CHAM에 대한 CPA 공격

https://youtu.be/eQSf-AbHuL8

Contents

CryptoCraft LAB

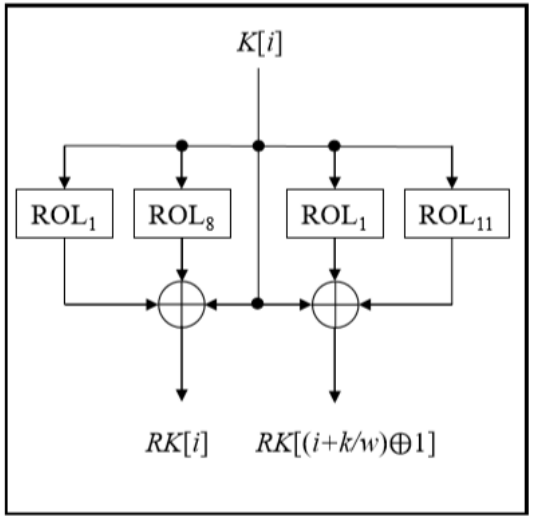
1 CHAM 이란

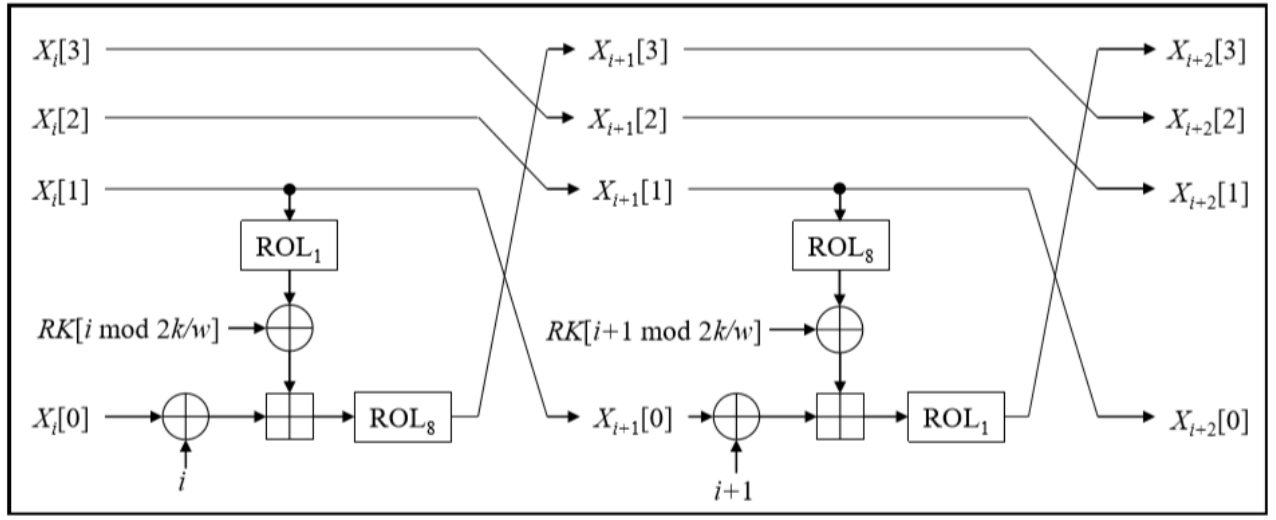
- 자원 제약을 받는 장치에 대해 효율적인 국산 경량 블록 암호
- ARX (더하기, 회전, XOR) 연산 기반 4- 분기 Feistel 구조
- LEA는 32 비트에선 적합하나 8 비트 및 16 비트 마이크로 컨트롤러에서는 그렇지 않다. 이를 개선하기 위해 만들어 졌다.
- CHAM-64 / 128, CHAM-128 / 128 및 CHAM-128 / 256 세 가지

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

CryptoCraft LAB

1 CHAM 이란



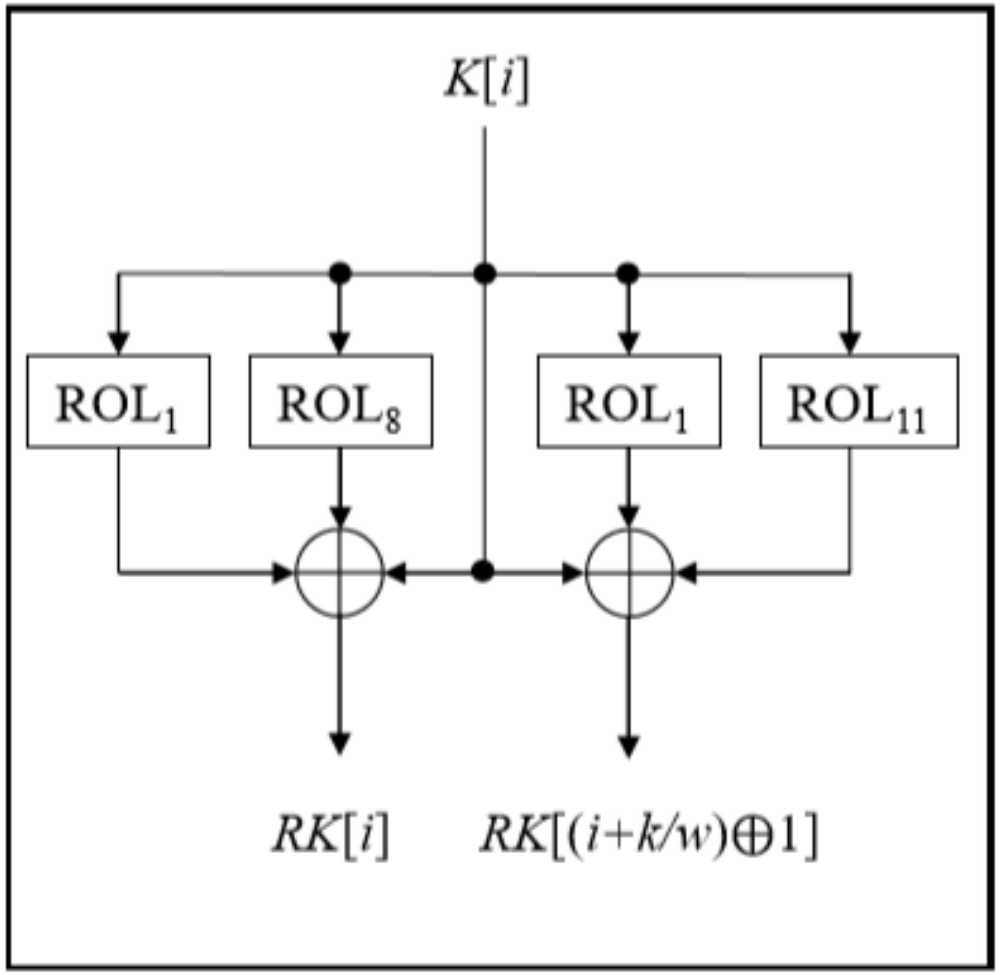


ü 정보 보호를 위한 암호 시스템을 임베디드 장치에서 개발할 경우 발생할 수 있는 구현상의 문제점을 이용하여 비밀 키를 추출하기 위한 여러 부채널 공격들이 시도되어 왔다.

ü 같은 ARX 구조 LEA와 SIMON / SPECK 암호가 부채널 공격 중 하나인 CPA에 대한 취약점이 제시되었다. CHAM 또한 CPA에 대한 취약성 확인이 필요하다.

CryptoCraft LAB

2 Key schedule특징

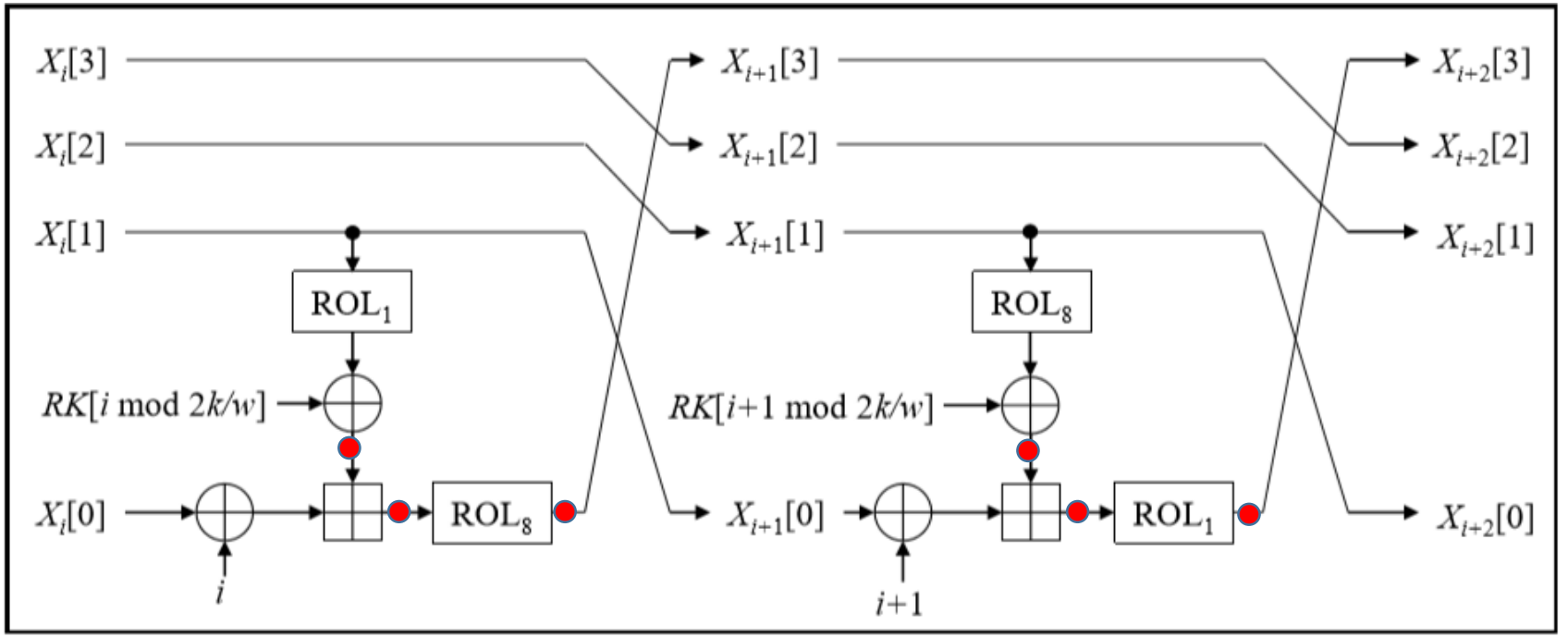


- 하나의 키 워드에 2개의 라운드 키 생성
- 모든 경우의 키 값에 대해서 겹치지 않게 라운드 키가 생성된다.

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

CryptoCraft LAB

3 Encryption 특징



● 공격 가능한 중간값 지점

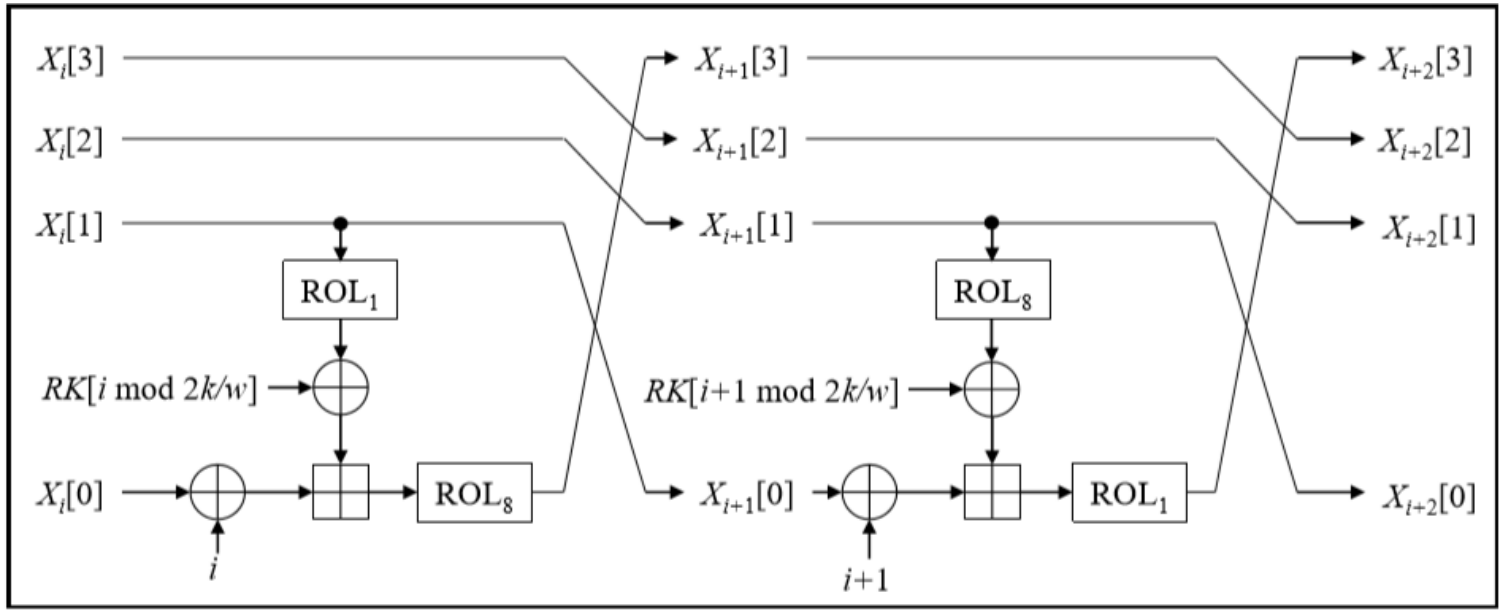
- 2라운드마다 각 연산이 반복되어 실행된다.
- 라운드마다 해당하는 하나의 라운드키를 사용한다.

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

✓ k/w 만큼의 개수의 라운드키를 알면 모든 키를 알 수 있다.

CryptoCraft LAB

4 제안 CPA 공격 기법



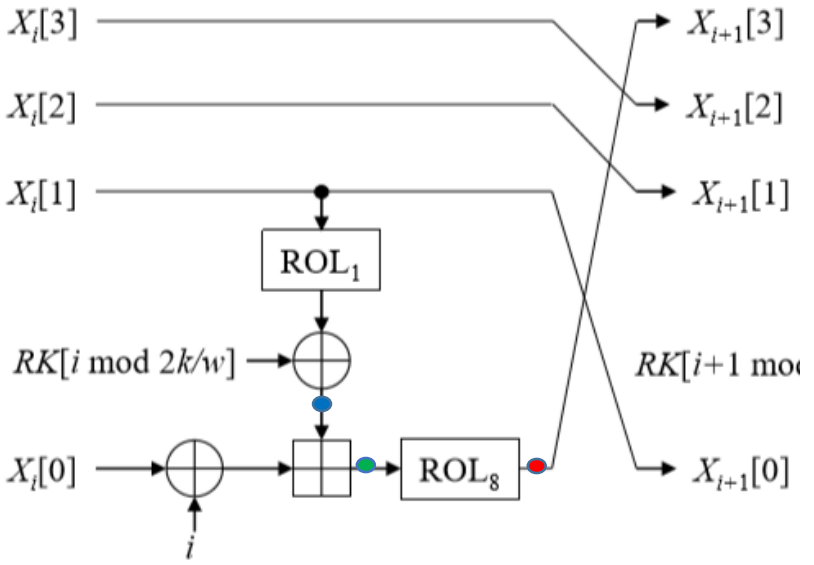
- 8bit 프로세서에서 돌아가는 CHAM-64/128 대상
- 8비트 씩 나눠서 공격한다.
- 8라운드까지의 파형을 수집
- 좌측 연산과 우측 연산을 다른 방법 사용하여 반복

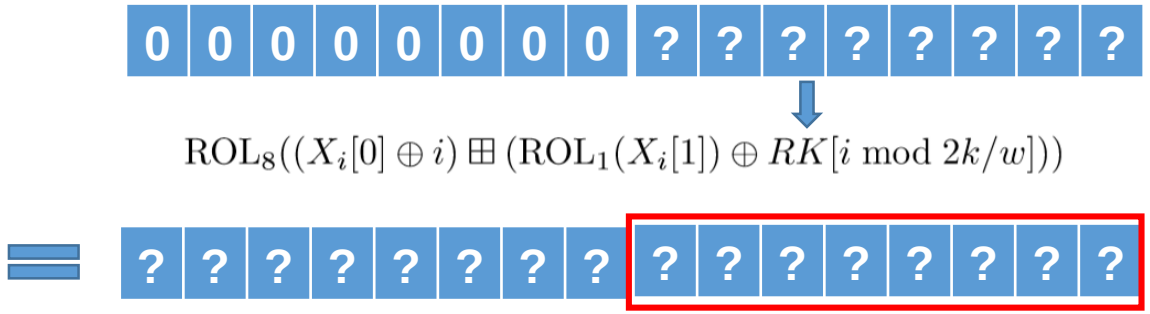
CryptoCraft LAB

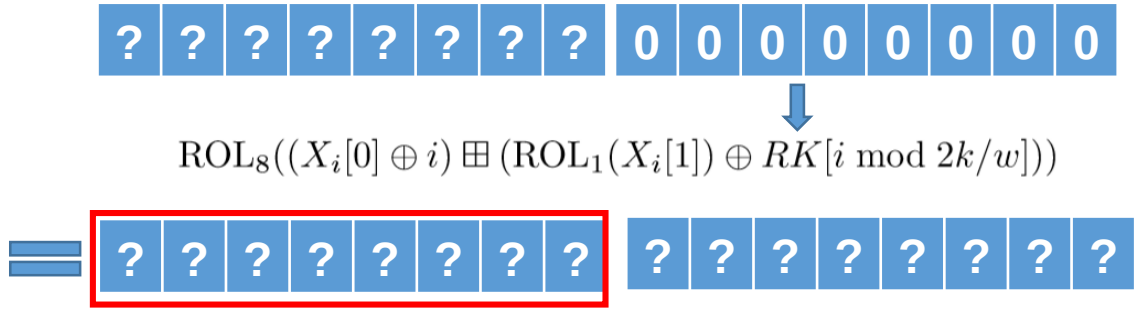
4 제안 CPA 공격 기법

좌측 연산

- ROL 연산 후(•) 지점에서
  가장 적은 트레이스로 라운드키를 획득 가능
- 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





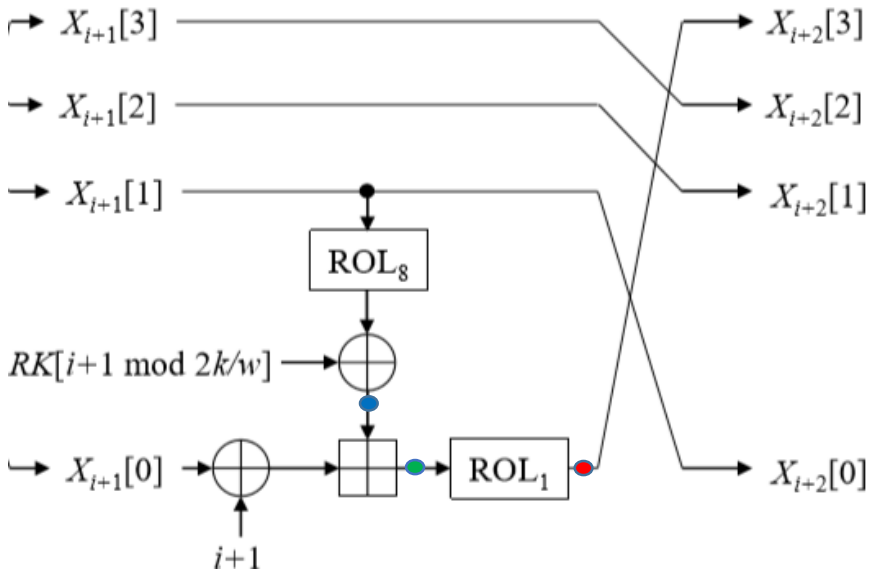


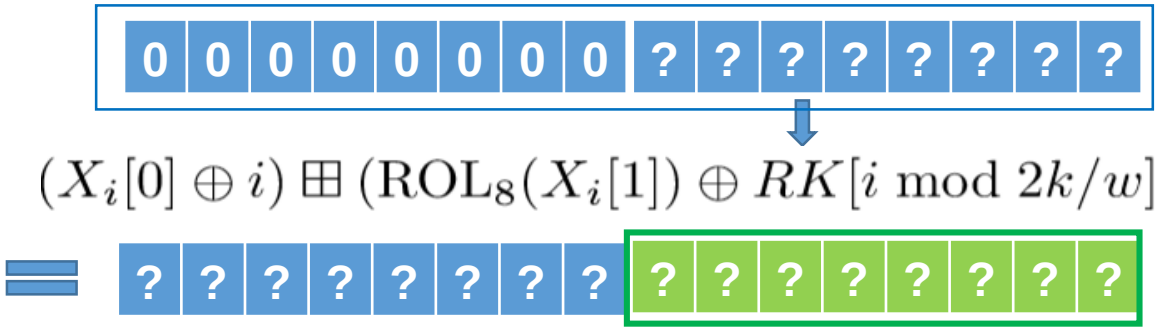
CryptoCraft LAB

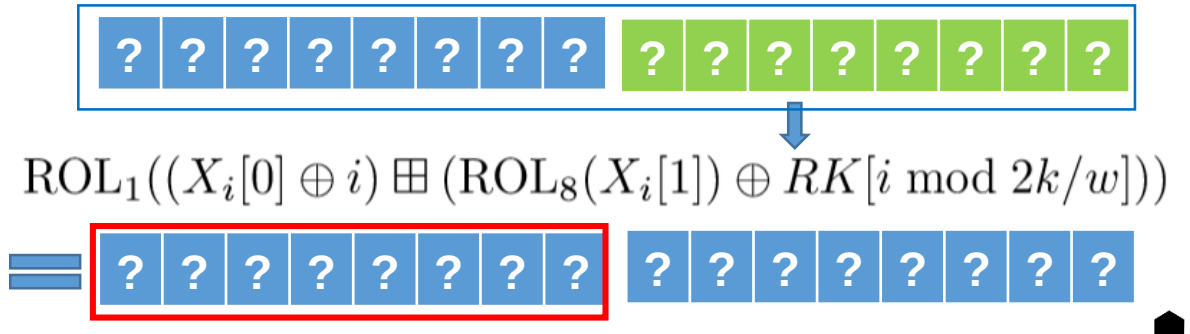
4 제안 CPA 공격 기법

우측 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







CryptoCraft LAB

5 실험 결과

- XMEGA 보드 사용
  Chipwhisperer로 파형 수집
- 8라운드까지 5000개의 파형 수집
- 모든 키를 알아내어 성공

```
void key_generic(uint16_t* rk, const uint16_t* k) {

    for (uint8_t i = 0;i < 8;i++) {
        rk[i] = k[i] ^ rol(k[i], 1) ^ rol(k[i], 8);
        rk[(i + 8) ^ 1] = k[i] ^ rol(k[i], 1) ^ rol(k[i], 11);
    }
}

void encrypt(uint16_t *ct, const uint16_t *pt, const uint16_t *rk) {
    ct[0] = pt[0];
    ct[1] = pt[1];
    ct[2] = pt[2];
    ct[3] = pt[3];
    uint16_t temp;
    for (uint8_t i = 0;i < 80;i++) {
        if (i % 2 == 0)
            temp = rol((rol(ct[1], 1) ^ rk[(i % (2 * 8))]) + (ct[0] ^ i), 8);
        else
            temp = rol((rol(ct[1], 8) ^ rk[((i) % (2 * 8))]) + (ct[0] ^ (i)), 1);
        ct[0] = ct[1];
        ct[1] = ct[2];
        ct[2] = ct[3];
        ct[3] = temp;
    }
}
```

CryptoCraft LAB

5

```
for K in range(0, 8):
    #좌측
    if K%2==0:
        leftRK = np.argmax(subattackLeft(_pt, traces, 0, K))
        rightRK = np.argmax(subattackRight(_pt, traces, 0, K))
        roundkey[K] = (leftRK << 8 | rightRK)
        for i in range(0, numtraces):
            temp = rol(((rol(_pt[i][1], 1) ^ roundkey[K]) + ((_pt[i][0] ^ K))) & 0xffff, 8)
            _pt[i][0] = _pt[i][1]
            _pt[i][1] = _pt[i][2]
            _pt[i][2] = _pt[i][3]
            _pt[i][3] = temp
    #우측
    else:
        rightRK = np.argmax(subattackRight(_pt, traces, 0, K))
        leftRK = np.argmax(subattackLeft(_pt, traces, rightRK, K))
        roundkey[K] = (leftRK << 8 | rightRK)
        for i in range(0, numtraces):
            temp = rol(((rol(_pt[i][1], 8) ^ roundkey[K]) + ((_pt[i][0] ^ K))) & 0xffff, 1)
            _pt[i][0] = _pt[i][1]
            _pt[i][1] = _pt[i][2]
            _pt[i][2] = _pt[i][3]
            _pt[i][3] = temp

    #subkey를 저장하고 나머지 roundkey 찾기
    key[K] = x.index(roundkey[K])
    roundkey[(K + 8) ^ 1] = roundkey_table(key[K])[1]
    #print(roundkey)
    print(str.join("", ("0x%02X " % i for i in key[:])), roundkey[K], leftRK, rightRK)
```

CryptoCraft LAB

5

```python
def intermediate1(pt,_pt, T, i):
    if (i % 2 == 0):
        x = rol(((rol(pt, 1) ^ T)+ ((_pt^i)))&0xffff,8)&0xff
    else:
        x = rol(((rol(pt, 8) ^ T)+ ((_pt^i)))&0xffff,1)&0xff
    return x
```

```python
def intermediate2(pt,_pt, T, i):
    if (i % 2 == 0):
        x = (rol(((rol(pt, 1) ^ T)+ ((_pt^i)))&0xffff,8)>>8)&0xff
    else:
        x = ((((rol(pt, 8) ^ T) + ((_pt ^ i))) & 0xffff) >> 8) & 0xff
    return x
```

```
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
```

CryptoCraft LAB

Thank You