**ACS 54500 Cryptography and Network Security – Lab 8**

In this lab, we have to set up and test secret key encryption.

**Task 1:**

In this task, the goal is to use frequency analysis to decipher a monoalphabetic cipher that is given, and use that to figure out what the encryption key is.

First, I created a new text file called plaintext.txt, and then used the watch command to detect, track and display changes made to this file.

Text

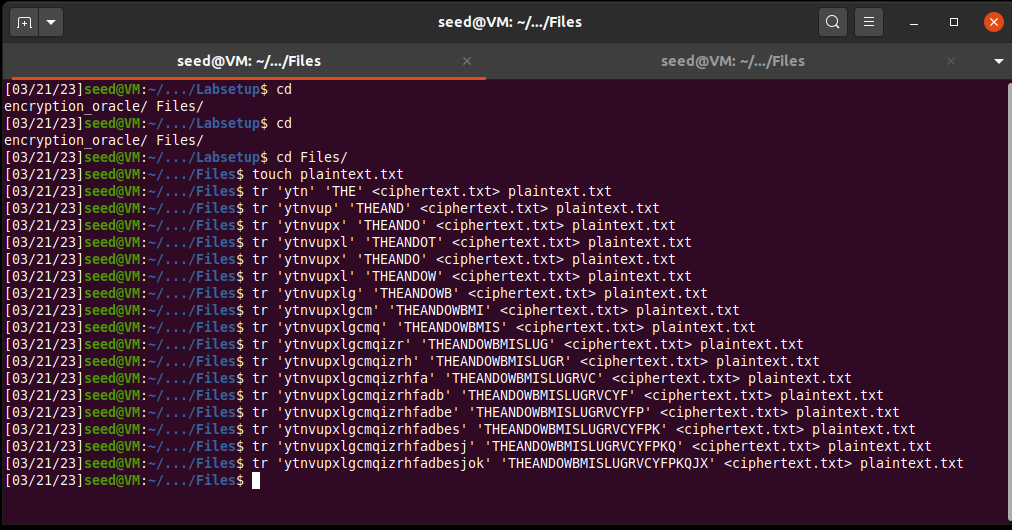
Description automatically generated

Here is the command that I used to watch the file – ***watch -n 5 cat plaintext.txt***.

Here, the 5 indicates an interval of 5 seconds between each refresh. Now, we use the given freq.py to figure out which 1-grams, 2-grams and 3-grams are the most common in the ciphertext to decipher it.Table

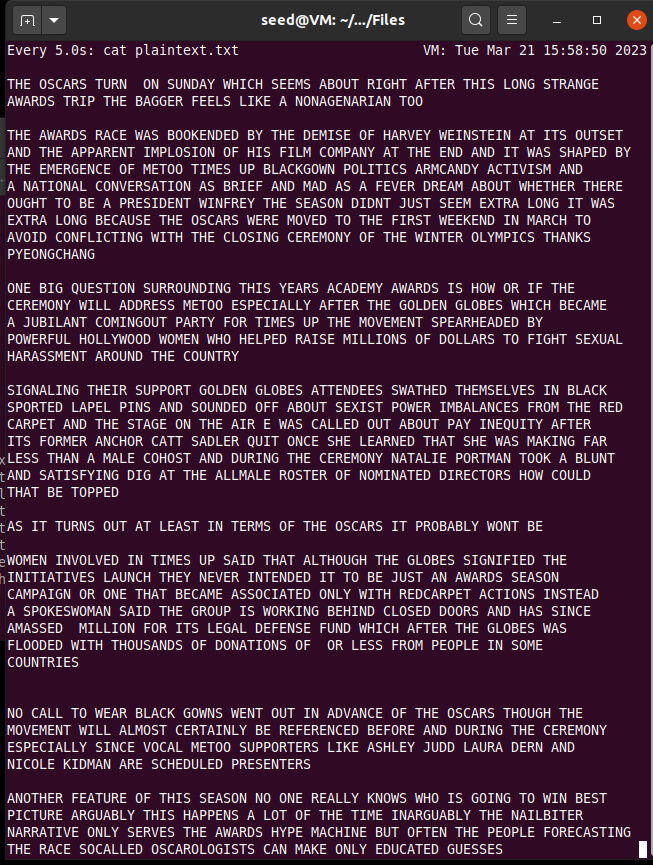
Description automatically generated

Below is a screenshot of me deciphering the cipher letter by letter -



Here was the encryption key that I found was used in encrypting the cipher - ***'ytnvupxlgcmqizrhfadbesjok' => 'THEANDOWBMISLUGRVCYFPKQJX'***.

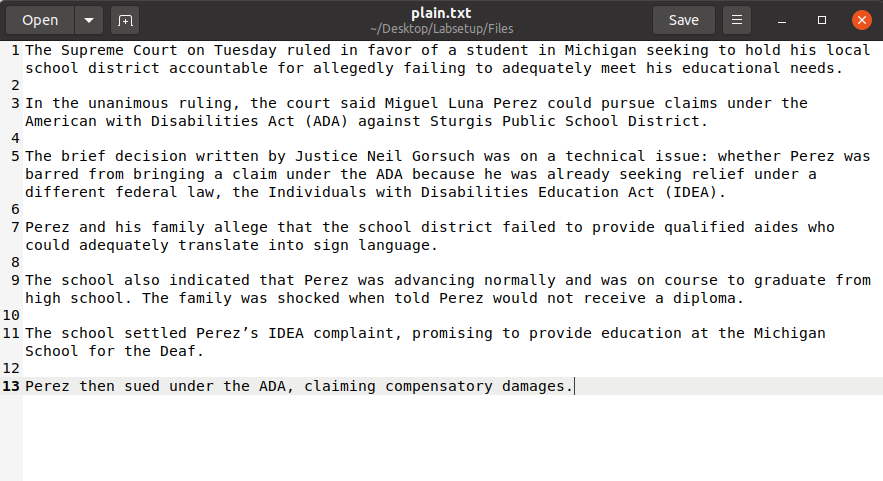
Finally, here is the deciphered plaintext – (it was too long, so I ended up putting only part of it)



Thus, the cipher has been successfully deciphered.

**Task 2:**

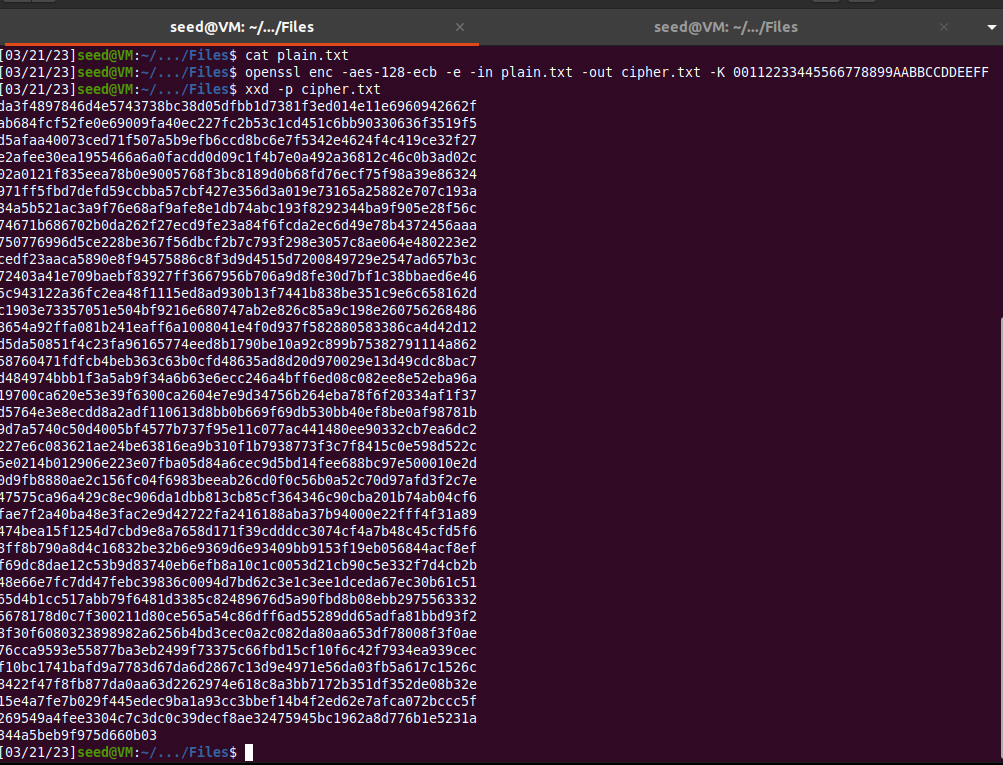
In this task, we experiment with 3 different kinds of encryption algorithms and modes. We had to choose any text to encrypt, so I picked a random news article and put some text from it in plain.txt -



I used the AES ECB, CBC and CTR algorithm modes.

**AES ECB**

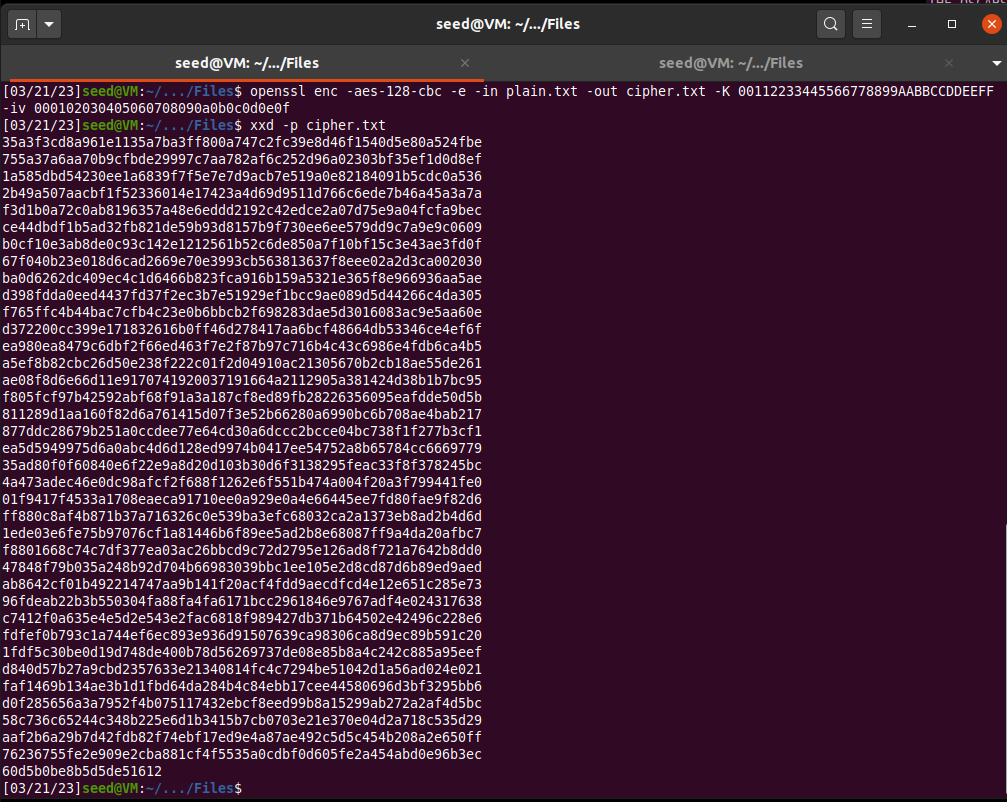
This algorithm only needs a 16 byte key, so below is a screenshot of how the commands used -



The screenshot also shows the encrypted text after we use the xxd -p command to convert the text into a hex string.

**AES CBC**

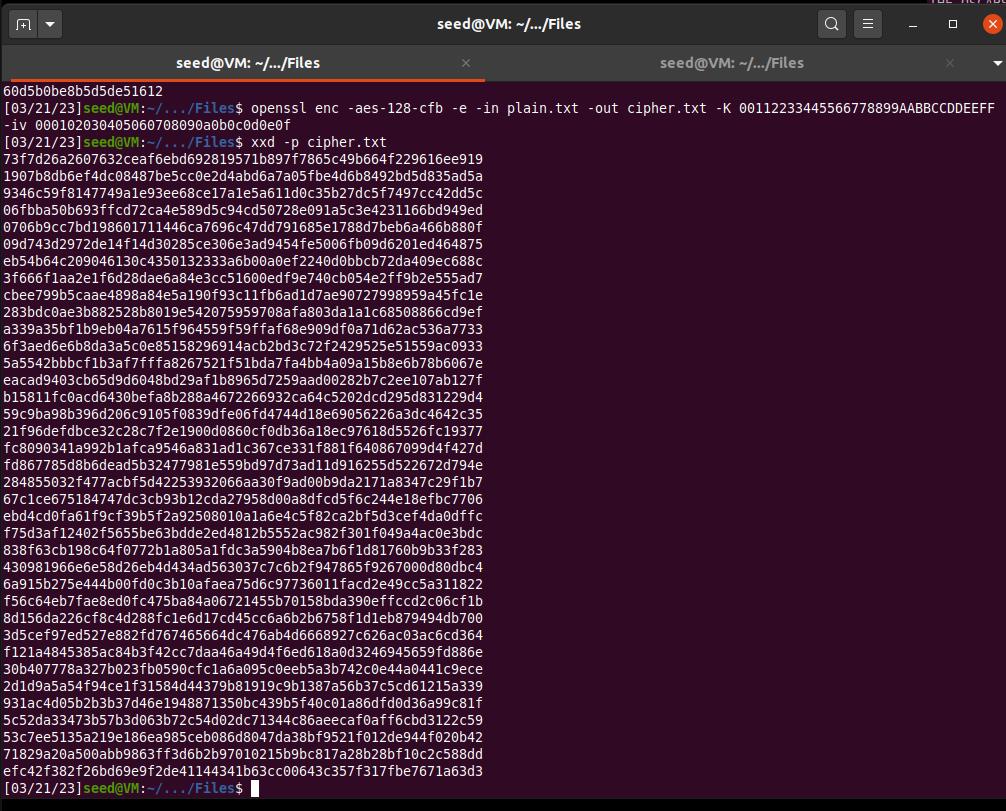
This mode needs an additional parameter of the initialization vector to be specified. Below is the screenshot of the commands used to encrypt the plaintext -



The above screenshot also shows the encrypted text in the form of a hex string.

**AES CTR**

This mode also needs an additional parameter of the initialization vector to be specified. Below is the screenshot of the commands used to encrypt the plaintext -



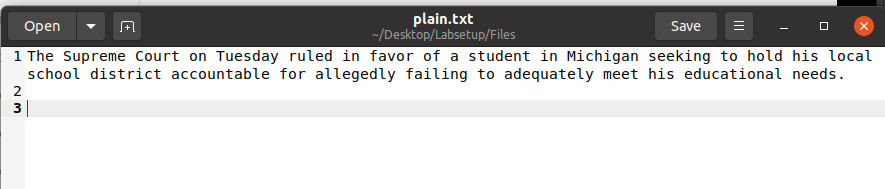
The above screenshot shows the encrypted text in the form of a hex string.

Thus, we have successfully demonstrated three different modes of the AES encryption algorithm and used them to encrypt a plaintext.

**Task 6:**

This task has three subtasks that deal with the initialization vector and common mistakes associated with it.

I shortened the plaintext in this task so make it just one sentence –

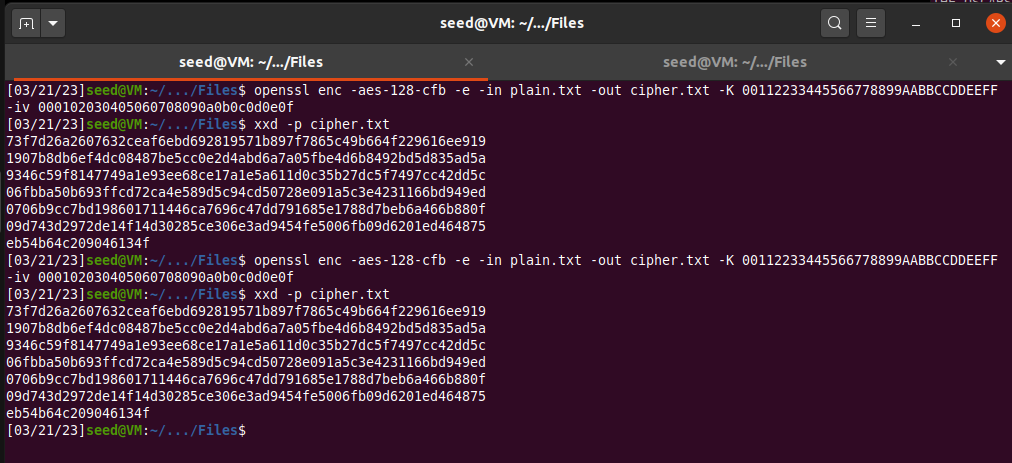


**Task 6.1:**

This subtask is an experiment with the uniqueness of IVs. We have to encrypt the plaintext first with the same IV twice, and then with two different IVs, and describe what is observed.

**Encryption using the same IV:**

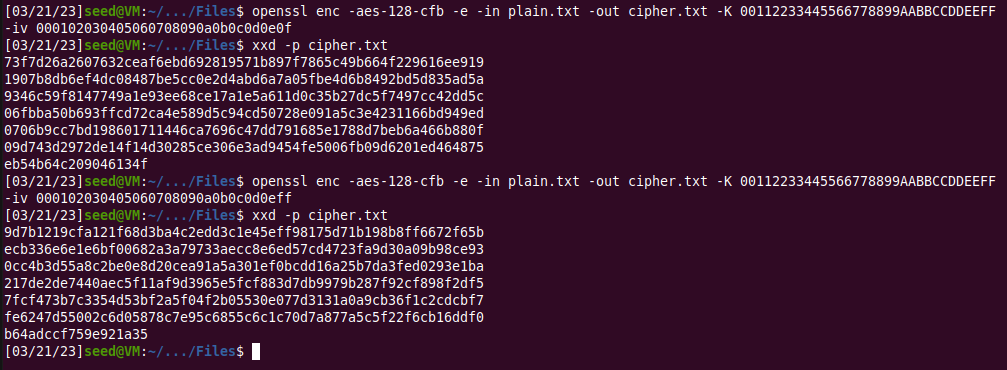
Below is a screenshot of me encrypting the plaintext with the same IV twice -



This shows that using the same IV to encrypt the same plaintext gives the same ciphertext regardless of the situation because the same mode of encryption and key are used. This makes it vulnerable to attacks. This means that no IV should be reused under the same key.

**Encryption using different IVs:**

Below is a screenshot of me encrypting the plaintext with two different IVs with a minor change of just two characters -



Even though the same key was used, using two different IVs to encrypt the same plaintext gave two very different encrypted texts. This shows that using a different IV each time leads to different encryptions.

**Task 6.2:**

In this subtask, the goal is to prove that when using the same key, we can derive P2 if we know what P1, C1 and C2 are. We modify the sample code provided to find P2 from the equations (the IVs cancel out) –

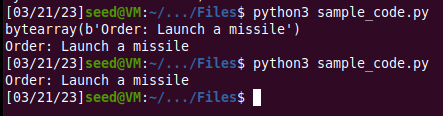
***P1 xor P2 = C1 xor C2***

***P2 = C1 xor C2 xor P1***

Here is the modified code with the given plaintext message and hex strings -

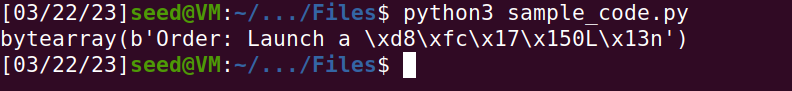


Now we run it after giving it the necessary permissions. This is the output -



Thus, we figured out that the second plaintext is “Order: Launch a missile”.

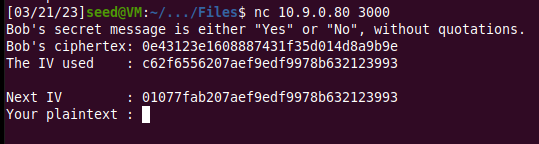
If we replaced OFB with CFB, then considering how CFB works, the attacker would not be able to fully recover the plaintext. I ran the same code by encrypting the two plaintexts and putting in the expected ciphertexts in the xor program to get the below result:



**Task 6.3:**

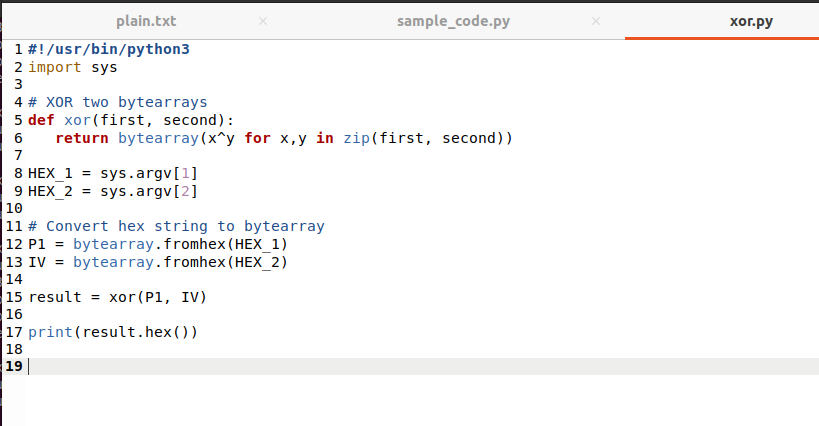
In this subtask, we need to figure out what Bob voted for (whether he voted “Yes” or “No”). Bob is simulated using an encryption oracle.

To connect to the oracle, we run the netcat command as shown below -



This shows Bob’s ciphertext, the IV used to encrypt his plaintext or vote, and the next IV.

To be able to xor two hex strings, I wrote the following code -



Now, I first used “Yes” as what I thought Bob voted. So, I saved this in P1, and converted it into a hex string using the xxd -p command -



The encryption mode being used is CBC, which requires padding. The hex string 596573 has 3 bytes, so we need to add 13 more bytes of padding, which means 13 in hexadecimal i.e., 0d repeated 13 times.

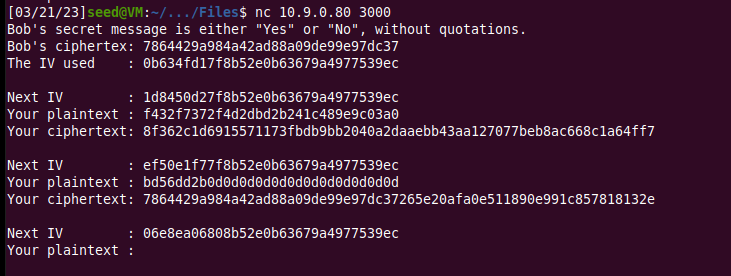
We now xor the IV used by Bob with the result -



The above result is now xor-ed with the next IV provided by the oracle.



We input this result in the oracle as our plaintext.

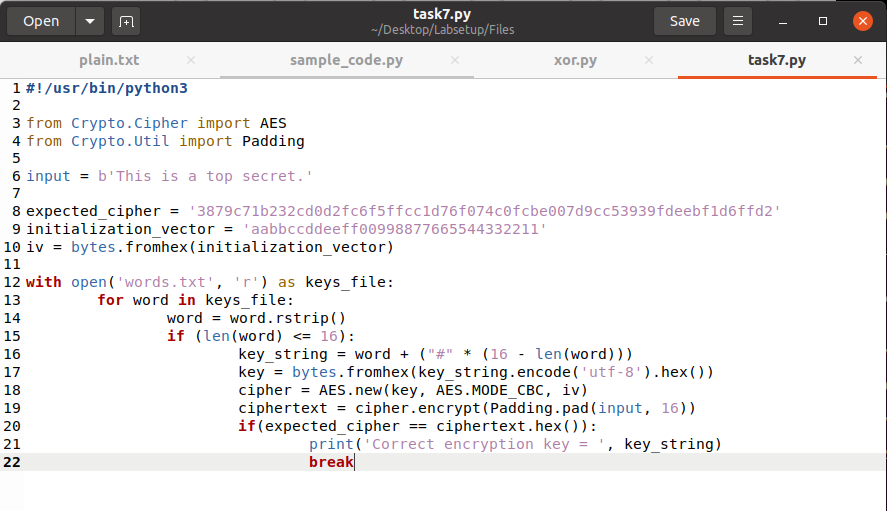


As shown above, when we omit the extra hex digits, Bob’s ciphertext and our ciphertext are the same. Thus we have guessed what the actual content of Bob’s message was.

**Task 7:**

In the last task, we have to find the key used to encrypt a plaintext from a file of encryption keys. I modified the in-class activity code to read from the file, remove new line characters, add padding of ‘#’ symbols and then encrypt using AES CBC mode.   
If the obtained cipher text matches the expected cipher text, then we print the key used for this encryption.

We are given a plaintext: “This is top secret.”, the expected ciphertext and the initialization vector. Here is the code -



Now, we run the above code and this shows that the correct encryption key used is “Purdue##########”.



Thus, we have successfully figured out the right encryption key used to encrypt the given plaintext.