Translated from Georgian to English - www.onlinedoctranslator.com

Translated from Georgian to English - www.onlinedoctranslator.com

Assignment 2

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 # 2.zip

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

1. Block Code Encryption

In this task you need to decrypt the message (challenge\_ciphertext.txt) received by AES in CBC-modePKCS # 7Padding.

You will have access to a server that will encrypt sent messages, but the answer will only tell you if an error occurred during the decryption.

Part of working with the network (sending and receiving packets) is implemented. To begin with, you can send a given encrypted message to the server and make sure it is decrypted without error.

Plaintext encoded message in ASCII is a readable English phrase, so you can easily test your solution.

See README for more information on uploaded files.

You must submit a file named decipher.py, which will run as follows:

python decipher.py <ciphertext\_filename>

And displays an encrypted message (plaintext) on stdout without paging.

2. CBC-MAC

In this task you have to attack raw CBC-MAC, thus showing that it is not protected by MAC for messages of different lengths.

Let me remind you how CBC-MAC works on the example of x -block messages:

You can get tags (generated by unknown key) for any 2-block (32-byte) message. Your goal is to get a tag for any message with an even block.

To test yourself, you will have access to the Verifier on the server, with which you will be able to check whether the received tag message is consistent.

As in the previous task, part of working with the network (sending and receiving packets) is implemented.

To begin with, you can check that Mac and Verifier are working on 2-block text\_message, and then test your solution on challenge\_message: "I, the server, hereby agree that I will pay $ 100 to this student" (4-block message).

See README for more information on uploaded files.

You need to submit a file named forge.py, which will run as follows:

python forge.py <challenge\_filename>

And displays a tag on the stdout for the corresponding message.