Homework 1 Report B07902127 資工二 羅義鈞

1. CIA

Confidentiality: 機密性。確保資料傳遞以及儲存的私密性，避免未經授權的使用者有意或是無意去得到或是揭露資料的內容。
實例: 網路封包未經加密就被攔截，或像是投影片裡面所提到的，期中考卷於網路管道被其他人竊走。

Integrity: 完整性。避免未經授權的第三者去竄改資料內容。要達到完整性，必須於資料傳送以及儲存過程中去證明內容未經竄改以及偽造。
實例：ATM 或是銀行軟體須保證任何提款、轉帳等資訊紀錄沒有遭到竄改，否則會出很大的問題。

Availability: 可用性。被授權的使用者應該在任何有需求的時間點下，都能去 access 這些資料。
實例：如 Denial of Service 這樣的攻擊，就是用大量的假資訊讓系統癱瘓，以至於使用者真的需要時無法使用。網路管理以及系統管理就會實際需要達成 Availability。例如定期系統維護、檢查負荷、備份等等，這些都是在確保能盡力達到 Availability。

2. Hash Function

(1) One-wayness:

The definition is that given y, it is computationally infeasible to find x such that y = H(x), which H is a hash function. That means it is nontrivial to obtain x with the information given of y. For example, Alice and Bob want to play rock paper scissors online. Since Alice cannot let Bob know what decision she makes before Bob makes his decision, she uses hash function with salt to hash her decision. Bob cannot know what strategy Alice made by knowing the hash value, then this is how one-wayness works.

(2) Weak Collision Resistance:

The definition is that given x, it is computationally infeasible to find $x' \neq x$ such that $H(x') = H(x)$. For example, NTU software downloads use MD5 to verify whether files have been modified or linked to other download sources.

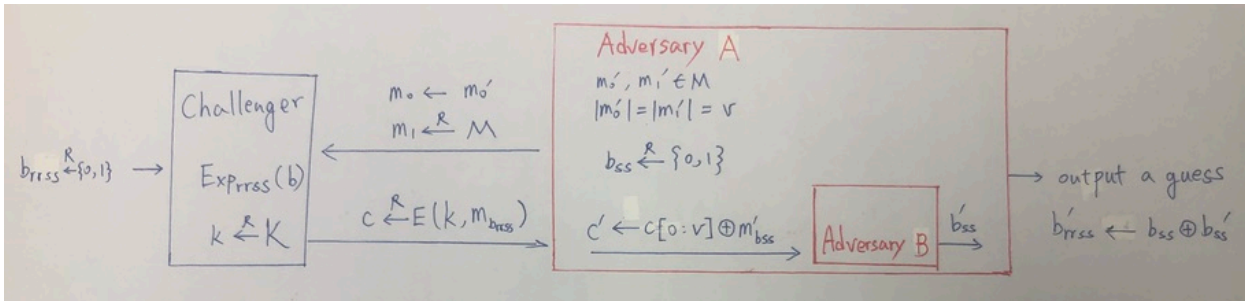
(3) Strong Collision Resistance:

The definition is that it is computationally infeasible to find x and x' such that H(x) = H(x'). For example, digital signature often signed over the hash of the message instead of the message itself. It can reach authentication due to strong collision resistance.

3. Semantic Security

Proof:

(⟸) 圖中，b'_{rrss}就是$\hat{b}$, rrss 為 rare/random semantic secure 的縮寫



$$\Pr[Exp_{rrss}(0) = 1] = \Pr\left[\hat{b} \neq b \mid Exp_{rrss}(0)\right]$$

$$= \Pr[Exp_{ss}(0) = 1] * \Pr[b_{ss} = 0] + Pr[Exp_{ss}(1) = 0] * \Pr[b_{ss} = 1]$$

$$= \frac{1}{2}(1 + Pr[Exp_{ss}(0) = 1] - Pr[Exp_{ss}(1) = 1])$$

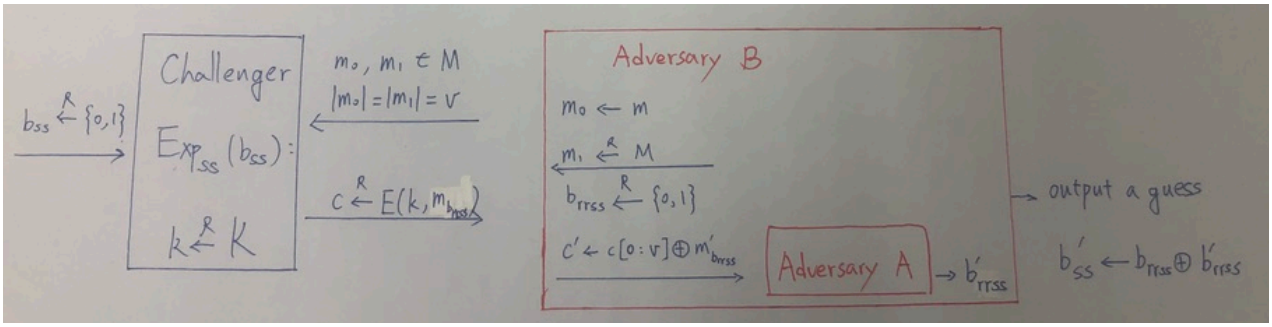
$$\Pr[Exp_{rrss}(1) = 1] = \Pr\left[\hat{b} \neq b \mid Exp_{rrss}(1)\right] = \frac{1}{2}$$

$$\mathcal{A}'s\ advantage = |\Pr[\hat{b} = b] - \frac{1}{2}| = \frac{1}{2}|\Pr[Exp_{rrss}(0) = 1] - \Pr[Exp_{rrss}(1) = 1]|$$

$$= \frac{1}{2} SS_{adv}[A, \mathcal{E}]$$

若$SS_{adv}[A, \mathcal{E}]$為 non-negligible，則$\mathcal{A}'s\ advantage$為 non-negligible，故得證。

(⟹)



$$\Pr[Exp_{ss}(0)=1]=\Pr\left[\widehat{b_{ss}}\neq b_{ss}\mid Exp_{ss}(0)\right]$$
$$=\Pr[Exp_{rrss}(0)=1]*\Pr[b_{rrss}=0]+Pr[Exp_{rrss}(1)=0]*\Pr[b_{rrss}=1]$$
$$=\frac{1}{2}(1+Pr[Exp_{ss}(0)=1]-Pr[Exp_{ss}(1)=1])$$

$$\Pr[Exp_{ss}(1)=1]=\Pr\left[\widehat{b_{ss}}\neq b_{ss}\mid Exp_{ss}(1)\right]=\frac{1}{2}$$

$$SS_{adv}[B,\mathcal{E}]=|\Pr[Exp_{ss}(0)=1]-\Pr[Exp_{ss}(1)=1]|$$

$$=|\Pr[Exp_{rrss}(0)=1]-\Pr[Exp_{rrss}(1)=1]|=2*\left|Pr\left[\hat{b}=b\right]-\frac{1}{2}\right|=2*A's\ advantage$$

若$A's\ advantage$為 non-negligible，則$SS_{adv}[B,\mathcal{E}]$為 non-negligible，故得證。

4. Simple Crypto

Flag: **CNS{CRYPT0_1S_S0_$IMP13}**

其中 round 3, round 4, round 5 需要手動輸入：round 3 輸入一組數字，round 4 輸入答案的句子，round 5 輸入一組數字。

Round 1: ROT-13 加密;　Round 2 & Round 3: 凱薩加密

Round 4: 亂序加密;　Round 5: 排列加密

最後再用 64base 去 decode，得到最終的 flag

5. Find The Secret

```
D1 is: 937126206405537801712293347433039190952528715630
D2 is: 347512686763893959654010162520158511966976470226
D3 is: 586887794480827544493863605619114998419481725887
a0 is: 22903031829579284005874185426584358429053
flag is: CNS{V3rIfi4BLe!!}
```

根據題目，D_1 = (1, A(1) mod q), D_2 = (2, A(2) mod q), D_3 = (3, A(3) mod q)

現在各 10000 組數據中只有一組是真正的，我們要用上 hint 給的 c_0, c_1, c_2。

首先，

$$c_0*c_1*c_2\equiv g^{(a_0+a_1+a_2)}\ (mod\ p)$$
$$\equiv g^{(a_0+a_1+a_2)\ mod\ q+k*q}\ (mod\ p),\qquad k\in\mathbb{N}$$
$$\equiv g^{(a_0+a_1+a_2)\ mod\ q}*g^{k*q}\ (mod\ p)$$

由題意得知，

$$g^q\equiv 1\ (\text{mod p})\ and\ \ q\mid(p-1)$$
$$\therefore c_0*c_1*c_2\equiv g^{(a_0+a_1+a_2)\ mod\ q}\ (mod\ p)$$

$$\equiv g^{A(1)\ mod\ q}\ (mod\ p)$$

同理，

$$c_0 * c_1^2 * c_2^4 \equiv g^{A(2)\ mod\ q}\ (mod\ p)$$

$$c_0 * c_1^3 * c_2^9 \equiv g^{A(3)\ mod\ q}\ (mod\ p)$$

因此我們可以藉由比對得到真正的 D_1, D_2, D_3。

接著，我們要找出正確的 secret a_0。經過方程式加減消去後，我們知道:

$a_0 = 3 * (A(1)\ mod\ q + k_1 * q) - 3 * (A(2)\ mod\ q + k_2 * q) + (A(3)\ mod\ q + k_3 * q)$

$a_1 = -5 * (A(1)\ mod\ q + k_1 * q) + 8 * (A(2)\ mod\ q + k_2 * q) + 3 * (A(3)\ mod\ q + k_3 * q)$

$a_0 = (A(1)\ mod\ q + k_1 * q) - 2 * (A(2)\ mod\ q + k_2 * q) + (A(3)\ mod\ q + k_3 * q)$

$k_1, k_2, k_3 \in \mathbb{N}$

理論上要窮舉所有可能的k_1, k_2, k_3，不過經過嘗試其實找到第一個小於 q 的a_0就行了。

最後再將數字a_0轉成二進制，每 8 個 bit 為一個字元，就得到 flag 了

6. Cute Baby Cat

(1)

```
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylen
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_yleno
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol_
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol_A
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol_A{
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol_A{S
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol_A{SN
now is: })=^._.^=(~woem~woeM.sdneirf_st:csed||0:nimdasi||0:pivsi||naw_tac_ylenol_A{SNC
flag is: CNS{A_lonely_cat_wan||isvip:0||isadmin:0||desc:ts_friends.Meow~meow~(=^._.^=)}
[*] Closed connection to cns.csie.org port 10202
```

Flag1: **CNS{A_lonely_cat_wan||isvip:0||isadmin:0||desc:ts_friends.Meow~meow~(=^._.^=)}**

觀察得到的 token，發現前 16 個為 IV，後面則是加密後的結果，每 16 個為一個 block。

利用上課所教的 CBC Padding Oracle Attack，先修改倒數第二個 block 中的最後一個 bytes，直到沒有出現 padding error 為止。接著以此類推，直到 reveal 所有的 plain text，轉成二進制後在每八個 bit 為一個字元，就得到第一個 flag 了。

(2)(3)

```
iteration 15...
iteration 16...
Here is a cat gif : CNS{B1t_FL1p_4T7Ack_15_Tr1ckY.a_cut_cat.gif}
Username :
Another cat! CNS{(=^.x.^=)W15H Y0U H4V3 FUN}
****************** user info ******************
*                                             *
* username    : Hi_from_TA_1?crM\x1e'D
d\x7f\x19\x1cf\x0c6\x1en    *
* description : ts_friends.Meow~meow~(=^._.^=)}  *
* isvip       : 1                             *
* isadmin     : 1                             *
*                                             *
***********************************************
======= menu =======
1. account info
2. create user
3. get secret
4. logout
====================
your choice :
[*] Closed connection to cns.csie.org port 10202
(base)
```

Flag2: **CNS{(=^.x.^=)W15H Y0U H4V3 FUN}**

Flag3: **CNS{B1t_FL1p_4T7Ack_15_Tr1ckY.a_cut_cat.gif}**

從第一個 flag 會發現說如果直接用它給的 token 的話，isadmin 和 isvip 都是 0。由於現在已知後面五個 block 的 plain text 了，因此先將 plain text 改為 isadmin:1(第四格 block)。這樣一來，該格 plain text 對應的前一個 cypher block(第三格)也必須修改一個 bytes。由於 AES 加密時，差一個 bytes 會讓通過 key 的結果差非常多，因此我們得出來的前一格 plain text(第三格)會整個爛掉。這時候又使用前面找第一個 flag 時的 Padding Oracle Attack，藉由不斷修改第二格的 cypher，使得它跟原本的 plain text XOR 之後會得到那個經過 key 加密後亂掉的結果，就可以去反推 cypher block。最後因為我們得到的 cypher block 可能會使得 XOR 出來的字元不是在 128~256 之間，因此去修改前面三個 bytes 直到不會出現 unicode decode error 為止，就能拿到第二個跟第三個 flag 了。

7. Resourceful Secret Agent

```
$ python3 code7.py
[+] Opening connection to cns.csie.org on port 10201: Done

CNS{T3xtb0ok_RSA_1s_uNSaF3}||12782629855016838226||info

===============
 1. Encrypt
 2. Decrypt
 3. Command
 4. Exit
===============
>
109640187244432374960240684899932570377535209749900965178385817049134527537511801770698425001603428159236966985908316667452351459800127253168121012330792521697466459890113120112973474597780142108551796680395373043178247340852919579097992201092385277071732362843543154707313706683313756692918292913266362141588||f012a93ddb409395afbb15dbaf1ac17f5408ec43fbe02ce0c48078860d09cf4cb6a5dca66e98ddc63f8426a4f58816e3c2374442dcc296efd7d9eacd7aacafa7

CNS{eNCrYp7_Y0uR_s3cr3T_w1sely}

[*] Closed connection to cns.csie.org port 10201
```

Flag1: **CNS{T3xtb0ok_RSA_1s_uNSaF3}**

這題一開始會拿到一組 RSA||AES 的 command，透過這個 command 能拿到 n 和 e。如果把前面 RSA 的部分直接拿去 decrypt 的話會失敗。因此，為了欺騙它，先將密文乘上一個加密過的常數，mod n 之後再拿去 decrypt，就可以得到 hex 過的 plain text 乘上常數了。再把它乘上該常數的乘法反元素，decode 之後就得到第一個 flag 了。

Flag2: **CNS{eNCrYp7_Y0uR_s3cr3T_w1sely}**

現在的 command 是 info，我們要改成 getflag 才能拿到第二個 flag。因此這邊分成兩個部分：一邊是 RSA，另一邊是 AES。RSA 只要將修改後的 hex 拿去 encrypt 就可以得到密文了。接著要讓 AES 一致，觀察 AES 加密過程，發現是使用 OFB mode，因此只要修改最後一個 cipher block，使得：

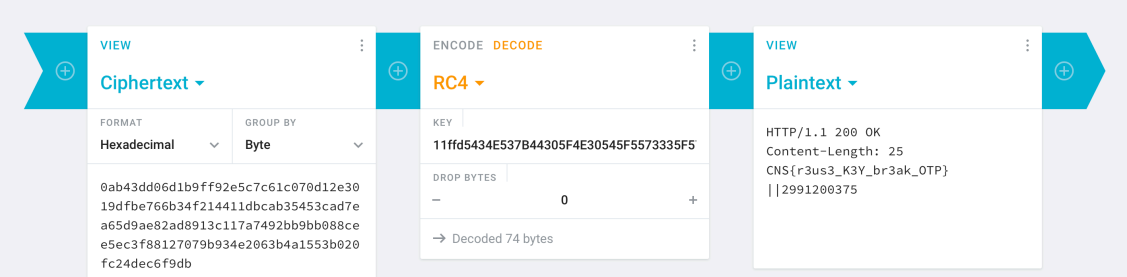
$$modified\ cypher\ block = plain\ text \oplus original\ cypher\ block \oplus modified\ plain\ text$$

就可以拿到改好的 cypher block，拼出 command: RSA||AES(modified)，得到第二個 flag。

8. Wired Equivalent Privacy

```
$ python code8.py
mykey: 11ffd5434E537B44305F4E30545F5573335F5745505F346E796D3072337D

flag is: CNS{r3us3_K3Y_br3ak_OTP}
bonus!!: CNS{D0_N0T_Us3_WEP_4nym0r3}
```



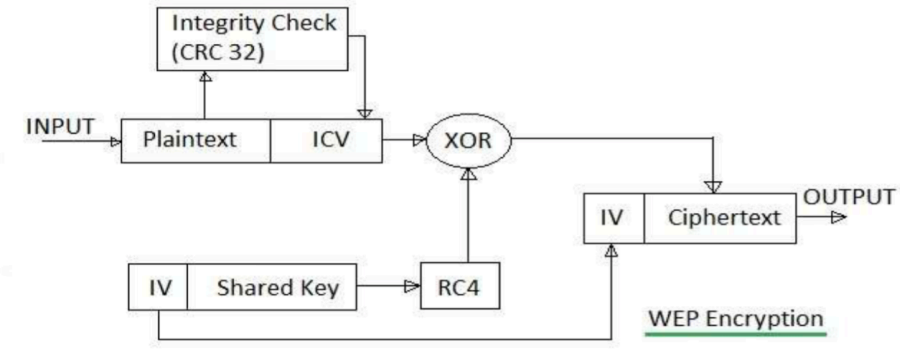
Reference:

https://github.com/jackieden26/FMS-Attack/blob/e397dc11740e283beec4fff1fd04eea77d01fbed/rc4.py#L8

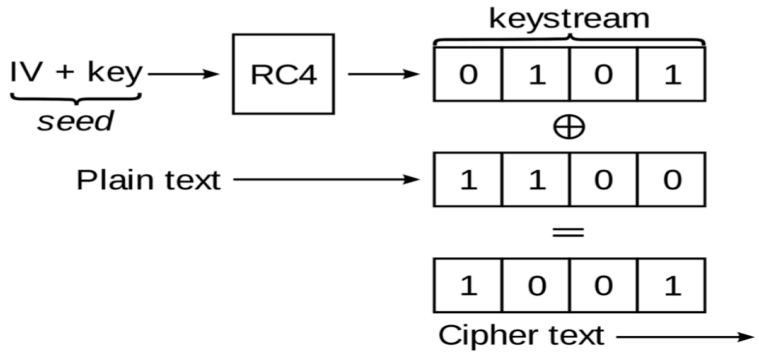
https://github.com/jackieden26/FMS-Attack/blob/master/keyRecover.py?fbclid=IwAR24VrssU-GuvGwPb9cTH7Y6EC9bjhbZNSnJvlIlD-2i5-S0DqISgSC4iUo

網路上直接查到找 rc4 的 key 的方法，所以把它拿來當工具使用。由於需要多筆的封包資訊才能破解，因此總共大概擷取了一萬多筆的封包，刪除掉重複 IV 的封包後大約剩下七千多筆，轉成 csv 的形式後跑網路上的 code 就能拿到 key 了(bonus flag)！接著把拿到的 key 前面加上 IV，連同密文拿到線上的 decoder 解譯，就得到一般的 flag 了。

至於這個工具的原理是 FMS-Attack。首先 WEP 的加密是使用 RC4：



RC4 是一個 stream cypher，目標是達到 PRG：



演算法為 KSA(Key Schedule Algorithm)和 PRGA(Pseudo Random Generation Algorithm)。

其弱點為每次的金鑰皆為 IV + key，而在 stream cypher 之中，多次使用同樣的 key 是十分危險的，因為會產生出相同的 key stream。由於封包的開頭是 HTTP 的 H，且每個攔截的封包都能獲得 IV，因此透過 FMS-Attack，隨著封包數量累積越來越多，分析過後破解得到 key 的機率也隨之提升。

FMS Attack:

首先，定義經過 t 次 iteration 之後，substitution box 的狀態為 S_t，index 為 i_t 及 j_t，當前 output key stream byte 為 Z_t。則：$Z_t = S_t[S_t[i_t] + S_t[j_t]]$

定義 L 為 session key 的長度，I 為 IV 的 bytes 數(這題是 I = 3)，則：

key K $= [\text{IV}, \text{key}[0], \text{key}[1], \ldots, \text{key}[\text{I} - 1 - \text{i}]]$

我們目標是一步步從第一個未知的 secret key bytes 去 recover，若其位置為 A，則在 key 中的實際 index 為 A+I = A+3。現在已知 plain text 為 H 開頭，也知道 IV 和 encrypted cypher stream。我們希望用這樣的形式去檢驗 IV: (A+3, N-1=255, V), V 可以是任意數。開始進行第一輪的 ksa:

$A+3$	$N-1$	V	$K[3]$		$K[A+3]$	
0	1	2			$A+3$	
$A+3$	1	2			0	
i_0					j_0	

這是此時的 substitution box status，下一輪會變成這樣：

$A+3$	$N-1$	V	$K[3]$		$K[A+3]$	
0	1	2			$A+3$	
$A+3$	0	2			1	
	i_1				j_1	

再下一輪時，j 會被多加上 V+2。之後的每一輪，我們都能計算直到出現第一個未知的 byte(A+3)。此時確認 S[0]和 S[1]是否有被改動。如果沒有，則我們知道 j 接下來又要加上 $S_{A+2}[i_{A+3}]$+K[A+3]，接著進行 swap，substitution box status 變為：

$A+3$	$N-1$	V	$K[3]$		$K[A+3]$	
0	1	2			$A+3$	
$A+3$	0	$S[2]$			$S_{A+3}[A+3]$	
					i_{A+3}	

由於已知 S_{A+2} 以及 j_{A+2}，如果我們知道 $S_{A+3}[A+3]$的值，就會知道它在 S_{A+2} 的位置，也就是 j_{A+3} 的值。如此一來，就能算出 K[A+3]了。

Formula:

$$Z_0 = S\left[S[1] + S[S[1]]\right] = S[0 + S[0]] = S[A+3] = S_A + 2[j_{A+3}]$$
$$= S_A + 2[j_{A+2} + S_{A+2}[A+3] + K[A+3]]$$

Acknowledgement: b07902002 連崇安、b07902135 黃申昌、b07902005 劉冠鴻