SPARKLE(2)

SCHWAEMM128-128 구현

발표자: 양유진

링크: https://youtu.be/Qd1vqOmZCWc

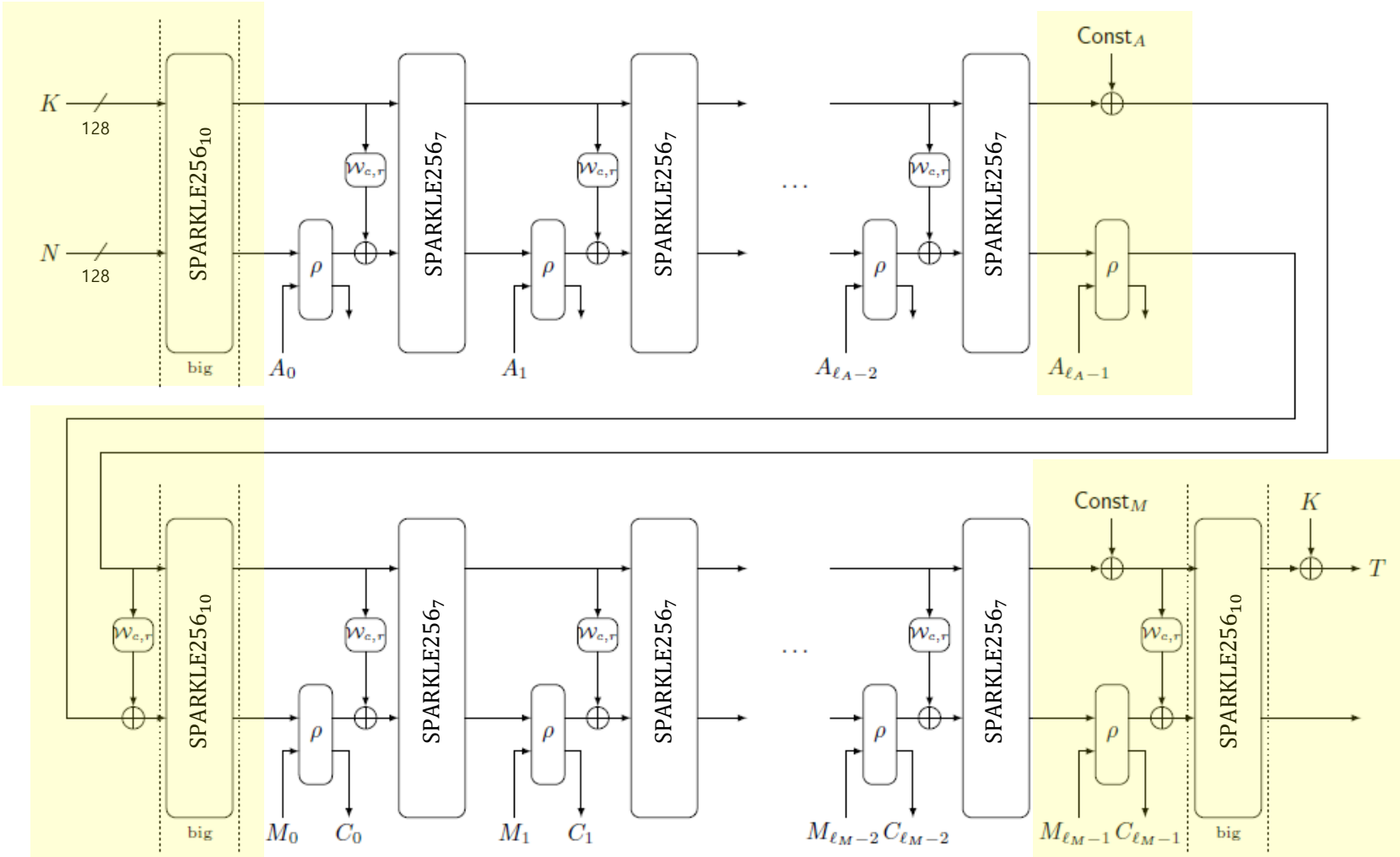
SCHWAEMM128-128 구현

Data block size: 16-bytes(128-bits)

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

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

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



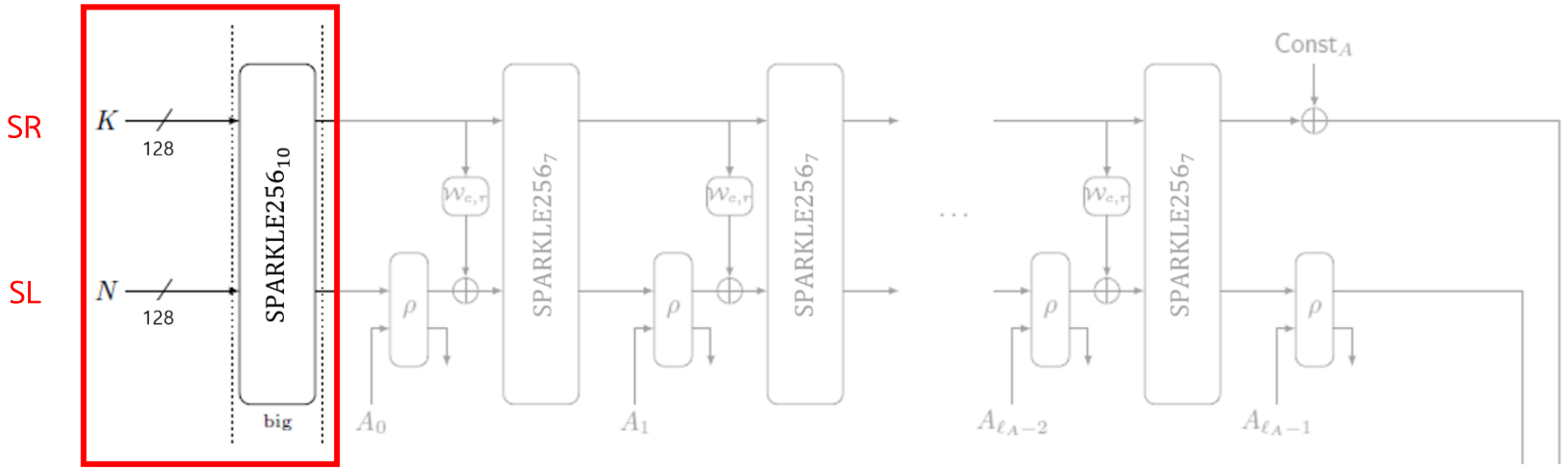
The Authenticated Encryption Algorithm SCHWAEMM128-128 with rate r=128 and capacity c=128

SCHWAEMM128-128 구현

▷ State initialization

$S_L \| S_R \leftarrow \text{SPARKLE256}_{10}(N \| K)$ with $|S_L| = 128$ and $|S_R| = 128$

▷ Processing of associated data




```
S = eng.allocate_qureg(256)
K = eng.allocate_qureg(128)
RC_XOR(eng, Key, K, 128)

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

```
SPARKLE(eng, S, carry, 10)
```

SCHWAEMM128-128 구현

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(128 - len)
X | padding[7] # 0000 0001 0000...
```

```
for i in range(len):
    A_last.append(A[i])
```

```
## padding
for i in range(128 - len):
    A_last.append(padding[i])
```

▷ Padding the associated data and message

if $A \neq \epsilon$ **then**

$A_0 \| A_1 \| \ldots \| A_{\ell_A - 1} \leftarrow A$ with $\forall i \in \{0, \ldots, \ell_A - 2\} : |A_i| = 128$ and $1 \leq |A_{\ell_A - 1}| \leq 128$

if $|A_{\ell_A - 1}| < 128$ **then**

$A_{\ell_A - 1} \leftarrow \text{pad}_{128}(A_{\ell_A - 1})$

$\text{Const}_A \leftarrow 0 \oplus (1 \ll 2)$

1<<2 = 4

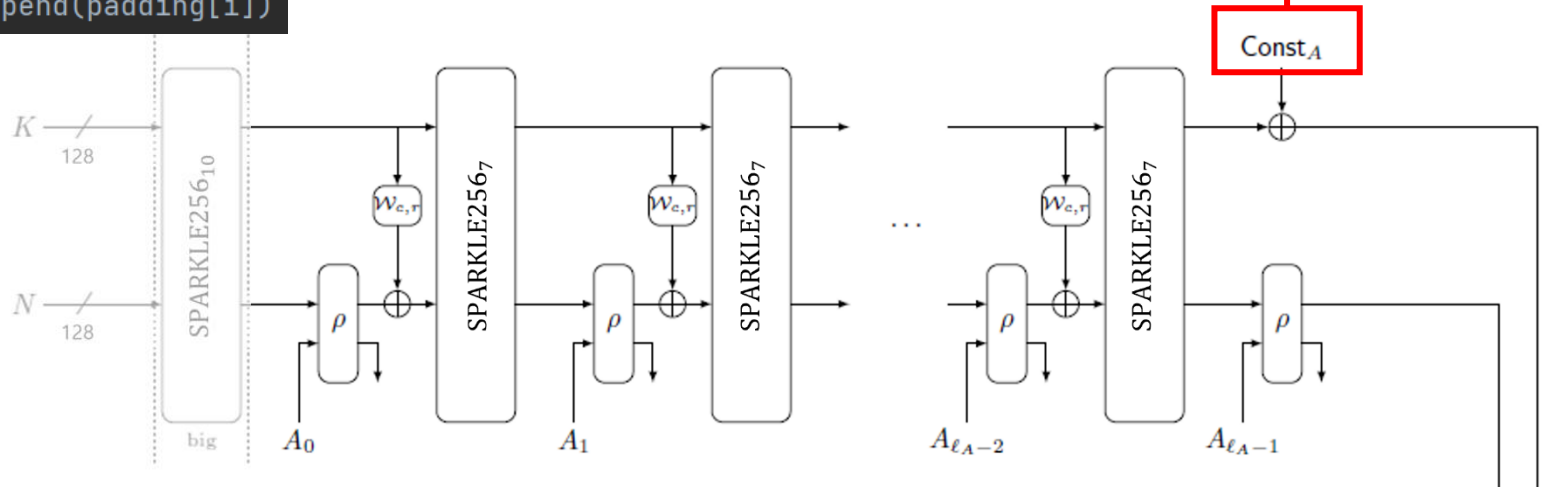
else

$\text{Const}_A \leftarrow 1 \oplus (1 \ll 2)$

end if

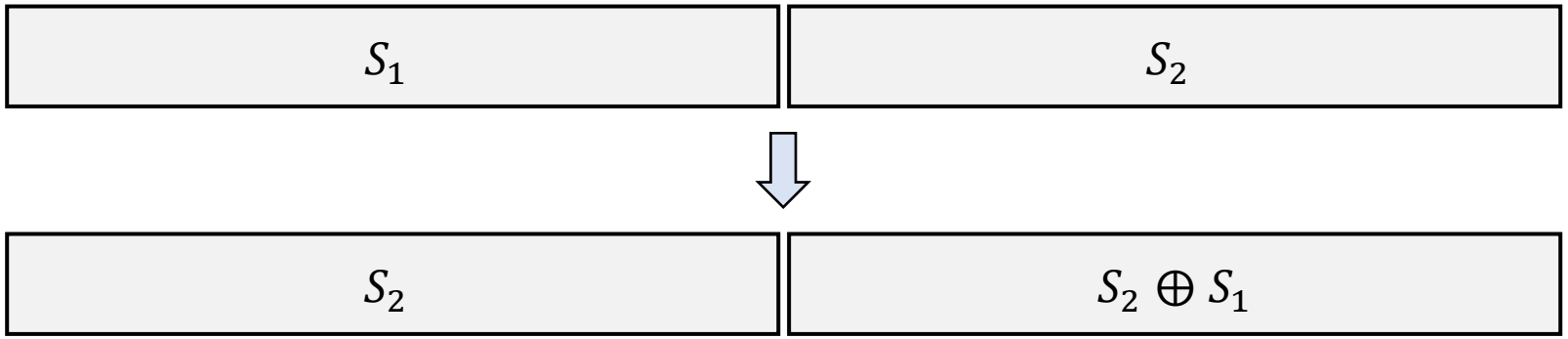
end if

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

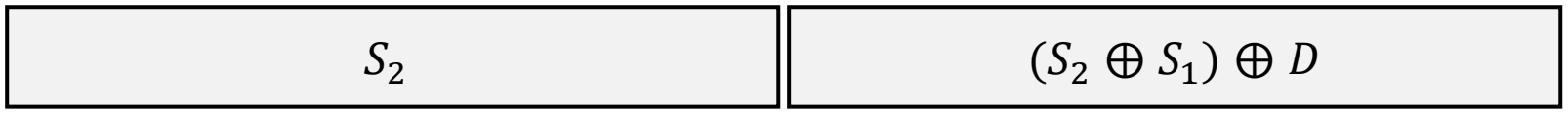


SCHWAEMM128-128 구현

$$\mathsf{FeistelSwap}(S) = S_2 \| (S_2 \oplus S_1)$$



$\rho_1 : (S, D) \mapsto \mathsf{FeistelSwap}(S) \oplus D$



$\rho_2 : (S, D) \mapsto S \oplus D$

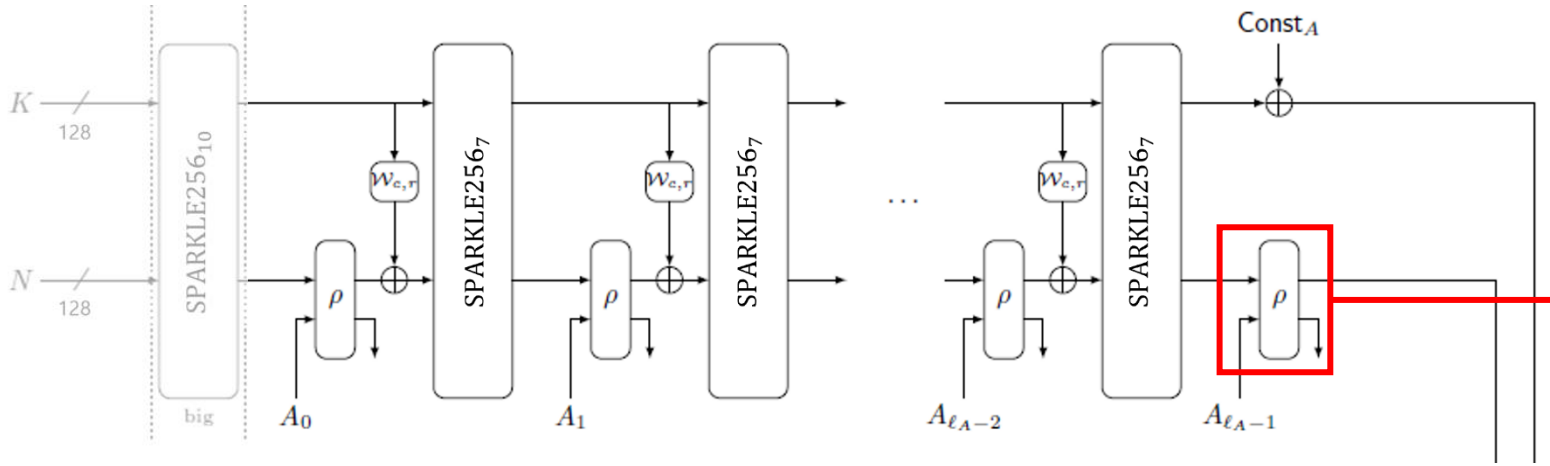
SCHWAEMM128-128 구현

▷ Processing of associated data

if $A \neq \epsilon$ **then**
  for all $j = 0, \ldots, \ell_A - 2$ **do**
    $S_L \| S_R \leftarrow \text{SPARKLE256}_7\big((\rho_1(S_L, A_j) \oplus S_R) \| S_R\big)$
  end for

▷ Finalization if message is empty

$S_L \| S_R \leftarrow \text{SPARKLE256}_{10}\big((\rho_1(S_L, A_{\ell_A-1}) \oplus S_R \oplus \text{Const}_A) \| (S_R \oplus \text{Const}_A)\big)$
end if




```python
def FeistelSwap(eng, s):
    for i in range(32):
        Swap | (s[191-i], s[255-i])
        Swap | (s[223-i], s[159-i])

    for i in range(32):
        CNOT | (s[255-i], s[191-i])
        CNOT | (s[223-i], s[159-i])
```

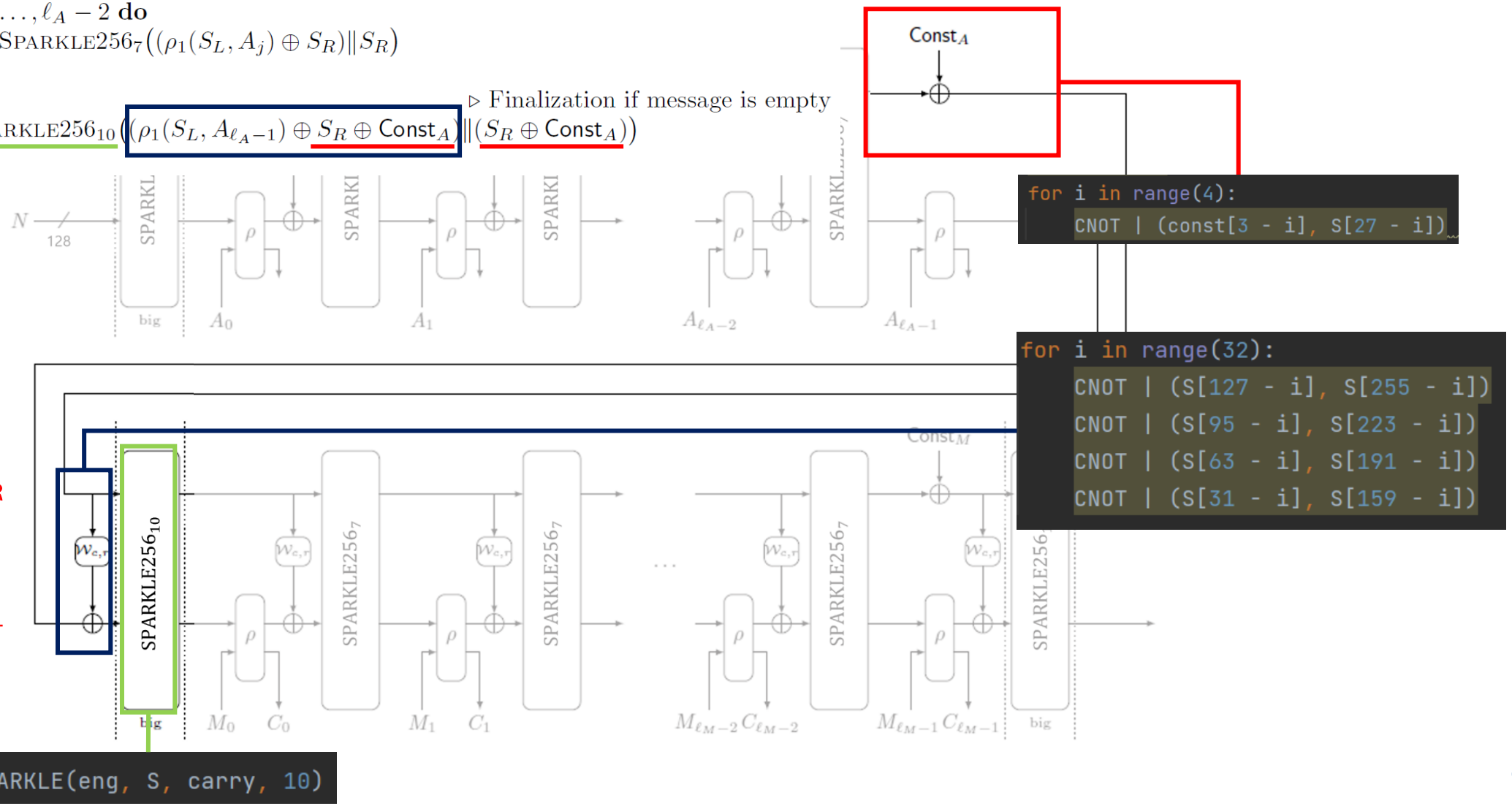
```python
def p1(eng, s, d, carry):
    FeistelSwap(eng, s)

    for i in range(32):
        CNOT | (d[31 - i], s[255 - i])
        CNOT | (d[63 - i], s[223 - i])
```

SCHWAEMM128-128 구현

▷ Processing of associated data

if $A \neq \epsilon$ **then**
  for all $j = 0, \ldots, \ell_A - 2$ **do**
    $S_L \| S_R \leftarrow \text{SPARKLE256}_7\big((\rho_1(S_L, A_j) \oplus S_R) \| S_R\big)$
  end for

  ▷ Finalization if message is empty
  $S_L \| S_R \leftarrow \text{SPARKLE256}_{10}\big((\rho_1(S_L, A_{\ell_A - 1}) \oplus S_R \oplus \text{Const}_A) \| (S_R \oplus \text{Const}_A)\big)$
end if




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

```
for i in range(32):
    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(eng, S, carry, 10)
```

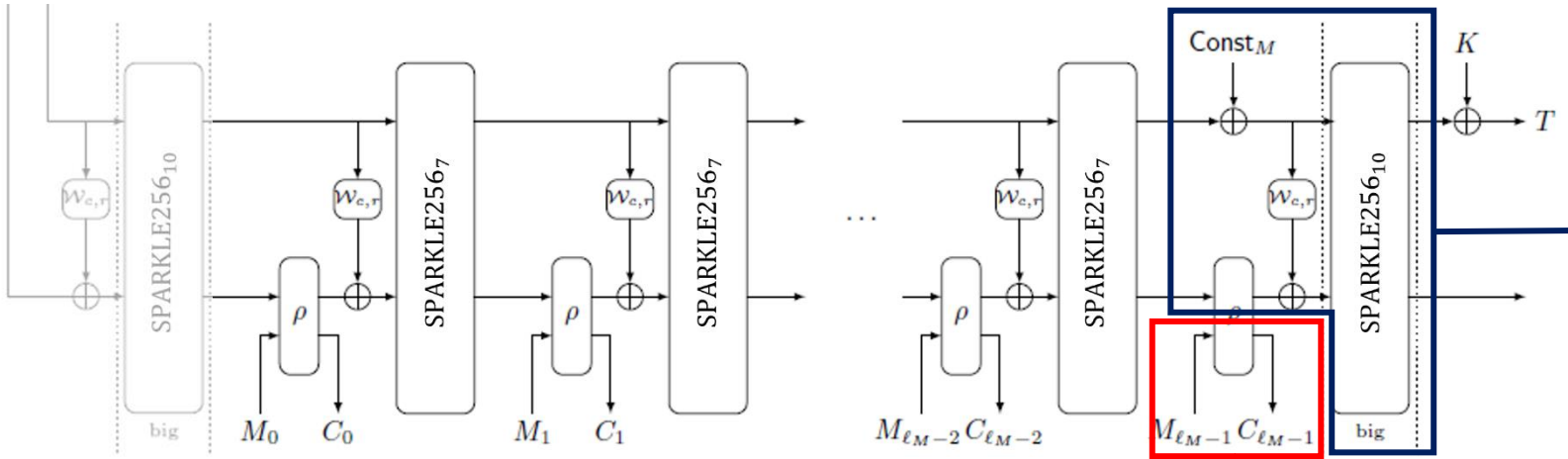
SCHWAEMM128-128 구현

▷ Encrypting

if $M \neq \epsilon$ **then**
  for all $j = 0, \ldots, \ell_M - 2$ **do**
    $C_j \leftarrow \rho_2(S_L, M_j)$
    $S_L \| S_R \leftarrow \text{SPARKLE256}_7\big((\rho_1(S_L, M_j) \oplus S_R) \| S_R\big)$
  end for
  $C_{\ell_M - 1} \leftarrow \text{trunc}_t\big(\rho_2(S_L, M_{\ell_M - 1})\big)$

▷ Finalization

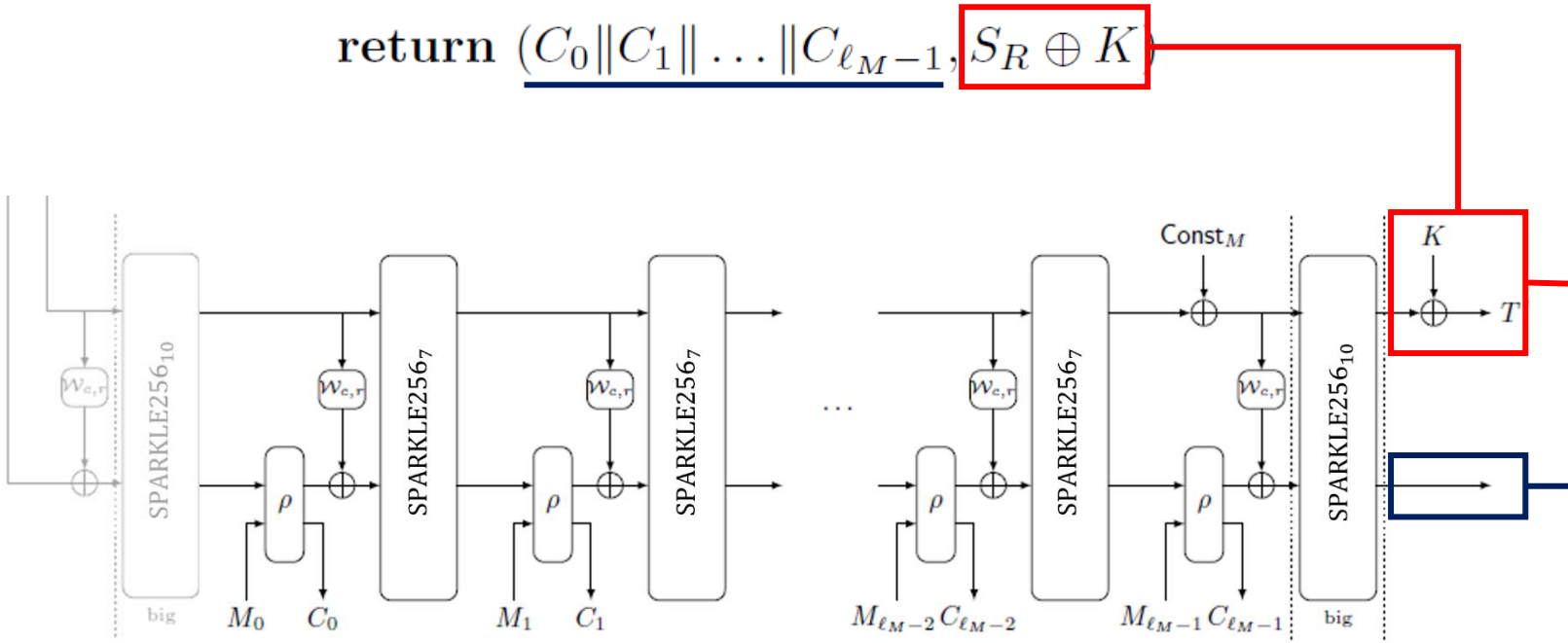
  $S_L \| S_R \leftarrow \text{SPARKLE256}_{10}\big((\rho_1(S_L, M_{\ell_M - 1}) \oplus S_R \oplus \text{Const}_M) \| (S_R \oplus \text{Const}_M)\big)$
end if




```
X | const[1] # 0110
for i in range(4):
    CNOT | (const[3 - i], S[27 - i])

p1(eng, S, M_last, carry)

## SL ^ SR
for i in range(32):
    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(eng, S, carry, 10)
```

```
def trunc(eng, s, c, n):
    for i in range(n):
        CNOT | (s[255 - i], c[31 - i])
```

SCHWAEMM128-128 구현




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

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

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

SCHWAEMM128-128 구현(결과)

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

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

[출력]
CipherText(128-bit): E21B16E5 3524DA18 39758C58 BA6AE225 ADEB5479

→ E5161BE2 18DA2435 588c7539 25E26ABA 7954EBAD

```
0xe5161be218da2435588c753925e26aba7954ebad


Process finished with exit code 0
```

[python code 결과]

감사합니다