Translated from Georgian to English - www.onlinedoctranslator.com

Translated from Georgian to English - www.onlinedoctranslator.com

Assignment 3

This assignment consists of two parts. You can write code for both parts in a modular way, the main thing is to put all the files that will be needed to run, in one zip file (without folders) and so on. In the zip archive name enter your freeuni email prefix to @, # and the number of this assignment. For example, gosip12 # 3.zip

The source code is given in Python 2. Therefore, you must write in this language.

You can not use external libraries for these assignments.

1. Meet-in-the-middle attack on discrete logarithm

In this task you have to compute a discrete logarithm where p is a prime number. You will be given four arguments:

*p*- Simple number

*g*- Generator

*h*- one of the elements of. Obviously

*max\_x*- Maximum value of x

You need to find the x for which the equation ends.

For example, when your code is tested, it is known that max\_x does not exceed. With a trivial algorithm (passing all possible values ​​of x) we would have to perform multiplication. But for this case we will use a faster attack - meet-in-the-middle attack.

To understand how this attack works, note:

If and, we can represent as, where and.

Therefore, .

We can divide both sides of the equation by and get:

The value of all variables except and and is known in the obtained equation. So we can calculate all the possible values ​​of the left hand side, store them in the hash table with the corresponding -s, and then check for which value of -s we have the corresponding entry in the hash-table.

(Approximately how many surgeries will we need to carry out this attack?)

Technical details:

You need to complete the filedlog.py, Which is launched with the following command:

python dlog.py

stdoutYou must display the value of x on.

You can check the code using the example stored in the file. If your code does not complete work in 1 minute, you will not receive points.

In the Linux command line, you can check the time of the program by using the time command.

2. Forgery of RSA-based signature

The following scheme is used to generate signatures for 63-byte messages: The public key is the RSA standard pair (N, e), and the secret key is also the default key (N, d), where N is a 128-byte (1024-bit) integer. . The 63-byte message m signature is calculated as :, where:

M = 0x00 m 0x00 m

(Make sure the size of M is 128 bytes).

If the size of m is less than 63 bytes, then 0s are added to the message to fill it up to the required size (obviously, a single-byte message with "x" and a two-byte message with "0x00 x" will have the same signature. Interesting in terms of this task).

Your task is to find the signature for the 63-byte challenge message:

Crypto is hard --- even schemes that look complex can be broken

You will have access to a server that will return any 63-byte or shorter message signatures except challenge messages.

You will also have access to a verification server that will receive a pair of notifications and signatures and will tell you whether the signature matches the message.

Part of working with the network (sending and receiving packets), as in the previous tasks, is implemented. To begin with, you can check if the connection code of the server works  
sample.pyBy running.

You need to create a filesign.py, Which is launched with the following command:

python sign.py

And outputstdoutSigning a challenge message on.