INGI2347 : Exercises

Lab session 2 *

Xiao Chen, Marco Canini

February 29, 2016

Exercise 1: Exhaustive Search for Symmetric Keys

Knowing that the specialized machine "DES-Cracker " takes on an average 4.5 days to find a 56-bit DES key by exhaustive search, how long would it take to find a 40-bit key? A Triple-DES 112-bit key? An AES 256-bit key? Here, we assume that this machine needs the same time to encrypt a block of data with DES, Triple-DES and AES.

Exercise 2: Symmetric and Asymmetric Encryption

A group of n people wish to use a cryptographic system to exchange confidential information pairwise. The information exchanged between two group members must not be readable by any other member.

The group decides to use a symmetric encryption system.

1. What is the minimum number of symmetrical keys required?

2. Name a well known symmetric encryption algorithm.

The group then decides to replace this system by an asymmetric one.

3. What is the minimum number of pairs of asymmetric keys required so that each member can send and receive encrypted and/or signed information?

4. Bob wishes to send encrypted and signed information to Alice (Bob and Alice both belong to the group). Which key(s) must Bob use?

5. Name a well known asymmetric encryption algorithm.

The group finally decides to use a hybrid system for encryption (i.e., one that uses asymmetric cryptography to establish a session key and then switches to symmetric cryptography).

6. Give a few reasons that motivated the group to use such a system.

Exercise 3: Loss of a Private Key

Alain Terrieur, who often uses his company's secure mail server, has just lost his private key, but still has the corresponding public key.

*A part of these exercises comes from the book "Computer System Security". The reproduction and distribution of these exercises or a part of them are thus forbidden.

1. Is he still able to send encrypted mails? What about receiving?

2. Is he still able to sign the mails he sends? What about verifying the signatures of mails he receives?

3. What must he do to again be capable of carrying out all the operations mentioned above?

Exercise 4: Kerckhoffs' Principles

In 1883, Auguste Kerckhoffs established a fundamental cryptography principle: "A cryptographic system must not require secrecy, and should not create problems should it fall into the enemy's hands". Explain this principle

Exercise 5: Symmetric Encryption Modes

Symmetric encryption algorithms, called "block ciphers ", such as, Triple- DES, IDEA or AES, can be seen as black boxes taking as input a data block of fixed size (usually 64 or 128 bits) as well as a key and returning as output an encrypted data block of same size as the input size. The good performance (with regards to speed) of these algorithms explains that they are used to encrypt large volumes of data. There are several ways of using these algorithms called "encryption modes". In this exercise, we will look into the security of the ECB (*Electronic Code Book*) and CBC (*Cipher- Block Chaining*) modes.

1. The ECB mode, shown in Figure 1 along with AES, simply consists in cutting up the data to be encrypted into n blocks $X_1, X_2, \ldots, X_n$ with block length of the encryption algorithm used (128 bits for AES) and to encrypt each block using the same key K, thus obtaining the encrypted blocks $Y_1, Y_2, \ldots, Y_n$. In certain applications, the ECB mode allows recovering information on the data. Describe a scenario that illustrates this weakness.

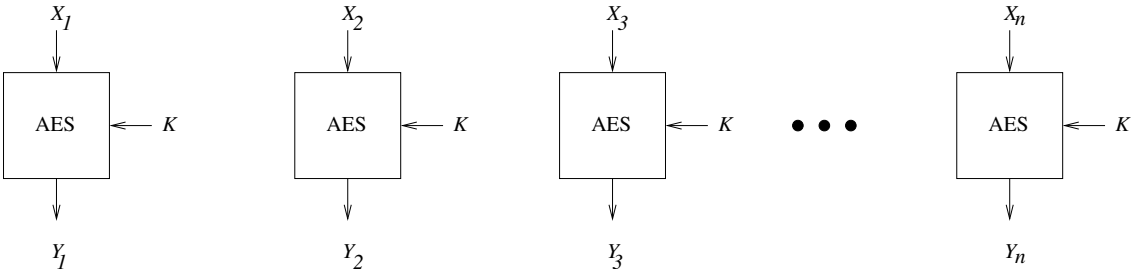

Figure 1: ECB encryption mode

2. The CBC mode, illustrated in the Figure 2, is slightly more complex than the ECB mode. It works in the following way: firstly, we randomly generate an initialization vector IV that will be transmitted in the clear to allow deciphering, we combine it with the first block of the clear text X_1 using exclusive-OR (denoted as XOR or $\oplus$)[1], then one encrypts the result using the key K, to obtain Y_1. Then, Y_1 is combined with X_2 using $\oplus$, the result is encrypted, then the process is repeated. We will assume that we encrypt the data using Triple-DES (that encrypts 64 bits blocks) in CBC mode and a pirate finds two blocks Y_i and Y_j such as $Y_i = Y_j$ and $i \neq j$. What information regarding the clear text can we deduce from this relation?

3. Supposing that we encrypt a hard disk's data using Triple-DES in CBC mode, what must

[1]XOR is defined as: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ and $1 \oplus 1 = 0$.

be the disk's size for the collision probability to be higher than 40%? (The birthday paradox indicates that the probability $p(n)$ to find, among n elements of S, two identical elements can be approximated by $p(n) \approx 1 - e^{-\frac{n^2}{2 \cdot S}}$)

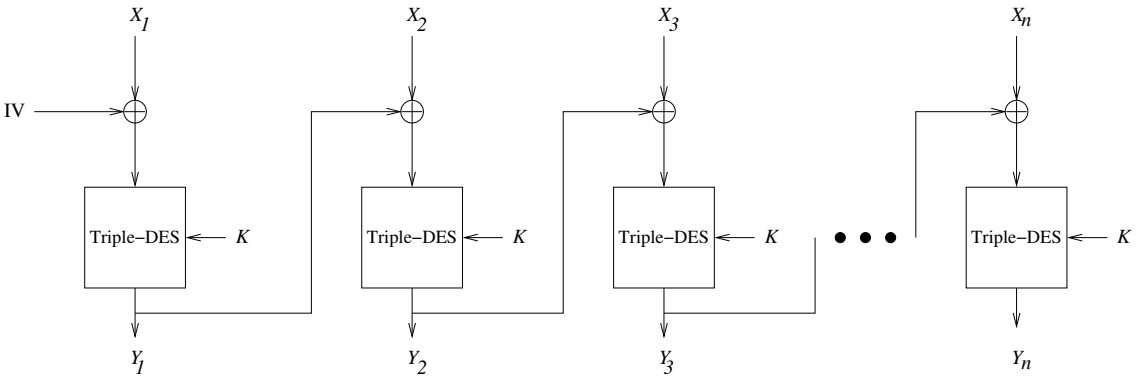

Figure 2: CBC encryption mode

Exercise 6: Simple Encryption with OpenSSL

Below we give a simple tutorial of how to use private and public key cryptography. You can refer to `http://tombuntu.com/index.php/2007/12/12/simple-file-encryption-with-openssl/` for additional information.
There is plenty of powerful encryption software, but what can you use if you just want to secure a couple files quickly? The OpenSSL toolkit works well for this. It comes installed with many OS distributions.

First, let's create a file with some secret text:
`echo ‘‘INGI2347’’ > secrets.txt`
Now, we start by using symmetric key cryptography to encrypt and then decrypt a file using a shared key. This is the basic command to encrypt a file:

```
openssl aes-256-cbc -a -salt -in secrets.txt -out secrets.txt.enc
```

How does this work?

* *openssl* is the command for the OpenSSL toolkit.
* *aes-256-cbc* is the encryption cipher to be used. (256-bit AES is what the United States government uses to encrypt information at the Top Secret level.)
* *-a* means that the encrypted output will be base64 encoded, this allows you to view it in a text editor or paste it in an email. This is optional.
* *-salt* adds strength to the encryption and should always be used.
* *-in* secrets.txt specifies the input file.
* *-out* secrets.txt.enc specifies the output file.
* You will be prompted for a password.

It is not much use unless you can decrypt it. Let's do that:
`openssl aes-256-cbc -d -a -in secrets.txt.enc -out secrets.txt.new`

How does this work?

* *-d* decrypts data.
* *-a* tells OpenSSL that the encrypted data is in base64.
* *-in secrets.txt.enc* specifies the data to decrypt.
* *-out secrets.txt.new* specifies the file to put the decrypted data in.
* Enter the encryption password when asked.

Exercise 7: Asymmetric Encryption practice
In the future, Prof. Marco Canini will send your grades to you, but wait... To find out your grade, you will put what you learned in the course to practice. Specifically, he will use public key cryptography to create an encrypted file that contains your grades. Only the person with the correct private key will be able to access his own grades.

In this lab session, you need to setup a few things in order to get familiar with this game. Simply follow these instructions:

* Create a 4096-bit RSA key:
  `openssl genrsa -out <your INGI username>.key 4096`
* Extract the public key:
  `openssl rsa -in <your INGI username>.key -pubout -out <your INGI username>.pub`
* Send your teammate (or your colleague) just your public key (and verify that he got the correct public key so that he/she knows the key is definitely your own).
* You will also receive the public key of a person in your group. Now let's encrypt a secret message with his public key:
  `openssl rsautl -encrypt -inkey <your teammate's key>.pub -pubin -in secrets.txt -out secrets.txt.enc`
* Now send the encrypted file to your teammate and similarly receive from him/her a file encrypted with your public key
* Decrypt the cipher text to discover what he/she has written:
  `cat secrets.txt.enc | openssl rsautl -decrypt -inkey <your INGI username>.key`

Exercise 8: Hybrid encryption practice
As the last exercise consider this scenario. You have a public key from your friends and you have a file you want to send them in a secure manner.

Typically you would use a software to do this such as a chat application with an OTR (Off-the-Record) encryption feature or send an encrypted email. But let's assume that you want to do it yourself. Let's try using what we have learned so far.

* Get their public key (recall the last exercise).

* Generate a 256 bit (32 byte) random key:
  `openssl rand -base64 32 > key.bin`

* Encrypt the key (try it on your own).

* Encrypt your file (try it on your own, you can use the argument `-pass file:./key.bin` to give a password from a file)/

* Send/decrypt the files. Send the .enc files to the other person and... Can you find out how to decrypt it?