Linear cryptanalysis

https://youtu.be/CFla5WhT9cI

IT융합공학부 송경주

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Linear cryptanalysis

- Linear cryptanalysis

- 1992년 Matsui에 의해 소개된 암호 공격 방식

- DES에서 수행된 암호화에 대해 유사한 선형 식을 찾아 2^{47}개의 known-pairs을 이용하여 공격에 성공하였음

- 선형식을 이용하는 Linear cryptanalysis는 다른 블록암호에도 적용할 수 있음

- Differential cryptanalysis(차분 공격)은 평문을 선택하여 공격하는 chosen plaintext attack이지만 Linear cryptanalysis는 non-chosen plaintext attack이므로 조금 더 강한 공격이다.

- 블록암호의 일반적인 유형 중 하나인 SPN(substitution permutation network)구조에서는 유일한 비선형 요소인 S-box를 선형화 하는데 초점을 두고 연구하고 있음.

Linear cryptanalysis

- 암호의 안전성
  - 일반적인 암호는 난수화가 잘 되어 있어야 한다.
  - 난수화가 잘 되어 있는 암호의 Key는 각 비트는 0과 1이 될 확률이 $\frac{1}{2}$이어야 한다.
  - 한쪽으로 편향된 확률을 가졌다면 난수화가 잘 되어 있지 않다고 판단한다.

→Linear cryptanalysis에서는 선형식을 통해 편향된 확률을 기반으로 key값을 찾을 수 있다.

[Linear cryptanalysis 공격의 아이디어]

1. 전체 암호에 대한 approximate equation(근사 방정식)을 찾는다. 이때, 이 approximate equation은 평문, 암호문, 키를 연결한다.
2. 찾은 approximate equation을 키를 복구하는 distinguisher로 사용한다.

Linear cryptanalysis

- Linear cryptanalysis는 확률을 기반으로 하는 공격이므로 approximate equation에 대한 probability(p)를 계산한다.

- Probability(b)는 입력과 출력에 대해 선형식의 적중 확률.

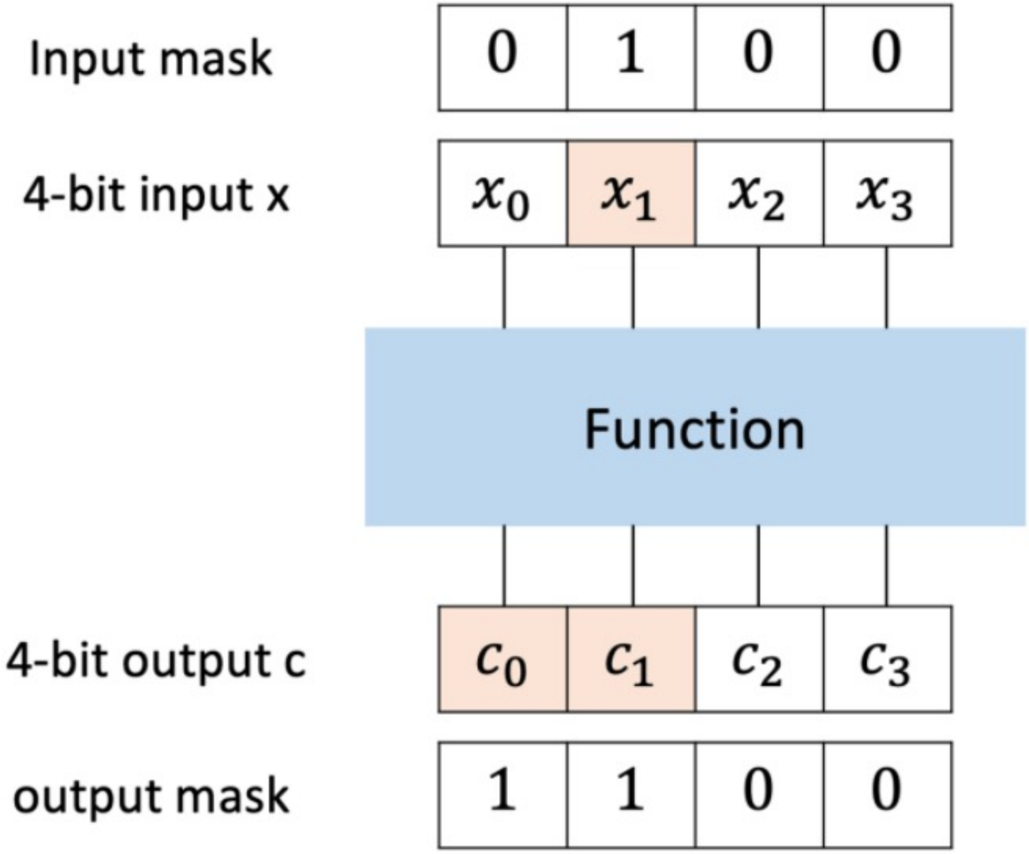
- Approximate equation이 non-linear 함수를 얼마나 linear 하게 잘 표현했는지 확인하는 지표로 bias(ε)를 활용.

$$\text{bias}(\varepsilon) = p - \frac{1}{2}$$

Linear cryptanalysis

Linear mask?

- Linear mask로는 input mask와 output mask가 있는데 이것은 각각 인풋과 곱해짐.
- Input mask = {0,1,0,0}, output mask = {1,1,0,0} 이면 x_1, c_0, c_1만 확률 계산에 사용된다.

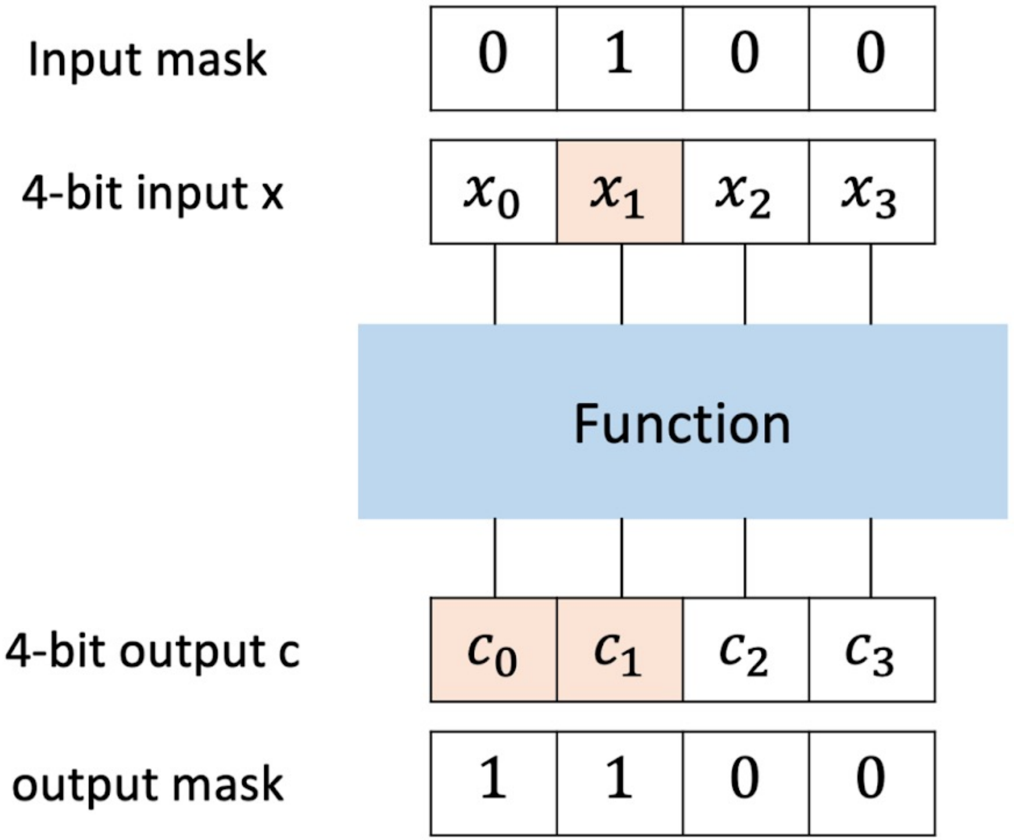


<4-bit의 인풋에 대해 4-bit 출력을 가지는 non-linear 함수>

Linear cryptanalysis

- 함수의 approximate equation이 α•x = β•F(x),(inner product, α•x := ⊕ $\alpha_i \cdot x_i$)라 가정하고 probability 계산 수행 결과 : approximate equation에 대한 probability는 p = ½ (ε = 0)

→ p = ½ (ε = 0) 는 0과 1이 될 확률이 같다는 것을 나타내므로 이 정보는 key를 찾는데 사용할 수 없다.



$x_0\ x_1\ x_2\ x_3$	$y_0\ y_1\ y_2\ y_3$	$x_1 = y_0 \oplus y_1$
0 0 0 0	0 0 0 0	O
0 0 0 1	0 0 0 1	O
0 0 1 0	0 0 1 0	O
0 0 1 1	0 0 1 1	O
0 1 0 0	0 1 0 0	O
0 1 0 1	0 1 0 1	O
0 1 1 0	0 1 1 0	O
0 1 1 1	0 1 1 1	O
1 0 0 0	1 0 0 0	X
1 0 0 1	1 0 0 1	X
1 0 1 0	1 0 1 0	X
1 0 1 1	1 0 1 1	X
1 1 0 0	1 1 0 0	X
1 1 0 1	1 1 0 1	X
1 1 1 0	1 1 1 0	X
1 1 1 1	1 1 1 1	X
Probability		$\frac{1}{2}$

Linear cryptanalysis

- Key를 찾는데 사용할 수 있는 정보?

- Linear approximation : $\alpha \cdot x = \beta \cdot F(x)$이라 가정하고 이에 대한 편향을 bias($\varepsilon$)라 하면, 다음과 같이 판단할 수 있다.

    1. if $\varepsilon = 0$, 0과 1이 될 확률이 같으므로 얻을 수 있는 정보가 없음.
    2. if $\varepsilon > 0$, $\alpha \cdot x = \beta \cdot F(x)$ 의 확률에 편향 되어 있으므로 정보 사용 가능.
    3. if $\varepsilon < 0$, $\alpha \cdot x = \beta \cdot F(x) \oplus 1$ 의 확률에 편향 되어 있으므로 정보 사용 가능.

- 그러므로 Linear mask 값을 바꿔가며 여러 input, output 조합에 대한 확률을 계산하고 사용 가능한 정보를 확인하여 활용해야 한다.

Linear cryptanalysis

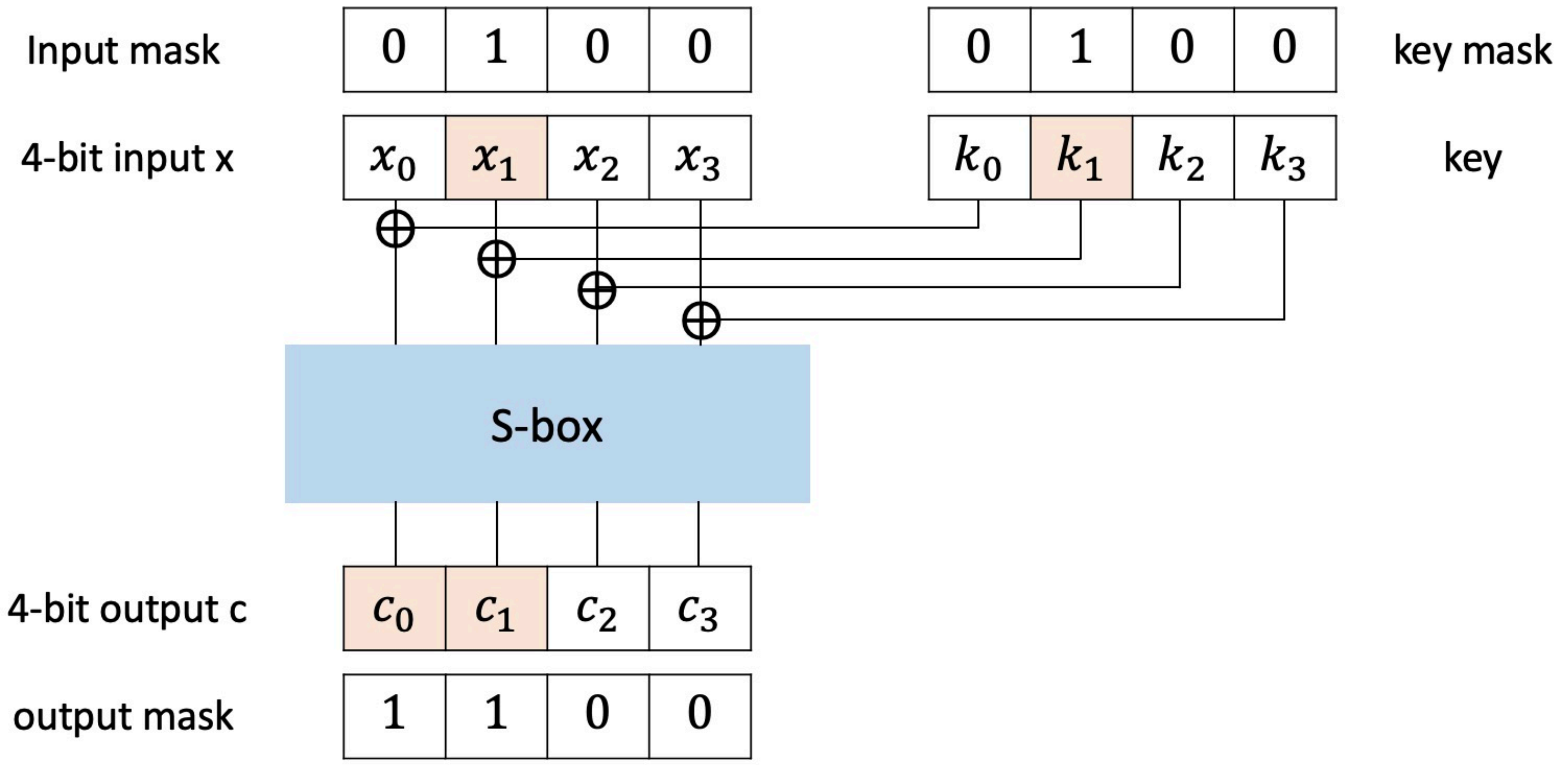
- 모든 linear mask 조합에 대한 값을 나타낸 표를 Linear Approximation Table(LAT) 라고 하고 LAT[α,β] = 2^b• ε 로 계산한다.
- LAT[α,β] = 0 이면, bias(ε)=0 이므로 데이터를 사용할 수 없다.
- 표의 값이 클수록 편향이 크므로 클수록 공격에 사용하기 적합한 정보이다.
- LAT 표를 사용하면 S-box의 강도를 정의할 수 있다.

$\alpha\backslash\beta$	0	1	2	3	⋯	F
0	0	-4	-2	0	⋯	8
1	2	0	2	4	⋯	2
2	0	2	2	0	⋯	4
3	0	-2	2	4	⋯	-4
⋮				⋮		
F	4	1	2	0	⋯	0

4-bit Linear Approximation Table (LAT)

Linear cryptanalysis

- 앞서 설명한 non-linear 함수에 대해 key를 추가하면 암호의 S-box로 표현될 수 있다. 입력에 key가 추가되므로 key mask가 필요하며 이것은 input mask와 같은 값을 가진다.
- 이전에 가정한 approximate equation : α•x = β•F(x) (inner product, α•x := ⊕ $\alpha_i \bullet x_i$)은 key 값을 추가하여 <u>approximate equation : α•x ⊕ K•k = β•F(x) (i.e. α•x ⊕ β•F(x) = K•k)</u> 로 가정할 수 있다.
- approximate equation을 구했다고 가정하면, 알고리즘을 수행하여 key recovey를 수행할 수 있다.



Linear cryptanalysis

- Key Recovery

[Matsui's Algorithm 1] - Key-recovery using an r-round approximation

approximate equation : α•x ⊕ β•F(x) ⊕ K•k = 0, (평문-암호문 쌍을 충분하고, bias ≠ 0)라고 가정.

1. $T_0 = 0, T_1 = 0$: count 상수 초기화
2. 평문-암호문 쌍 만큼 반복
   - If α•x ⊕ β•F(x) = 0, T_0++
   - If α•x ⊕ β•F(x) = 1, T_1++
3. 식을 통해 key의 1-bit 를 알 수 있음
   - If $T_0 > T_1$, → K•k=0
   - If $T_0 < T_1$, → K•k=1

장점 : 정보를 직접 사용하여 기밀성을 공격할 수 있음

단점 : 많은 데이터에 대해 한 비트만 알 수 있으므로 하나 이상의 approximation 이 필요함.

Linear cryptanalysis

- Key Recovery

[Matsui's Algorithm 2] - Last-rounds attack

1. 충분히 많은 known-plaintext pairs을 확보
2. 마지막 라운드 키의 모든 후보 k_r에 대한 연산 수행
   - $T_0^{k_r} = 0, T_1^{k_r} = 0$: Count 상수를 초기화
   - c(ciphertext)에 대한 1-round 를 복호화 하여 중간 C'을 얻음
   - If α•x ⊕ β•C', T_0++ // else T_1++
   - Key가 정답일수록 T_0와 T_1의 차 ($T_0^{k_r} - T_1^{k_r}$)가 큼 → T_0와 T_1의 차가 가장 큰 key를 찾음

장점 : (r-1) round에 대해서만 approximation이 필요함, 한번에 많은 key 정보를 알 수 있음

단점 : 더 많은 비트를 추측해야 하므로 비용이 많이 필요할 수 있음.

Q & A