8-비트 프로세서에서 CHAM-64/128 CPA 공격과 마스킹 기법 최적화 적용

https://youtu.be/mfwyjrG-ubM

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

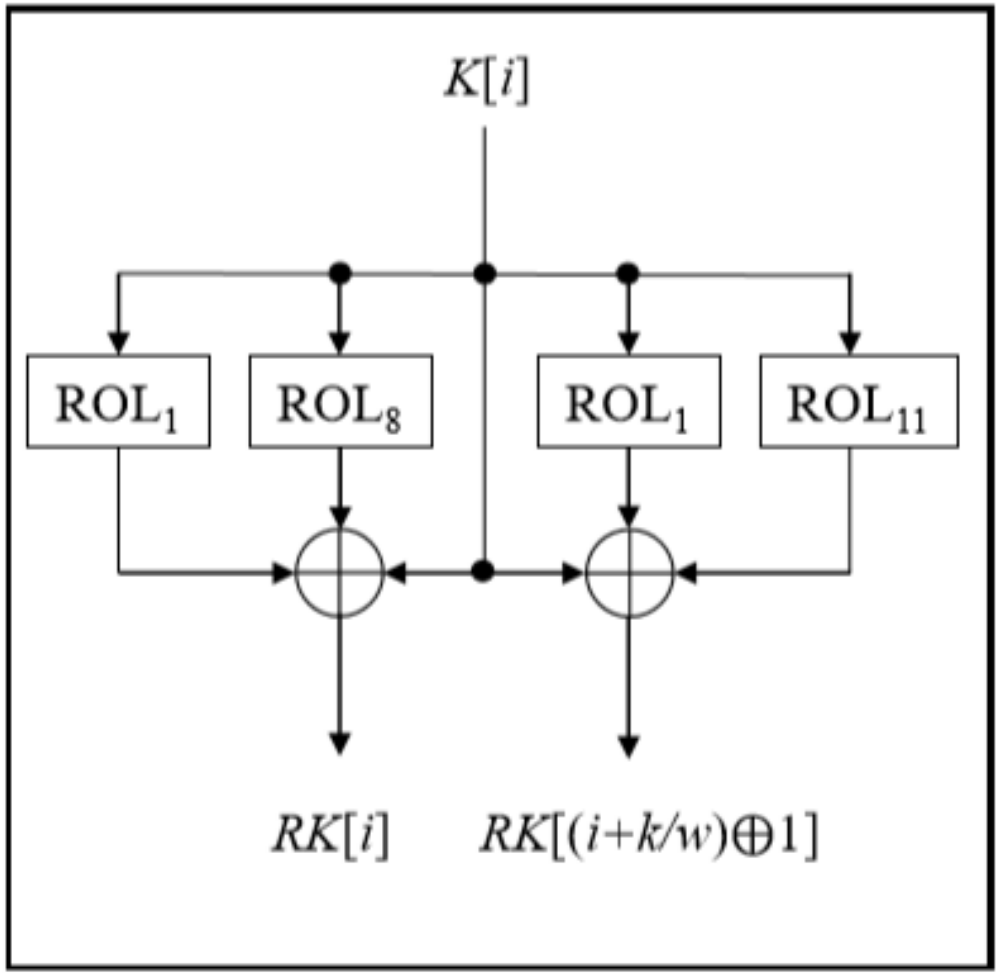
CryptoCraft LAB

CHAM

- ARX (더하기, 회전, XOR) 연산 기반 4- 분기 Feistel 구조
- 구조의 상태 정보를 유지하지 않는 stateless 키 스케줄
- 8 비트 AVR 마이크로 컨트롤러의 작업 수를 최소화하기 위해 1 비트 및 8 비트 두 가지 유형의 왼쪽 회전을 사용
- 자원 제약을 받는 저사양 디바이스 장치에서 효율적인 국산 경량 블록 암호
- 32 비트에 적합하만 8 비트 및 16 비트 환경에서는 효율적이지 못한 LEA에 대한 개선판
- CHAM-64 / 128, CHAM-128 / 128 및 CHAM-128 / 256 세 가지

cipher	n	k	r	w	k/w
CHAM-64/128	64	128	80	16	8
CHAM-128/128	128	128	80	32	4
CHAM-128/256	128	256	96	32	8

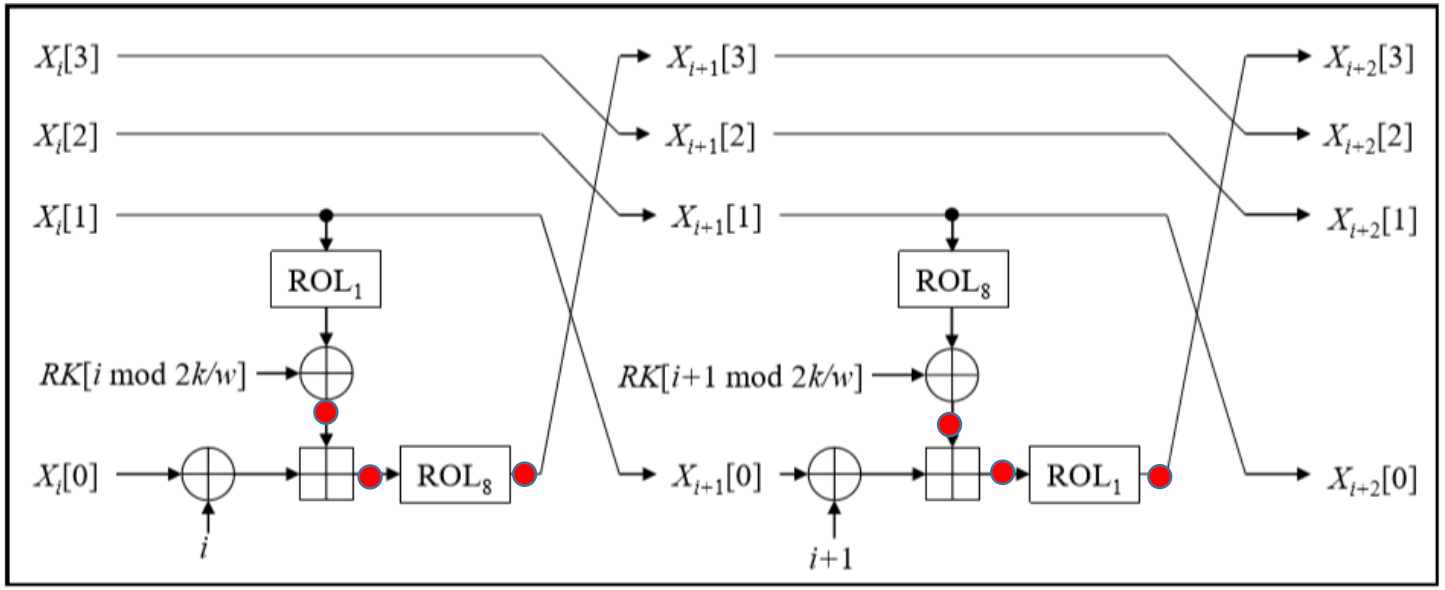
Key schedule 특징



- 하나의 키 워드에 2개의 라운드 키 생성
- 모든 경우의 키 값에 대해서 겹치지 않게 라운드 키가 생성된다.

✓ 하나의 라운드키를 획득한다면 전탐색 기법을 통해서 키를 알아 낼 수 있다.

Encryption 특징



● 공격 가능한 중간값 지점

- 2라운드마다 각 연산이 반복되어 실행된다.

✓ 다음 라운드키 값을 찾기 위해서는 전 단계 라운드키를 알아내고 연산 결과 값을 계산 해야 한다.

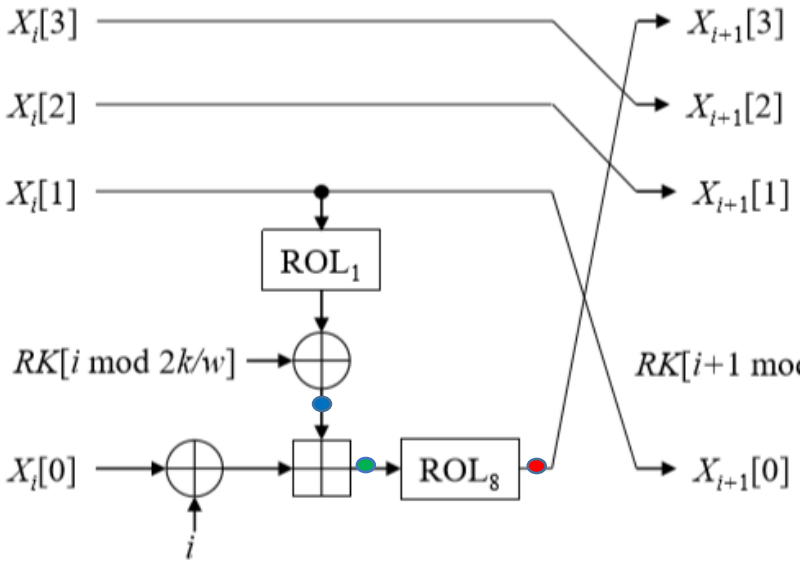
- 라운드마다 해당하는 하나의 라운드 키를 사용한다.
- 라운드 키를 저장하는 데 필요한 메모리 크기를 줄이기 위하여 반복적으로 재사용

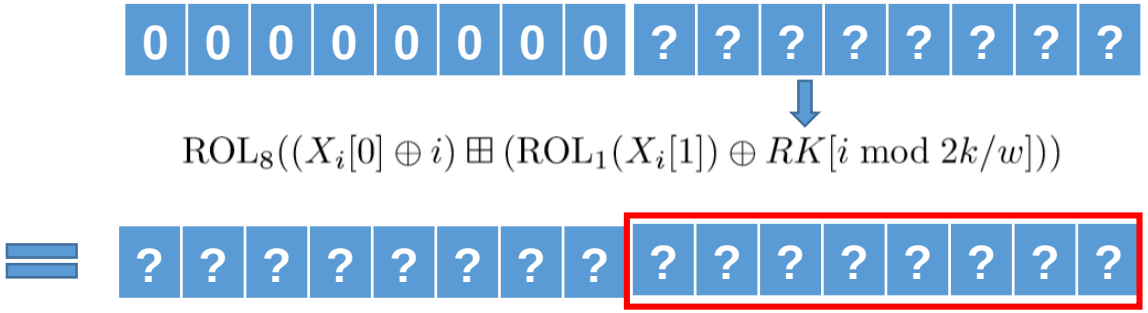
✓ k/w 만큼의 개수의 라운드키를 알면 모든 키를 알 수 있다.

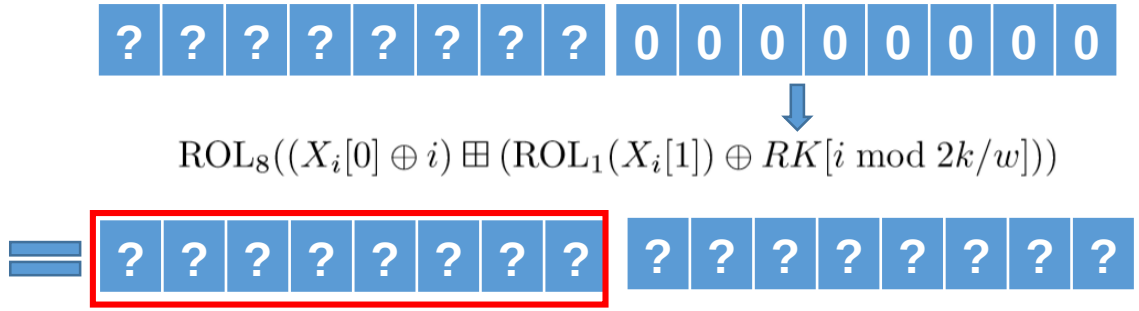
제안 CPA 공격 기법

홀수 라운드 연산

- ROL 연산 후(•) 지점 공격
- RK에 우측부분에 8bit($RK_{8\text{-}15}$)만 추측값를 넣고 결과값 중 가장 상관계수 값이 높은 추측값를 라운드키의 우측 8bit로 선택
- RK에 좌측부분에 8bit($RK_{0\text{-}7}$)만 추측값를 넣고 결과값 중 가장 상관계수 값이 높은 추측값를 라운드키의 좌측 8bit로 선택
- 획득한 라운드키값으로 마스터키값 획득



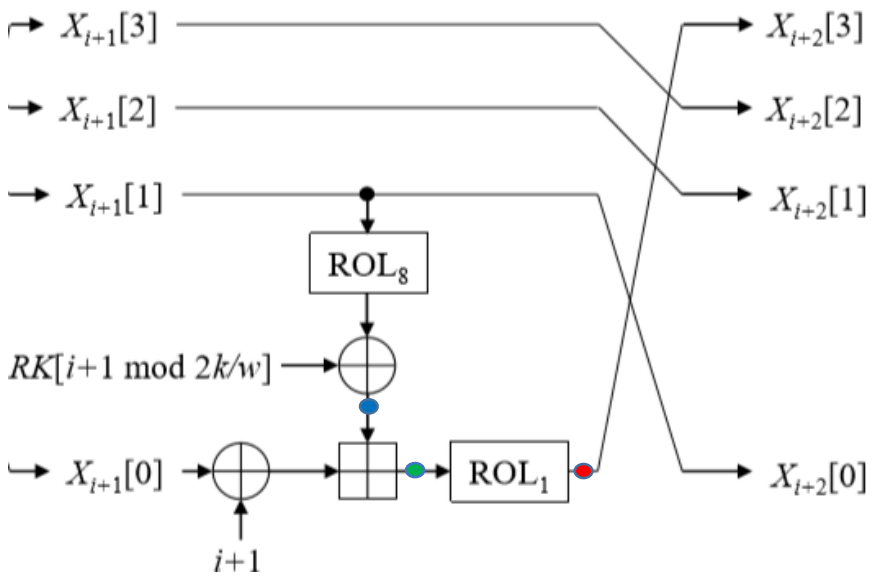


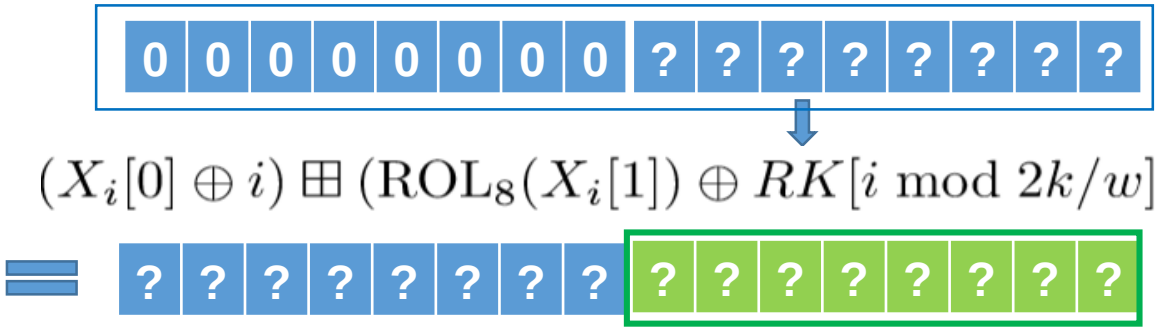


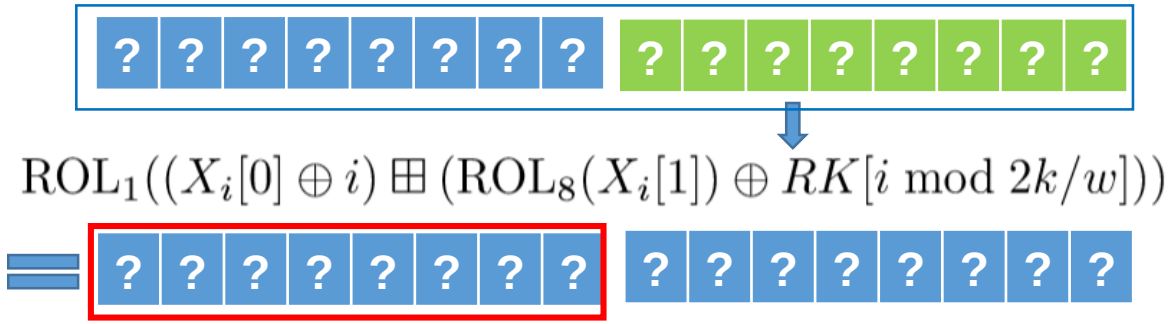
제안 CPA 공격 기법

짝수 라운드 연산

- $RK_{8\text{-}15}$는 ● 지점을 공격, $RK_{0\text{-}7}$은 ● 지점을 공격한다.
- 먼저 $RK_{8\text{-}15}$만 추측값를 넣고 결과값 중 가장 상관계수 값이 높은 추측값를 라운드키의 우측 8bit로 선택
- 앞부분에서 구한 $RK_{8\text{-}15}$값과 $RK_{0\text{-}7}$에 추측값를 넣고 결과값 중 가장 상관계수 값이 높은 추측값를 라운드키의 좌측 8bit로 선택
- 획득한 라운드키값으로 마스터키값 획득



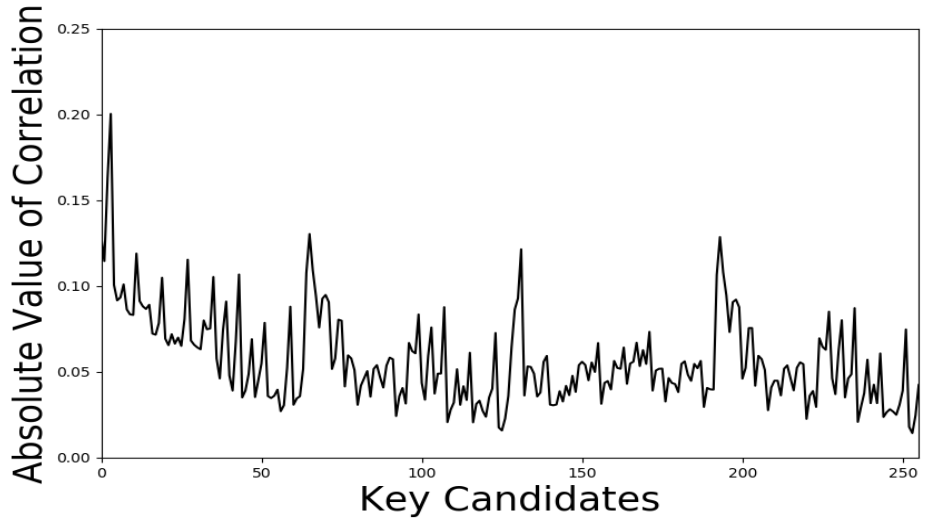


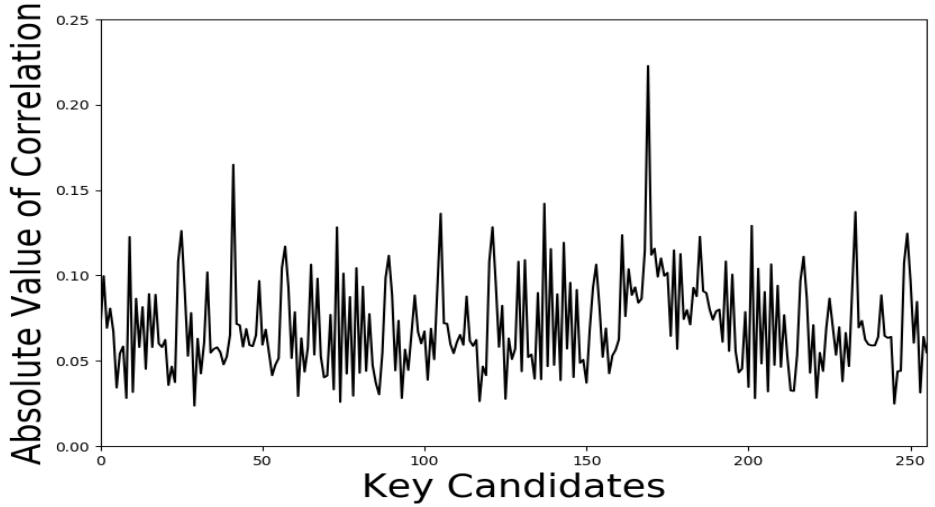


실험 결과

- 8bit 프로세서에서 돌아가는 CHAM-64/128 대상
- 8비트 씩 나눠서 공격한다.
- 1~8라운드까지의 파형을 수집
- 좌측 연산과 우측 연산을 다른 방법 사용하여 반복

```
0x2B7E 0x00 0x00 0x00 0x00 0x00 0x00 0x00  937 3 169
0x2B7E 0x1516 0x00 0x00 0x00 0x00 0x00 0x00  10543 41 47
0x2B7E 0x1516 0x28AE 0x00 0x00 0x00 0x00 0x00  55258 215 218
0x2B7E 0x1516 0x28AE 0xD2A6 0x00 0x00 0x00 0x00  53561 209 57
0x2B7E 0x1516 0x28AE 0xD2A6 0xABF7 0x00 0x00 0x00  2995 11 179
0x2B7E 0x1516 0x28AE 0xD2A6 0xABF7 0x1588 0x00 0x00  46733 182 141
0x2B7E 0x1516 0x28AE 0xD2A6 0xABF7 0x1588 0x9CF 0x00  54616 213 88
0x2B7E 0x1516 0x28AE 0xD2A6 0xABF7 0x1588 0x9CF 0x4F3C  60683 237 11
[937, 10543, 55258, 53561, 2995, 46733, 54616, 60683, 36754, 36057, 16766, 2231, 32308, 16711, 13117, 25119]
0x2B7E 0x1516 0x28AE 0xD2A6 0xABF7 0x1588 0x9CF 0x4F3C
0x2B7E 0x1516 0x28AE 0xD2A6 0xABF7 0x1588 0x9CF 0x4F3C
280.34166120000003
```



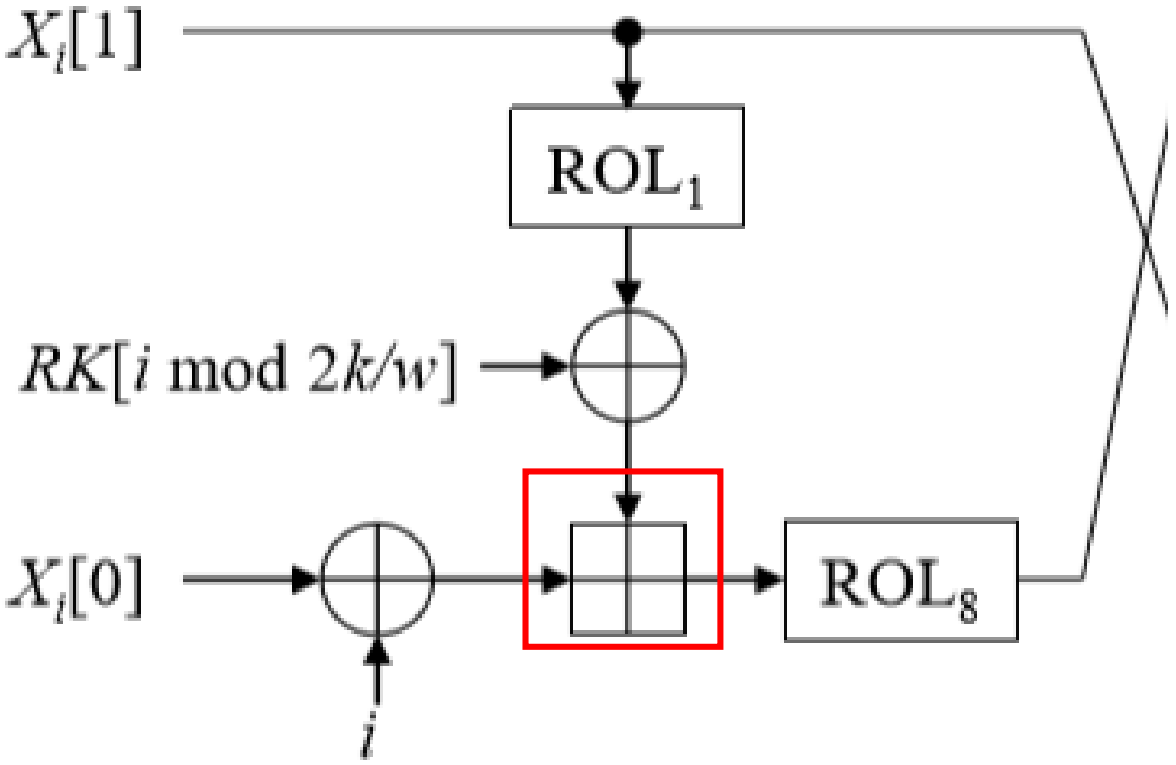


마스킹 기법

- 사이드 채널 분석 공격을 막기위한 일반적인 대책으로 알고리즘 마스킹
- 부울 마스킹 : $x'=(x \oplus r)$
- 산술 마스킹 : $A=x-r \bmod 2^k$
- CHAM과 같은 ARX 구조의 알고리즘에 마스킹 기법이 적용되는 경우 불 마스킹 기법과 산술 마스킹을 상호 변환 하는 과정이 필요

마스킹 기법

- (x⊕r)+(x' ⊕r') r, r'?
- x-r -> x ⊕r
- (x-r)+r ⊕r
- +r 연산시에 x값이 노출됨
- 안전한 변환 기법이 필요



부울 – 산술 마스킹 변환 기법

- 부울-산술 마스킹 변환 기법은 Messerges가 처음 제안
- 그러나 취약성이 발견되어 그 후 Goubin에 의해 새로운 변환 기법이 제안됨
- 연산량이 매우 적기 때문에 많은 암호 알고리즘에서 이 부울-산술 마스킹 변환 기법을 사용한다.
- 입력값 : x ⊕r ,r 출력값 : x-r

Input : x', r_x

Output: A

1. $\Gamma = \gamma$	5. $\Gamma = \Gamma \oplus r_x$
2. $T = x' \oplus \Gamma$	6. $A = x' \oplus \Gamma$
3. $T = T - \Gamma$	7. $A = A - \Gamma$
4. $T = T \oplus x'$	8. $A = A \oplus T$

산술-부울 마스킹변환 기법

- Goubin 기법
- 내부 루프 때문에 데이터의 비트 크기에 비례하여 연산량이 증가
- 5k+5번의 연산 필요
- 입력값 : x-r,r 출력값 : x ⊕r

Input : A, r_x

Output: x'

1. $\Gamma = \gamma$	6. $\Gamma = \Gamma \oplus x'$
2. $T = 2\Gamma$	7. $\Gamma = \Gamma \wedge r_x$
3. $x' = \Gamma \oplus r_x$	8. $\Omega = \Omega \oplus \Gamma$
4. $\Omega = \Gamma \wedge x'$	9. $\Gamma = T \wedge A$
5. $x' = T \oplus A$	10. $\Omega = \Omega \oplus \Gamma$
11. for $i = 1$ to $k-1$ do	
11.1 $\Gamma = T \wedge r_x$	11.4 $\Gamma = \Gamma \oplus T$
11.2 $\Gamma = \Gamma \oplus \Omega$	11.5 $T = 2\Gamma$
11.3 $T = T \wedge A$	
12. $x' = x' \oplus T$	

산술-부울 마스킹변환 기법

- CT 산술-불 마스킹 변환

  Goubin이 제안한 산술-부울 마스킹 변환 기법의 연산량 개선을 위해
  룩업 테이블 (look-up table) 기반의 산술-부울 마스킹 변환 기 법

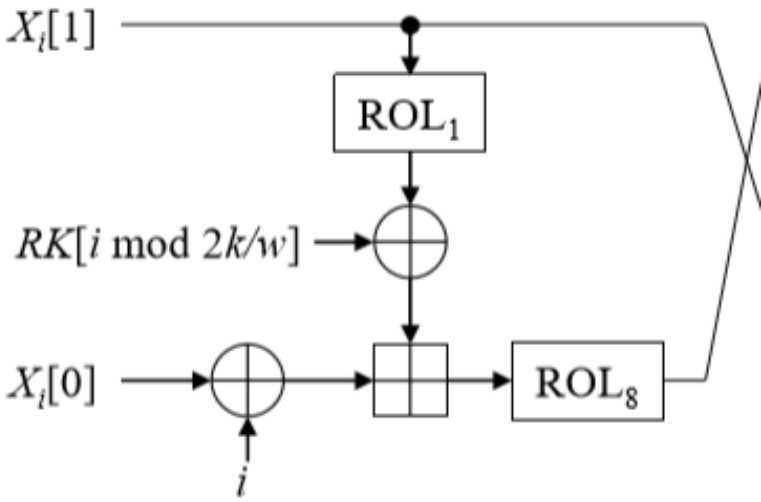
- 산술 연산을 부울 연산으로 변환하는 G-테이블,
  모듈라 덧셈 시 발생하는 캐리를 저장하는 C-테이블사용
  $2^k + 2^k$개의 태이블 값을 저장할 공간 필요

- Debraize 산술-불 마스킹 변환

  Debraize는 안전성과 효율성을 개선한 룩업 테이블 기반의 새로운 방법을 제안

- 2^{k+1} 개의 테이블 값을 저장할 공간 필요

Secure Addition 마스킹 기법



- 불 마스킹된 두 개의 데이터를 더하기 해서는 불산술 마스킹 변환, 덧셈, 산술-불 마스킹 변환의 3단계 를 거쳐야 한다.
- 마스킹 변환의 번거로움을 이고 연산량을 개선시키고자 Secure Addition 제안됨
- 두 개의 불 마스킹된 데이터를 입력받아 불 마스킹 된 덧셈 결과를 출력

KRJ SA 마스킹 기법

- Karroumi는 기존의 마스킹 변환의 번거로움과 연산량을 개선하기 위해 Secure Addition 마스킹 기법을 처음으로 제안
- Goubin의 기법은 5k+21의 연산 필요
- 5k+8의 연산 소요
- VG SA 마스킹 기법
  - KRJ 기법에 룩업 테이블을 이용한 SA 마스킹 기법을 제안

Input : x', y', r_x, r_y

Output: z'

1. $C = \gamma$	10. $B = \Omega \ll 1$
2. $T = x' \wedge y'$	11. $C = C \ll 1$
3. $\Omega = C \oplus T$	12. $z' = x' \oplus y'$
4. $T = x' \wedge r_y$	13. $r_z = r_x \oplus r_y$
5. $\Omega = \Omega \oplus T$	14. $T = C \wedge r_z$
6. $T = y' \wedge r_x$	15. $\Omega = \Omega \oplus T$
7. $\Omega = \Omega \wedge T$	16. $T = C \wedge r_z$
8. $T = r_x \wedge r_y$	17. $\Omega = \Omega \oplus T$
9. $\Omega = \Omega \wedge T$	
18. for $i = 2$ to $k-1$ do	
18.1 $T = B \wedge z'$	18.4 $B = B \oplus T$
18.2 $B = B \wedge r_z$	18.5 $B = B \ll 1$
18.3 $B = B \oplus \Omega$	
19. $z' = z' \oplus B$	20. $z' = z' \oplus C$

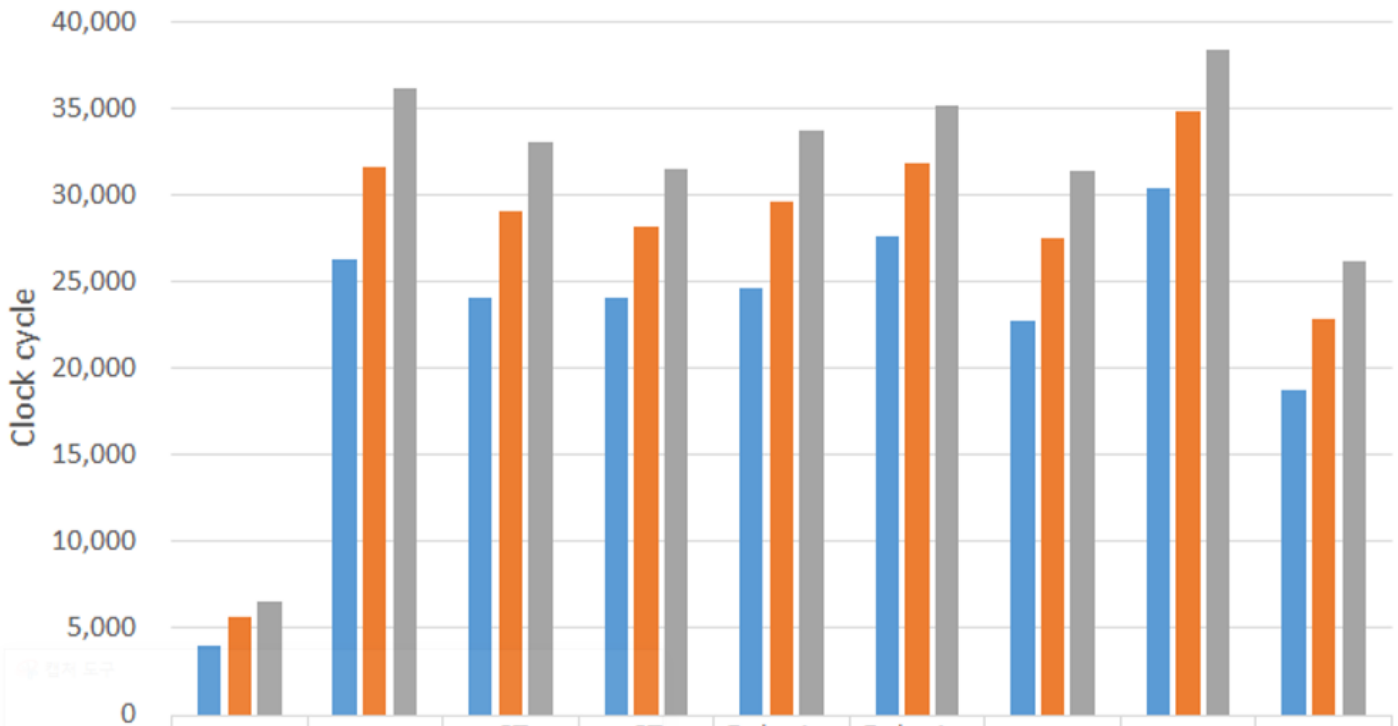
CGTV SA 마스킹 기법

- Kogge-Stone 올림수 예견 덧셈기 기반의 마스킹 변환 기법을 제안
- Goubin 기법이나 KRJ 기법의 경우 $O(k)$
  CGTV 기법은 $O(logk)$ 의 연산 복잡도
- $k = 2^n$일때 24n+4의 연산 소요

Input : x', y', r_x, r_y
Output : z'

1. Generate random number k-bit s_1, s_2
2. $P' = SecXor(x', y', r_y)$
3. $G' = SecAnd(x', y', r_x, r_y, u)$
4. $G' = G' \oplus r_x$ 5. $G' = G' \oplus u$
6. for $i = 1$ to $n-1$ do
6.1 $H = SecShift(G', r_x, s_1, 2^{i-1})$
6.2 $U = SecAnd(P', H, r_x, s_1, s_2)$
6.3 $G' = SecXor(G', U, s_2)$
6.4 $H = SecShift(P', r_x, s_1, 2^{i-1})$
6.5 $P' = SecAnd(P', H, r_x, s_1, s_2)$
6.6 $P' = P' \oplus r_x$ 6.7 $P' = P' \oplus s_2$
7. $H = SecShift(G', r_x, s_1, 2^{n-1})$
8. $U = SecAnd(P', H, r_x, s_1, s_2)$
9. $G' = SecXor(G', U, s_2)$
10. $z' = SecXor(y', x', r_x)$
11. $z' = z' \oplus 2G'$ 12. $z' = z' \oplus 2r_x$
13. $z' = z' \oplus r_x$

CHAM에 마스킹 적용



	Non	Goubin	CT 4-bit	CT 8-bit	Debraize 4-bit	Debraize 8-bit	KRJ	VG	CGTV
128-bit	4,025	26,281	24,107	24,057	24,641	27,600	22,753	30,347	18,798
192-bit	5,701	31,632	29,038	28,220	29,632	31,799	27,516	34,810	22,901
256-bit	6,508	36,108	33,094	31,508	33,748	35,123	31,404	38,398	26,129

참고논문에서 전체 사이클 비교시에 lea에 대한 마스크 적용시 CGTV 기법이 제일 낮고 KRJ이 2번째

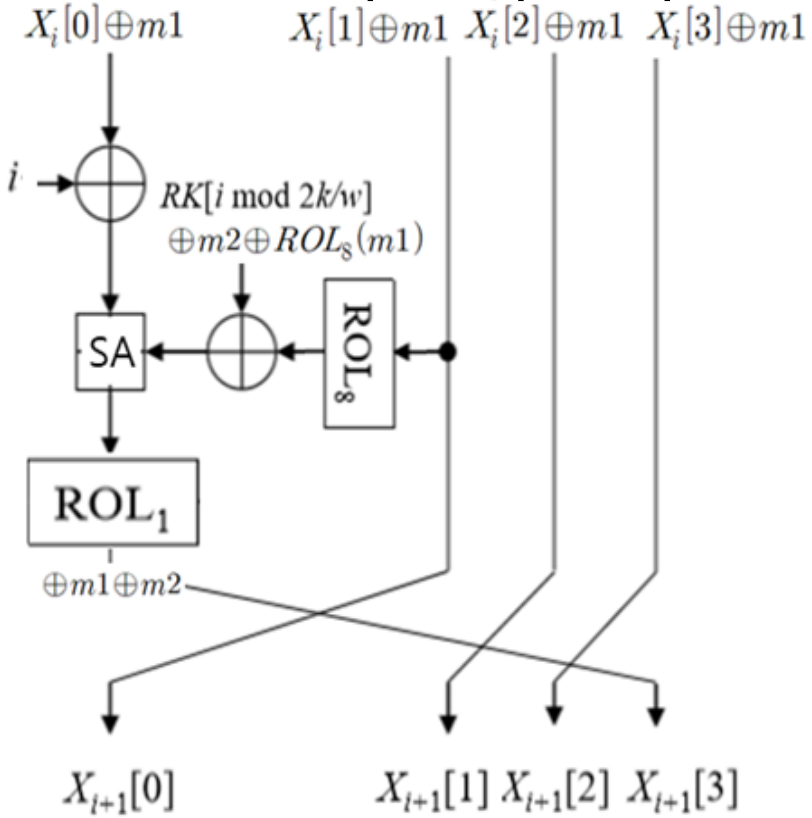
Goubin, KRJ, CGTV 기법은 기본 연산을 레지스터 간 연산으로 표현이 가능

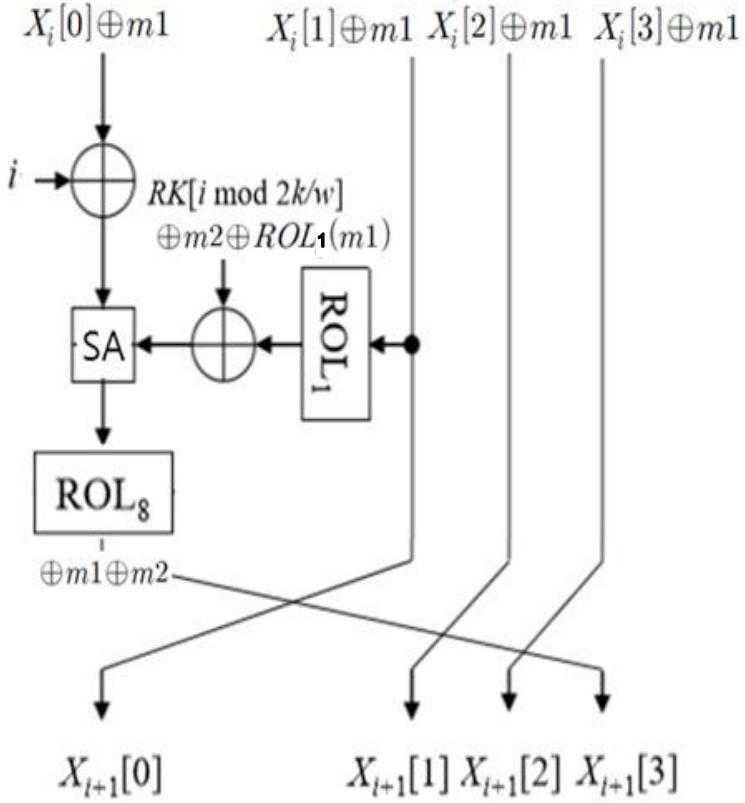
8비트상에서
CGTV보다 KRJ가 더 적은 연산

CGTV 레지스터 수가 부족

제안 마스킹 적용 기법

- KRJ 기법을 사용
- 마스크 값은 16비트의 난수 2개를 사용





CHAM에 마스킹 적용

- 레퍼런스 코드에 비하여 13배 느린 성능
- 최적화 전에 비하여 약 60%의 성능 향상됨을 확인

	Non masking	4Round v1	4Round v2	Goubin	KRJ v1	KRJ v2
Lenguage	C	C	ASM	C	C	ASM
Cycle clock	2,424	2,269	1,513	145,405	87,405	32,221

Q & A