Cryptanalysis
(암호분석)

Chapter 5 – Part 3

2020.5

Contents

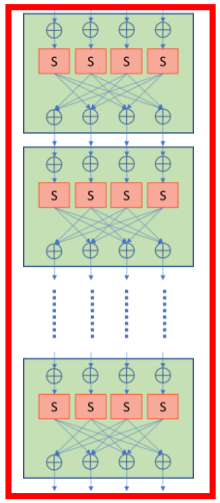
Chapter 5
- Part 1

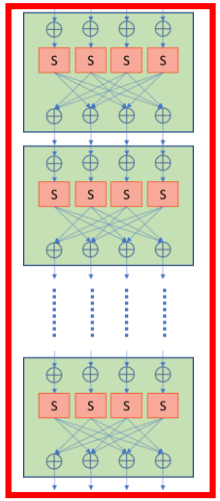
- Generic Attack
- Brute force attack: Exhaustive key search
- Meet-in-the-Middle Attack

Chapter 5
- Part 2

- TMTO: Time Memory Trade Off

여기까지
중간고사
범위





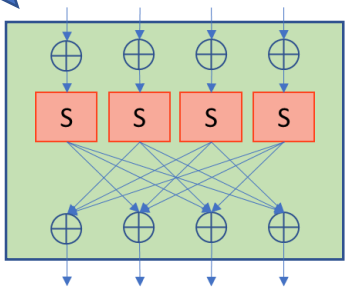
Chapter 5
- Part 3

- Slide Attack

Weak round function

- 블록암호의 라운드 함수 F
  - F: (입력, 라운드 키) → 출력
  - 입력, 출력: 블록크기

TC20:
weak round
function ?



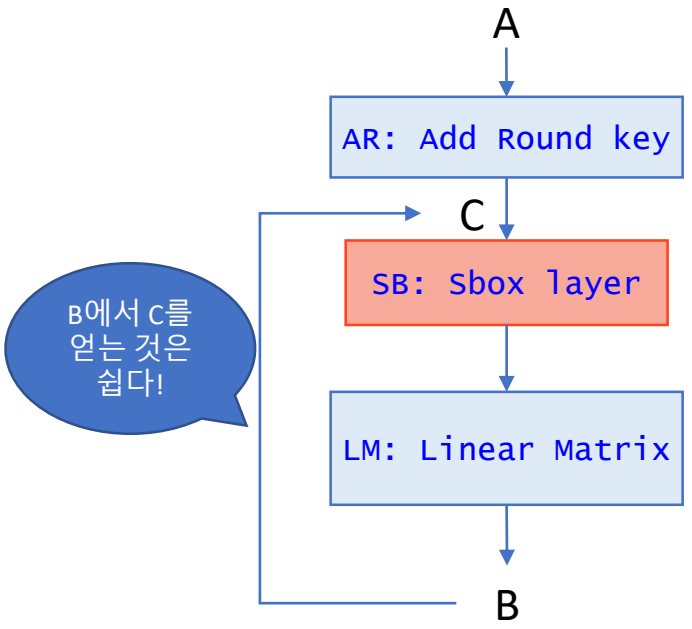
- (일반적인) 블록암호의 설계사상
  - F함수는 단순한 구조
  - F함수를 반복하여 복잡한 암호문 생성

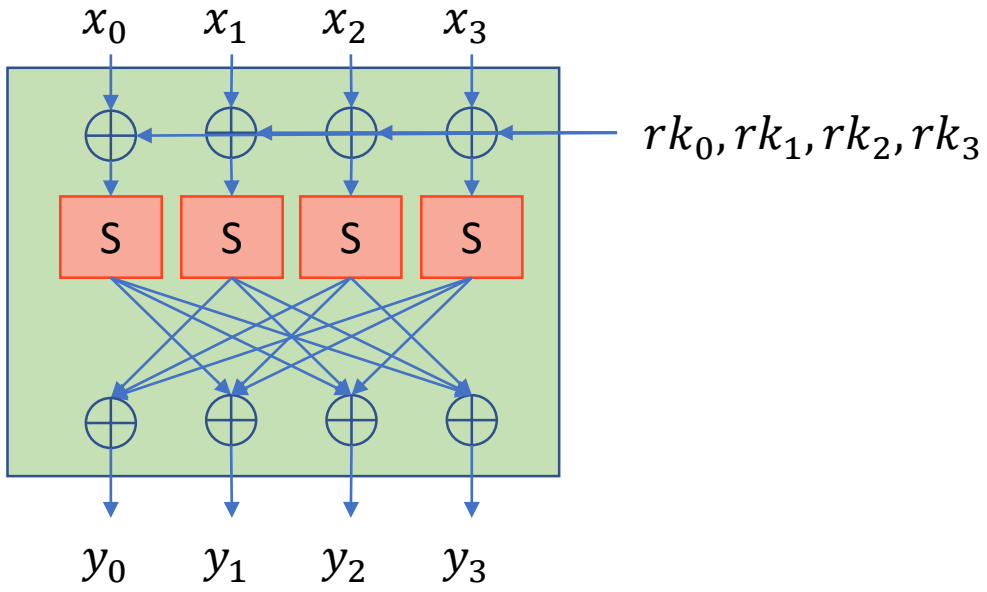
- 'Weak' 라운드 함수
  - 주어진 A, B에 대하여,
    F(A,k)=B를 만족하는 k를 **찾기 쉬운** 라운드 함수

주관적인
설정(?)

TC20의 Weak Round Function

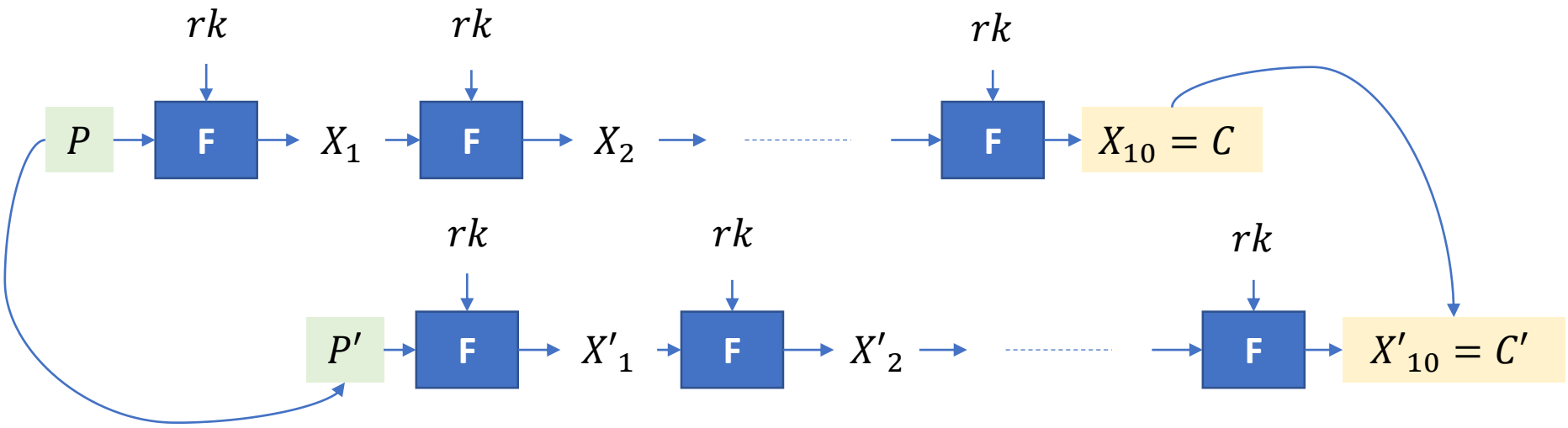
- F(A,rk) = B 로부터 rk를 얻는 과정
  - 출력 B로부터 LM^{-1}(=LM), SB^{-1}를 거쳐 C를 얻는다.
  - 입력 A와 C를 XOR하면 rk를 얻는다.





Slid Pair

- 라운드 키가 모두 동일하다면?
  - 두 개의 평문-암호문 쌍: $C = E(P), C' = E(P')$
  - $P' = F(P)$을 만족하면, $C' = F(C)$를 만족한다.



- Slid Pair (모든 라운드 키가 동일한 암호에 대하여)
  평문-암호문 쌍 (P, C), (P', C')이 다음 조건을 만족하면 Slid Pair라고 정의 한다.

$$P' = F(P),\ C' = F(C)$$

라운드 입출력 → 라운드 키

- F(A,rk) = B 로부터 rk를 얻는 프로그램

```
import TC20_Enc_lib as TC20E
import TC20_Dec_lib as TC20D
#------------------------------
# 주어진 A, B에 대하여
#  F(A, rk) = B (F: 라운드 함수)
# 를 만족하는 rk를 찾는 함수
#------------------------------
def Extract_RK(A,B):
    in_state= A
    state1 = TC20D.LM(B)
    state2 = TC20D.ISB(state1)
    rk = TC20D.AR(state2, in_state)
    return rk
```

```
def LM(in_state):
    out_state = [0, 0, 0, 0]
    all_xor = in_state[0] ^ in_state[1] ^
in_state[2] ^ in_state[3]
    for i in range(len(in_state)):
        out_state[i] = in_state[i] ^ all_xor
    return out_state
```

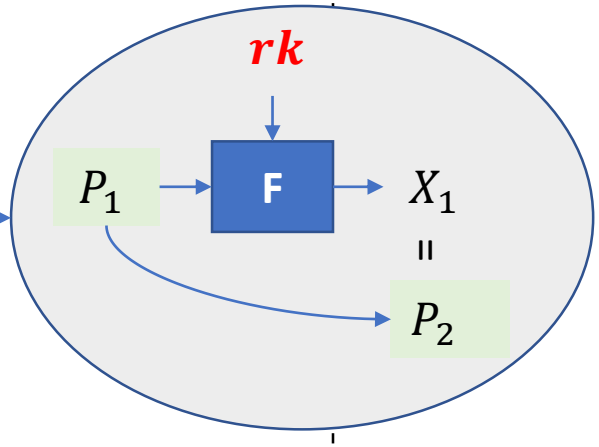
```
def ISB(in_state):
    out_state = [0, 0, 0, 0]
    for i in range(len(in_state)):
        out_state[i] = ISbox[in_state[i]]
    return out_state
```

```
#-- AR: Add Roundkey
def AR(in_state, rkey):
    out_state = [0, 0, 0, 0]
    for i in range(len(in_state)):
        out_state[i] = in_state[i] ^ rkey[i]
    return out_state
```

Slid Pair인지 확인하는 함수

- IsSlidPair()
  - 입력: (P1, C1), (P2, C2) – 두 개의 (평문, 암호문) 쌍
  - 출력: Slid pair 이면 (True, 암호키), 아니면 (False, [0,0,0,0])

```
#-------------------------
# 주어진 두 쌍의 (평문, 암호문)이 Slid pair 인지 확인
# (P1, C1), (P2, C2)
#-------------------------
def IsSlidPair(P1, C1, P2, C2):
    rk = [0,0,0,0]
    flag = False

    rk = Extract_RK(P1, P2)
    rnd_out = TC20E.Enc_Round(C1, rk)
    if rnd_out == C2:
        flag = True
    else:
        flag = False

    return (flag, rk)
```



Slid Pair와 라운드 키 추출

- 공격 대상 : 라운드 키가 모두 동일한 블록암호

- Slid pair → 라운드 키
  - Slid pair (P, C), (P', C') 가 주어지면,
  - 조건식: $P' = F(P)$, $C' = F(C)$ 에서 라운드 키 추출

- Slid Pair 찾기 ⇔ 라운드 키 찾기

따라서
Slide Pair만 찾으면
공격을 성공한다.

Slid Pair 확률

- 임의의 (평문, 암호문) 쌍 (P, C), (P', C')
  - Slid Pair 일 확률 $\Leftrightarrow P' = F(P)$ 일 확률
  - 왜냐하면, $P' = F(P)$ 이면 $C' = F(C)$를 만족한다.
  - 따라서 Slid Pair 일 확률 $= 2^{-n}$ (n: 블록크기 비트)

- M개의 (평문, 암호문) 쌍 $\{(P_i, C_i)\}_{i=1,2,\ldots,M}$
  - 순서쌍 $\{(P_i, C_i), (P_j, C_j)\}$을 만드는 방법: M^2 가지
  - Slid Pair 개수의 기대값 $= 2^{-n} \times M^2$
  - $M = 2^{n/2}$ 이면 1개의 Slid Pair가 발견될 것으로 기대됨

- 모든 라운드 키가 동일한 n비트 블록암호의 경우 $2^{n/2}$ 개의 (평문, 암호문) 쌍을 얻으면, 암호 키를 찾을 수 있을 것으로 예상됨.

Slide Attack 구현

실제로는
고정된 키로
암호화한
메시지를 수집

- 공격에 사용할 (평문, 암호문) 쌍 준비하기

```
import TC20_Enc_lib as TC20E
import TC20_Dec_lib as TC20D
import pickle    # 변수 저장
import random    # 난수 생성
import copy      # 딥 카피 (깊은 복사)

def Generate_Known_pt_ct(num_pair, key):
    list_pool = []
    print('Generating PT-CT pairs', end='')
    for i in range(0,num_pair):
        PT = [ random.randint(0,255), random.randint(0,255), \
               random.randint(0,255), random.randint(0,255)]
        CT = TC20E.TC20_Enc(PT, key)
        item = copy.deepcopy([PT,CT])
        list_pool.append(item)

    print('  Done! \n')
    return list_pool
```

랜덤한
(평문, 암호문)
쌍을 만든다.

파일로
저장하기
불러오기를
하려면?

```
save_var_to_file(pool, 'known_ptct.var')
load_pool = load_var_from_file('known_ptct.var')
```

Slide Attack 구현

- 수집한 (평문, 암호문) 쌍으로 암호 키 구하기

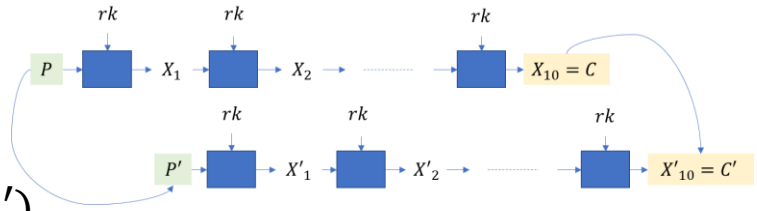
```
pool = [  [[0, 170, 160, 229], [55, 129, 88, 151]], [[0, 203, 83, 32], [180, 26, 160, 97]],
          ... , [[0, 39, 206, 64], [75, 58, 29, 43]]  ]
```

```
def Slide_Attack(pool):
    key = [0,0,0,0]

    print('Start Slide Attack', end='')
    for pair1 in pool:
        P1 = pair1[0]
        C1 = pair1[1]
        for pair2 in pool:
            P2 = pair2[0]
            C2 = pair2[1]
            flag, key = IsSlidPair(P1, C1, P2, C2)
            if flag:
                return (True, key)
        print('.', end='')

    print('  Done!\n')
    return (False, [0,0,0,0])
```

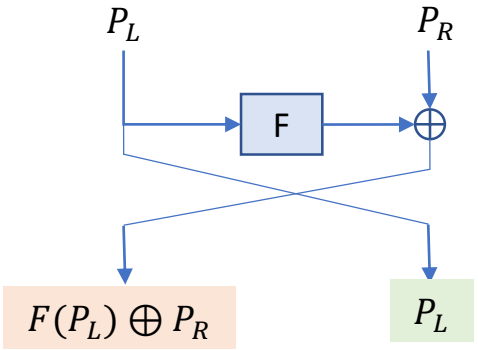
실행결과

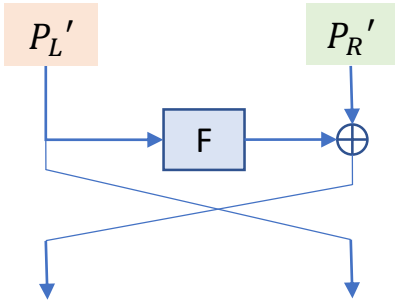
```
Start Slide Attack.....
.....True [1, 2, 3, 4]
```

Feistel 암호의 Slid Pair



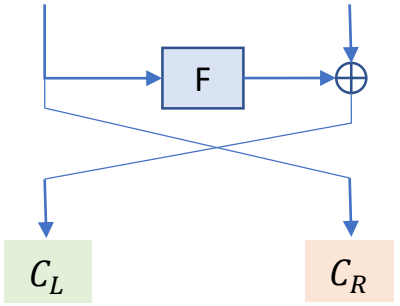
- Feistel 암호의 Slid Pair (P, C), (P', C')
  - 평문의 표현: $P = (PL, PR)$, $P' = (PL', PR')$
  - Feistel 암호에서 조건식 $P' = F(P)$은
    $\boldsymbol{F(PL) \oplus PR = PL'} \cdots$ (식1) $PL = PR' \cdots$ (식2)
- (평문, 암호문) 쌍을 수집하는 단계에서 $PL = PR'$을 만족하는 것만 모은다면, $\boldsymbol{F(PL) \oplus PR = PL'}$ 만 만족되면 Slid Pair 이다. (확률$2^{-n/2}$)

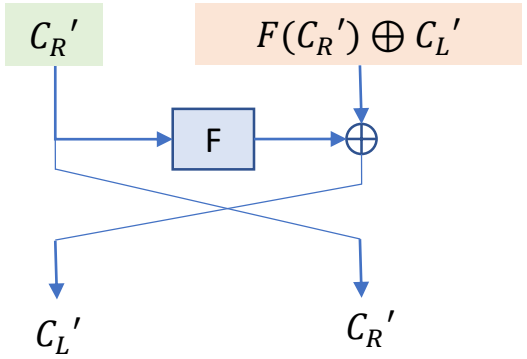




Feistel 암호의 Slid Pair

- Feistel 암호의 Slid Pair (P, C), (P', C')
  - $P = (PL, PR)$, $P' = (PL', PR')$
  - $C = (CL, CR)$, $C' = (CL', CR')$
  - Slid Pair의 경우 암호문은 다음 조건을 만족한다.
    $\boldsymbol{F(CR') \oplus CL' = CR} \cdots$ (식3) $CL = CR' \cdots$ (식4)





Feistel 암호의 Slide Attack

- 공격에 사용할 (평문, 암호문) 쌍 준비하기
  - $PL = \boldsymbol{P}$(고정) 를 선택한다.
  - 두 종류의(평문, 암호문) 쌍

$$A = \{((\boldsymbol{P}, PR_i), (CL_i, CR_i))\}_{i=1,2,\ldots,M}, \qquad B = \{((PL_j, \boldsymbol{P}), (CL_j, CR_j))\}_{i=1,2,\ldots,M}$$

  - 집합 A와 B에서 하나씩을 선택:
    $((P, PR), (CL, CR)) \in A, ((PL', P), (CL', CR')) \in B.$
  - 조건 $\boldsymbol{F(PL) \oplus PR = PL'}$을 만족하면 Slid Pair (확률$2^{-n/2}$)
  - Slid Pair 라면 다음 조건도 만족 (각 확률$2^{-n/2}$)
    $\boldsymbol{F(CR') \oplus CL' = CR} \cdots$ (식3) $\qquad \boldsymbol{CL = CR'} \cdots$ (식4)
  - (식4)만 만족하는 것을 Slid Pair 후보로 한다. (확률$2^{-n/2}$)
  - $M = 2^{n/4}$ 로 하면, 한 개의 Slid Pair를 기대함

$\boldsymbol{F(PL) \oplus PR = PL'} \cdots$ (식1) $\qquad PL = PR' \cdots$ (식2)