Assignment 3: Block Ciphers

CSCI3381-Cryptography

Due October 3, 2014

If you're concerned about the size of this assignment, rest assured that I don't expect you to do all the problems. It's just that I thought of a zillion interesting ones, tossed out some of my favorites, but still left a large assortment to provide you with some variety and choice. A 'perfect' score on this assignment is 100 points, and you won't get more than 100 for the assignment, but there is a total of 200 points to choose from. Great at math but struggling with programming? The written problems in the first section will give you a chance to shine, and you can achieve a perfect score doing just these. Love to code but a tad mathophobic? You'll find an assortment of programming challenges, from the trivial to the sublime. Don't like math **OR** programming? Maybe you should reconsider...

The written problems refer to block ciphers with a variety of parameters: DES (block size 64 bits, key size 56 bits); AES (block size = key size = 128 bits); and the baby block cipher I've provided (block size = key size = 16 bits). The programming problems all use the baby block cipher, because the small key size allows you to carry out some experiments that would be impossible with the industrial-strength versions. For problems that require you to decrypt an extensive ciphertext, you will receive the reward of illuminating and entertaining reading on completion of a successful decryption.

(But it would be nice to be able to use industrial-strength crypto in some of our programming work. I have recently installed the package pycrypto and am experimenting with it. Stay tuned.)

1 Written Problems

1. Block Cipher Modes of Operation *6 points per part, 30 total.* Parts (a)-(d) refer to the cipherblock chaining mode (CBC), and part (e) is about a variant. These problems do not depend at all on the specific block cipher used. Explain your answers carefully.

(a) Suppose a message of 100 plaintext blocks is being encrypted with CBC mode. Suppose that, prior to encryption, one bit of the tenth plaintext block is inadvertently changed. How many blocks of plaintext, after decryption, are certain to be correct?

(b) Same setup as part (a), but this time, one bit of the tenth *ciphertext* block is changed after encryption but before decryption. How many blocks of plaintext, after decryption, are certain to be correct?

(c) Same set up as (b), but this time the tenth and eleventh ciphertext blocks are swapped before decryption.
(d) Suppose we encrypt two multi-block messages with CBC mode but use the same IV each time. What information about the plaintext is leaked by the ciphertext? (You also might try to imagine a scenario in which this information is useful to an attacker.)

(e) Here is a variant of CBC mode called 'Infinite Garble Extension' (I don't know why!) As usual, the plaintext blocks are denoted $P_1, P_2, \ldots$, the ciphertext blocks $C_1, C_2, \ldots$, and an initialization vector block IV. The encryption algorithm is given by

$$C_1 = E_K(P_1) \oplus IV,$$

$$C_i = E_K(P_i \oplus C_{i-1}) \oplus P_{i-1}.$$

Write analogous equations defining the decryption operation.

2. Modifications to Substiution-Permutation Networks. *Point values: 7,8,10; 25 total.* This problem is about the internal structure of block ciphers. The model here is the generic substitution-permutation network described in the notes. (While you don't need to explicitly consider AES in this problem, you should be aware that conclusions also apply to AES, where the mixing permutation is replaced by multiplication by an invertible matrix. Problem 4 of Chapter 5 is relevant to part (c) below.)

(a) Suppose our encryption algorithm uses only *one round* of the SP network. Explain how we can completely decrypt any block given a single known plaintext-ciphertext pair of blocks. Describe the procedure in detail. Does this attack recover the key?

*(b)*Suppose that instead of alternating AddKey, Sub, and Mix in the r rounds of our substitution cipher, we do all r rounds of AddKey, then all r rounds of substitution, then all r rounds of the mixing permutation, in that order. Show that if we have a single known plaintext-ciphertext pair of blocks, we can decrypt any block. Does this attack recover the key?

(c) Suppose we got rid of the S-boxes, and just retained the AddKey and Mix phases of each round. Show how, given a single known plaintext-ciphertext pair of blocks, we can decrypt any other block of ciphertext. HINT: Show that this forces

$$E_K(M_1) \oplus E_K(M_2) = \pi(M_1 \oplus M_2),$$

where π is some permutation of the bits, for any two plaintext blocks M_1 and M_2. Does this attack recover the key?

3. Block Cipher Statistics. *9 points per part; 45 total.* An obvious requirement of cryptographic systems is that if you have two different plaintexts $M_1 \neq M_2$ and encrypt them under the same key K, then the ciphertexts have to be different: $E_K(M_1) \neq E_K(M_2)$. However, if you encrypt the same plaintext M under two different keys $K_1 \neq K_2$, it is entirely possible that $E_{K_1}(M) = E_{K_2}(M)$. This was important in figuring out how many known plaintext-ciphertext pairs we would need to successfully mount a brute-force attack, or a meet-in-the-middle attack against double encryption.

In this problem you will study the statistics of such 'collisions'. Some pointers to relevant probability tools that you may have forgotten are given in each part. *In general,* use k to represent the number of bits in the key, m to denote the number bits in a block, and then use your general answers to obtain specific numbers for DES, AES, and BabyBlock.

(a). Fix a plaintext block M and a candidate ciphertext block M'. What is the probability that there is *no* key K such that $E(K, M) = M'$? HINT: Choosing one random key and encrypting M under it to see if you get M' is , or should be, the same as choosing a random block and seeing if you get M'. Then think of multiple independent trials of this experiment.

(b) Use the result of *(a)* to estimate, for each M, the number of ciphertext blocks that cannot be gotten by encrypting M. Do this for each of our three example block ciphers. What does this tell you about the likelihood of colliding keys for a given plaintext M, at least in the case of AES and the Baby Block cipher? (HINT: It is very helpful to know that $(1 - 1/n)^n$ is close to e^{-1} for large n, and that more generally $1 - x$ is well approximated by e^{-x} for x close to 0. This problem might give a rather inconclusive result for DES unless you are really clever about it.)

(c) This is an extension of parts (a) and (b) above. Estimate the number of ciphertext blocks M' such that *exactly one* key maps M to M', exactly two keys, *etc.* (HINT: This uses the Poisson distribution.)

(d) Parts (a)-(c) should tell you that if you pick a plaintext block at random, there are likely to be many pairs of distinct keys that encrypt it to the same ciphertext block. Suppose I go looking for such a collision, testing keys at random until I discover a match. About how many keys will I have to test to have better than even odds of finding a colliding pair? (HINT: Do you know the problem about the chances of two people in a room having the same birthday?)

(e) What we've been showing is that although the function

$$M \mapsto E_K(M)$$

is one-to-one for each fixed key K, the function

$$K \mapsto E_K(M)$$

for a fixed is block M is almost surely not, at least not if we want our keys to behave like random permutations of the set of plaintext blocks. This may come as a bit of a surprise after the one-time pad. Estimate (do this just for AES and BabyBlock) the probability that this map is one-to-one. (HINT: Stirling's formula for $n!$). The answer, as you might expect after all this, is 'very small'. But is it very small like one in a million, or very small like one in a googol, or like one in a googolplex?

2 Computer Problems.

Problems 4,5, and 6 (as well as 7 and 9) refer to the base 64-encoded ciphertexts in the accompanying page on the course website. These were produced by representing

ASCII texts as sequences of bytes, and encrypting them with the baby block cipher using different modes of operation. See the accompanying notes for the description of the cipher, and the code for block encryption and decryption, and for encryption under the different modes of operation. Your job in each case is to decrypt the ciphertext and produce the original ASCII plaintext. It may be that the number of bytes in the original plaintext is odd: when this happens, the plaintext is padded with a 0 byte so that it fills an entire block. In these cases, the ciphertext will be one byte longer than the plaintext.

ECB Decryption.*(8 points)* The plaintext for Problem 4 was encrypted using the baby block cipher in ECB mode with key 12345. Decrypt it. You should write a function `decrypt_ecb(key,ciphertext)` that takes as parameters an integer key and the base64-encoding of a ciphertext (in that order) and returns the decryption as an ASCII string. You should apply your function to the given ciphertext to obtain the result.

5. CBC Decryption.*(12 points)*The plaintext for Problem 5 was encrypted using the baby block cipher in CBC mode with key 12345. The first two bytes of the displayed ciphertext are the initialization vector, and not actual ciphertext. You should write a function `decrypt_cbc(key,ciphertext)` that takes as parameters an integer key and the base64 encoding of a ciphertext, whose first two bytes are the IV, and returns the decryption as an ASCII string. You should apply your function to the given ciphertext to obtain the result.

6. CTR Decryption.*(10 points)* The plaintext for Problem 6 was encrypted using CTR mode with key 12345. The first two bytes of the displayed ciphertext are the initial value of the counter, and not actual ciphertext. You should write a function `decrypt_ctr(key,ciphertext)` that takes as parameters an integer key and the base64 encoding of a ciphertext, whose first two bytes are the the initial value of the counter, and returns the decryption as an ASCII string. You should apply your function to the given ciphertext to obtain the result. Remember that in CTR mode, decryption and encryption are the same.

7. Brute-force Attack.*(15 points)* The plaintext for this problem was encrypted using ECB mode. Find the plaintext by trying out all keys. (We would usually do this with a known-plaintext attack, so I would have to tell you a couple of plaintext blocks. Instead, I will tell you that the plaintext is ordinary ASCII. Since ASCII bytes have values in the range 0 to 127 and random bytes range from 0 to 255, there is little possibility that more than one key will decrypt the ciphertext to something consisting only of ASCII characters.) You should present your solution as a function `decrypt_brute(ciphertext)` that takes as a parameter the base64-encrypted ciphertext. A good strategy is to check the first n characters of plaintext for each choice of key and see if they are all in the range 0 to 127. You may need to experiment to find a value of n that is not too large, but that nonetheless leads to a unique key. Your function should return the plaintext as an ASCII string.

8 Collision Statistics. *(15 points)* This problem complements Problem 3(c) in the written section, and it is probably best that you work that problem if you're going to do this one.. Write a function `collision_statistics(M)` that takes as input a plaintext block M (an integer between 0 and 65535), encrypts it under all possible keys,

and returns a list or a dictionary telling how many ciphertext blocks are the encryption of M under exactly 0 keys, 1 key, 2, keys, etc. (You'll need to go up to about 9 or 10.) In other words, if the first entry in your table is 21000, it means that there are 21000 blocks M' such that there are NO keys K such that $E(K, M) = M'$. The next entry in the table tells how many ciphertexts are mapped to by a unique key, then how many by exactly two keys, *etc.*

Execute for various values of M, and compare to the results predicted by 3(c). This tells you in some respects how closely the statistics of the cipher resemble what one would get for a randomly selected permutation.

Now repeat the experiment but restrict the number of rounds to 2, and compare the results.

9. Meet-in-the-middle Attack *(40 points.)* The plaintext for this problem was encrypted using CBC mode, but the block encryption function is

$$M \mapsto E_{K_1}(E_{K_2}(M)).$$

The first four characters of plaintext are 'We h'. Find the plaintext by mounting the meet-in-the middle attack as described in both the notes and the textbook.