Homework 4: Chapter 2 Questions

Mudit Vats
mpvats@syr.edu
5/5/2020

Table of Contents

Overview

This lab report presents observations and explanations for the exercises described in Professor Weissman's HW4 assignment.

Problem 2.1

The stream cipher described in Definition 2.1.1 can easily be generalized to work in alphabets other than the binary one. For manual encryption, an especially useful one is a stream cipher that operates on letters.

Answer:

1. Develop a scheme which operates with the letters A, B,. . ., Z, represented by the numbers 0,1,. . .,25. What does the key (stream) look like? What are the encryption and decryption functions?
    a. Encryption Function: esi(xi) = xi + si mod 26
    b. Decryption Function: dsi(yi) = yi - si mod 26

The key is added to the plain text during encryption then mod 26 to fit into the set of keys. During decryption, the key is subtracted from the cipher text and then mod 26 per the set of keys (letters 0..25).

2. Decrypt the following
    a. cipher text:
        i. bsaspp kkuosp
    b. which was encrypted using the key:
        i. rsidpy dkawoa
    c. Decode (using Number/Letter table below)

```
 i.  b - r mod 26,  1 - 17 mod 26 = -16 = 10 => K
ii.  s - s mod 26, 18 - 18 mod 26 = 0        => A
iii. a - i mod 26,  0 -  8 mod 26 = -8 = 18  => S
iv.  s - d mod 26, 18 -  3 mod 26 = 15       => P
v.   p - p mod 26, 15 - 15 mod 26 = 0        => A
vi.  p - y mod 26, 15 - 24 mod 26 = -9 = 17  => R
vii. k - d mod 26, 10 -  3 mod 26 = 7        => H
viii.k - k mod 26, 10 - 10 mod 26 = 0        => A
ix.  u - a mod 26, 20 -  0 mod 26 = 20       => U
x.   o - w mod 26, 14 - 22 mod 26 = -8 = 18  => S
xi.  s - o mod 26, 18 - 14 mod 26 = 4        => E
xii. p - a mod 26, 15 -  0 mod 26 = 15       => P
```

    d. Final Decryption: KASPAR HAUSEP

Number	Letter
0	A
1	B
2	C
3	D
4	E

5	F
6	G
7	H
8	I
9	J
10	K
11	L
12	M
13	N
14	O
15	P
16	Q
17	R
18	S
19	T
20	U
21	V
22	W
23	X
24	Y
25	Z

3. How was the young man murdered?

The link to "KASPAR HAUSER" is below. This is what shows up when searching for "KASPAR HAUSEP". The decoded solution was "KASPAR HAUSEP", however.

There is an errata (http://www.crypto-textbook.com/ Book→Errata List) in the book which states "The last letter of the cipher text should be a "r", not a "p"". With that change, we have: r - a mod 26, 17 - 0 mod 26 = 17 => R, which then produces the correct solution "KASPER HAUSER".

https://en.wikipedia.org/wiki/Kaspar_Hauser

Death by stabbing!

Problem 2.2

Assume we store a one-time key on a CD-ROM with a capacity of 1 Gbyte. Discuss the real-life implications of a One-Time-Pad (OTP) system. Address issues such as life cycle of the key, storage of the key during the life cycle/after the life cycle, key distribution, generation of the key, etc.

Answer:

1. Sharing the OTP key is difficult. This would require sending the CD-ROM to the person needing to decrypt the text. Of course, this can be done via courier or

personally giving it them, but it's time consuming (i.e. not very real-time) and cumbersome (since reading from a CD-ROM can be slow, possibly error prone and just a huge chunk of data to handle). Also, the carrier would have to be trusted such that it doesn't get copied or tampered with along the way (MITM attack).
2. The key size would limit the data size, so if the key is 1 GB, the max amount of data that can be streamed (securely) would be up to 1 GB. This may be ok for some applications, but limiting for others. Either way, not very flexible.
3. The key bits can only be used once so if the sender wishes to send data beyond the size of the key size, they would need to also send a newly generated key. That would result in #1 having to happen. Which, again, can be cumbersome, error prone and time consuming.
4. Generating the key would require a true random number generator (TRNG) which would require special hardware or devices that are fit for such a purpose. PCs do not have TRNG (per lecture notes/book), therefore generating the key may not be that easy or convenient.

These issues make OTP an interesting goal, but impractical to use.

Problem 2.3

Assume an OTP-like encryption with a short key of 128 bit. This key is then being used periodically to encrypt large volumes of data. Describe how an attack works that breaks this scheme.

Answer:

If the key is reused, then we are not OTP. Of course, the question says "OTP-like" and key bits are reused. When we do this, the attacker can discover patterns.

If we consider the case where the attacker knows the file type of data, they can use the header info to unravel the key bits being used since he knows the plain text which corresponds to the cipher text of the header.

128 bits is only 16 bytes, which is very "reasonable" for metadata about a file to precede the actual data of a file. This metadata would be the plain text and the cipher text is the file itself. By knowing both and knowing the algorithm, OTP-like, the attacker can discover the key bits. Once the 16 byte key is discovered, it can then be reused to decrypt the rest of the volume data.

Key bits that repeat are not secure and by repeating the 128 bit key, we have a pattern, which, once the pattern is discovered, can compromise (i.e. reveal) the entire data set.

Problem 2.4

At first glance it seems as though an exhaustive key search is possible against an OTP system. Given is a short message, let's say 5 ASCII characters represented by 40 bit, which was encrypted using a 40-bit OTP. Explain exactly why an exhaustive key search

will not succeed even though sufficient computational resources are available. This is a paradox since we know that the OTP is unconditionally secure. That is, explain why a brute-force attack does not work.

Note: You have to resolve the paradox! That means answers such as "The OTP is unconditionally secure and therefore a brute-force attack does not work" are not valid.

Answer:

1. In the OTP definition, it states that "the key stream s0, s1, s2, … is generated by a true random number generator". This means that each bit of the key stream is independent of the next bit. There is no pattern or predictability which can be relied upon to discover the next bits of the key. So, if an attacker used the file header info attack by guessing the plaintext, unraveling any part of the first set of bits of the key bits will not reveal any other parts of the key bits.
2. The OTP definition also states that "every key stream bit si is only used once". This has the implication that, which supports the previous point as well, that there is no pattern for the 40-bit OTP and any of the bits beyond the 40-bits (if this were considered). Once again, there is no pattern within the key stream which can be used to help an attacker unravel the key.
3. A True Random Number Generator is "characterized by the fact that their output cannot be reproduced". Once again, zero predictability. Since the OTP is based on used a TRNG, by definition, the key is truly random.

The short answer is that there is no pattern within the OTP key stream which can be relied upon which the attack can use to help assist them to unravel the cipher text even if some part of the cipher text, like the header of a file, is predicted.

Problem 2.5

Using 2.6 LFSR below, answer the questions.

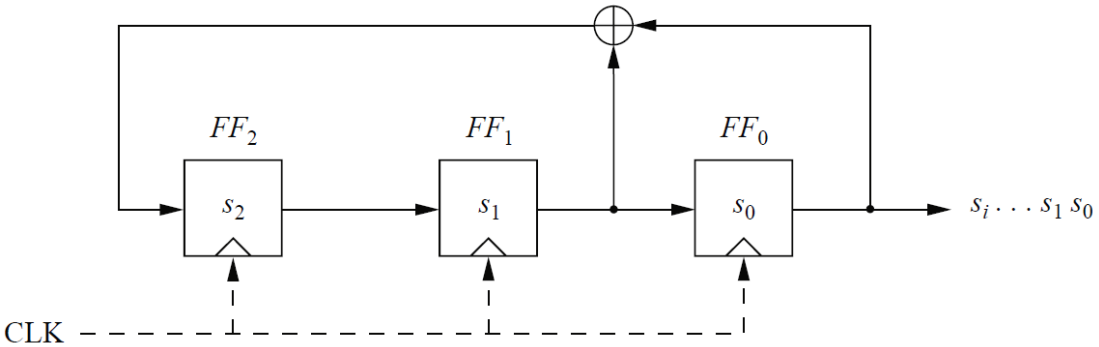

Fig. 2.6 Linear feedback shift register of degree 3 with initial values s_2, s_1, s_0

Answer:

1. What is the sequence generated from the initialization vector (s2=1, s1=0, s0=0)?

CLK	FF2=S2	FF1=S1	FF0=S0
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0
9	1	0	1
10	1	1	0

Maximum sequence length is 2^3 – 1 => 7. So after the 6th iteration above, the sequence repeats.

2. What is the sequence generated from the initialization vector (s2=0, s1=1, s0=1)?

CLK	FF2=S2	FF1=S1	FF0=S0
0	0	1	1
1	0	0	1
2	1	0	0
3	0	1	0
4	1	0	1
5	1	1	0
6	1	1	1
7	0	1	1
8	0	0	1
9	1	0	0
10	0	1	0

Maximum sequence length is 2^3 – 1 => 7. So after the 6th iteration above, the sequence repeats.

3. How are the two sequences related?

Sequence 2 starts at the CLK 5 of Sequence 1. Then they follow the same sequence. They simply differ on the initialization vector. But the follow-on pattern is the same.

Problem 2.6

Assume we have a stream cipher whose period is quite short. We happen to know that the period is 150–200 bit in length. We assume that we do not know anything else about the internals of the stream cipher. In particular, we should not assume that it is a simple LFSR. For simplicity, assume that English text in ASCII format is being encrypted.

Describe in detail how such a cipher can be attacked. Specify exactly what Oscar has to know in terms of plaintext/ciphertext, and how he can decrypt all ciphertext.

Answer:

The tricky part of this problem is that we don't know the internals of the stream cipher. We do not know if it's a simple single XOR based stream cipher or something more complex feedback-oriented implementation. What we do know, however, is that the period of the stream is 150-200 bits. This is important since it implies that the key will repeat every 150-200 bits.

The idea is that if we have some known plain text, we can XOR that with the ciphertext to reveal the key. When considering stream ciphers, the last step of the key stream bit and steam bit is to XOR them together. So, if we XOR the cipher text with the known plain text, we can recover the key bit. We can repeat this for characters we know.

Further, we can use the segment of "revealed key" to decrypted other parts of the cipher text. Where? Well, we can brute force and attempt to decrpt chunk after chunk shifting one bit. If a decryption is *potentially* successful, we'll know if we get ASCII characters. If the ASCII characters make sense, we'll have revealed more of the cipher text and by nature of XOR'ing the ciphertext again, we can potentially reveal more of the key.

For 150 – 200 bit key length, we would have ~21-25 characters assuming a 7-bit representation of each ASCII character. Even if we have a few characters of known plaintext we would reveal 10-15% of the key. Repeating decryption attempts an "reverse XOR'ing" would be simple to repeat throughout the stream.

Ultimately, with the key period repeating and revealing more of the key as more ciphertext is decrypted with known parts of the key, the entire key will be recovered, which can be used to decrypt the entire ciphertext.