CPE-321
Introduction to Computer Security

Symmetric Key Cryptography Exploration

Assignment Designed by Dr. Peterson, Revised by Dr. DeBruhl

4/10/23 (Updated Hartman/Ngo)

Objectives

The objectives for this lab assignment are as follows:

- To explore symmetric key cryptography security with different modes
    - Electronic Codebook Mode (ECB)
    - Cipher Block Chaining Mode (CBC)
- Explore the limits of block ciphers in their use
- Performance Study of Public vs Symmetric key algorithms

Tasks

Task 1: Modes of Operation – In this task, you will explore the differences in security attained by the ECB and CBC modes of encryption. Using the AES-128 primitive provided by your cryptographic library, implement your own versions of ECB and CBC modes of encryption (do not use the built-in methods for modes of operation). Your program should take a (plaintext) file, generate a random key (and random IV, in the case of CBC), and write the encryption of the plaintext in a new file.

Note: Regarding utility of built-in methods from your chosen cryptographic library, the AES methods often require you to specify a block cipher mode as an argument.

You may use the supplied AES encrypt/decrypt primitives with ECB mode specified, but you should only encrypt 128 bits at a time. Specifically, if you are using PyCryptodome cryptography package in python, you can use `AES.new(key, AES.MODE_ECB)` to generate the AES ciphers in your implementations of ECB and CBC mode.

Although you are specifying `AES.MODE_ECB` you will not be using the built-in ECB cipher block mode because you will be encrypting only one 128 bit block at a time (and XOR sequent blocks with previous ones in CBC) and concatenating them together. You also need to implement your own key generation and PKCS padding.

AES is a block cipher, and expects a 128-bit plaintext message and produces a 128-bit ciphertext – nothing more, nothing less. Putting a block cipher like AES, into a "**mode of operation**" is the way to enable encrypting data larger than 128-bits using a single key. However, a problem arises in certain modes of operation if the files are not evenly divisible by 128-bits. To account for plaintexts that are not an integral size of AES's block size, implement PKCS#7 padding. Details of this scheme can be found here: http://tools.ietf.org/html/rfc5652#section-6.3 but perhaps a more easily understood description is here: http://en.wikipedia.org/wiki/Padding_(cryptography)#PKCS7.

Encrypt one of the BMP files listed in this Canvas assignment using your ECB and CBC implementations, creating two different ciphertexts. Be sure to the preserve and re-append the plaintext BMP headers. The header size should be 54 bytes for the BMP format we used, but we have also seen 138 bytes work for Apple Macs when 54 bytes didn't.

Task 2: Limits of confidentiality – In this task, you will explore some of the limits of block ciphers in their use within a secure system. Based on the PKCS#7 padding and CBC encryption code from Task 1, write two "oracle" functions that emulate a web server that wants to use cryptography to protect access to a site administration page. First, at the start of your program generate a random AES key and IV, which will be used in both functions, keeping it constant for the execution of your program (do not generate a new key or IV for every encryption and decryption). The first function, called submit(), should take an arbitrary string provided by the user, and prepend the string:

userid=456; userdata=

and append the string:

;session-id=31337

For example, if the user provides the string:

You're the man now, dog

submit() would create the string:

userid=456;userdata=You're the man now, dog;session-id=31337

In addition, submit() should: (1) URL encode any ';' and '=' characters that appear in the user provided string; (2) pad the final string (using PKCS#7), and (3) encrypt the padded string using the AES-128-CBC you implemented in Task 1. Submit() should return the resulting ciphertext.

The second function, called verify(), should: (1) decrypt the string (you may use a AES-CBC decrypt library or implement your own CBC decrypt); (2) parse the string for the pattern ";admin=true;" and, (3) return true or false based on whether that string exists. If you've written submit() correctly, it should be impossible for a user to provide input to submit() that will result in verify() returning true.

Now the fun part: use your knowledge of the way CBC mode works to modify the ciphertext returned by submit() to get verify() to return true. **Hint:** Flipping one bit in ciphertext block c_i will result in a scrambled plaintext block m_i, but will flip the same bit in plaintext block m_{i+1}.

Task 3: Performance comparison – In this task, we will quantify the performance differences between public and symmetric key algorithms. Fortunately, OpenSSL provides a simple interface for doing so.

Openssl speed rsa

Openssl speed aes

can perform and measure their respective public and symmetric key operations using different parameters. One of the returned results for both operations is a **measure of throughput:** Throughput for AES is measured in 1000s of bytes/sec. Throughput for the four RSA functions is measured as signatures/second, verifies/second, encryptions/second, and decryptions/second. To convert RSA throughput numbers to bytes/second, multiply the provided numbers by the corresponding RSA key size in bytes (e.g., 512 bits = 64 bytes). Converting RSA throughput to bytes/second will allow you to compare AES and RSA bytes/second performance. Run these operations and report your findings. **Include two graphs**: one that plots the **block size vs. throughput for the various AES key sizes** and one that plots the **RSA key size vs. throughput for the four RSA functions**.

Questions:

1. For task 1, looking at the resulting ciphertexts, what do you observe? Are you able to derive any useful information about either of the encrypted images? What are the causes for what you observe?

2. For task 2, why this attack possible? What would this scheme need in order to prevent such attacks?
3. For task 3, how do the results compare? Make sure to include the plots in your report.

Submission:

Write a brief report describing what you did and what you observed. Include any code that you wrote, as well as answers to any questions. Please include any explanations of the surprising or interesting observations you made. Write at a level that demonstrates your technical understanding, and do not shorthand ideas under the assumption that the reader already "knows what you mean". Think of writing as if the audience was a smart colleague who may not have taken this class.

Describe what you did in sufficient detail that it can be reproduced. Please do not use screenshots of the VM to show the commands and code you used, but rather paste (or carefully retype) these into your report from your terminal. Submit your completed write up to Canvas in PDF format. Any generated images should be included in your report.

Modes of operations

The modes of operation explored in this assignment (ECB and CBC) are all NIST standards. For diagrams see:

Electronic Codebook Mode (ECB): https://en.wikipedia.org/wiki/File:ECB_encryption.svg

Cipher Block Chaining Mode (CBC): https://en.wikipedia.org/wiki/File:CBC_encryption.svg