Camellia 알고리즘 분석

https://youtu.be/Dp_EFGX0Bs8

1. 알고리즘 개요

- 일본의 통신회사 NTT(Nippon Telegraph and Telephone)와 Mitshubishi(미쓰비시) 기업이 공동으로 개발한 블록암호.

- CRYPTREC(일본 암호기술 평가 프로젝트)
  - 일본 정부기관과 산업계에서 사용될 암호 알고리즘을 평가 및 선정하는 공식 프로젝트. 2003년, 일본 정부의 공식적인 암호 평가 프로젝트인 CRYPTREC에서 추천 알고리즘으로 선정.

CRYPTREC
Cryptography Research and Evaluation Committees

FY	Guideline	Document ID
2024	"CRYPTREC Cryptographic Technology Guideline - Post-Quantum Cryptography - 2024 Edition"	CRYPTREC GL-2007-2024
2023	"CRYPTREC Cryptographic Technology Guideline - Lightweight Cryptography - 2023 Edition"	CRYPTREC GL-2006-2023
2022	"CRYPTREC Cryptographic Technology Guideline - Post-Quantum Cryptography -"	CRYPTREC GL-2004-2022
	"CRYPTREC Cryptographic Technology Guideline - Advanced Cryptography -"	CRYPTREC GL-2005-2022
2018	"CRYPTREC Cryptographic Technology Guideline - SHA-1"	CRYPTREC GL-2001-2013R1
2016	"CRYPTREC Cryptographic Technology Guideline - Lightweight Cryptography -"	CRYPTREC GL-2003-2016JP
	"CRYPTREC Cryptographic Technology Guideline - Lightweight Cryptography - (English Version)"	CRYPTREC GL-2003-2016EN
2013	"CRYPTREC Cryptographic Technology Guideline - Countermeasures against recent attacks on SSL/TLS"	CRYPTREC GL-2002-2013

1. 알고리즘 개요

- ISO/IEC 국제 표준화
  - 2005년, 국제 표준화 기국 ISO/IEC의 블록 암호 표준(ISO/IEC 18033-3)에 포함
  - ISO/IEC 18033-3: 국제표준화기구(ISO)와 국제전기기술위원회(IEC)가 공동으로 제정한 국제 표준으로, 데이터 암호화 알고리즘에 대한 명세를 정의한 표준 시리즈 중 하나
    - 블록 암호 알고리즘의 선정 및 명세화

알고리즘	개발 국가/기관	블록 크기(bit)	키 길이(bit)
AES	미국(NIST)	128	128, 192, 256
Camellia	일본(NTT, Mitsubishi)	128	128, 192, 256
SEED	대한민국(KISA)	128	128
SM4	중국(국가암호관리국)	128	128

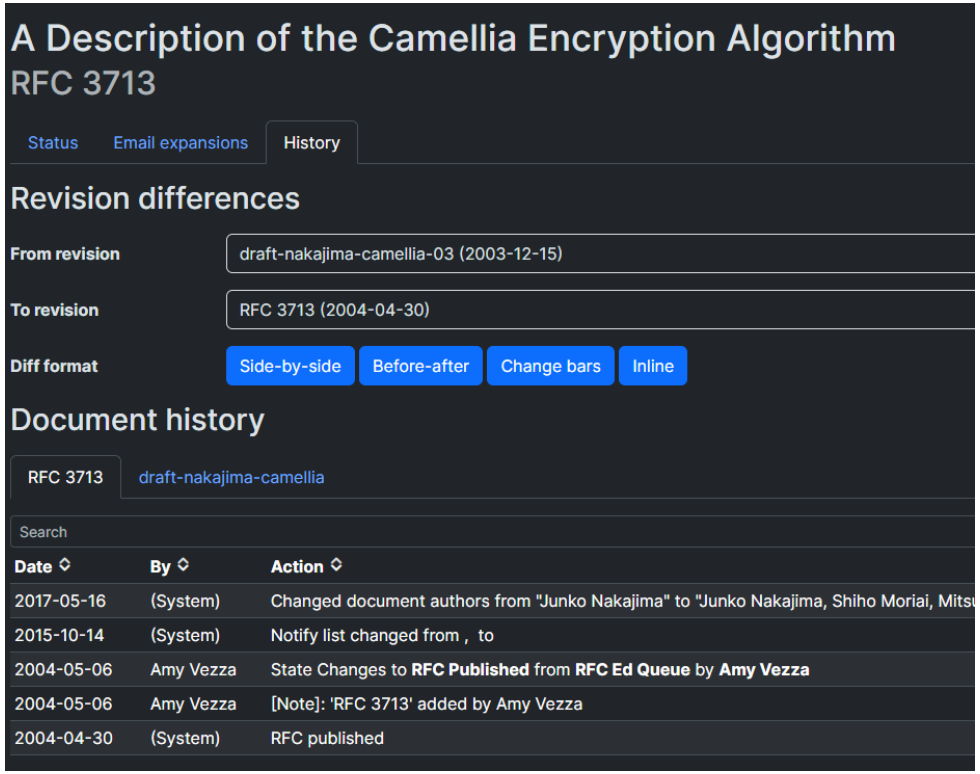
1. 알고리즘 개요

- IETF RFC 채택
  - 인터넷 프로토콜의 기술 표준을 관리하는 기관 IETF에서 Camellia를 TLS, IPsec등 주요 보안 프로토콜에 적용할 수 있도록 RFC 3713을 통해 공식적으로 권장.

IETF는 인터넷 표준 프로토콜의 개발과 기술적 지침 제공을 담당하는 국제 비영리 단체

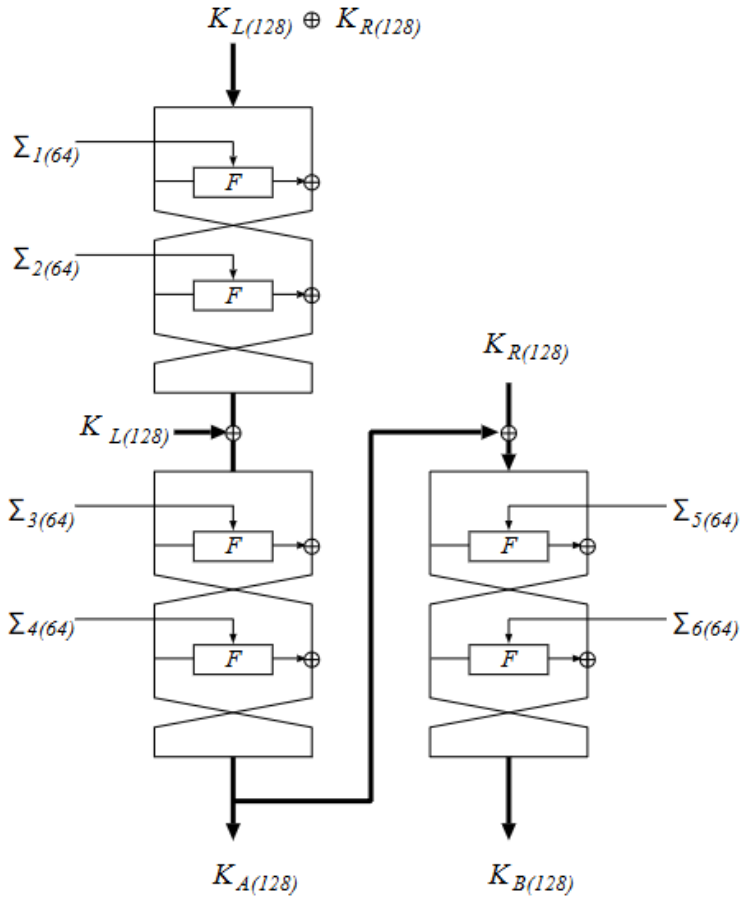
RFC(Request for Comments)는 IETF에서 관리하는 일련의 문서 시리즈로, 인터넷 기술 및 프로토콜의 표준을 정의하고 기술적 세부 사항을 명세화한 문서

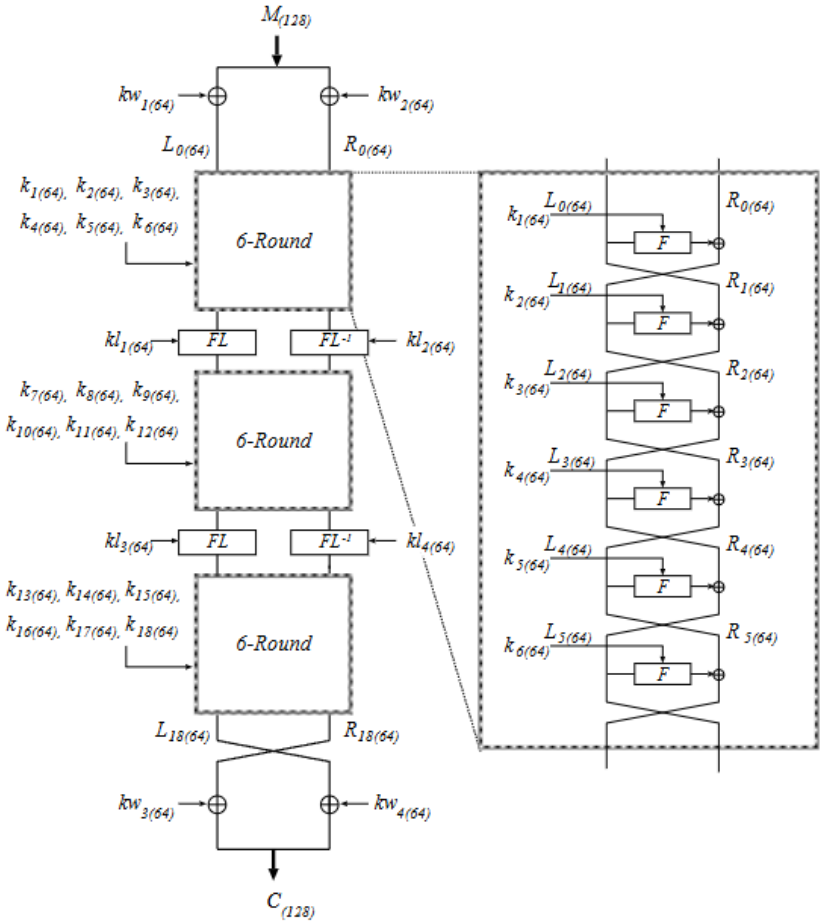
RFC 문서로 등록되었다는 것은, 인터넷 프로토콜 환경에서 해당 암호 알고리즘을 구현할 때 표준 참조 문서로서 활용할 수 있음을 의미



2. Camellia

- 기본 특징
  - Feistel 구조, 128비트 블록크기, 128/192/256 키크기, 18라운드/24라운드





2. Camellia

- 키 스케쥴

$\lll_n$ The left circular rotation of the operand by n bits.

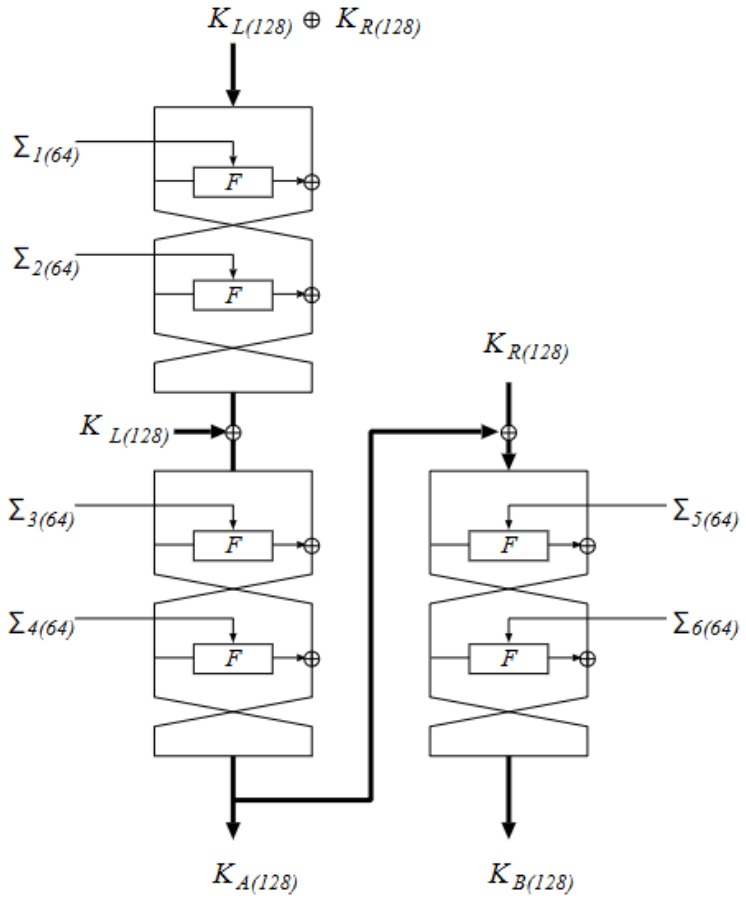
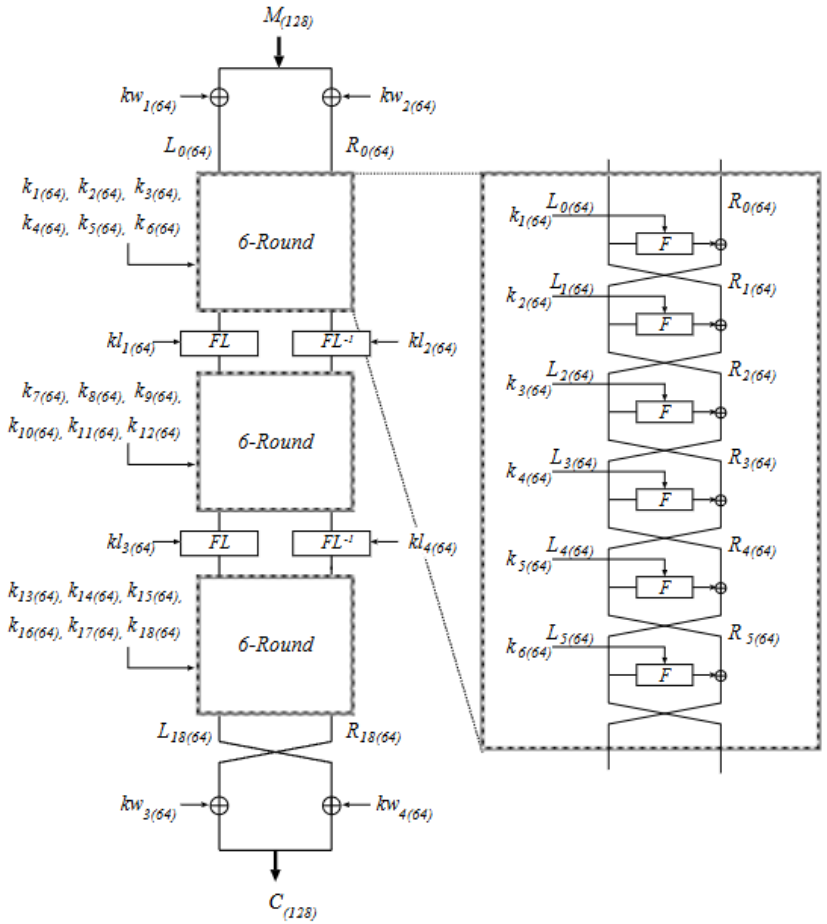


Table 2: Subkeys for 128-bit secret key

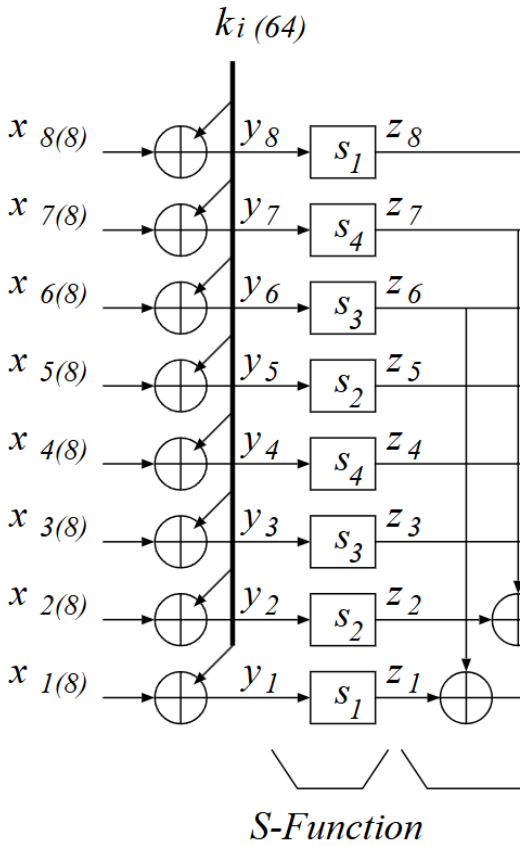
	subkey	value
Prewhitening	$kw_{1(64)}$	$(K_L \lll_0)_{L(64)}$
	$kw_{2(64)}$	$(K_L \lll_0)_{R(64)}$
F (Round1)	$k_{1(64)}$	$(K_A \lll_0)_{L(64)}$
F (Round2)	$k_{2(64)}$	$(K_A \lll_0)_{R(64)}$
F (Round3)	$k_{3(64)}$	$(K_L \lll_{15})_{L(64)}$
F (Round4)	$k_{4(64)}$	$(K_L \lll_{15})_{R(64)}$
F (Round5)	$k_{5(64)}$	$(K_A \lll_{15})_{L(64)}$
F (Round6)	$k_{6(64)}$	$(K_A \lll_{15})_{R(64)}$
FL	$kl_{1(64)}$	$(K_A \lll_{30})_{L(64)}$
FL^{-1}	$kl_{2(64)}$	$(K_A \lll_{30})_{R(64)}$
F (Round7)	$k_{7(64)}$	$(K_L \lll_{45})_{L(64)}$
F (Round8)	$k_{8(64)}$	$(K_L \lll_{45})_{R(64)}$
F (Round9)	$k_{9(64)}$	$(K_A \lll_{45})_{L(64)}$
F (Round10)	$k_{10(64)}$	$(K_L \lll_{60})_{R(64)}$
F (Round11)	$k_{11(64)}$	$(K_A \lll_{60})_{L(64)}$
F (Round12)	$k_{12(64)}$	$(K_A \lll_{60})_{R(64)}$
FL	$kl_{3(64)}$	$(K_L \lll_{77})_{L(64)}$
FL^{-1}	$kl_{4(64)}$	$(K_L \lll_{77})_{R(64)}$
F (Round13)	$k_{13(64)}$	$(K_L \lll_{94})_{L(64)}$
F (Round14)	$k_{14(64)}$	$(K_L \lll_{94})_{R(64)}$
F (Round15)	$k_{15(64)}$	$(K_A \lll_{94})_{L(64)}$
F (Round16)	$k_{16(64)}$	$(K_A \lll_{94})_{R(64)}$
F (Round17)	$k_{17(64)}$	$(K_L \lll_{111})_{L(64)}$
F (Round18)	$k_{18(64)}$	$(K_L \lll_{111})_{R(64)}$
Postwhitening	$kw_{3(64)}$	$(K_A \lll_{111})_{L(64)}$
	$kw_{4(64)}$	$(K_A \lll_{111})_{R(64)}$



2. Camellia

$$S : \mathbf{L} \longrightarrow \mathbf{L}$$
$$l_{1(8)}||l_{2(8)}\ |l_{3(8)}||l_{4(8)}|\ l_{5(8)}||l_{6(8)}\ |l_{7(8)}\ |l_{8(8)} \longmapsto l'_{1(8)}\ |l'_{2(8)}||l'_{3(8)}||l'_{4(8)}||l'_{5(8)}\ |l'_{6(8)}||l'_{7(8)}||l'_{8(8)}$$

- S-Function



S-Function

$$s_1 : \mathbf{B} \longrightarrow \mathbf{B}$$
$$x_{(8)} \longmapsto \mathbf{h}(\mathbf{g}(\mathbf{f}(\mathtt{0xc5} \oplus x_{(8)}))) \oplus \mathtt{0x6e},$$
$$s_2 : \mathbf{B} \longrightarrow \mathbf{B}$$
$$x_{(8)} \longmapsto s_1(x_{(8)}) \lll_1,$$
$$s_3 : \mathbf{B} \longrightarrow \mathbf{B}$$
$$x_{(8)} \longmapsto s_1(x_{(8)}) \ggg_1,$$
$$s_4 : \mathbf{B} \longrightarrow \mathbf{B}$$
$$x_{(8)} \longmapsto s_1(x_{(8)} \lll_1),$$

$$\mathbf{g} : \mathbf{B} \longrightarrow \mathbf{B}$$
$$a_{1(1)}|\ a_{2(1)}||a_{3(1)}\ |a_{4(1)}||a_{5(1)}||a_{6(1)}\ |a_{7(1)}||a_{8(1)}$$
$$\longmapsto b_{1(1)}||b_{2(1)}||b_{3(1)}||b_{4(1)}||b_{5(1)}\ |b_{6(1)}||b_{7(1)}|\ b_{8(1)},$$

$$(b_8 + b_7\alpha + b_6\alpha^2 + b_5\alpha^3) + (b_4 + b_3\alpha + b_2\alpha^2 + b_1\alpha^3)\beta$$
$$= 1/((a_8 + a_7\alpha + a_6\alpha^2 + a_5\alpha^3) + (a_4 + a_3\alpha + a_2\alpha^2 + a_1\alpha^3)\beta).$$

$$\mathbf{f} : \mathbf{B} \longrightarrow \mathbf{B}$$
$$a_{1(1)}|\ a_{2(1)}||a_{3(1)}\ |a_{4(1)}||a_{5(1)}||a_{6(1)}\ |a_{7(1)}||a_{8(1)}$$
$$\longmapsto b_{1(1)}||b_{2(1)}||b_{3(1)}||b_{4(1)}||b_{5(1)}\ |b_{6(1)}||b_{7(1)}|\ b_{8(1)},$$

$$\mathbf{h} : \mathbf{B} \longrightarrow \mathbf{B}$$
$$a_{1(1)}|\ a_{2(1)}||a_{3(1)}\ |a_{4(1)}||a_{5(1)}||a_{6(1)}\ |a_{7(1)}||a_{8(1)}$$
$$\longmapsto b_{1(1)}||b_{2(1)}||b_{3(1)}||b_{4(1)}||b_{5(1)}\ |b_{6(1)}||b_{7(1)}|\ b_{8(1)},$$

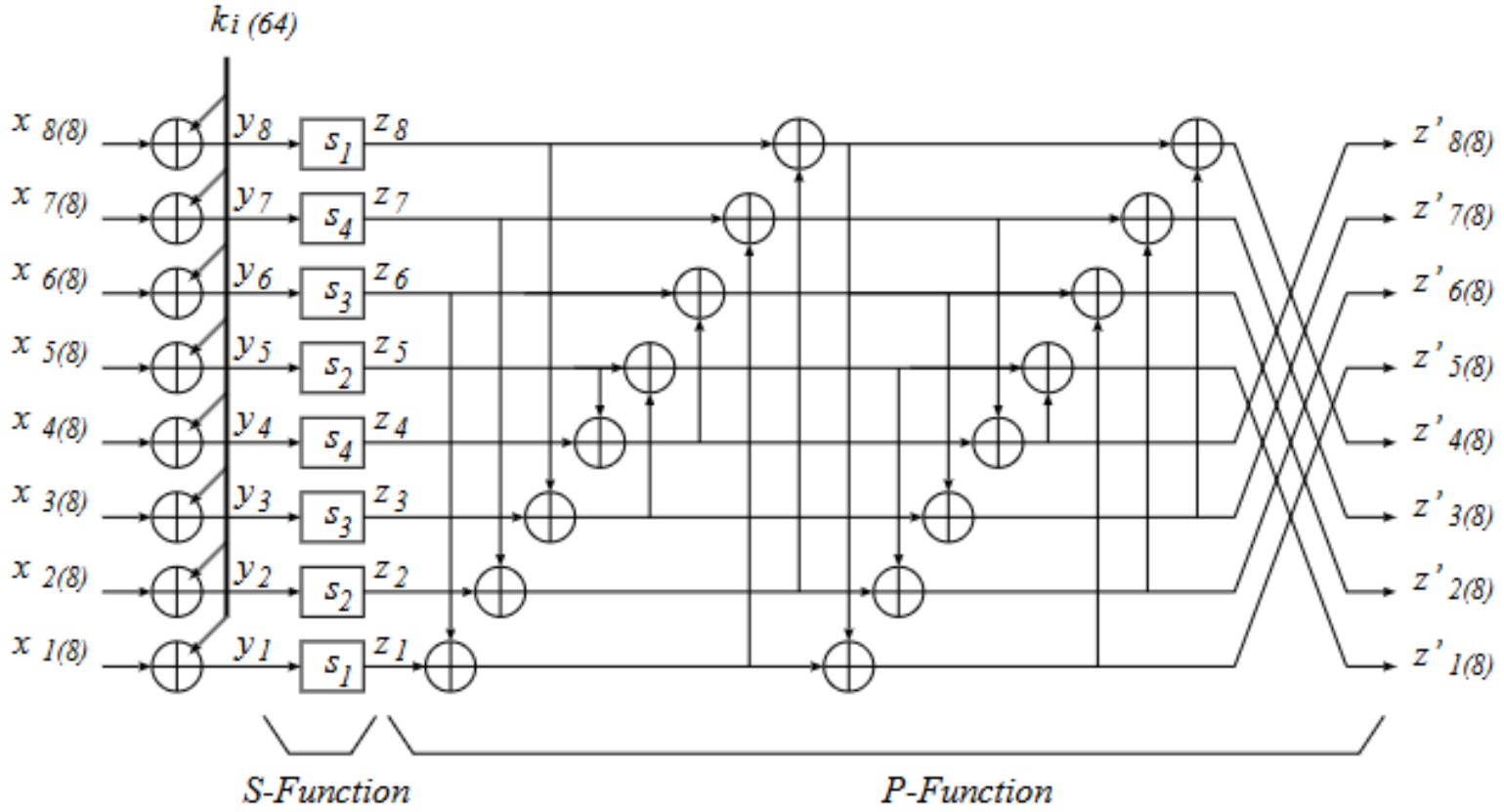
$$b_1 = a_6 \oplus a_2,$$
$$b_2 = a_7 \oplus a_1,$$
$$b_3 = a_8 \oplus a_5 \oplus a_3,$$
$$b_4 = a_8 \oplus a_3,$$
$$b_5 = a_7 \oplus a_4,$$
$$b_6 = a_5 \oplus a_2,$$
$$b_7 = a_8 \oplus a_1,$$
$$b_8 = a_6 \oplus a_4.$$

$$b_1 = a_5 \oplus a_6 \oplus a_2,$$
$$b_2 = a_6 \oplus a_2,$$
$$b_3 = a_7 \oplus a_4,$$
$$b_4 = a_8 \oplus a_2,$$
$$b_5 = a_7 \oplus a_3,$$
$$b_6 = a_8 \oplus a_1,$$
$$b_7 = a_5 \oplus a_1,$$
$$b_8 = a_6 \oplus a_3.$$

2. Camellia

- openSSL Camellia F function 코드

```
#define Camellia_Feistel(_s0, _s1, _s2, _s3, _key) \
    do                                             \
    {                                              \
        register u32 _t0, _t1, _t2, _t3;           \
                                                   \
        _t0 = _s0 ^ (_key)[0];                     \
        _t3 = SBOX4_4404[_t0 & 0xff];              \
        _t1 = _s1 ^ (_key)[1];                     \
        _t3 ^= SBOX3_3033[(_t0 >> 8) & 0xff];      \
        _t2 = SBOX1_1110[_t1 & 0xff];              \
        _t3 ^= SBOX2_0222[(_t0 >> 16) & 0xff];     \
        _t2 ^= SBOX4_4404[(_t1 >> 8) & 0xff];      \
        _t3 ^= SBOX1_1110[(_t0 >> 24)];            \
        _t2 ^= _t3;                                \
        _t3 = RightRotate(_t3, 8);                 \
        _t2 ^= SBOX3_3033[(_t1 >> 16) & 0xff];     \
        _s3 ^= _t3;                                \
        _t2 ^= SBOX2_0222[(_t1 >> 24)];            \
        _s2 ^= _t2;                                \
        _s3 ^= _t2;                                \
    } while (0)
```



3. 구현 계획

- 기존의 ARIA 코드 활용
  - Sbox 병합 후 공유메모리 활용과 __byte_perm() 활용

- GPU CUDA CTR 구현
  - CISC-S 학술대회 제출
  - CUDA CTR/ES -> AES / ARIA / SEED / Camellia 합쳐서 내도 괜찮을거 같음.

- Bitslice 구현 및 ARMv8 병렬 구현
  - 많이 사용되는 거 같지 않아서, 시간될 때?

감 사 합 니 다