SPARKLE(3)

SCHWAEMM256-128 구현

발표자: 양유진

링크: https://youtu.be/N_rQ_0muubY

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

SCHWAEMM256-128 구현

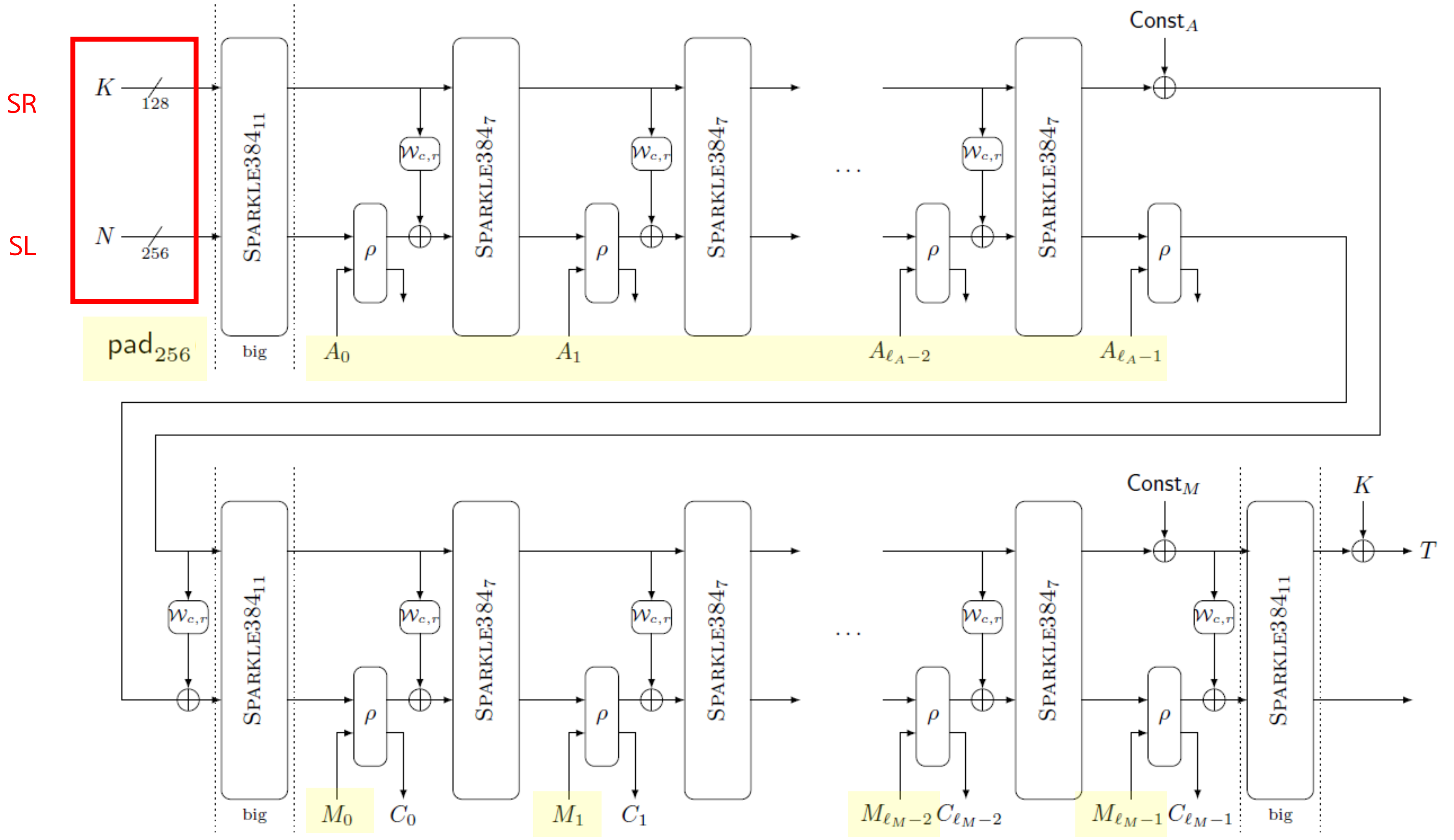


Figure 2.5: The Authenticated Encryption Algorithm SCHWAEMM256-128 with rate $r = 256$ and capacity $c = 128$.

SCHWAEMM256-128 구현

Data block size: 256-bits

Key length: 128-bits
Nonce length: 256-bits
Tag length: 128-bits

plain text: 32-bits (가정)
cipher text: 32-bits (가정)
authenticated data: 32-bits (가정)

n(Internal state size): 384-bit
r(size of rate): 256-bit
c(size of the capacity): 128-bit

SCHWAEMM256-128 구현 0) Padding AD and M

Algorithm 2.8 pad_r

Input/Output: $M \in \mathbb{F}_2^*$, with $|M| < r$

$i \leftarrow (-|M| - 1) \mod r$
$M \leftarrow M \| 1 \| 0^i$
return M

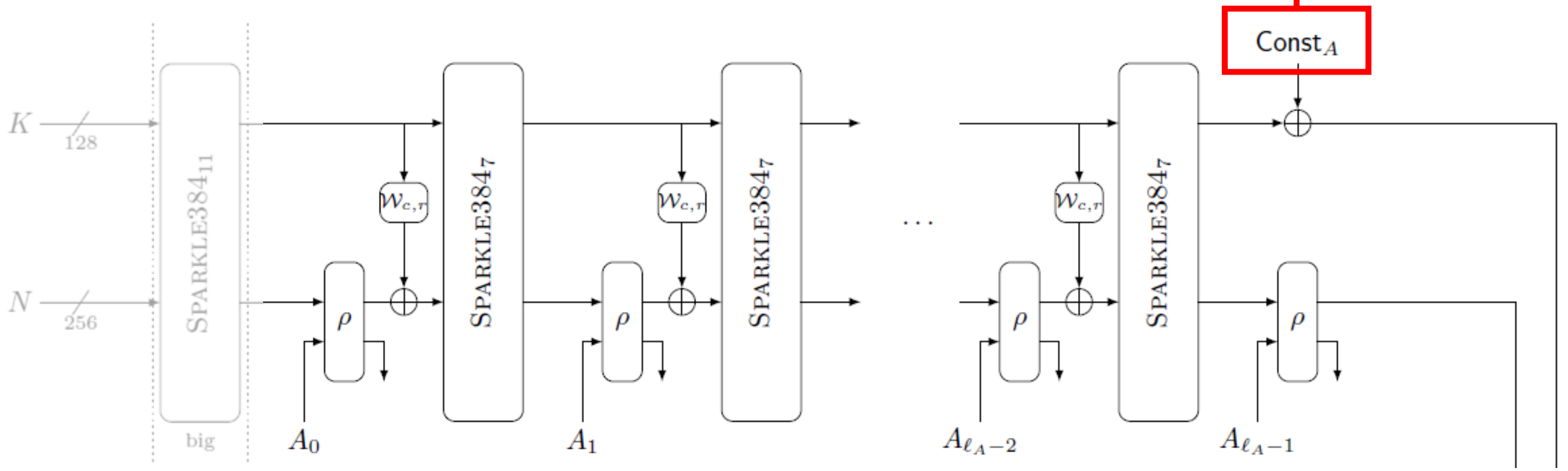
```
padding = eng.allocate_qureg(256 - len)
X | padding[7]  # 0000 0001 0000...

## padding
for i in range(256 - len):
    A_last.append(padding[i])
```

▷ Padding the associated data and message

if $A \neq \epsilon$ **then**
  $A_0 \| A_1 \| \dots \| A_{\ell_A - 1} \leftarrow A$ with $\forall i \in \{0, \dots, \ell_A - 2\} : |A_i| = 256$ and $1 \leq |A_{\ell_A-1}| \leq 256$
  if $|A_{\ell_A-1}| < 256$ **then**
    $A_{\ell_A-1} \leftarrow \mathsf{pad}_{256}(A_{\ell_A-1})$
    $\boxed{\mathsf{Const}_A \leftarrow 0 \oplus (1 \ll 2)}$
  else
    $\mathsf{Const}_A \leftarrow 1 \oplus (1 \ll 2)$
  end if
end if

1<<2 = 4

```
## AD 계산
const = eng.allocate_qureg(4)
X | const[2]  # 0100
```



SCHWAEMM256-128 구현 0) Padding AD and M

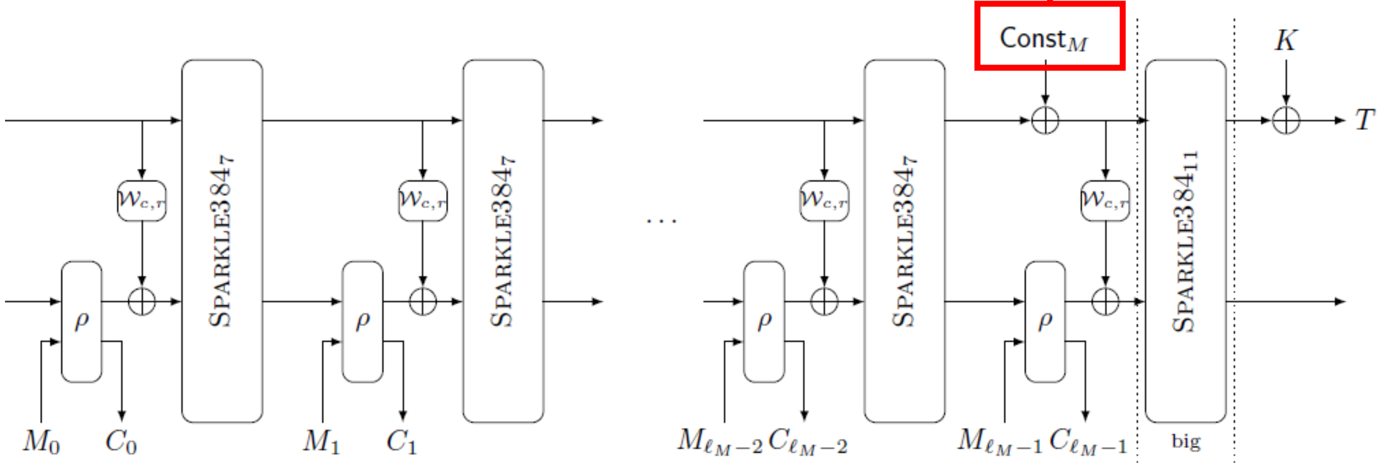
```
padding = eng.allocate_qureg(256 - len)
X | padding[7]  # 0000 0001 0000...

## padding
for i in range(256 - len):
    M_last.append(padding[i])
```

$\triangleright$ Padding the associated data and message

if $M \neq \epsilon$ **then**
  $M_0 \| M_1 \| \ldots \| M_{\ell_M - 1} \leftarrow M$ with $\forall i \in \{0, \ldots, \ell_M - 2\} : |M_i| = 256$ and $1 \leq |M_{\ell_M-1}| \leq 256$
  $t \leftarrow |M_{\ell_M - 1}|$
  if $|M_{\ell_M-1}| < 256$ **then**
    $M_{\ell_M-1} \leftarrow \mathsf{pad}_{256}(M_{\ell_M-1})$
    $\boxed{\mathsf{Const}_M \leftarrow 2 \oplus (1 \ll 2)}$
  else
    $\mathsf{Const}_M \leftarrow 3 \oplus (1 \ll 2)$
  end if
end if

(1<<2 = 4), 2^4=6

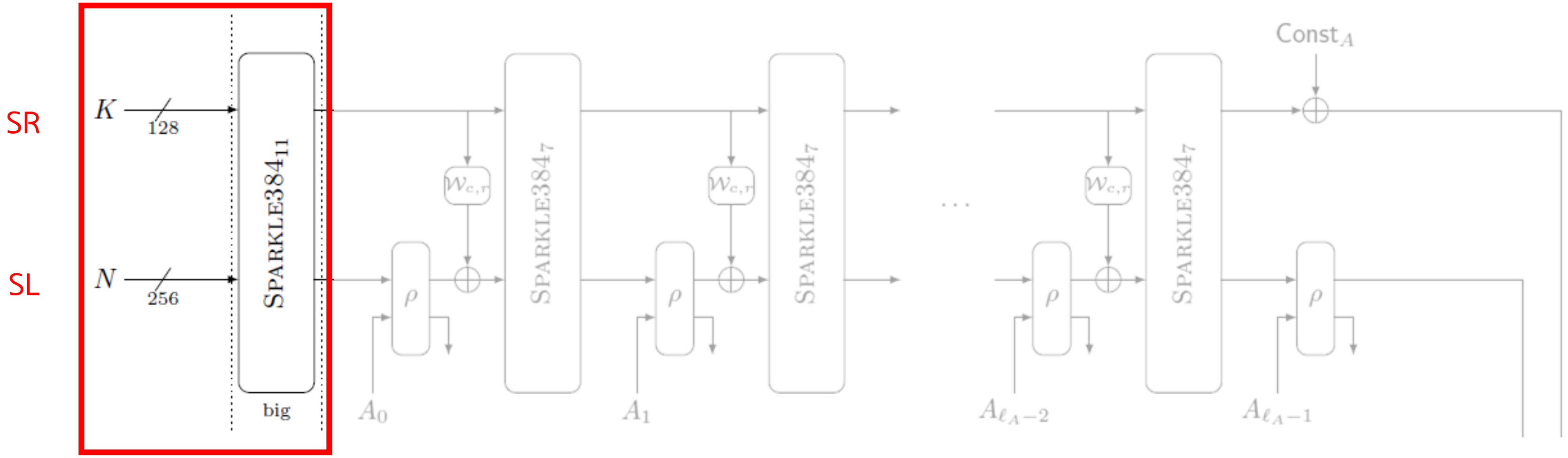
```
X | const[1]  # 0110
```



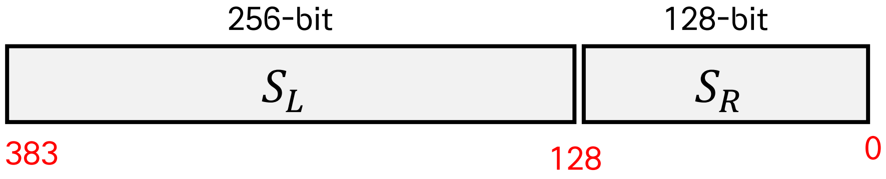
SCHWAEMM256-128 구현 1) State initialization

▷ State initialization

$S_L \| S_R \leftarrow \text{SPARKLE384}_{11}(N \| K)$ with $|S_L| = 256$ and $|S_R| = 128$




```
S = eng.allocate_qureg(384)
K = eng.allocate_qureg(128)
N = eng.allocate_qureg(256)

# N || K
concat_NK(eng, K, S[:128], 128)  SR
concat_NK(eng, N, S[128:384], 256)  SL
```



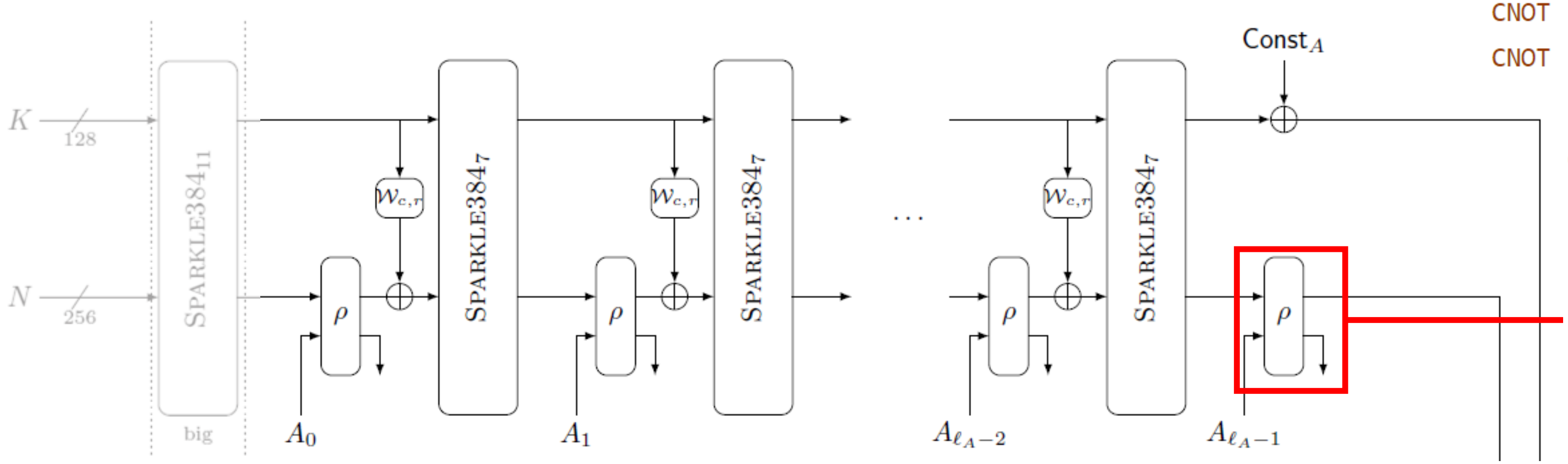
SCHWAEMM256-128 구현 2) Processing of AD

▷ Processing of associated data

if $A \neq \epsilon$ **then**
  for all $j = 0, \ldots, \ell_A - 2$ **do**
    $S_L \| S_R \leftarrow \text{SPARKLE384}_7((\rho_1(S_L, A_j) \oplus \mathcal{W}_{128,256}(S_R)) \| S_R)$
  end for

▷ Finalization if message is empty

  $S_L \| S_R \leftarrow \text{SPARKLE384}_{11}(\boxed{\rho_1(S_L, A_{\ell_A - 1})} \oplus \mathcal{W}_{128,256}(S_R \oplus \mathsf{Const}_A)) \| (S_R \oplus \mathsf{Const}_A))$
end if

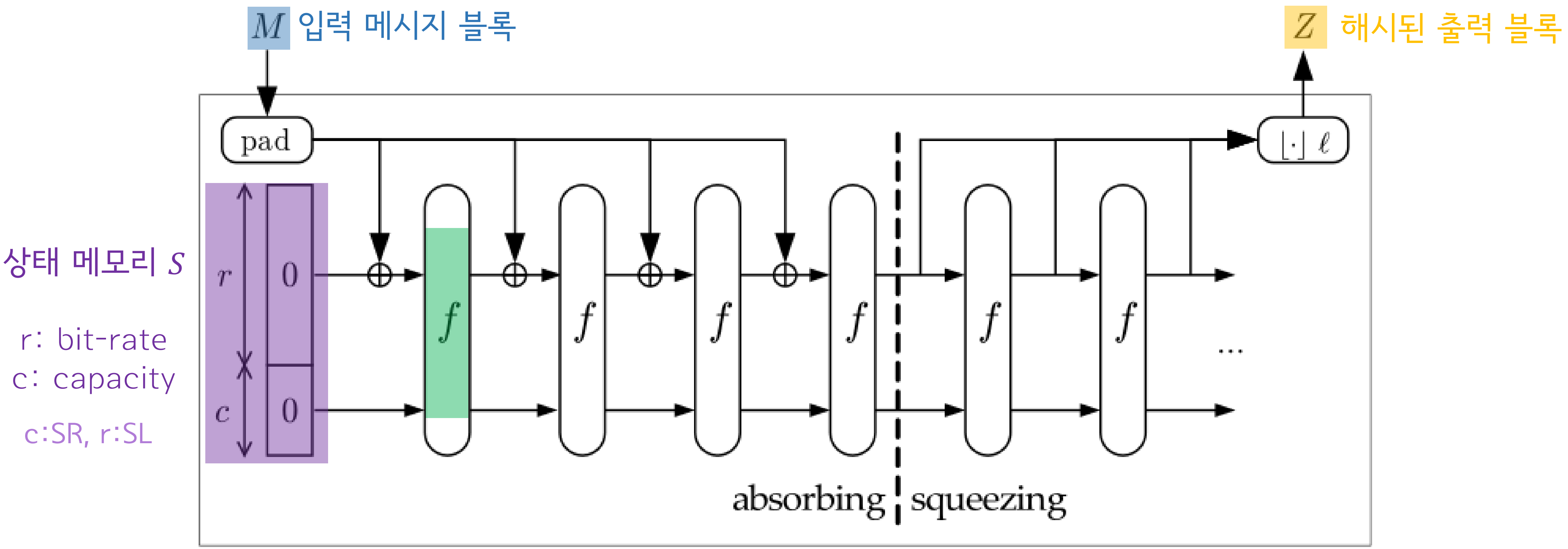



```
def FeistelSwap(eng, s):
    for i in range(32):
        Swap | (s[255 - i], s[383 - i]) # x2 <-> x0
        Swap | (s[223 - i], s[351 - i]) # y2 <-> y0
        Swap | (s[191 - i], s[319 - i]) # x3 <-> x1
        Swap | (s[159 - i], s[287 - i]) # y3 <-> y1

    for i in range(32):
        CNOT | (s[383 - i], s[255 - i])  # x2 ^= x0
        CNOT | (s[351 - i], s[223 - i])  # y2 ^= y0
        CNOT | (s[319 - i], s[191 - i])  # x3 ^= x1
        CNOT | (s[287 - i], s[159 - i])  # y3 ^= y1
```

```
def p1(eng, s, d, carry):

    if(resource_check != 1):
        FeistelSwap(eng, s)

    for i in range(32):
        CNOT | (d[31 - i], s[383 - i])
        CNOT | (d[63 - i], s[351 - i])
```

Sponge 구조

[문제점]외부 부분을 통해 순열의 일부 계산 가능



sponge

SPARKLE384
상태 메모리 갱신함
$f: \{0,1\}^b \rightarrow \{0,1\}^b$

SCHWAEMM256-128 구현 2) Processing of AD

Race-Whitening 사용하면 내부 상태를 파악하지 않을 경우 순열의 일부 계산 불가능

$\triangleright$ Processing of associated data

if $A \neq \epsilon$ **then**
  for all $j = 0, \dots, \ell_A - 2$ **do**
    $S_L \| S_R \leftarrow \textsc{Sparkle384}_7((\rho_1(S_L, A_j) \oplus \boxed{\mathcal{W}_{128,256}(S_R)}) \| S_R)$
  end for

$\triangleright$ Finalization if message is empty

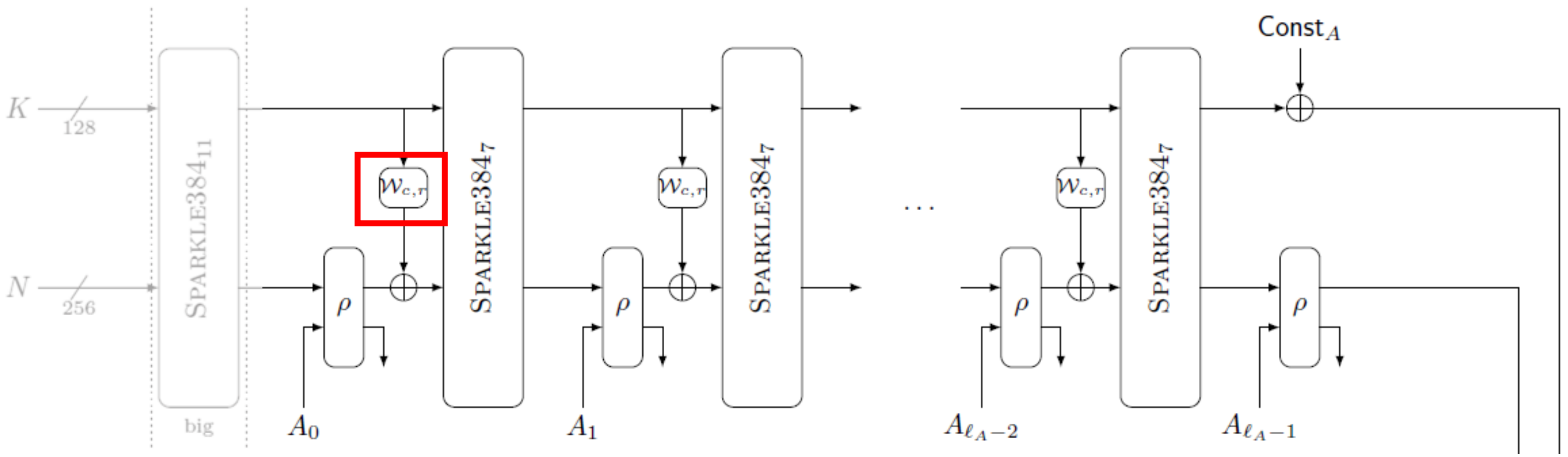
  $S_L \| S_R \leftarrow \textsc{Sparkle384}_{11}((\rho_1(S_L, A_{\ell_A - 1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \mathsf{Const}_A)) \| (S_R \oplus \mathsf{Const}_A))$
end if

내부 상태SR

$\mathcal{W}_{c,r} : \mathbb{F}_2^c \rightarrow \mathbb{F}_2^r$

Race-Whitening

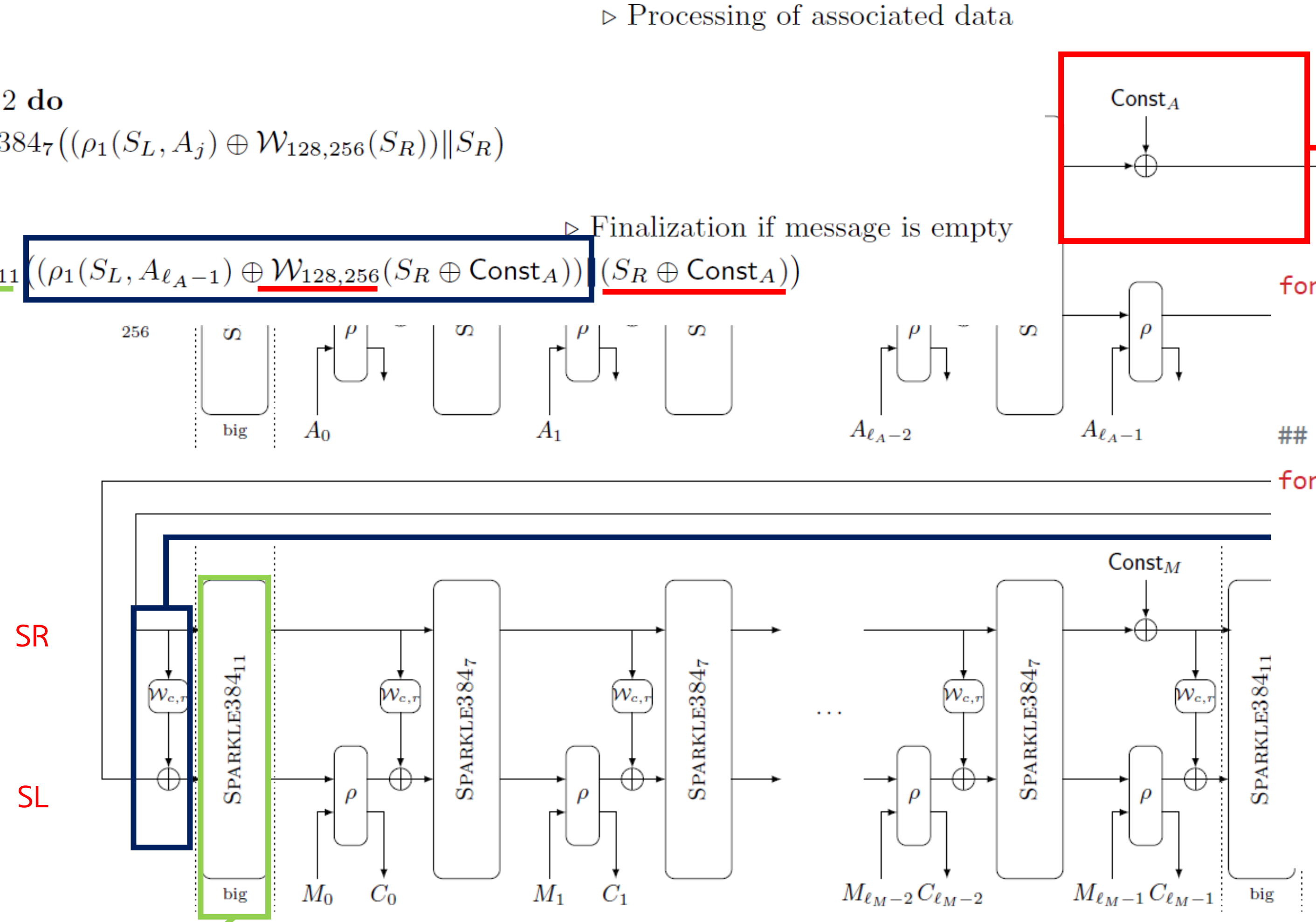
$\mathcal{W}_{128,256}(x, y) = (x, y, x, y)$



SCHWAEMM256-128 구현 2) Processing of AD

if $A \neq \epsilon$ **then**
  for all $j = 0, \ldots, \ell_A - 2$ **do**
    $S_L \| S_R \leftarrow \textsc{Sparkle384}_7((\rho_1(S_L, A_j) \oplus \mathcal{W}_{128,256}(S_R)) \| S_R)$
  end for

  $S_L \| S_R \leftarrow \textsc{Sparkle384}_{11}((\rho_1(S_L, A_{\ell_A - 1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \mathsf{Const}_A)) \| (S_R \oplus \mathsf{Const}_A))$
end if




```
for i in range(4):
    CNOT | (const[3 - i], S[27 - i])

## SL ^ SR
for i in range(32):
    CNOT | (S[127 - i], S[383 - i])
    CNOT | (S[95 - i], S[351 - i])
    CNOT | (S[63 - i], S[319 - i])
    CNOT | (S[31 - i], S[287 - i])
    CNOT | (S[127 - i], S[255 - i])
    CNOT | (S[95 - i], S[223 - i])
    CNOT | (S[63 - i], S[191 - i])
    CNOT | (S[31 - i], S[159 - i])
```

```
SPARKLE_384(eng, S, carry, 11)
```

SCHWAEMM256-128 구현 3) Encrypting

▷ Encrypting

if $M \neq \epsilon$ **then**

for all $j = 0, \ldots, \ell_M - 2$ **do**

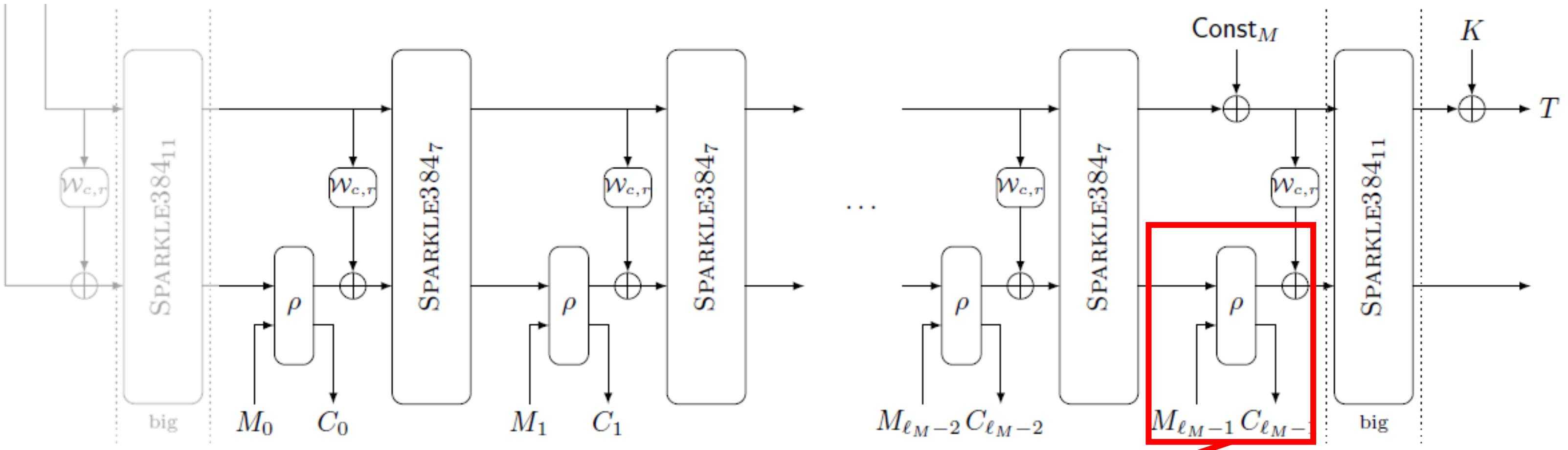
$\boxed{C_j \leftarrow \rho_2(S_L, M_j)}$

$S_L \| S_R \leftarrow \text{SPARKLE384}_7((\rho_1(S_L, M_j) \oplus \mathcal{W}_{128,256}(S_R)) \| S_R)$

end for

$\boxed{C_{\ell_M - 1} \leftarrow \mathsf{trunc}_t(\rho_2(S_L, M_{\ell_M - 1}))}$

$\boxed{S_L \oplus M_j}$




```
def trunc(eng, s, c, n):
    for i in range(n):
        CNOT | (s[383 - i], c[31 - i])  # c[0] ^= x0
```

SCHWAEMM256-128 구현 4) Finalization

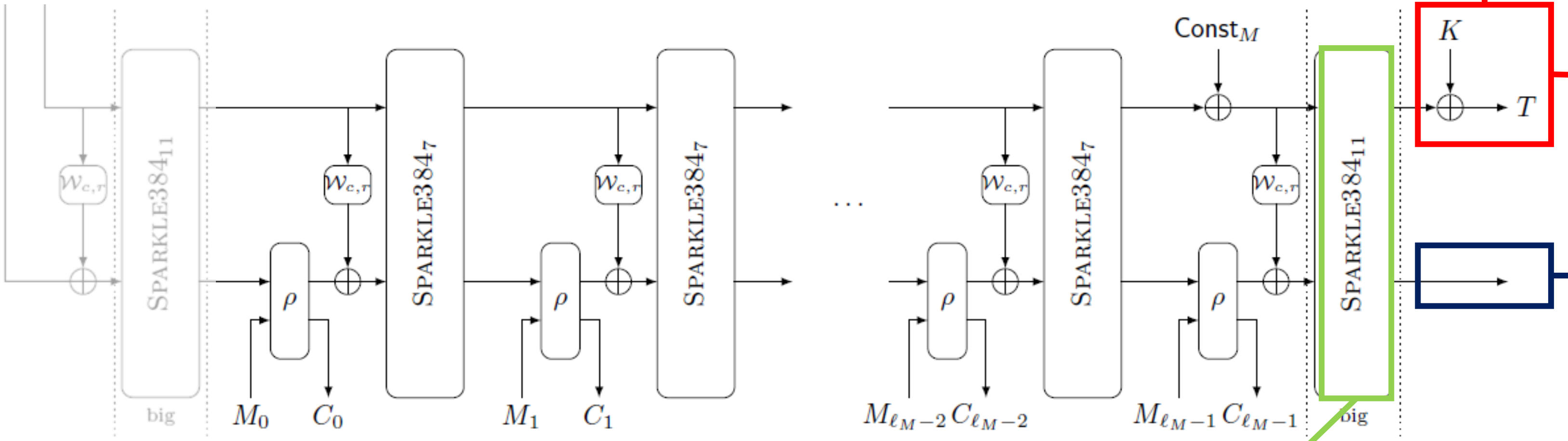
$\triangleright$ Finalization

$$S_L \| S_R \leftarrow \text{SPARKLE384}_{11}((\rho_1(S_L, M_{\ell_M - 1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \mathsf{Const}_M)) \| (S_R \oplus \mathsf{Const}_M))$$

end if

return $(C_0 \| C_1 \| \ldots \| C_{\ell_M - 1}, \boxed{S_R \oplus K})$

```
result = eng.allocate_qureg(128 + len)

## TAG생성해서 result에 담기
for i in range(127, -1, -1):
    CNOT | (K[32 * (3 - int(i / 32)) + i % 32], S[i])
    CNOT | (S[i], result[i])
```




```
SPARKLE_384(eng, S, carry, 11)
```

```
for i in range(len - 1, -1, -1):
    CNOT | (C[i], result[128 + i])
```

SCHWAEMM256-128 결과

```
All(Measure) | result
```

[입력]
PlainText(32-bit): 0x03020100
Authenticated Data(32-bit): 0x03020100
Key(128-bit): 0x0F0E0D0C0B0A09080706050403020100
Nonce(256-bit): 0x1F1E1D1C1B1A191817161514131211100F0E0D0C0B0A09080706050403020100

[출력]
CipherText(128-bit): 8711A728 679B5F30 A61C4712 9308B9AD 6A61C33C

→ 28A71187 305F9B67 12471CA6 ADB90893 3CC3616A

```
0x28a71187305f9b6712471ca6adb908933cc3616a
```

[python code 결과]

감사합니다