UNIVERSITY OF MARYLAND
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENEE 459C
Computer Security
Instructor: Charalampos Papamanthou

Homework 2

Out: 09/26/14 Due: 10/07/14 11:59pm

Instructions

1. Strictly adhere to the University of Maryland Code of Academic Integrity.
2. Submit your solutions as a pdf document at Canvas. Include your full name in the solutions document. Name the solutions document as x-hw2.pdf, where x is your last name.
3. For Problem 5, submit your code files in a tarball at Canvas.

Problem 1 (20 points) Alice and Bob shared an n-bit secret key some time ago. Now they are no longer sure they still have the same key. Thus, they use the following method to communicate with each other over an insecure channel to verify that the key k_a held by Alice is the same as the key k_b held by Bob. Their goal is to prevent an attacker from learning the secret key.

1. Alice generates a random n-bit value r.
2. Alice computes $x = k_a \oplus r$, and sends x to Bob.
3. Bob computes $y = k_b \oplus x$ and sends y to Alice.
4. Alice compares r and y. If $r = y$, she concludes that $k_a = k_b$, that is, she and Bob have indeed the same secret key.

Show how an attacker eavesdropping on the channel can gain possession of the shared secret key. Also show how an attacker who can do more than eavesdropping can make Alice and Bob believe they do not share the same key.

Problem 2 (20 points)

1. Write the definition of the three security properties of cryptographic hash functions that we mentioned in class. Prove that collision resistance implies second-preimage resistance. Does second-preimage resistance always imply preimage resistance? Why? Why not?
2. State the security property of a message authentication code (MAC) that we mentioned in class (existential forgery under chosen plaintext attacks).
3. Let $f_k(m)$ be a secure message authentication code, i.e., it satisfies the previous definition (k is the secret key). Consider the following MAC that is based on $f_k(.)$. On input a message $a||b$ with $|a| = |b| = n - 1$ and key $k \in \{0,1\}^n$, the MAC is computed as

$$f_k(0||a)||f_k(1||b) ,$$

where $||$ denotes concatenation. Is the new MAC secure? If yes, prove it. If not, give an attack. How about this MAC

$$f_k(0||a)||f_k(f_k(1||b))\,?$$

Again, either prove security or give an attack.

Problem 3 (20 points) Let $\mathcal{E}_k(\cdot)$ and $\mathcal{D}_k(\cdot)$ be the encryption and decryption algorithms of a symmetric key cryptosystem with ℓ-bit keys and n-bit plaintexts and ciphertexts. We derive another symmetric key cryptosystem with a 2ℓ-bit key by applying twice $\mathcal{E}_k(\cdot)$ and $\mathcal{D}_k(\cdot)$:

$$\mathcal{E}'_{(k_1,k_2)}(M) = \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(M))$$

$$\mathcal{D}'_{(k_1,k_2)}(C) = \mathcal{D}_{k_1}(\mathcal{D}_{k_2}(C))$$

Consider an adversary who wants to perform a brute-force attack on the above cryptosystem to recover keys k_1 and k_2 but knows only a single valid pair of plaintext M and ciphertext C.

1. Suppose that the adversary has only $O(n)$ bits of space. Describe the attack in pseudocode and estimate the number of encryption and decryption operations performed in addition to the overall runtime of the algorithm.
2. Suppose now that the adversary has $O(n \cdot 2^\ell)$ bits of space. Describe, again in pseudocode, a more efficient way to perform the attack and estimate the number of encryption and decryption operations performed and the overall runtime.
3. In response to increases in computational power, the Data Encryption Standard (DES) was extended to support longer keys without designing a new algorithm. This was accomplished by developing Triple DES (3DES), which encrypts the plaintext three times with three different keys, with decryption applying the same keys in reverse order (the same as previously described scheme, but with one more key). Given again $O(n \cdot 2^\ell)$ bits of space, describe how to modify the attack of (2) to break 3DES. Again, estimate the number of decryption and encryption operations performed along with the overall runtime.

All answers estimating the number of cryptographic operations should be in big-O notation in terms of n and ℓ, and all answers should have at least a short justification.

Problem 4 (20 points) In class, we talked about the DES algorithm for symmetric encryption. Recall that the output of a single round of the DES algorithm with input (L_i, R_i) (where L_i and R_i are 32-bit blocks) is

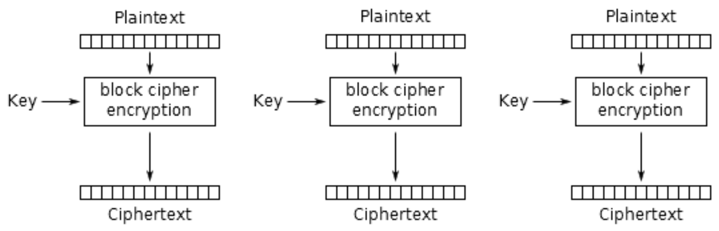
$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus f_{k_i}(R_i)$$

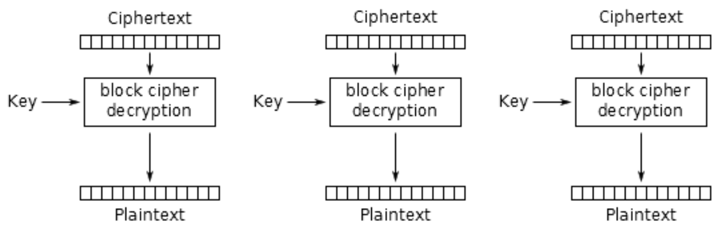
where k_i is the 48-bit round key. Recall also that the output of the round function $f_{k_i}(.)$ is computed with the S-boxes, that map a 48-bit string (which equals $k_i \oplus e_i$ where e_i is the expansion of R_i into 48 bits) to a 32-bit string.

1. Why does DES use eight S-boxes, each mapping 6 bits to 4 bits, and not a single S-box that directly maps 48 bits to 32 bits? How many bits would be required to store such an S-box?
2. Devise an algorithm that breaks a single-round DES and retrieves the round key k_1. Assume you are given *one pair* of valid input/output (i.e., L_1, R_1 and L_2, R_2). How many operations does the algorithm require?

3. Devise an algorithm that breaks a double-round DES when you are given one pair of valid input/output (i.e., L_1, R_1 and L_3, R_3). Can this attack be extended to a triple-round DES (in this case you are given L_1, R_1 and L_4, R_4)? Justify your answer.
4. In class we talked about different modes of block cipher encryption. Below are diagrams detailing ECB, CBC, CFB, and OFB modes of operation. For each mode of operation, explain what happens, in terms of which block(s) and bit(s) get corrupted, when you flip a single bit in the ciphertext and then try to decrypt it using the same mode.
   (a) ECB mode:

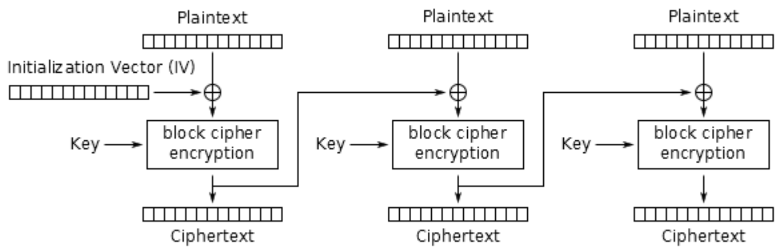


Electronic Codebook (ECB) mode encryption

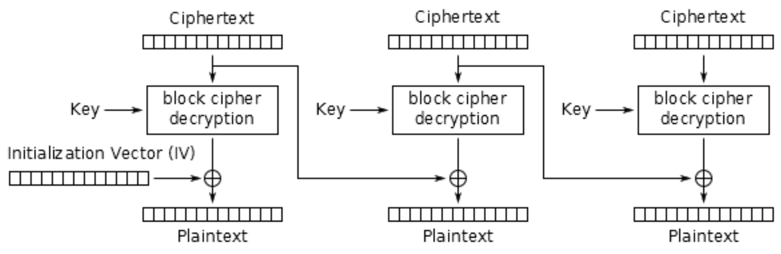


Electronic Codebook (ECB) mode decryption

   (b) CBC mode:

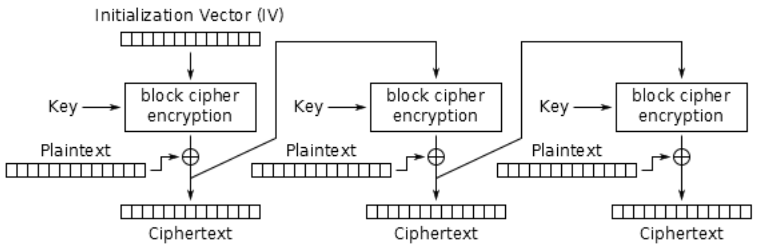


Cipher Block Chaining (CBC) mode encryption

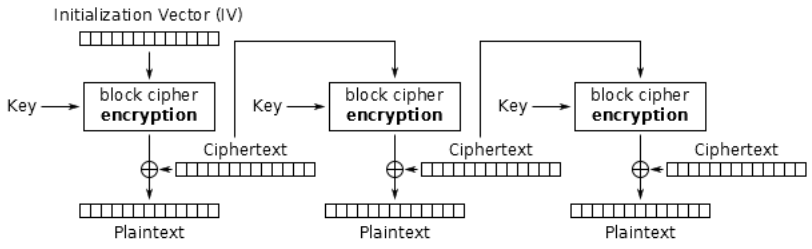


Cipher Block Chaining (CBC) mode decryption

(c) CFB mode:

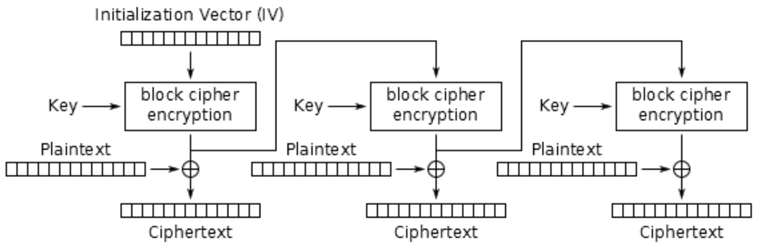


Cipher Feedback (CFB) mode encryption

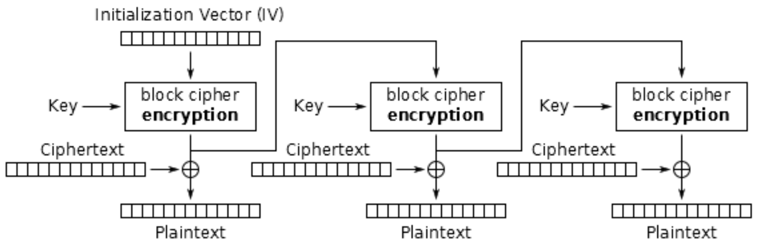


Cipher Feedback (CFB) mode decryption

(d) OFB mode:



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Problem 5 (20 points) You may use Java's crypto libraries for parts 1 and 2.

1. Write a java program that takes a file name as a command line argument and outputs the SHA-256 hash of the file. For example, with this input file, the program should output:
```
$ java SHA2 input_file
6896D9EA3F73A4434F5832BC65714E7D066F177373F36F34DC8A6F735DAA41B1
```
2. Write a java program that performs AES encryption using CBC mode and PKCS5Padding. It should take as command line arguments a file containing a 16 byte binary key, a file containing

a 16 byte binary initialization vector, an input file to encrypt, an output file name, and either "enc" or "dec" to encrypt or decrypt. For example, with this key file, this initialization vector file and this input file, it should output a ciphertext equal to this, i.e.,:

```
$ java AES key_file iv_file input_file outfile enc
$ diff outfile input_file.encrypted
$ # outfile is equal to input_file.encrypted
$ java AES key_file iv_file input_file.encrypted outfile dec
$ diff outfile input_file
$ # outfile is equal to input_file
```

3. Alice and Bob formed a standards committee to design a secure encryption scheme. Mallory wants to be able to spy on their communications, so she joined their standards committee and suggested they encrypt files by splitting the file into blocks of size equal to the length of they key, and then compute a XOR of each block of the plaintext with the key to get the ciphertext. Their encryption program in C is given below. It reads the plaintext from stdin and outputs the ciphertext to stdout:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    char key[8] = {'K', 'E', 'Y', ' ', 'H', 'E', 'R', 'E'};
    int keylen = 8;
    char *buf = malloc(keylen * 512);
    int nread;

    while (nread = read(0, buf, sizeof(buf))) {
        int i;
        for (i = 0; i < sizeof(buf); i ++)
            buf[i] ^= key[i % keylen];
        write(1, buf, nread);
    }
}
```

Your task is to help Mallory spy on their communications by writing a program to crack this encryption scheme. You are given two ciphertexts 1.enc and 2.enc that you need to find the key and plaintext for. In 1.enc, the length of the key is 8 bytes. In 2.enc, the length of the key could be any value less than 30. The bytes of the key can be any value from 0x00 to 0xff.
Hints: First you will probably want to code and test a program to break the following simpler encryption scheme:
Given a plaintext and a 1-byte key, XOR every byte of the plaintext with the key to get the ciphertext. This encryption scheme can be attacked by trying to decrypt the ciphertext with all 256 possible keys, and picking the key that gives a character frequency distribution most similar to english text. The following array will probably be useful:

```
double freq[] = {0.0651738, 0.0124248, 0.0217339, 0.0349835,
                 0.1041442, 0.0197881, 0.0158610, 0.0492888,
                 0.0558094, 0.0009033, 0.0050529, 0.0331490,
                 0.0202124, 0.0564513, 0.0596302, 0.0137645,
                 0.0008606, 0.0497563, 0.0515760, 0.0729357,
                 0.0225134, 0.0082903, 0.0171272, 0.0013692,
                 0.0145984, 0.0007836, 0.1918182};
```

In the above array, `freq[0]` contains what fraction of characters in average english text are the letter 'a', `freq[1]` is for the letter b and so on.

You are supposed to hand in a tar ball that contains (i) SHA2.java and AES.java (the programs of the first question) (ii) 1.plain and 1.key (plaintext and key files for 1.enc); (iii) 2.plain and 2.key (plaintext and key files for 2.enc); (iv) the program(s) that you wrote, in any language, to crack the ciphertexts 1.enc and 2.enc respectively.