撰寫：資管三 B04705001 陳約廷
日期：2018.03.21
標題：密碼學與資訊安全 Homework 1 report

---

1. CIA (10%)

Confidentiality：保密性，使內容不被其他人所看見。

例：電子郵件的傳輸需要有保密性，不被收件人以外的使用者查閱，否則不具保密性

Integrity：完整性，使內容不被其他人所竄改。

例：帳號需要被保持完整性，不能遭其他人盜用。如果帳號遭到盜用則並不具有完整性。

Availability：可用性，確保一般使用者可以正常使用服務。

例：網頁伺服器需要持續保持可用性，DDoS攻擊可能導致網站癱瘓而不具有可用性。

---

2. Hash Function (10%)

y=H(x)。

y：Hash值；H(x)：雜湊函式；x：原文；

One-wayness：擁有Hash值，難以將其還原為原文。

例：給定一函式輸出 n bit。Pr[H(x)=y]=2^(-n)。試使用暴力搜尋法。則每次隨機嘗試猜中的機率為2^(-n)。在經過m次隨機嘗試猜中原文的機率為 1 - (1-2^(-n))^m。（極低！！）

Weak collision resistance：給定一原文，難以找到不同的原文使兩種原文具有相同的Hash值。

例：給定一函式輸出 n bit。Pr[H(x)=y]=2^(-n)。試使用暴力搜尋法。則每次隨機嘗試使得Hash值重複的機率為2^(-n)。在經過m次隨機嘗試與原文的Hash值相同的機率為 1 - (1-2^(-n))^m。（極低！！）

Strong collision resistance：難以找到兩種原文有相同的Hash值。

例：給定一函式輸出 n bit。Pr[H(x)=y]=2^(-n)。試使用暴力搜尋法。則經過m次隨機嘗試使其中有兩種原文的Hash值重複的機率為 ：

1 - (1 - 1/(2^n))*(1 - 2/(2^n))*(1 - 3/(2^n))* … *(1 - (m-1)/(2^n))。（極低！！）

---

3. ElGamal Threshold Decryption (15%)

setup:

- large safe prime: ***p***
- generator: ***g***
- private key: ***a***
- public key: ***(p, g, A = g^a (mod p))***
- share private key among ***n*** parties, using Shamir's secret sharing with q ∈ {1, … , p − 1}

  —> randomly choose k − 1 coefficients a_1, … , a_{k-1}

  —>set a_0 = ***a***

  —>build polynomial $f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{(k-1)}$

—>set $i = 1, \ldots, n$ and calculate Points $s_i = (i, f(i)) \bmod q$

- now every party gets one point s_i

encryption:

- plaintext: m
- random value: x
- ciphertext1: $c_1 = g^x \pmod p$
- ciphertext2: $c_2 = m(g^a)^x \pmod p$

decryption:

- at least k parties have to compute decryption share $d_j = A^{s_i}$
- can compute m with set S of $j \in \{1, \ldots, n\}$ which returned their d_j

$$m = \left(\prod_{j \in S} d_j^{\ell_{j,0,S}} \right)^{-1} \cdot c \bmod p$$

---

4. How2Crypto (10%)

Round 1
Round 1 的加密是 a1z26 ，把 c1 輸入進去由 round1.cpp 編譯成的 1 ，程式會直接輸出解密後的明文。
Round 2
Round 2 的加密是 caesar cipher ，把 m1, c1, c2 輸入進去由 round2.cpp 編譯成的 2 ，程式會直接輸出解密後的明文。
Round 3
Round 3 的加密是 caesar cipher ，但是沒有現有的明文跟密文，因此無法直接算出key。將 c1 輸入由 round3.cpp 編譯成的3，程式會輸出所有 caesar cipher 解密後的可能明文，再用肉眼辨認哪個是正常的英文句子即可。
Round 4
Round 4 的加密是 affine cipher ，把 m1, c1, c2 輸入進去由 round4.cpp 編譯成的 4 ，程式會直接輸出解密後的明文。
Round 5
Round 5 的加密是 columnar transposition cipher，把 m1, c1, c2 輸入進去由 round5.cpp 編譯成的 5 ，程式會將解密後的所能明文放在 output.txt 。查看 output.txt 裡唯一正常的英文就是真正的明文。**但是因為 dfs 十分微妙，因此有一定的機率會解碼錯誤。**
Round 6
Round 6 的加密是 rail fence cypher ，因為已經懶得寫出 decoder 了，所以將 c2 放進 https://www.dcode.fr/rail-fence-cipher 中進行 decode ，再把最合理的解密輸入進去。
再來解密 flag，將 b64decode 後的內容輸出成檔案。在 finder 一看！是個圖片！

最後得到 FLAG！！！！ BALSN{You_are_Crypto_Expert!!!^_^}

---

5. Mersenne RSA (10%)

N 是由兩個巨大的 Mersenne prime 所組成。 N 是 341 位數，在查查 Mersenne prime 的表格之後，就知道是由 (2^521 - 1) 與 (2^607 - 1) 相乘而得。
（以下以 python 運算）

p = 2^521 - 1
q = 2^607 - 1

ø (n) = (p-1)*(q-1)

已知 e = 65537 ，因此可以求模逆元——— d 。（5_inverse.py）

d =
6053094400294447976320793659223519037789446365042906288156878044475184017252020458305874289930722003787987322053896081784688690023706520534088384630615907689760597552116667228685901745467621650239551711584236818097010586049340003820338363935591315640935201706010863329781787620091365012718740272353963795227079333552244173964044160340893473

c：ciphertext，n已提供。（5_getm.py）
pow (c,d,n) = m =
2610582713491357871820194044994779023773381319354468767608494452620902902774824365856303235678876526498380465521916285

hex (m) =（5_decode.py）
0x42414c534e7b69665f4e5f69735f666163746f72697a65645f796f755f6765745f7468655f707269766174655f6b65797d
得到 secret key，就可以解密了～～

最後得到 FLAG ！！！！　　BALSN{if_N_is_factorized_you_get_the_private_key}

---

6. OTP (10%)

在只知道密文的情況下，設加密函數為 f (x) ，則 f(x) xor f(y) = x xor y 。則再利用明文都是字母的特性，如果是正確猜中 key length 的話，那麼 xor 的出來的小寫字母都是會是小於 32 ，而大寫字母都會小於 64，加上標點符號的話，所有的值都應該小於 128 。（利用 guess_length.cpp 編譯出的 guess_length)

經過計算得知 key length 應該是 13 。以下有幾個好性質：（ observe.cpp 編譯出的 observe）
1. 所有字母（含大小寫）間互相的 xor，值會掉在 0~63 之間，結果不會是字母。
2. 如果是空白與字母 xor ，大小寫字母會互換。

由 (1), (2) 我們可以大概知道哪些地方是空白鍵了，搭配原來的 ciphertext ，我們可以嘗試還原 key 。
key 長這樣：
　　key[13]={169,109,201,15,92,226,255,144,212,123,223,119,148};
這樣就可以解密了～～～

最後得到 FLAG ！！！！　　BALSN{NeVer_U5e_0ne_7ime_PAd_7wIcE}

7. Double AES (10%)

運用兩個 n 長度的 key 兩層加密的可能空間跟運用一個 2n 長度的 key 加密的空間複雜度相差甚遠。假設有兩個 n 位元的 key，那麼如果使用 meet in the middle 技巧的話暴力搜尋只需要花 2^32 * 2 的時間，也就是 2^33。然而如果是 2n 位元的 key ，那麼需要花的時間是 2^64 ，是相差非常多的，於是這樣的兩層加密並不是有效的。

在已知一組密文與明文的情況下，可以分別試著做第一層加密與從第二層解密，觀察結果是否相同，二分搜尋第一層加密後的 key_space 是否有第二層解密的 key1 ，就可以用來解密來得到 flag。（要注意的是要把 ciphertext 先 unhexlify 。要不然他就以為是很長很長的 ciphertext，會吐出兩倍 size 的 ciphertext 給你）

得到 key0： 6298659
得到 key1： 4272711
接著解密～～～

最後得到 FLAG ！！！ BALSN{so_2DES_is_not_used_today}'

8. Time Machine (10%)

觀察網路上公布的 shattered attack 中，可以發現做出來 pdf 的前 320 byte就發生碰撞。因此將 A.pdf B.pdf 兩個的前 320 bit 拿出來就會是兩個註定會碰撞的 message。

接著查看要求的後 24 bit 是什麼，然後把兩個 message 後面加上數字直到後 24 bit 符合要求，就求出 x 跟 y 了！

終於完成這最後一題了！

最後得到 FLAG！！！！ BALSN{P0W_1s_4_w4st3_0f_t1m3_4nd_3n3rgy}

9. Future Oracle (10%)

1. 在 second() 函式中他將正確的 password 與 使用者輸入的一起回傳，這樣就暴露出 password 被 hash 之後的樣子了！而且在 third() 中檢查時要求的是給訂的 random number，如果是這樣子的話，就可以再開一個使用者來將那個 random number 輸入，讓伺服器幫我們與 password 一起 hash 。於是就有辦法或的被 hash 過後的字串。
2. 在檢查 action 的時候要求的是 plaintext 的最後一個項目，這就會變成弱點！因為這樣就可以利用 length extension attack 來追加動作 printflag 。

經過以上兩步驟就可以通過檢查得到 FLAG了！！
最後得到 FLAG！！！！ BALSN{Wh4t_1f_u_coul1d_s33_th3_futur3}

10. Digital Saving Account (15%)

首先第一層關卡是想希望用 cut and paste attack 來獲得 admin 權限。因為無法打出 & 還有 = ，這兩種字元，因此透過換行來讓 token 中出現兩次 role。要注意的是 AES ECB Mode 中行數會影響 encode。

因此利用 record.txt 裡面的 LEFT_PART 來組成上兩行，與 RIGHT_PART 組成的下兩行來增加 admin 權限。要先把得來的兩個 token 做 b64decode之後再切割，然後上下組合之後再 encode 回 token 的樣子。由此就可以通過第一道關卡。

第二道關卡是利用 DSA 中的危險之處，也就是 k 如果被重複使用則可以被用來回推出 private key。要注意的是需要先把 transaction 先做 sha1 的 hash之後再開始做 private key 的還原。從 https://github.com/tintinweb/DSAregenK fork 過來之後我寫了個 interface 來將給訂的 public key 還有 signature 來還出 private key。因此利用執行 solve.py 然後輸入 admin 模式下給的東東。最後成功造出 sig(“FLAG”) 的 r 跟 s ！
最後得到FLAG！！！！ BALSN{s3nd_m3_s0m3_b1tc01n_p13as3}