proposed method: (3) modulo



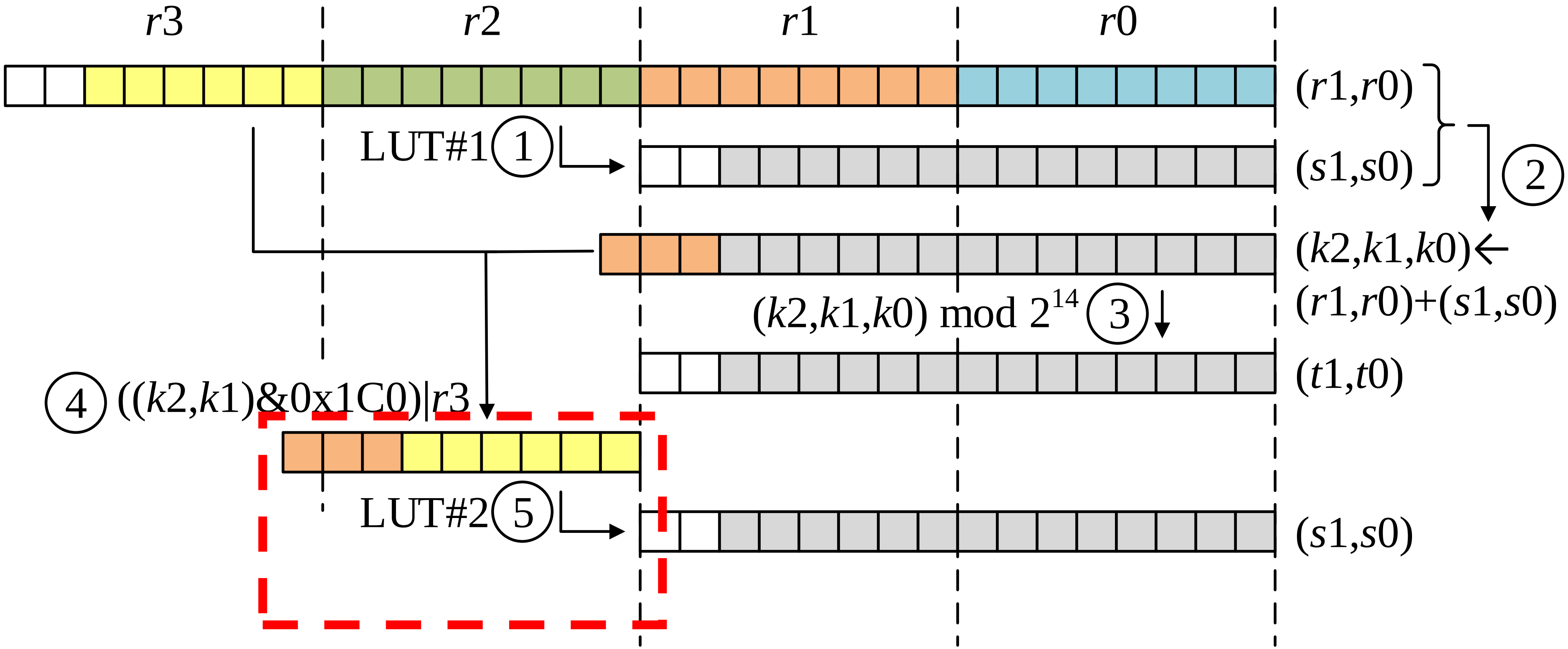
# Optimization Techniques for Modular Reduction

- Proposed method: (4) concatenation



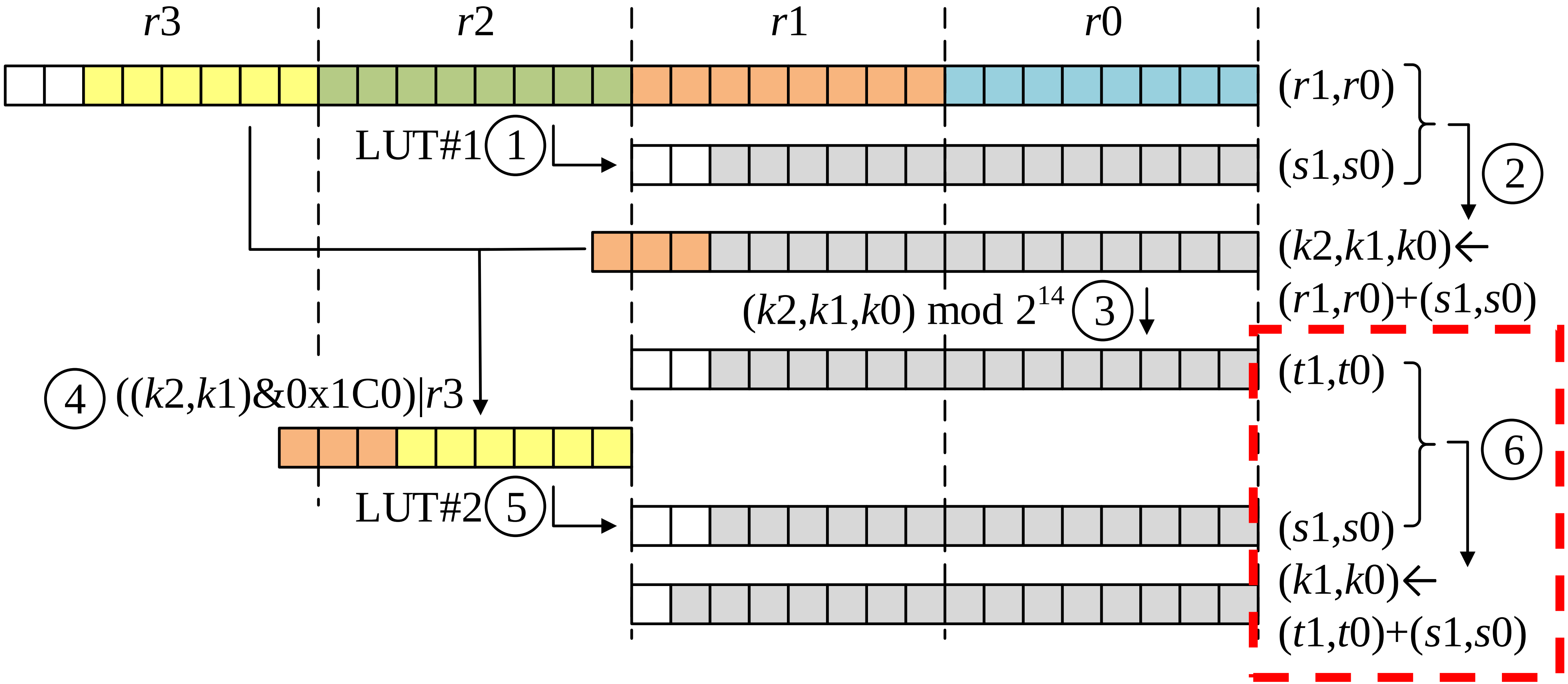
# Optimization Techniques for Modular Reduction

- Proposed method: (5) LUT access



# Optimization Techniques for Modular Reduction

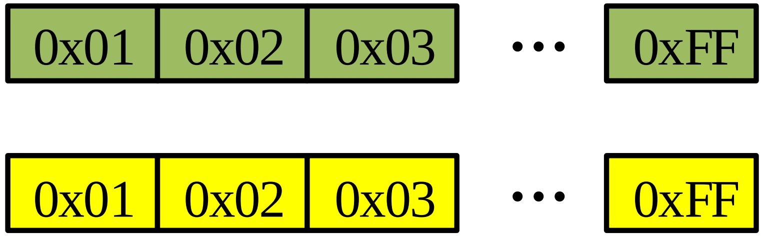
- Proposed method: (6) addition



# LUT access optimization

- Previous work: storing data in 16-bit wise


- Proposed work: storing data in 8-bit wise in separated way
  → memory offset calculation is 1 clock cheaper than previous work



# Evaluation

| Imp. | 256-bit security | | |
|---|---|---|---|
| | Mod MUL | NTT | Const |
| Boorghany | N/A | 2,207,787 | X |
| Boorghany | N/A | N/A | X |
| Poppelmann | N/A | 855,595 | X |
| Liu | N/A | 441,572 | X |
| Liu | 70 | 516,971 | O |
| Seo | 66 | 403,224 | O |
| This work | 47 | 344,288 | O |

# Evaluation

| Imp. | 256-bit security | | |
|---|---|---|---|
| | Mod MUL | NTT | Const |
| Boorghany | N/A | 2,207,787 | X |
| Boorghany | N/A | N/A | X |
| Poppelmann | N/A | 855,595 | X |
| Liu | N/A | 441,572 | X |
| Liu | 70 | 516,971 | O |
| Seo | 66 | 403,224 | O |
| This work | 47 | 344,288 | O |

# Evaluation

| Imp. | NTT/FFT | Key-Gen | Enc | Secure |
|---|---|---|---|---|
| **Implementations of 256-bit security level** | | | | |
| Boorghany | 2,207,787 | N/A | N/A | X |
| Poppelmann | 855,595 | N/A | 3,279,142 | X |
| Liu | 441,572 | 2,165,239 | 2,617,459 | X |
| **Liu** | **516,971** | **N/A** | **1,975,806** | **O** |
| **Seo** | **403,224** | **N/A** | **1,754,064** | **O** |
| **This work** | **344,288** | **1,325,171** | **1,430,601** | **O** |

# Conclusion

- **Contribution**
  - **Secure and fast NTT** computation
  - New speed record for **Ring-LWE encryption on 8-bit AVR**

- **Future works**
  - Implementations on **other low-end processors** (8-bit PIC / 16-bit MSP)
  - Investigation of **side channel attack model**

# Q & A