CSE 5473 — Network Security Autumn 2020

Assignment #1

Symmetric Cryptography *Due September 23th, 11:59PM*

1 Secret Key Cryptography - DES [30 pts]

1. Search the web for a DES implementation (e.g., `http://des.online-domain-tools.com/`) and use it to find the result of encrypting the plaintext `A23B716BA14C32C9` with the key `D43CB19AF490D7CE` in hexadecimal format

   Initialization Vector (IV): `00 31 0d 31 f8 df 90 df`

   (a) des-ecb: `08 83 53 71 26 38 52 45 8e 1e f7 96 c2 d3 0c 12`

   (b) des-cbc (with IV): `1b 9f 37 6d be 78 fe 0a 4c 6c 02 d8 39 9d c2 73`

   (c) des-cfb (with IV): `d9 eb 3b 13 9d 5a 5a 70 3f 0a 8a ea 26 de a6 9f`

   (d) des-ofb (with IV): `d9 3a e8 55 58 76 7d 37 be 4d 60 44 37 b2 3e e3`

   (e) des-nofb (with IV): `d9 20 00 6e 71 a1 5a 68 77 19 f0 d0 72 27 1d de`

2. The keys `0000000000000000` and `FFFFFFFFFFFFFFFF` (in hexadecimal format) are considered "weak" keys for DES (`http://en.wikipedia.org/wiki/Weak_key`). Explain why these keys are weak. (Hint: Consider the per-round keys that they generate and the effect these per-round keys have on each round of the algorithm – You might want to check your textbook or do a google search to understand what a "weak" key means.)

   When we are using DES algorithm to encrypt a message, DES is going to take the plaintext m with sixteen round Feistel network to get the ciphertext c. To be more specific, after the Initial Permutation (IP), the 64-bit message will be equally splited to two parts, L_{i-1} and R_{i-1}. Then, DES will following function to encrypt the message:

   $$L_i = R_{i-1}$$
   $$R_i = Li - 1 \oplus f(R_{i-1}, K_i)$$

   The Key k is not being used directly, instead, there is a Key Schedule which generate 16 Subkeys. The 64-bit k will go through Permutated Choice 1 ($PC1$), and output two 28-bit key C_i and D_i (every 8th bit in the key is not used). Then DES will use Permutated Choice 2 ($PC2$) to get a 48-bit key K_i (LS means Left Shift):

$$C_i = LS_i(C_i - 1)$$
$$D_i = LS_i(D_i - 1)$$
$$K_i = PC2(C_i D_i)$$

If we are using keys like `0000000000000000` and `FFFFFFFFFFFFFFFF` through $PC1$, the value of $C_i D_i$ will be all 0 or all 1 and casue $K_i = K_{15-i}$ (Subkeys are symmetric). Since DES is using reversing function to decrypt, which means we can use same k to encrypt the ciphertext c again to get the plaintext m.

3. If DES keys had been 128 bits as IBM originally proposed, how long would one million computers, each trying one trillion keys per second, take to examine all possible keys?

   DES is using 64-bit key, but only 56-bit has been used to encryption because every 8th bit in the key is not used. Once we increased the key length to 128 bits, there is only 112-bit can be used to encryption. Let's be clear, 1 million = 10^6 and 1 trillion = 10^{12}.

$$\begin{aligned}
\frac{2^{128} \text{ keys}}{10^{12} \text{ keys/second}} / 10^6 \text{ computer} &= 3.40282 \times 10^{20} \text{ second/computer} \\
&= 5.67137 \times 10^{18} \text{ minute/computer} \\
&= 9.45229 \times 10^{16} \text{ hour/computer} \\
&= 3.93845 \times 10^{15} \text{ day/computer} \\
&= 1.07903 \times 10^{13} \text{ year/computer}
\end{aligned}$$

2 Encrypting Large Messages [40 pts]

1. Suppose a sequence of plaintext blocks, x_1; x_2; ...; x_n yields the ciphertext sequence y_1; y_2; ...; y_n. Suppose that one ciphertext block, say y_i, is transmitted incorrectly (i.e., some 1's are changed to 0's and vice versa). What is the number of plaintext blocks that will be decrypted incorrectly when the following modes are used: (1) ECB, (2) CBC, (3) OFB, and (4) CFB modes are used.

   (a) ECB: 1 block
   (b) CBC: 2 blocks
   (c) OFB: 1 block
   (d) CFB: 3 blocks

2. What pseudo-random block stream is generated by 64-bit OFB with a weak DES key?

   When we are using OFB, we will use the output from pervious block stream to encrypt the next block stream. The sequence would be $E_i(IV), E_i(E_i(IV)), E_i(E_i(E_i(IV))), \cdots$. As we discussed before in section 1, week keys are symmetric, and we can get the plaintext by using the same algorithm to encrypt the ciphertext again. Therefore, the pseudo-random block stream generated by 64-bit OFB with a weak DES key would be $E_i(IV), IV, E_i(IV), IV, \ldots$

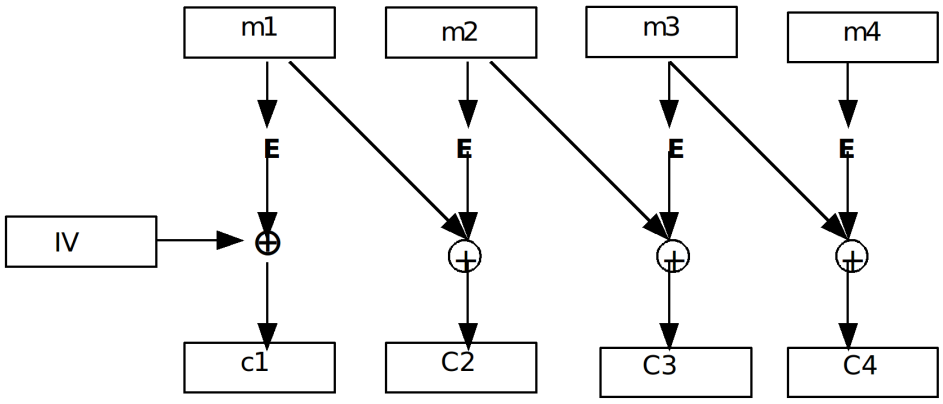
3. The pseudo-random stream of blocks generated by 64-bit OFB must eventually repeat (since at most 2^{64} different blocks can be generated). Will $k\{IV\}$ necessarily be the first block to be repeated? Explain.

   $k\{IV\}$ is necessarily be the first block to be repeated since OFB will using the sequence $E_i(IV), E_i(E_i(IV)), E_i(E_i(E_i(IV))), \cdots$ to encrypt blocks which means the first block stream is used to encrypt the next block stream (and the next block stream is used to decrypt the pervious block stream).

4. A common message integrity check is DES-CBC, which means encrypting the message in CBC mode and using the final ciphertext block as the checksum. This final block is known as the "CBC residue". This is what is used in the banking industry. Show how you can construct a message with a particular CBC residue, with the only constraint that somewhere in the message you have to be able to embed 64 bits of "garbage".

   We can leave the last block empty, and working from the last block (CBC residue) to the first block to find the bit who must be exist in order to maintain the CBC residue correct.

5. Consider the following alternative method to CBC for encrypting large messages. In this CBC variant, the encryption is done as shown in the Figure below. Is this variant reversible (can you decrypt the ciphertext generated by this scheme)? How? What are the security implications of this scheme, if any, as opposed to the "normal" CBC? (Hint: Think about passive and active attacks that are possible to carry on this scheme.)



1. Use $c_1 \oplus IV$ to get m_1.
2. Use $m_1 \oplus c_2$ to get m_2.
3. Use $m_2 \oplus c_3$ to get m_3.
4. $\cdots$
5. Use $m_i \oplus c_{i+1}$ to get m_{i+1}.

- It is very easy to find m_i if we have m_{i+1} and $c_i + 1$ which means the entire message will be compromised when only one of the block compromised.
- For passive attackers, if we know the plaintext and ciphertext, we can break the encryption easily by solving simple math problem.

3 Multiple Encryption DES [30 pts]

In answering this question, assume that an average close to 2^{32} steps to break an encryption system (find the corresponding key(s)) is tractable, while an average close to 2^{64} steps or larger is not.

1. Let's assume that you do DES double encryption by encrypting with k_1 and doing DES in decrypt mode with k_2. Does the same meet-in-the-middle attack described in the textbook work as with double encryption with k_1 and k_2? If not, how could it be made to work? (Hint: it is possible but requires some tweaks. Please describe your tweaks)

   The meet-in-the-middle attack works.

2. Devise a meet-in-the-middle attack to break EDE. That is, find the keys k_1 and k_2 assuming knowledge of a set of plaintext/ciphertext pairs $<m_i, c_i>$obtained by applying EDE on each p_i. (Hint: you meet either after ED, or before DE)

   (a) Generate all possible keys $k_{1...56}$, there are 2^{56} values.
   (b) Use the k_i to encrypt the first block of the plaintext m_1.
   (c) Use the k_j to decrypt the first block of the ciphertext c_1.
   (d) If there is a key k when $Enc(k, m_1) = Dec(k, c_1)$, store it for further verification.
   (e) Test all values from pervious step, find the only value who meets $Enc(m_i) = Dec(c_i)$.

3. Assume that EDE was deployed using three keys k_1 (for step 1 encryption), k_2 (for step 2 decryption), and k_3 (for step 3 encryption). Is this new version of EDE more secure than the traditional EDE? Explain.

   (a) Assume $k_1 \neq k_2 \neq k_3$.
   (b) Generate all possible keys $k_{1...56}$, there are 2^{56} values.

(c) Use the k_x to decrypt the first block of the plaintext m_1.

(d) Use the k_y to decrypt the first block of the ciphertext c_1.

(e) If $Dec(k_x, m_1) = Dec(k_y, c_1)$, store k_x, k_y.

(f) Use the k_z to encrypt $Dec(k_x, m_1) \mid Dec(k_y, c_1)$, store all of them.

(g) Test all values from pervious step, find the plaintext.

EDE has same security as the tridational EDE, since they have same complexity $O\left(n^{112}\right)$.

For tridational EDE, we need to use $O\left(n^{56}\right)$ to solve the outer layer of encryption, and use $O\left(n^{112}\right)$ to get the plaintext.

The new version of EDE has same security with tridational, since we need to take $O\left(2 \times n^{56}\right)$ to solve the outer layers, and use $O\left(n^{112}\right)$ to get the plaintext from middle layer with the outer layer.

4. Consider the following approach to increase the security of DES. Encryption is done by encrypting the plaintext twice using the same shared secret. Decryption is done by decrypting the ciphertext twice using the same shared secret. Is this approach more secure than the traditional DES?

Encrypt twice with same shared secret will not secure than the traditional DES because we have demonstrated the meet-in-the-middle attack in question 3.2. We can use the same key k_i to encrypt the plaintext and decrypt the ciphertext then get the value of k_i which takes same complexity.

4 Breaking Monoaphabetic Cipher Using Frequence Analysis [30 points Bonus. There will be partial credit if you only partially solve it.]

A monoalphabetic cipher is one where each symbol in the input (called the "plaintext") is mapped to a fixed symbol in the output (called the "ciphertext"). Let's go back to the age where we use statistic analysis (or frequence analysis) to break ciphers (esentially the keys). It will be a fun assignment, and you can use modern techniques to break old ciphers.

While you can entirely finish this assignment by using pencil and paper, it is more efficient to use (existing) programs. Please **Find**[1] or **write** a program[2] that reads an input ciphertext, and outputs the (sorted) letter frequencies, diagram (consecutive pairs of letters) frequencies, and trigram (consecutive triplets of letters) frequencies.

[1] For instance, `http://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html`, which I recommend especially if you don't wanna write code.

[2] Please note that you don't have to write your program in our VM. You can use whatever computing environment convenient to you, e.g., Windows, Linux, Mac, etc., using Java, C, Python, etc.

Using this frequence analysis program, decode the following ciphertext. The original plaintext that produced this ciphertext contains only upper case letters; all punctuation and spaces were removed before encoding. It was encrypted with a mono-alphabetic cipher. Explain how you decoded the ciphertext in your answer.

```
WMTLXJAVQNXBTECJIZMWDMEIGXXVTSXRAWHXHTNHVMTVW
KVMXBTECQGXUCXSWXNDXWYNQZQGNQZTIQGPAHRWNRQGVM
XGAVASXJAVJIVMXCTKVTNHPAHRWNRJIVMXCTKVWZWKMVQ
YNQZZMTVVMXSXMTKJXXNWNVMXDQNHADVQGVMXJSWVWKME
WNWKVSIGQSVMXBTKVVZQIXTSKVQPAKVWGIVMQKXMQCXKZ
WVMZMWDMRXNVBXEXNMTLXJXXNCBXTKXHVQKQBTDXVMXEK
XBLXKTNHVMXMQAKXWKWVVMTVWNKWHWQAKKEWBXZWVMZMW
DMQASCXVWVWQNMTKJXXNBTVXBISXDXWLXH
```

```
IHAVEBUTONELAMPBYWHICHMYFEETAREGUIDEDANDTHATI
STHELAMPOFEXPERIENCEIKNOWOFNOWAYOFJUDGINGOFTH
EFUTUREBUTBYTHEPASTANDJUDGINGBYTHEPASTIWISHTO
KNOWWHATTHEREHASBEENINTHECONDUCTOFTHEBRITISHM
INISTRYFORTHELASTTWOYEARSTOJUSTIFYTHOSEHOPESW
ITHWHICHGENTLEMENHAVEBEENPLEASEDTOSOLACETHEMS
ELVESANDTHEHOUSEISITTHATINSIDIOUSSMILEWITHWHI
CHOURPETITIONHASBEENLATELYRECEIVED
```

```
str = """
WMTLXJAVQNXBTECJIZMWDMEIGXXVTSXRAWHXHTNHVMTVW
KVMXBTECQGXUCXSWXNDXWYNQZQGNQZTIQGPAHRWNRQGVM
XGAVASXJAVJIVMXCTKVTNHPAHRWNRJIVMXCTKVWZWKMVQ
YNQZZMTVVMXSXMTKJXXNWNVMXDQNHADVQGVMXJSWVWKME
WNWKVSIGQSVMXBTKVVZQIXTSKVQPAKVWGIVMQKXMQCXKZ
WVMZMWDMRXNVBXEXNMTLXJXXNCBXTKXHVQKQBTDXVMXEK
XBLXKTNHVMXMQAKXWKWVVMTVWNKWHWQAKKEWBXZWVMZMW
DMQASCXVWVWQNMTKJXXNBTVXBISXDXWLXH
""".replace("\n", "")

# find most common character
def check_freq(x):
   freq = {}
   for c in x:
      freq[c] = str.count(c)
   return freq


check_freq(str)

# abcdefghijklmnopqrstuvwxyz
# TJDHXGRMWPYBENQCFSKVALZUIO

# found 47 "X", replace as "e"
text = str.replace("X", "e")

# try to find most common double "e" to identify the word
```

```
# found mutiple "JeeN", suspect can be replaced by "been"
text = text.replace("J", "b").replace("N", "n")

# found mutiple "MTK" before "been"
text = text.replace("M", "h").replace("T", "a").replace("K", "s")

# found mutiple "haLe", only "L" did not be replaced, replace it as "v"
text = text.replace("L", "v")

# found "W", "have", replace "W" as "I"
text = text.replace("W", "i")

# found mutiple "Vhe"
text = text.replace("V", "t")

# found "bAt", replace "A" as "u"
text = text.replace("A", "u")

# found "Qne", replace "Q" as "o"
text = text.replace("Q", "o")

# found "DonHuDt", replace "D" as "c", "H" as "d"
text = text.replace("D", "c").replace("H", "d")

# found mutiple "oG", replace "G" as "f"
text = text.replace("G", "f")

# found "futuSe", replace "S" as "r"
text = text.replace("S", "r")

# found "bI", replace "I" as "y"
text = text.replace("I", "y")

# found "Zhich", replace "Z" as "w"
text = text.replace("Z", "w")

# found "Ey feet", replace "E" as "m"
text = text.replace("E", "m")

# found "wishtoYnow", replace "Y" as "k"
text = text.replace("Y", "k")

# found "BamC", replace "B" as "l", "C" as "p"
text = text.replace("B", "l").replace("C", "p")

# found "Ruided", replace "R" as "g"
text = text.replace("R", "g")

# found "Pudging", replace "P" as "j"
text = text.replace("P", "j")

# found "eUperience", replace "U" as "x"
text = text.replace("U", "x")

print(text)
```

```
# split by knowing words
rep = [
     "have",
     "what",
     "feet",
     "been",
     "house",
     "which",
     "with",
     "wish",
     "and",
     "know",
     "conduct",
     "but",
     "one",
     "for",
     "last",
     "to",
     "of",
     "years",
     "there",
     "them",
     "way",
     "those",
     "are",
     "that",
     "received",
     "insidious",
     "british",
     "by",
     ]

for item in rep:
   text = text.replace(item, " {} ".format(item))

" ".join(text.split())
```

Reference http://en.wikipedia.org/wiki/Frequency_analysis

5 Submiting your report

Please write a report describing how you solve each of the problem above. You can use WORD, or Latex to write the report. The Latex source of this assignement is available at https://www.overleaf.com/read/xfwppdqjbzbt. Regarding how to use Latex,

please see this tutorial if you have 0-experience with it `https://www.andy-roberts.net/writing/latex/absolute_beginners`. Please note that latex can be installed in all modern OSes such as Windows, Linux, and Mac.