# SECURITY

Assignment 4, Friday, October 9, 2017

S1013793 Carlo Jessurun

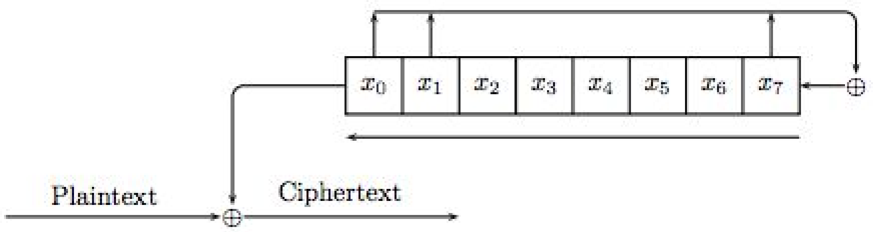
S1013792 Tony Lopar

Radboud University

Teaching assistant: Joost Rijneveld

# Assignment 1

(30 points) Consider the following simple Linear Feedback Shift Register (LFSR). The plaintext is bitwise XOR-ed with the output bits of the LFSR which first computes $x_0 \oplus x_1 \oplus x_7$ and then shifts such that $x_0$ falls out.



**Example:**

| | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| The initial state | **0** | **1** | 1 | 1 | 0 | 1 | 0 | **1** |
| is followed by | **1** | **1** | 1 | 0 | 1 | 0 | 1 | **0** |
| and outputs | **0** | | | | | | | |

A. Describe the next five states of the LFSR if it is initialized according to the box above. The first successor state is already given as illustration, so you have to give the four subsequent ones.

B. Also do a "rollback" and compute the previous four states, starting from the initial state.

C. Assume you know that the LFSR is in the initial state given above. After four shifts you intercept 0110 as resulting ciphertext. Reconstruct the 4 bits of plaintext that were encrypted to this ciphertext.

# Solutions 1

A.

| Next five states | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **S** | $x^0$ | $x^1$ | $x^2$ | $x^3$ | $x^4$ | $x^5$ | $x^6$ | $x^7$ |
| **INI** | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| **F** | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| **1** | 1 | 1 | 0 | 1 | 0 | 1 | 0 | **0** |
| **2** | 1 | 0 | 1 | 0 | 1 | 0 | **0** | **0** |
| **3** | 0 | 1 | 0 | 1 | 0 | **0** | **0** | **1** |
| **4** | 1 | 0 | 1 | 0 | **0** | **0** | **1** | **0** |

We've color coded the state in red. Added values are marked as bold. In case it wasn't clear yet. INI stands for initial state, F for follows and the numbers reflect the next four states.

B.

| Previous four states | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **S** | $x^0$ | $x^1$ | $x^2$ | $x^3$ | $x^4$ | $x^5$ | $x^6$ | $x^7$ |
| **INI** | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| **1** | **0** | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| **2** | **0** | **0** | 0 | 1 | 1 | 1 | 0 | 1 |
| **3** | **1** | **0** | **0** | 0 | 1 | 1 | 1 | 0 |
| **4** | **1** | **1** | **0** | **0** | 0 | 1 | 1 | 1 |

C.

1. The 0 comes from a XOR with zero so the plaintext bit is **0**
2. The 1 from the XOR with 1, so plaintext **0**
3. The 1 from the XOR with 1, so plaintext **0**
4. The 0 from XOR with 1, so plaintext was **1**
5. So the plain bits where **0001**

# Assignment 2

(30 points) Alice wants to send Bob a confidential message. In the arrow protocol notation introduced in the lecture, this would be expressed as $A \rightarrow B : K_{AB}\{m\}$ (i.e. using a block cipher). However, this assumes that Alice and Bob already share a (symmetric) key $K_{AB}$. This is usually not the case for the pair of any two random people.
Instead, we assume that a trusted third party, Trudy, shares keys with everyone (i.e. a large group of users can rely on the same Trudy). Using these keys, Alice can send m to Trudy and ask her to confidentially forward it to Bob.
For this exercise, we assume a passive attacker: Eve can only eavesdrop messages, but not insert, delete or modify them.

A. Write the above description of the two-step protocol down in arrow notation.

The crucial downside of the protocol described in (a) is the fact that Trudy learns the content of every message. Ideally, Alice and Bob would set up a shared symmetric key $K_{AB}$ without trusting a third party. Consider the following protocol. Note that $K_A$ is a key only Alice knows.

Step 1. Alice picks 1 million different key candidates $K^i_{AB}$ and 20-bit puzzle keys $K^i_{puz}$ (i.e. i = 0, 1, 2, . . .).
Step 2. Alice computes identifiers $ID_i = K_A\{i\}$ and cipher texts $c_i = K^i_{puz}\{ID_i , K^i_{AB}\}$.
Step 3. Alice sends all cipher texts $c_i$ to Bob, in a random order.
Step 4. Bob picks a random i and decrypts $c_i$ by trying all possible puzzle keys.
Step 5. Upon success, Bob sends Alice $ID_i$ .
Step 6. Alice uses $K_A$ to decrypt $ID_i = K_A\{i\}$ to find i.
Step 7. Alice and Bob now use $K^i_{AB}$ as their shared key.

(b) How many encryptions does Alice have to perform?
(c) How many keys (and thus: decryptions) does Bob have to try to find $K^i_{AB}$?
(d) Describe what Eve would have to do to find the key $K_{AB}$. How many encryption or decryption operations does this include?

Say that Alice picks n key candidates, and uses puzzle keys of log (n) bits.
(e) Express the number of operations Eve has to perform (and thus: the security of the scheme) in terms of n.

*The solution to the 'key exchange problem' described in this exercise dates back to a paper written in 1974 by Ralph Merkle, and is often referred to as Merkle's Puzzles1 . Later in this course, we will revisit this problem by looking a much more secure solution posed by Diffie and Hellman2 , only two years later.*

# Solutions 2

A. Let Trudy be T
   Let {m} be the message for Bob
   A → T : $K_{AT}${m}
   T → B : $K_{AB}${m}
   I'm also assuming here that the connection between Alice and Trudy is protected by a key as well since the text clearly states that "*we assume that a trusted third party, Trudy, shares keys with everyone*"
   A → T: key request for A to B
   T → A: $K_B\{K_{AB}\}$ and ticket $K_A\{K_{AB}\}$
   A → B: ticket secured with $K_{AB}$

B. Not sure about this one.
   1million key candidates
   20bit puzzle keys
   The she 'computes' the identifiers (i don't think this requires encryption?)
   Then she 'computes' the cipher texts so i'd say that requires encryption?
   We think about 2 million encryptions?
   Or is every key candidate covered as a bit so 1000000(id)/20bit=50000 encryptions?

C. So if alice then has to do 1 million encryptions then maybe bob has to do 50000 decryptions?

Bob tries to decrypt a random cipher with all possible puzzles. A puzzle is 20 bit which gives $2^{20}$ possible solutions. Since the shared key is in the puzzle Bob will find the key as soon as he succeeds to decrypt the puzzle.

D. Eve can do a man-in-the-middle attack. For this action Eve should be able to intercept communication from Alice or Bob. Alice will think that she's talking to Bob and Bob will think that he's talking to Alice. First, Eve will redirect all $c_i$ to Bob. When Bob sends $ID_i$ back to Alice, Eve can intercept this and decrypt a cipher with all possible puzzles. When she cracks a puzzle for a certain i which has the same $ID_{i,}$ then she also has the key. Since Eve doesn't know i, she will have to try all puzzles for all 1 million possible i. Eve should have to try 1 million times $2^{20}$ decryptions in the worst scenario. A major disadvantage of this action is that Eve cannot take initiative and should wait for A or B to take initiative.

E. First of all, Eve needs to intercept all ciphertexts which is already one operation. When Eve gets $ID_i$ from Bob, she should perform n * $2^{\log(n)}$ operations, because for every ciphertext she wants to know whether decrypting it results in $ID_i$.

# Assignment 3

(40 points) Recall the DES block cipher, which has a block length of 64 bits and operates with 56-bit keys. In the lecture, you learned that by the end of the 1980's, the key length of DES was considered too short, and DES was eschewed in favor of triple-DES. In this exercise, you will learn the real reason why so-called "double-DES" was omitted. In consistency with the lectures' notation, we define the three functions as follows:

$$\mathrm{DES} = \left( \cdot \xrightarrow[\text{Encrypt}]{K} \cdot \right).$$
$$\mathrm{2DES} = \left( \cdot \xrightarrow[\text{Encrypt}]{K_1} \cdot \xrightarrow[\text{Decrypt}]{K_2} \cdot \right).$$
$$\mathrm{3DES} = \left( \cdot \xrightarrow[\text{Encrypt}]{K_1} \cdot \xrightarrow[\text{Decrypt}]{K_2} \cdot \xrightarrow[\text{Encrypt}]{K_3} \cdot \right).$$

In this exercise, you are the adversary.

A. Suppose that you are given a single plaintext-ciphertext tuple (P, C) of $\mathrm{DES}_k$ for secret key K. Explain the exhaustive key search algorithm on DES to recover key K.
B. Suppose that you are given a single plaintext-ciphertext tuple (P, C) of 2DES $K_1$,$K_2$ for secret key ($K_1$, $K_2$). How many evaluations of 2DES would exhaustive key search take? How many *unique* evaluations of the Encrypt functionality of DES would this attack take? Explain the difference.
C. Explain how you can recover the entire key ($K_1$, $K_2$) in $2{\cdot}2^{56}$ evaluations of DES. Hint: only perform encryptions with DES, and no decryptions. Feel free to describe the steps procedurally, or even in pseudo-code.
D. Explain how the attack extends to 3DES to recover the entire 192-bit key in $2{\cdot}2^{112}$ + $2^{56}$ evaluations of DES.

# Solutions 3

A. The algorithm for retrieval of the key should try all the possible keys. Since the key is 56 bit long, we have $2^{56}$ options. By encrypting the plaintext with every possible key, we can check whether the output equals the paired ciphertext. If this is the case, then we have found the key that has been used.

B. We already know that we have to try $2^{56}$ options for single DES. For 2DES with different keys, we would not only have $2^{56}$ options for encrypting the plaintext, but also $2^{56}$ to decrypt with $k_2$. This means we have to do $2^{56}$ x $2^{56}$ evaluations to try all possibilities. The evaluation of the encrypt function would take the same amount of evaluations as in DES.

C. We may encrypt the plaintext and ciphertext we know from the pair with all possible keys and store all the outputs of this encryptions. When we find a key with a result that has already been discovered in the search to the other key, then we have found the encrypted version of the plaintext with $k_1$ and of the ciphertext with $k_2$. This means we have found the cipher intersecting in the two iterations of DES and in this way also the keys.
Procedurally this process could be:
   1. We try all the possible keys to encrypt the known plaintext with a single DES and store the combination of the key and output.
   2. We try to encrypt the known ciphertext with all possible keys with a single DES and store these combinations also.
   3. After these two key searches we will perform an intersect on the output of both searches. This should only be one result with a key from both encryptions. The two keys that which resulted this output are the used keys. The difference between $k_1$ and $k_2$ can be found by looking in which of the two iterations they resulted the intersecting output.

D. The main difference compared with 2DES is that the ciphertext has been encrypted with a third key. We may do an encryption of the known plaintext with all the possible keys and store all the output with the keys en decrypt this output with all possible keys. Now we have stored all possible outputs for the first two processes. These processes had a complexity of $2^{112}$. Since we know the ciphertext we can decrypt this with all possible keys and compare it to the output of the previous process. This encryption process has a complexity of $2^{56}$. After this we search for the intersection between the two stored sets with all the outputs and get the associated keys that generated this outputs.