1주차 실전 암호학 스터디 정리

안녕하세요 여러분, 오랜만이네요.

원래는 당일날 정리를 올려드리려고 했는데, 추석 때문에 귀찮아져서 오늘 올리게 됩니다.

1주차는 OT같이 했는데, 별 이야기는 안했어요.

10페이지 정도 될 것 같습니다. 그런데 꼭 읽어주세요

가장 중요합니다.

1. 암호의 목적은?

사전적인 정의를 일단 제쳐두고 생각해보죠, 일단 우리는 암호를 다른사람이 모르게 하려고 씁니다.

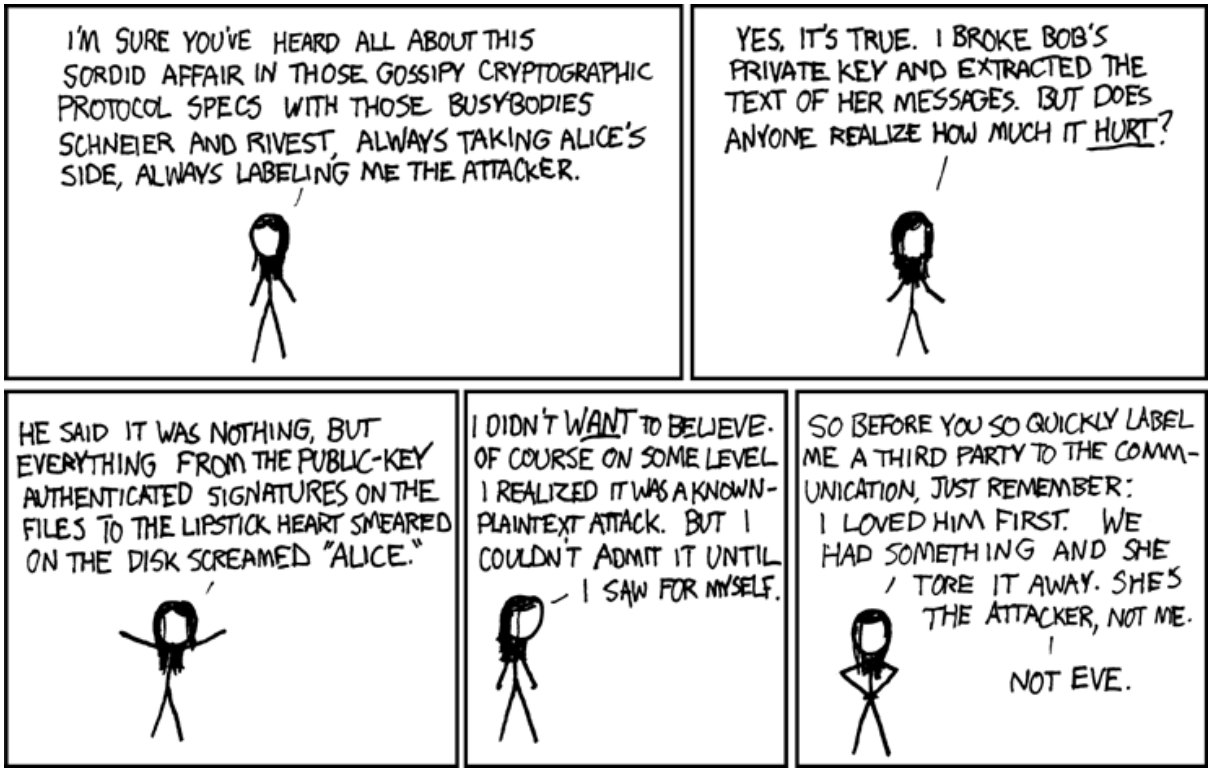
일단 우리는 '비밀(key)를 모르는 사람이 암호문을 봤을 때, 어떠한 정보도 얻을 수 없어야 하는 것'을 암호의 목적으로 합시다. 이 정도만 생각하고 가도 충분합니다.

2. 용어 정리

흔히 A, B가 메세지를 주고받는데 보통 암호학에서는 Alice, Bob이라고 합니다.

주로 암호문(ciphertext)을 주고 받습니다. 이와 반대되는 개념은 평문(plaintext)입니다.

중간에 Eve라는 사람이 등장하는데, 흔히 암호문을 가로채서 보는 사람이라고 생각하면 됩니다.



이정도만 알고가면 뒤는 쉽게 이해할 수 있습니다.

3. mod와 합동

스터디를 들으면서 mod라는 글자와 짝대기 3개 있는거(≡)를 많이 보게 될거에요.

(짝대기 3개 있는거는 ===으로 쓸게요, 수식 붙여넣기 귀찮습니다ㅠㅠ)

Mod는 여러분이 컴프I 배울 때 봤던 %연산이라고 생각하면 됩니다.

나머지를 나타내는 것이죠

예를 들어 3 mod 2 하면 1이겠죠.

(3을 2로 나눈 나머지)

===(≡)는 다음과 같이 씁니다.

7 === 4 mod 3

10 === 2 mod 2

12 === 16 mod 4

22 === 71 mod 7

아실 것이라 생각합니다.

a === b mod c는

a를 c로 나눈 나머지 = b를 c로 나눈 나머지 라고 생각하시면 됩니다.

4. Caesar cipher

이러한 톱니바퀴를 보신 분이 있을겁니다. 고전암호중에는 유명한 카이사르 암호입니다. 원하는 만큼 안쪽 톱니바퀴를 돌리고 암호화를 합니다.

예를 들어, 톱니바퀴가

(바깥쪽) : A B C D ... Z

(안쪽) : B C D E ... A

와 같이 세팅되어 있다고 하면

HELLO WORLD는

IFMMP XPSME가 되겠죠?

(H -> I, E -> F, ... , D -> E)

(한글자씩 밀렸습니다)

안쪽 바퀴를 한번 더 밀어 봅시다.

그렇다면

(바깥쪽) : A B C D ... Z

(안쪽) : C D E F ... B

가 되어서

HELLO WORLD는

JGNNQ YQTNF가 될 것입니다.

이와 같이 평문(원래 문장)은 톱니바퀴를 얼마나 돌렸는지에 따라 수많은 암호문(바뀐 문장)이 될 수 있습니다.

그렇다면 계속 밀면 계속 다른 암호문이 생성될까요?

안쪽에 톱니바퀴를 계속 밀면

톱니바퀴는

(바깥쪽) : A B C D ... Z

(안쪽) : Z A B C ... D

를 지나서

(바깥쪽) : A B C D ... Z

(안쪽) : A B C D ... Z

가 됩니다. 평문과 암호문이 똑같아 지는 순간이죠

이 이후는

(바깥쪽) : A B C D ... Z

(안쪽) : B C D E ... A

이 되면서 이전 상황이랑 똑같아집니다.

평문과 암호문이 같은 상황을 제외하면 톱니바퀴의 세팅에 따라 25개의 암호문이 만들어집니다.

이때 톱니바퀴를 몇 번 밀었는지가 비밀(key)가 될 수 있겠죠

5. 수학적으로,

암호학은 수학입니다.

웬만한 암호화 방식은 다 수학으로 표현할 수 있습니다.

A = 0, B = 1, … , Z = 25라고 생각하고

아까 봤던 Caesar cipher를 수학적으로 표현한다면,

$C = P + K \bmod 26$

$P = C - K \bmod 26$

이 됩니다.

(C : 암호문, P : 평문, K : 키(돌린 횟수))

(추가 : 정확히 쓰려면 $C = (P + K) \bmod 26$으로 써야 하지만 그냥 mod의 계산순위를 가장 낮게 생각하세요)

K가 0이 된다면,

$C = P \bmod 26$이 되면서

암호문과 평문이 같습니다.

그렇다면 K에 대해서 생각해봅시다

K는 톱니바퀴를 몇 번 밀었는지를 나타내는 수입니다.

만약 K = 26이 된다면

C = P + 26 mod 26이 되면서

C = P mod 26이 됩니다.

(이유 : 어떤 수 x에다가 26을 더한 수를 26으로 나눈 나머지는 x를 그냥 26으로 나눈 나머지와 같습니다)

26을 넘어가도 막 새로운 암호문이 생성되지 않습니다. 톱니바퀴를 27번 민 상태는 1번 민 상태와 같고, 28번 민 상태는 2번 민 상태와 같고... 이렇게 생각하면,

실질적으로 K가 될 수 있는 수는

0을 제외한다면

{1, 2, ... , 25} -> 25가지입니다.

6. Key space

이와 같이 키가 될 수 있는 경우의 수를 Key space라고 합니다.

아까와 같은 Caesar cipher에서는

Key space = 25 (K=0 제외)입니다.

7. Bruteforce Attack

이제 문제를 생각해봅시다. 암호화 방식이 아까와 같은 Caesar 암호고, JGNNQ YQTNF와 같은 암호문이 주어져있다면 우리는 어떻게 원래 문장을 알 수 있을까요?

가장 쉽게 생각할 수 있는 방법은, 그냥 K는 1부터 25까지 모든 경우에 대해서 복호화를 해보는 것입니다.

실제로 모든 경우에 대해서 복호화를 하면,

JGNNQ YQTNF <- K = 0

IFMMP XPSME <- K = 1

HELLO WORLD <- K = 2

GDKKN VNQKC <- K = 3

FCJJM UMPJB <- K = 4

EBIIL TLOIA <- K = 5

DAHHK SKNHZ <- K = 6

CZGGJ RJMGY <- K = 7

BYFFI QILFX <- K = 8

AXEEH PHKEW <- K = 9

ZWDDG OGJDV <- K = 10

YVCCF NFICU <- K = 11

XUBBE MEHBT <- K = 12

WTAAD LDGAS <- K = 13

VSZZC KCFZR <- K = 14

URYYB JBEYQ <- K = 15

TQXXA IADXP <- K = 16

SPWWZ HZCWO <- K = 17

ROVVY GYBVN <- K = 18

QNUUX FXAUM <- K = 19

PMTTW EWZTL <- K = 20

OLSSV DVYSK <- K = 21

NKRRU CUXRJ <- K = 22

MJQQT BTWQI <- K = 23

LIPPS ASVPH <- K = 24

KHOOR ZRUOG <- K = 25

가 됩니다.

K = 2일 때, HELLO WORLD라는 의미있는 문장이 나오고

이것이 우리가 찾던 평문임을 알 수 있겠네요

이와 같이 key가 될 수 있는 모든 경우에 대해 복호화를 하는 공격을 Bruteforce Attack(무차별 대입 공격)이라고 합니다.

모든 암호는 이 공격에 의해 뚫립니다. 모든 가능성을 계산하는데 어떻게 안 뚫립니까.

하지만, key가 될 수 있는 경우가 많아진다면, 복호화를 하는데 시간이 더 들겠죠

컴퓨터의 처리성능은 계속 늘어납니다, 그에 따라 무차별 대입 공격에 드는 시간이 줄어듭니다.

결국, 암호는 시간이 지남에 따라 Key space가 커지는 방향으로 갈 수 밖에 없습니다.

Caesar cipher같은 #key = 25(key space를 #key로 줄여 쓰겠습니다)인 암호는 그냥 손으로도 몇 분 내에 손쉽게 계산이 가능합니다.

#key 가 1000이라면? 조금 시간은 들겠지만 풀 수는 있겠네요

#key 가 1000000이라면? 사람 손으로는 풀 수 없겠네요.

하지만 컴퓨터는 쉽게 풀 수 있습니다.

~~요새 컴퓨터는 여러분이 알고리즘을 이상하게 짜도 잘 돌아갑니다.~~

~~이게 다 성능이 좋아서 그렇습니다.~~

#key 가 10^1000이라면? 컴퓨터도 짧은시간 내에 계산하기는 힘들겠네요.

암호는 Key space가 늘어나는 방향으로 발전할 수 밖에 없습니다. 컴퓨터에 처리성능이 좋아짐에 따라서, 무차별 대입 공격의 속도 또한 늘어나고, 모든 가능성을 계산하는 시간이 줄어들기 때문이죠.

옛날에 쓰던 암호인 DES의 key space는 2^56입니다. 56비트짜리 키를 쓴다는 뜻이죠.

1998년 기준으로 무차별 대입 공격을 진행하는데 56시간이 걸렸습니다.

하지만 요즘 쓰는 AES의 Key space는 2^256입니다. 256비트짜리 키를 쓴다는 뜻입니다.

아마 이것도 컴퓨터의 처리성능이 발전함에 따라 Key space가 늘어난 새 암호로 바뀌겠죠.

Key space가 작으면 무차별대입으로 쉽게 풀 수 있다는 말을 하고싶었습니다.

하지만 이 공격방법이 언제나 통하지는 않죠.

다음 시간에는 실습과 함께 다른 내용을 알아보도록 하겠습니다.

8. 참고

위에 있는 Caesar cipher의 무차별 대입 공격 코드입니다.

파이썬으로 작성했습니다. 파이썬이 익숙하지 않아면 지금부터 친해지면 됩니다.

```
>>> def disk(k):
...     book = {}
...     for i in range(0, 26):
...             disk[chr(i + 65)] = chr(((i - k) % 26) + 65)
>>> for i in range(0, 26):
...     plain = ""
...     dec = disk(i)
...     for j in cipher:
...             if j in dec:
```

```
...                             plain += dec[j]
...                     else:
...                             plain += j
...         print(plain, "<- K =", i)
```