NIST 경량암호 후보 ACE에 대한 부채널 분석

https://youtu.be/Le1myl-wx6Q

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

NIST Lightweight Cryptography

- 제약된 환경의 장치가 서로 연결되어 일부 작업을 수행하기 위해 무선 통신으로 협력하는 새로운 영역이 생겨남
  (센서 네트워크, 의료, 분산 제어 시스템, 사물 인터넷)
- 스마트 카드가 점점 더 일반화
- 데스크탑 컴퓨터에서 소형 컴퓨터로의 전환 대부분
  암호화 알고리즘의 대부분은기존의 데스크톱 및 서버 환경에 최적화 되어있음
- NIST 암호화 표준을 수용 하지 못하는 제한된 환경에서 적합한 경량 암호화 알고리즘을 요청

제출된 57개
Round1 56개
Round 2 32개 (현재)
(2019 년 8 월 발표)
12 개월동안 지속될 것으로 예상

NIST Lightweight Cryptography

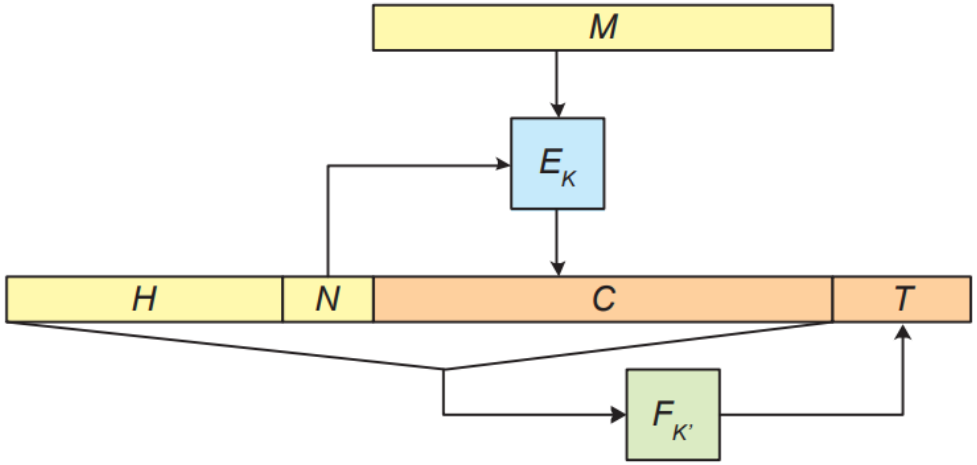
- 관련 데이터를 통한 인증된 암호화 (AEAD) 기능을 사용하여 인증 된 암호화를 구현
  (선택적으로 해싱 기능도 구현)
- 현재 NIST 표준과 비교하여 제한된 환경에서 훨씬 더 나은 성능을 발휘 할 것
- 단문 메시지 (예 : 8 바이트)에 효율적으로 최적화 할 것
- RAM 및 ROM 사용량이 적은 소형 하드웨어 구현 및 임베디드 소프트웨어 구현이 가능해야 할 것
- ASIC 및 FPGA의 성능은 광범위한 표준 셀 라이브러리를 고려해야함
- 다양한 구현 전략 (저에너지, 저전력, 낮은 대기 시간)을 지원할 수 있도록 알고리즘이 유연해야 할 것
- 8 비트, 16 비트 및 32 비트 마이크로 컨트롤러 아키텍처를 고려해야함
- AEAD 알고리즘 및 선택적 해시 함수 알고리즘의 구현은 타이밍 공격, SPA / DPA 및 SEMA / DEMA를 포함한 다양한 사이드 채널 공격에 대한 대책에 적합해야 함

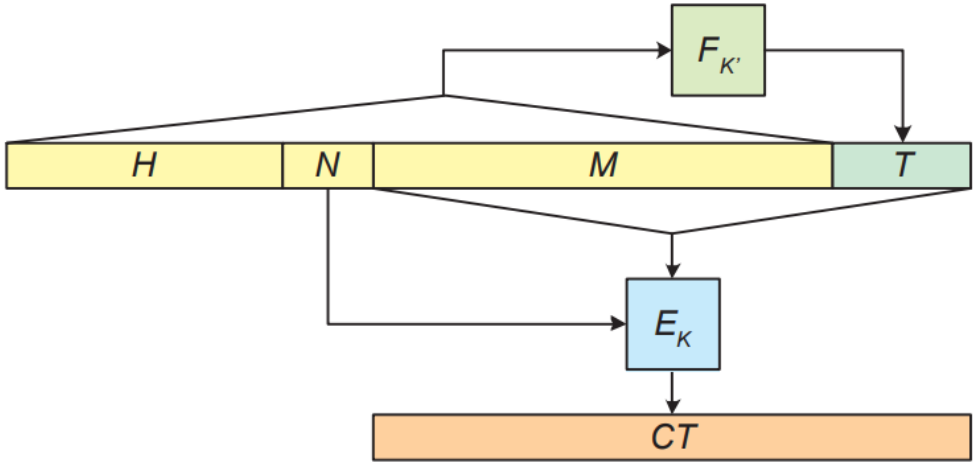
제출된 57개
Round1 56개
Round 2 32개 (현재)
(2019 년 8 월 발표)
12 개월동안 지속될 것으로 예상

AEAD(Authenticated Encryption with Associated Data)

- AEAD기능을 사용하여 인증 된 암호화를 구현이 조건
- AEAD는 연관 데이터(Associated Data,AD)를 암호문 및 표시될 컨텍스트에 결합
- 기밀성(confident), 무결성(integrity)및 진위성(authenticity)을 보장

$$(K, N, AD, M) = (C, T)$$





ACE

- Bitwise XOR 및 AND, 왼쪽 순환 이동 및 64비트 워드 셔플
- 비선형 레이어 : 64 비트의 블록 크기의 키가 없는 라운드 축소 Simeck 블록 암호 사용
  우수한 암호화 속성과 낮은 하드웨어 비용
- 선형 레이어 : 5 개의 64 비트 단어가 (3,2,0,4,1) 순서로 표시
  이는 다양한 암호화 및 선형 암호화 분석에 대해 우수한 저항을 제공
- 일체형 프리미티브, 동일한 하드웨어 회로를 사용하여 AEAD 및 해싱 기능을 모두 제공
- 통합 스폰지 이중 모드 사용, 128 비트 보안

ACE

ACE-$\mathcal{AE}$-128

$$\text{ACE-}\mathcal{E}(K, N, AD, M) = (C, T)$$
$$\text{ACE-}\mathcal{D}(K, N, AD, C, T) \in \{M, \perp\}$$
$$H = \text{ACE-}\mathcal{H}\text{-}h(M, IV)$$

Functionality	Algorithm	r	k	n	t	$\log_2(d)$	h	iv
AEAD	ACE-$\mathcal{AE}$-128	64	128	128	128	124	-	-
Hash	ACE-$\mathcal{H}$-256	64	-	-	-	-	256	24

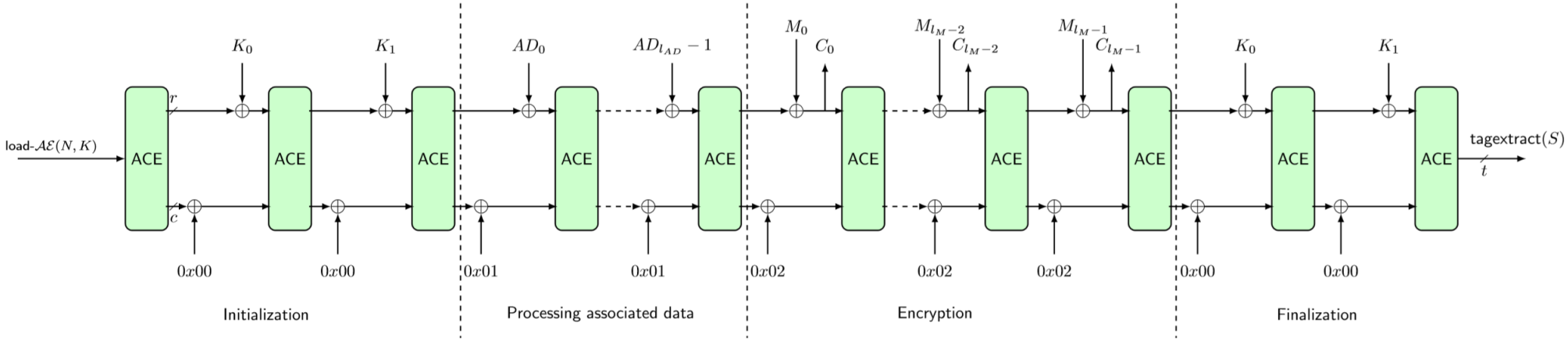
ACE-AEAD

- Initialization : 초기화 단계는 암호화 키와 IV를 입력. 먼저 전체 상태가 0으로 초기화
- Additional Data Processing : 초기화 단계 후 관련 데이터 상태를 업데이트
- Encryption : 암호화
- Decyption : 복호화
- Finalization : Tag 계산

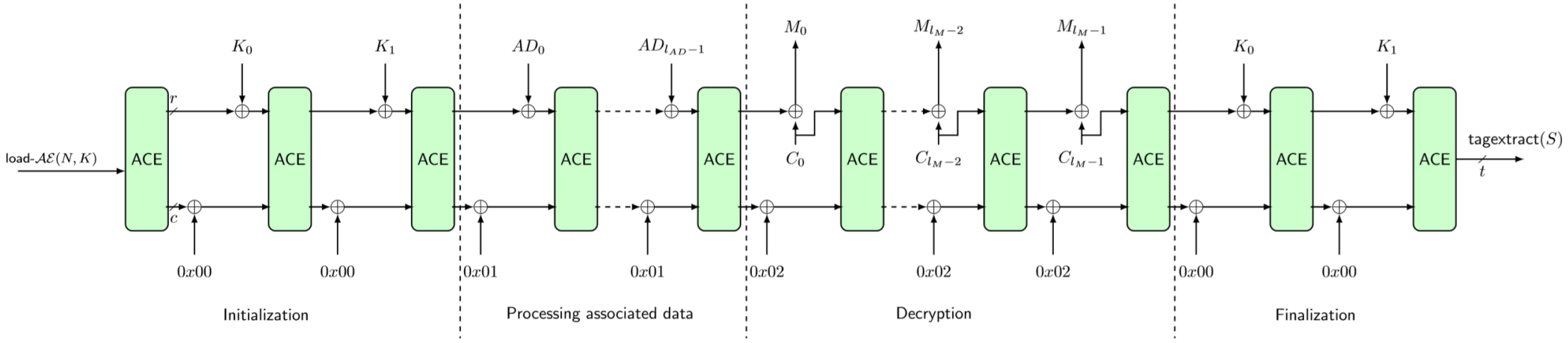
1: Authenticated encryption ACE-$\mathcal{E}(K, N, AD, M)$:
2: $S \leftarrow$ Initialization(N, K)
3: **if** $|AD| \neq 0$ **then:**
4: $S \leftarrow$ Processing-Associated-Data(S, AD)
5: $(S, C) \leftarrow$ Encyption(S, M)
6: $T \leftarrow$ Finalization(S, K)
7: **return** (C, T)

1: Verified decryption ACE-$\mathcal{D}(K, N, AD, C, T)$:
2: $S \leftarrow$ Initialization(N, K)
3: **if** $|AD| \neq 0$ **then:**
4: $S \leftarrow$ Processing-Associated-Data(S, AD)
5: $(S, M) \leftarrow$ Decyption(S, C)
6: $T' \leftarrow$ Finalization(S, K)
7: **if** $T' \neq T$ **then:**
8: **return** $\perp$
9: **else:**
10: **return** M

ACE-AEAD



(a) Authenticated encryption algorithm ACE-$\mathcal{E}$



(b) Verified decryption algorithm ACE-$\mathcal{D}$

ACE-AEAD 부채널 대응

- 캐시 타이밍 공격에 대한 내성을 제공하고
- 여러 개의 독립적 인 ACE 인스턴스를 병렬로 실행할 수있는 SIMD 명령 세트를 사용하여 비트 슬라이스 방식으로 ACE를 구현
- SSE 및 AVX 명령어 세트가 각각 XMM 및 YMM이라고하는 128 비트 및 256 비트 SIMD 레지스터를 지원하는 Intel 프로세서의 SSE 및 AVX 명령어 세트를 고려

ACE 부채널 공격

- ACE와 유사하며 AEAD 구조인 ACORN 암호에 대한 공격
  AEPractical Algebraic Side-Channel Attacks Against ACORN: 21st International Conference, Seoul, South Korea, November 28–30, 2018, Revised Selected Papers
- algebraic side channel attack (ASCA)을 사용 Initialization 함수 부분을 공격
- IV와 Xor 하는 부분, f128에서 f209까지 처음 82bit의 IV에 대한 지식이 있으면 된다

Algorithm 1 $\texttt{StateUpdate}(S^i, m_i, ca, cb)$

$S^i_{289} \leftarrow S^i_{289} \oplus S^i_{235} \oplus S^i_{230}$ ▷ Update using six LFSRs
$S^i_{230} \leftarrow S^i_{230} \oplus S^i_{196} \oplus S^i_{193}$
$S^i_{193} \leftarrow S^i_{193} \oplus S^i_{160} \oplus S^i_{154}$
$S^i_{154} \leftarrow S^i_{154} \oplus S^i_{111} \oplus S^i_{107}$
$S^i_{107} \leftarrow S^i_{107} \oplus S^i_{66} \oplus S^i_{61}$
$S^i_{61} \leftarrow S^i_{61} \oplus S^i_{23} \oplus S^i_0$
$ks_i \leftarrow \kappa(S^i)$
$c_i \leftarrow ks_i \oplus m_i$ ▷ Encryption of the input
$f_i \leftarrow \varphi(S^i, ks_i, ca, cb)$ ▷ Nonlinear feedback bit generation
for j from 0 to 291 **do**
  $S^{i+1}_j \leftarrow S^i_{j+1}$ ▷ Shift the state
$S^{i+1}_{292} \leftarrow f_i \oplus m_i$ ▷ Injection of the input

Algorithm 2 $\texttt{AcornInit}(S^0, K, IV)$

$(S^0_0, ..., S^0_{292}) \leftarrow (0, ..., 0)$ ▷ Initialize the state to zero
for i from 0 to 127 **do**
  $S^{i+1} \leftarrow \texttt{StateUpdate}(S^i, K_i, 1, 1)$ ▷ Update the state with key bits as input
for i from 0 to 127 **do**
  $S^{129+i} \leftarrow \texttt{StateUpdate}(S^{128+i}, IV_i, 1, 1)$ ▷ Update the state with IV bits as input
$S^{257} \leftarrow \texttt{StateUpdate}(S^{256}, K_0 \oplus 1, 1, 1)$
for i from 1 to 1535 **do**
  $S^{257+i} \leftarrow \texttt{StateUpdate}(S^{256+i}, K_{i \bmod 128}, 1, 1)$ ▷ Update the state with key bits as input

ACE 부채널 공격

- ACE와 유사하며 AEAD 구조인 ACORN 암호에 대한 공격
  AEPractical Algebraic Side-Channel Attacks Against ACORN: 21st International Conference, Seoul, South Korea, November 28–30, 2018, Revised Selected Papers
- algebraic side channel attack (ASCA)을 사용 Initialization 함수 부분을 공격
- IV와 Xor 하는 부분, f128에서 f209까지 처음 82bit의 IV에 대한 지식이 있으면 된다

$$\left(S_0^{128}, ..., S_{164}^{128}\right) = (0, ..., 0)$$
$$\left(S_{165}^{128}, ..., S_{198}^{128}\right) = (\neg K_0, ..., \neg K_{33})$$
$$\left(S_{199}^{128}, ..., S_{201}^{128}\right) = (K_{34} \oplus K_0, ..., K_{36} \oplus K_2)$$
$$\left(S_{202}^{128}, ..., S_{218}^{128}\right) = (\neg K_{37} \oplus K_3 \oplus K_0, ..., \neg K_{53} \oplus K_{19} \oplus K_{16})$$
$$\left(S_{219}^{128}, ..., S_{223}^{128}\right) = (K_{54} \oplus K_{20} \oplus K_{17} \oplus K_0, ..., K_{58} \oplus K_{24} \oplus K_{21} \oplus K_4)$$
$$\left(S_{224}^{128}, ..., S_{229}^{128}\right) = (\neg K_{59} \oplus K_{25} \oplus K_{22} \oplus K_5 \oplus K_0, ..., \neg K_{64} \oplus K_{30} \oplus K_{27} \oplus K_{10} \oplus K_5)$$
$$\left(S_{230}^{128}, ..., S_{261}^{128}\right) = (\neg K_{65} \oplus K_{11} \oplus K_6, ..., \neg K_{96} \oplus K_{42} \oplus K_{37})$$
$$\left(S_{262}^{128}, ..., S_{272}^{128}\right) = (K_{97} \oplus K_{43} \oplus K_{38} \oplus f_{97}, ..., K_{107} \oplus K_{53} \oplus K_{48} \oplus f_{107})$$
$$\left(S_{273}^{128}, ..., S_{288}^{128}\right) = (\neg K_{108} \oplus K_{54} \oplus K_{49} \oplus K_0 \oplus f_{108}, ..., \neg K_{123} \oplus K_{69} \oplus K_{64} \oplus K_{15} \oplus f_{123})$$
$$\left(S_{289}^{128}, ..., S_{292}^{128}\right) = (\neg K_{124} \oplus f_{124}, ..., \neg K_{127} \oplus f_{127})$$

$$f_i = \begin{cases} 1 & \text{if } 0 \le i \le 96 \\ K_{i-97} & \text{if } 97 \le i \le 99 \\ \left(\neg K_{i-58}\right) \wedge \left(\neg K_{i-100}\right) \oplus K_{i-97} & \text{if } 100 \le i \le 111 \\ \left(K_{i-58} \oplus K_{i-112}\right) \wedge \left(\neg K_{i-100}\right) \oplus K_{i-97} & \text{if } 112 \le i \le 116 \\ \neg \left(K_{i-58} \oplus K_{i-112} \oplus K_{i-117}\right) \wedge \left(\neg K_{i-100}\right) \oplus K_{i-97} & \text{if } 117 \le i \le 127 \end{cases}$$

ACE 부채널 공격

- Initialization 부분에서

Initialization(N, K):

$S \leftarrow$ load-$\mathcal{AE}(N, K)$

$S \leftarrow \mathsf{ACE}(S)$

for $i = 0$ to 1 **do:**

$S \leftarrow (S_r \oplus K_i, S_c)$

$S \leftarrow \mathsf{ACE}(S)$

return S

load-$\mathcal{AE}(N, K)$

$$A[7], A[6], \cdots, A[0] \leftarrow K_0[7], K_0[6], \cdots, K_0[0]$$
$$C[7], C[6], \cdots, C[0] \leftarrow K_1[7], K_1[6], \cdots, K_1[0]$$
$$B[7], B[6], \cdots, B[0] \leftarrow N_0[7], N_0[6], \cdots, N_0[0]$$
$$E[7], E[6], \cdots, E[0] \leftarrow N_1[7], N_1[6], \cdots, N_1[0]$$
$$D[7], D[6], \cdots, D[0] \leftarrow 0x00, 0x00, \cdots, 0x00$$

ACE 부채널 공격

- Initialization 부분에서

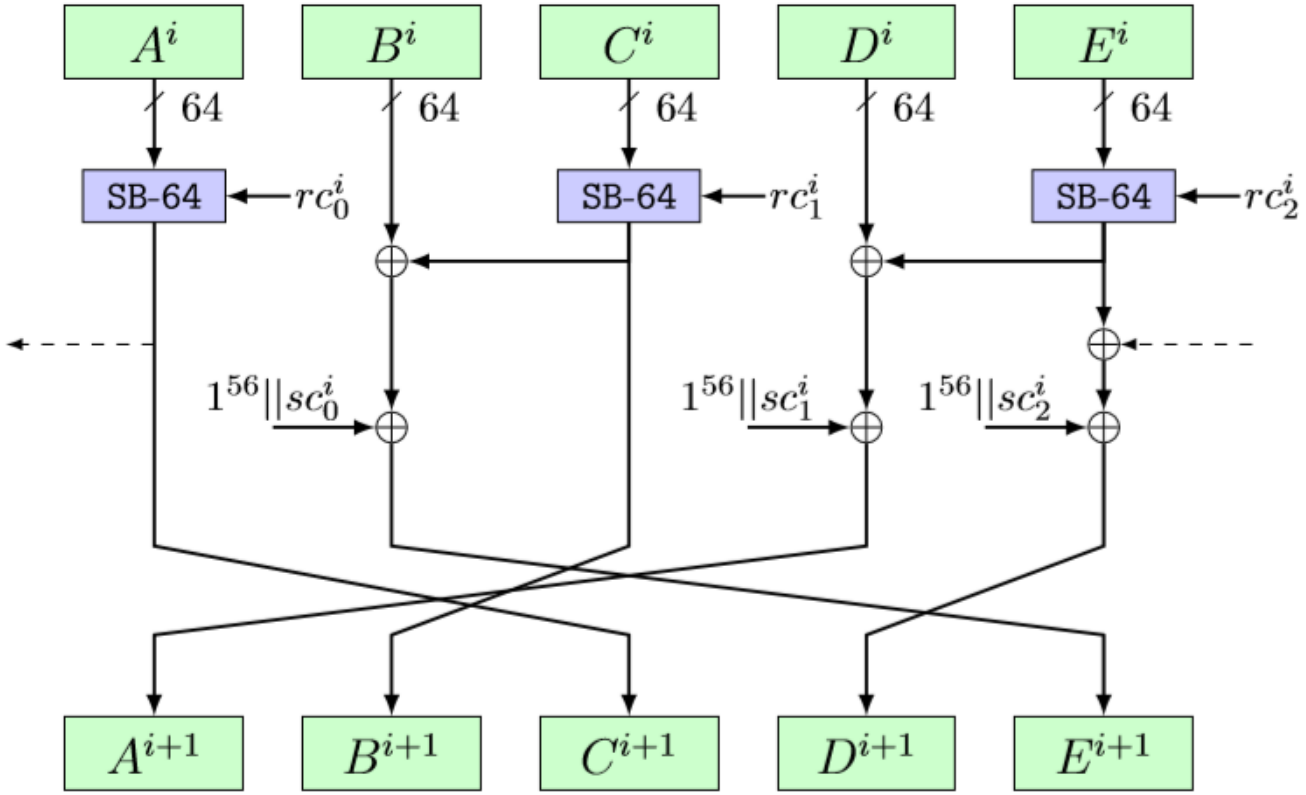


Figure 2.1: ACE-step

ACE 부채널 공격

- 공격자가 nonce값을 알수 있다고 가정 실제로 설정하기 어려울 수 있는 강력한 가정
  Bricklayer Attack: A Side-Channel Analysis on the ChaCha Quarter Round

- TLS의 경우에는 nonce 의 일부를 강제로 무작위값으로 사용

- 기존 프로토콜은 이런 식으로 정의되어 있지 않음

- SSL(Secure Shell) 프로토콜은 패킷 시퀀스 번호를 64비트 IV로 사용하는 반면 나머지 64비트는 블록 카운터에 사용되며, 각 패킷에 대해 재설정된다.
- 전체 SSH 세션을 관찰하면 전체 nonce를 예측할 수 있으므로 공격자는 충분한 패킷이 전송되는 즉시 모든 핵심 단어를 복구할 수 있는 기회를 갖게 된다.

ACE 부채널 공격 대응

- Bitwise XOR 및 AND, 왼쪽 순환 이동 및 64비트 워드 셔플 마스킹에 효율적인 구조
- 공격이 가능하다고 추측되는 부분에 부분적으로 마스킹 적용 마스킹으로 인한 속도 저하 최소화
- 유사한 방법으로 다른 NIST Round2의 암호도 가능할 것

Q & A