Cryptography and Network Security - Homework #1

B03902015 簡瑋德

1. CIA

Confidentiality

- Able to hide the information from people unauthorised to view it
- “Eavesdropping on messages” is an example that violates confidentiality

Integrity

- Make sure that the data is an accurate and unchanged representation of the original secure information
- “Modification of messages” is an example that violates integrity

Availability

- Ensure that authorized parties are able to access the information when needed
- “Denial of Service” is an example that violates availability

2. Hash Function

One-wayness

- Given the hash result y, it’s hard to find the origin value x such that $H(x) = y$
- Password hashing needs one-wayness property, making it hard for attackers to recover the passwords in case of data leak

Weak Collision

- Given origin value x, it’s hard to find another $x' \neq x$ such that $H(x) = H(x')$
- Checking file integrity needs weak-collision property, expecting that different file content results in different hash code

Strong Collision

- Hard to find a pair of value (x, x'), where $x \neq x'$ but $H(x) = H(x')$
- Commitment needs strong-collision property, since the committer should not be able to change his previous answer

3. ElGamal Threshold Decryption

Setup

- 完整的私鑰是b，計算$y = (g^b \mod p)^x$
- 隨機產生$(t-1)$個亂數，以r_2到r_t表示，且r_1即為y，得方程$f(x) = \sum_{i=1}^{t} r_i x^{i-1} \mod p$
- 第k個私鑰片段S_k就是$f(k)$且索引$I_k = k$，其中$k \in \{1, 2, \ldots, n\}$

Decryption

- 第k個人負責計算$d_k = c_1^{S_k} \mod p$，其中t個數值分別是d_{u_1}到d_{u_t}
- 第u_j個人還需計算$V_{u_j} = f_{u_j}(0) = \prod_{i=1}^{j-1} \frac{I_{ui}}{I_{ui} - I_{u_j}} \cdot \prod_{i=j+1}^{t} \frac{I_{ui}}{I_{ui} - I_{u_j}}$，得到$V_{u_1}$到$V_{u_t}$
- 當t個人彼此合作，即可計算出$d = \prod_{i=1}^{t} (d_{u_i})^{V_{ui}} \mod p$
- 得到d後，可進行解密，$m = c_2 d^{-1} \mod p$

4. How2Crypto

`BALSN{You_Are_Crypto_Expert!!!^_^}`

`$ python3 code4.py` (python 3.6.4) (pwntools v2.2.1) (若腳本錯誤，請多執行幾次)

Description - Piece #1

1. `c1` 兩個數字為一組， `01` 轉成字元 `a` ， `02` 轉成字元 `b` ， `26` 轉成字元 `z` ，以此類推
2. 空白轉換後仍是空白，即得 `m1`

Description - Piece #2

1. 明文的 `abc` 會被平移到 `cde` 或 `mno` 之類，作為密文
2. 透過 `m1` 與 `c1` 的關係，找出平移的距離，套用到 `c2` 上面即得解

Description - Piece #3

1. 與上一題相似，但不再提供 `m1` 和 `c1` 作為對照
2. 用一層迴圈從 `1` 跑到 `26` ，猜測平移的距離，並以字典 `word.txt` 輔助，找出最有可能的平移距離，因為明文應該是正常的英語句子，所以理論上在字典中可以找到這些單詞 (仍有機會因為猜錯而使腳本終止，需要多嘗試幾次)

Description - Piece #4

1. 與上題相似，但本題中 `m1` 和 `c1` 的平移關係變為「一次線性方程」
2. 用兩層迴圈猜測線性方程的參數，套用到 `c2` 上面即得解

Description - Piece #5

1. 明文的長度為 `t` ，密文的第 `i` 個字元即是明文的第 `(d*i+1)%t` 個字元
2. 透過 `m1` 和 `c1` 的關係，找出 `d` 的正確數值，套用到 `c2` 上面即得解

Description - Piece #6

1. 假設有一參數 `d` ，序列的一開始就 `d` 個 `1` ，例如 `1_1_1_1_1_`
2. 偶數是向左填空，奇數則是向右填空，且跳過第一個位置，例如 `1_1_12_12_1_` 和 `12_123_123_1234_13_`
3. 不斷填空，直到序列的長度與明文密文一樣長
4. 按照序列產生的順序，回去密文中拿取對應位置的字元，加到明文的尾巴
5. 舉例來說，已知 `d=5` 且密文為 `abcdefg` ，可得序列 `1_1_12_12_1_` 與，並推得明文 `abcegfd`
6. 利用 `m1` 和 `c1` 的關係，先找出 `d` 的正確數值，再套用到 `c2` 上面即得解

Description

1. 把六個片段接起來，用 `base64` 解碼成 `binary` 的形式
2. 將值寫進一張 `png` 圖檔中，打開圖片即可得解

5. Mersenne RSA

`BALSN{if_N_is_factorized_you_get_the_private_key}`

`$ python2 code5.py` (python 2.7.14)

Description

1. 去看看「Mersenne Generator」會產出甚麼樣的質數，猜出 `p` 跟 `q`
2. 用 `p` 跟 `q` 算出私鑰，對 `flag` 解密，並轉成十六進位再用「ascii」解碼

6. OTP

`BALSN{NeVer_U5e_0ne_7ime_PAd_7wIcE}`

`$ python3 code6.py` (python 3.6.4)

Description

1. 先猜測文章中可能出現的單詞，在程式中使用的即 `_that_`
2. 用一層迴圈，猜測單詞出現的位置，並用另一層迴圈猜測金鑰的長度
3. 用上述的方法，取得片段一（單詞），和另外三個片段（間距都是金鑰長度的倍數），分別和片段一、 `_that_` 的十六進位結果進行 `XOR` ，並轉回 `ascii`
4. 如果猜測正確，這三段結果應該都是正常的英文字符、標點或空格，例如 `tes fr` 、 `os on` 和 `in w`
5. 重複同樣的步驟，把片段越解越長，最後就可以得到完整的內文

7. Double AES

`BALSN{so_2DES_is_not_used_today}`

`$ python3 code7.py` (python 3.6.4) (180 seconds) (pycrypto v2.6.1)

Description

1. 先用一個迴圈，遍歷所有金鑰，記錄第二個 `AES` 解密的結果並記錄下來
2. 在用另一個迴圈，遍歷所有金鑰，確認第一個 `AES` 加密的結果是否在步驟一出現過
3. 如果找到匹配的結果，即破解了 `Double AES` 系統

8. Time Machine

`BALSN{P0W_1s_4_w4st3_0f_t1m3_4nd_3n3rgy}`

`$ python3 code8.py` (python 3.6.4) (60 seconds) (pwntools v2.2.1)

Description

1. 下載谷歌提供的兩個 `pdf` 檔案，讀取前 `320(bytes)` ，分別為 `m1` 跟 `m2`
2. 隨機產生短字串 `r` ，並試看看 `sha1(m1|r)` 的末六碼是否正確
3. 一旦找到符合條件的 `r` ， `m1|r` 和 `m2|r` 即為本題要求的字串（他們會有同樣的 `sha1` 哈希值）

9. Future Oracle

`BALSN{Wh4t_1f_u_cou1d_s33_th3_futur3}`

`$ python3 code9.py` (python 3.6.4) (pwntools v2.2.1)

- 需要使用 `./hexpand` 這個指令工具，參考 github連結 (https://github.com/amlweems/hexpand) 安裝
- 將 `hexpand` 的執行檔放置到當前目錄 `.` 底下即可

Description

1. 假裝自己是 `admin` ，先問出這一次的 `Ns` 亂碼是多少
2. 開一個分身，問出 `admin` 搭配正確的 `Ns` 的哈希值是什麼
3. 透過「哈希長度擴展攻擊」，找出追加了 `padding` 和 `||printflag` 的新哈希值
4. 由於長度擴展攻擊實現時需要知道原始資訊的長度，所以用一層迴圈去猜看原本的密碼長度是多少（每次猜測都需要等十秒）

10. Digital Saving Account

`BALSN{s3nd_m3_s0m3_b1tc01n_p13as3}`

`$ python3 code10.py` (python 3.6.4) (pwntools v2.2.1)

Description

1. 先註冊兩組帳號密碼，分別是 `xxxx/xxxxxx` 和 `xxxxxxxxxxadmin/xxxxxxxxxxx` ，並記下它們的加密結果
2. 取第一組加密結果的前十六位，和第二組加密結果的後四十八位，接起來得到新的加密結果，搭配第一組帳號與第二組密碼，即可以管理者身分登入系統
3. 登入後，拿到公鑰的 `p`, `q`, `g` 和 `y` 和兩組「可用同一個 `y` 驗證」的轉帳資訊
4. 這兩組轉帳資訊，因為驗證使用的 `y` 相同，可以回推得到 `k`
5. 參考維基，$k = (m_1 - m_2)(s_1 - s_2)^{-1} \mod q$
6. 拿到 `k` 之後，可以再進一步推回 `x` ，即可得私鑰
7. 參考維基，$x = (s_1 \cdot k - m_1) \cdot r_1^{-1} \mod q$
8. 有了私鑰，就可以對字串 `"flag"` 進行簽章