NETWORK AND TELECOMMUNICATIONS SYSTEMS SECURITY ASSIGNMENT

Deadline: June 28th, 2020

This assignment is mandatory. Your submission could be in *either English or Greek.*

Your final grade will be mainly determined by your achievement A ($0 \leq A \leq 10$) in this assignment. Of course you are allowed to exchange views in order to facilitate your understanding and your learning. However, each submission should reflect your personal work. You may be asked for short oral exams (via skype meeting, with cameras) at the beginning of the July, depending on the overall impression of your submission; in such a scenario, the grade A will be also determined by the oral exams. (Almost) identical answers between different submissions will be scrutinized.

In case that someone has submitted the optional Homework 1 and/or Homework 2, then the final grade will be equal to: max{10, A+H1+H2}, where H1 (resp. H2) is the grade of Homework 1 (resp. Homework 2).

If you use any additional source from literature, this should be explicitly referenced.

For problems 1-4, you will work on a Ubuntu system. You may download the Ubuntu image from https://www.ubuntu.com/download/desktop (version 20.0.4) and subsequently install it on a virtual machine. To this end, the Virtual Box software (https://www.virtualbox.org/) is suggested (you should download and install the appropriate version, based on your host machine). When Virtual Box first starts, you should select to create a new virtual machine, of the type Linux-Ubuntu.

Moreover, for problems 1-3, you will also use the openssl program, whish is pre-installed on the Ubuntu system. Guidelines on using openssl can be found on the file "Openssl_guidelines.pdf".

1. **Symmetric encryption [1.25]**

   (please check some hints at the end).

i) Create a text file being called *name.txt* containing the first 16 characters of your first name concatenated with your last name (if less than 16 characters are needed, you should add random characters at the end, until you get exactly 16 characters). The file should be exactly 16 bytes in length. There must be no newline character.

Encrypt the file name.txt using AES (with 256-bit key size), in the modes of operation ECB, CBC and CTR by using the same arbitrary key for all encryptions. Therefore, three encrypted files will be created, which should be respectively called as *encrypted_ecb.bin*, *encrypted_cbc.bin* and *encrypted_ctr.bin*. You should describe the commands that you used and submit all the created files.

ii) Create a text file being called *repeated_name.txt*, consisting of the 16 characters from the file name.txt, repeated by 5 times, such as its final size is exactly 80 bytes. With this new file, do exactly the same work as in case i, by using the same encryption key that you previously used; you should get three new ciphertexts, which should be respectively called as *new_encrypted_ecb.bin*, *new_encrypted_cbc.bin* και *new_encrypted_ctr.bin*. Again, you should describe the commands that you used and submit all the created files.

Compare the ciphertexts from case i with the ciphertexts from case ii and discuss your observations, based on the expected outcome according to the theory of modes of operations.

iii) In each of the three ciphertexts from case ii, modify its first bit and subsequently decrypt the modified ciphertext (with the same key as previously). You should get three messages, being called *new_decrypted_ecb.bin*, *new_decrypted_cbc.bin* and *new_decrypted_ofb.bin* which you should also submit. How are they related with the initial plaintext *repeated_name.txt*? Please discuss and explain your conclusions.

Hints

1) In all encryption/decryption procedures, you should use the arguments –nopad –nosalt (since openssl by default uses "padding" και "salting" even if it is not needed – to avoid any confusion on the resulting ciphertexts, you should use the aforementioned arguments).

2) For creating a text file of exactly 16 characters (e.g. for name Claude Shannon), you may proceed as follows: echo –n "claudeshannoncla" > name.txt

3) To handle binary files, you need a hex editor. You may choose any hex editor you like – a convenient one that can be freely installed is Bless).

2. Mounting a modification attack [1.5]

The file *ciphertext1.bin* corresponds to a ciphertext that has obtained by AES-CTR. You don't know the secret key. You do know though that the initial plaintext is a message that is being sent from Alice to Bob containing the bank account (for money transfer) of Alice, which is 1586120871445081 (more precisely, the file *ciphertext1.bin* is the AES-CTR encryption of the plaintext in the file "message.txt"). Your bank account is 1586120871441198. Describe in detail how you can appropriately modify the transmitted ciphertext so as to ensure that that when Bob decrypts it (Bob knows the secret key) he will get your bank account instead of Alice's bank account.

What could Alice do in order to avoid such an attack?

3. Public key encryption [0.75]

i) Create your own RSA pair "public key-private key", with size of modulus N being equal to 3072 bits. Submit your public key (not the private), describing all the commands that you executed.

ii) Create a random symmetric key of size 128 bits and save it in a file " key.bin". Using this key, encrypt with AES-CTR, and zero IV, the file «name.txt» of the previous problem A.i. The corresponding ciphertext will be called "encrypted_name.bin". Describe all the commands that you use and submit the encrypted file "encrypted_name.bin" (please do not submit your key – i.e. the file key.bin).

iii) Use appropriately the RSA encryption in order to securely send the file "key.bin" to your tutor. In other words, submit an appropriately encrypted version of the file "key.bin". using RSA, so as only your tutor is able to decrypt it, in order to subsequently decrypt the

file “encrypted_name.bin”. Describe the process (commands) of appropriately encrypting the file “key.bin”. Please note that you should submit only the absolutely necessary files and not more. Your tutor’s public key is the file publickey-tutor.pem.

4. Password cracking in Linux [2]

(please check some hints at the end).

The salted hashed versions of the passwords in Linux are being created by the function crypt. You may see information of this function in the following: http://manpages.ubuntu.com/manpages/bionic/man3/crypt.3.html (i.e. how we call this function, its arguments, how to “read” its output etc.)

You managed to get access to the secret shadow file of an Ubuntu Linux συστήματος (see file shadow). This system has two users, with names bob και alice.

i) Based on what you have read on the above link about the crypt function, which hash function has been used by the system to generate the salted hashed values of the password?

ii) You wish to recover the original passwords. To achieve this, you think of some personal information of them that you know:

   a. Bob lives in Athens, his wife is Maria and he loves Pink Floyd.

   b. Alice is born on August 3rd 1984, her car plate is ZKA5231 and her phone number is 6955345671.

Try to recover their original passwords, by appropriately using the function crypt (we assume that Alice and Bob are not familiar with security issues and they pick very “easy” passwords, just to be able to easily remember them).

(even if you don’t manage to recover their passwords, please describe your work).

iii) Could any of the passwords that you found above be also recovered by a rainbow attack, in case that the same hash function (and, of course, no salt) had been used? Please justify your answer via an example through the tool https://crackstation.net/

(Hint: To execute the crypt function, you may incorporate it within a small C (or C++) program. Within the source code, you should write at the beginning the following:

#include <crypt.h>

In compiling the source file into an executable, you should use the –lcrypt parameter; namely:

gcc crack_passwd.c –lcrypt –o crack_passwd

where crack_passwd.c (as an example) is the name of the file with the source code and crack_passwd the name of the executable file that will be created. Once the executable file is being created, you may execute this as following:

./crack_passwd

There are several free automated tools that perform password cracking; your answer though should not be based on them).

5. Using the GPG software [1]

By using GPG, create a pair of public-private key and then encrypt any message you want so as only your tutor (K. Limniotis) is able to decrypt it. You should also digitally sign your encrypted message. Your submission should contain only the ciphertext and the information that your tutor needs to verify your signature, as well as a description on your steps you followed. Your tutor's public key is the file: AA77401D3F369DB1B1E6AC64BEFB73866CD8DBE8.asc .

6. TLS security [1.5]

a) Visit three https websites of your own choice (one being an e-shop, one being a University, one being a news site) and check its TLS strength by using any TLS scanner tool. Submit the output of the scanning and discuss your main findings.

b) Some of users claim that after visiting the following web site: listing.saleofast.com, some days ago they noticed that their personal data that they had entered have been leaked. Please explain the possible reason.

c) Similarly as b), but for the web site https://www.blackboxresale.com/login.aspx

7. Nework security [1]

A. Consider the following threats to network security and describe – in simple words - how each of them is being addressed (if it is addressed) by IPSec.

   a. Password sniffing
   b. SYN flooding
   c. DNS cache poisoning attack

B. In the next two possible security incidents, please formulate shortly – for each of them, individually - a potential attack that is taking place (note that it may not necessarily be an attack but something else; in any case, please justify your claim).

   a. Your employees claim that, when they enter a valid URL address (https) in their web browsers, they are being re-directed into a strange website of an online pharmacy.
   b. Although your company uses a VPN (IPSec in tunnel mode, with the header AH), it seems that data that have been exchanged between two gateways have been leaked to non-authorised users.

8. Post-quantum one-time signatures [1]

a. Create your own OTS Lamport signature scheme, for signing a message of size 4 bits. Describe in detail which will be your public key, which will be your private key and how you constructed them. (The hash function that you use could be any).
b. Please sign, with the above signature scheme, the message m1=1001.
c. Next, please sign, with the above signature scheme, the message m2=0010.
d. Let us consider that an attacker (Eve) observes the above two signatures, for these two given (known) messages m1 and m2. Is Eve able to sign another 4-bit message, imitating yourself, without having knowledge of your private key? Explain your answer.