Midterm Questions from Previous Times the Course was taught

CS381-Cryptography

February 20, 2017

1 2012

Useful Items

$||$ denotes concatenation

$\oplus$ denotes exclusive-or

$$2^{10} \approx 10^3 = 1000,$$

and likewise for any (smallish) positive integer k,

$$2^{10\times k} \approx 10^{3\times k},$$

so, for instance, 2^{30} is about 10^9, or one billion. (In fact, a ‘gigabyte’ is not one billion bytes, but 2^{30} bytes.)

1.1 Exhaustive search attack against symmetric block cipher

Suppose we have a block cipher with a 64-bit key and an 64-bit block size. As usual, we assume that the encryption algorithm E is known to the eavesdropper Eve, but that Eve does not know the key that was used to encrypt the messages she intercepts. We saw in class that it is usually sufficient for Eve to have two ciphertext blocks C_1, C_2 for which she knows the corresponding plaintext blocks P_1, P_2 in order to recover the key through a brute-force attack. (The reason, in a nutshell, is this: While it is quite likely that two different keys for an ideal block cipher encrypt P_1 to C_1, it is extremely unlikely that two different keys encrypt both P_1 to C_1 and P_2 to C_2.)

(a) What kind of attack (apart from 'brute force') is this? (i.e., is it ciphertext-only? chosen plaintext? Something else?)

(b) How many encryptions of blocks will be performed carrying out this attack? (The number will vary—you might get lucky and discover the key very quickly—but what should you expect roughly?)

(c) Is this attack feasible? That is, is this attack (i) possible to carry out over the course of a day or so on a laptop computer? (ii) within the realm of possibility if one harnesses massive computational resources, including distributed processing and dedicated hardware? (iii) completely outside the realm of possibility?

(d) Suppose the block cipher is constructed this way: Both the plaintext block P and the key K are split into two 32-bit subblocks, and each half of the plaintext is encrypted separately with the corresponding half of the key, using a 32-bit block cipher E'. That is,

$$P = P' || P''$$
$$K = K' || K''$$
$$E(K, P) = E'(K', P') || E'(K'', P'').$$

(See Figure 1.) Does this structure change the feasibility of a brute-force attack? Explain.

1.2 Block cipher modes of operation

Parts (a) and (b) refer to a 4-bit block cipher with a fixed key K. The encryption function E_K is given by the table on the last page. In part (c), the key is unknown. The problems refer to CBC and CTR mode. For your convenience, block diagrams for these modes of operation are included with the exam. Show all your calculations carefully.

(a) A 2-block plaintext is encrypted using CBC mode with the accompanying table. The three blocks 0101 1110 0010 are received. The first block is the initialization vector. Find the **second plaintext block**.

(b) A 2-block plaintext is encrypted using CTR mode with the accompanying table. The three blocks 0101 1110 0010 are received (the same as in part (a)). The first block is the initial setting CTR of the counter, and the subsequent values $CTR+1, CTR+2$ are used to produce the pad. Find the **second plaintext block**.

(c) A 2-block plaintext is encrypted using CTR mode. (The accompanying table is **not** relevant here.) The three blocks 0101 1110 0010 are received. You do not know the key, but later you learn that the second plaintext block of the original message was 1111. You then intercept a new encrypted message 0110 0001 0001. What is the **first plaintext block**?

(d) The attack in (c) says something important about the proper use of CTR mode, regardless of block size. What is it?

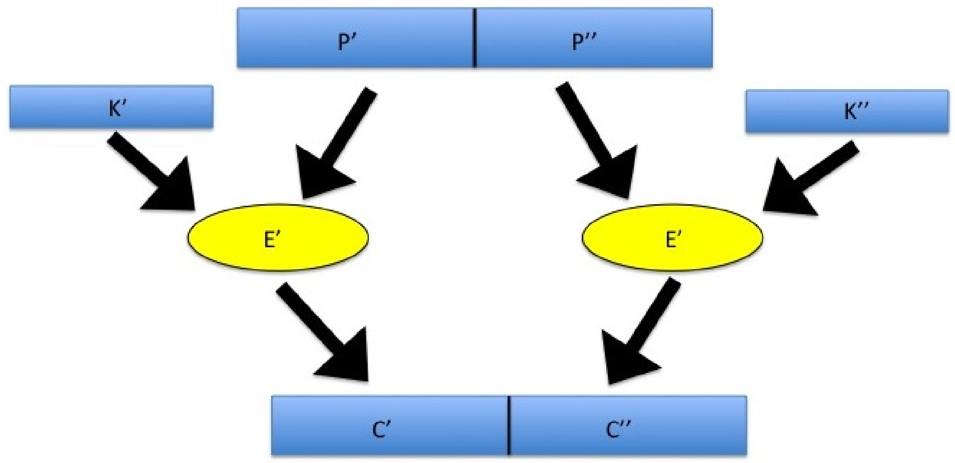

Figure 1: *The structure of the block cipher in Problem 1(d). The two halves of the plaintext block are encrypted separately, each with half the key.*

2 2014

Useful Items

$\oplus$ denotes exclusive-or, applied either to individual bits or to sequences of bits. The same operation in Python is denoted ˆ.

$$2^{10} \approx 10^3 = 1000,$$

and likewise for any (smallish) positive integer k,

$$2^{10\times k} \approx 10^{3\times k},$$

so, for instance, 2^{30} is about 10^9, or one billion. (In fact, a ‘gigabyte’ is not one billion bytes, but 2^{30} bytes.)

Security Benchmarks

- Through use of a custom hardware unit that costs about $10,000 to build, a 56-bit DES key can be recovered by exhaustive search in about a week.

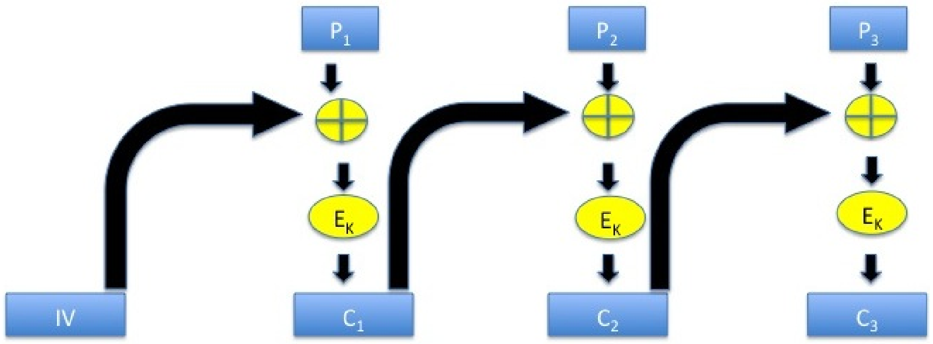

Figure 2: *CBC Encryption.*

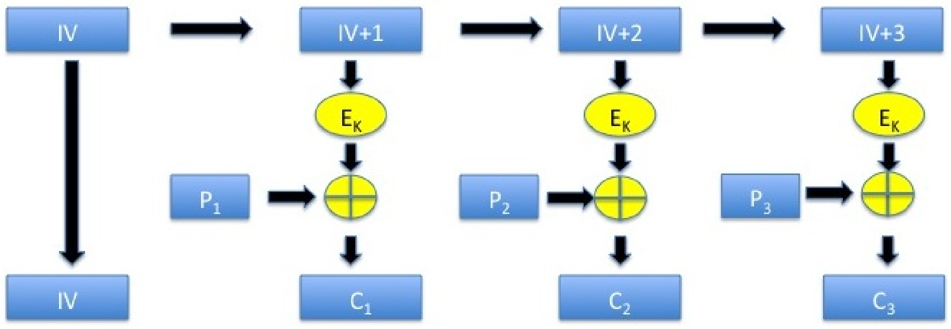

Figure 3: *CTR Encryption.*

P	**E(K,P)**
0000	0011
0001	0001
0010	0111
0011	1100
0100	1001
0101	1111
0110	0000
0111	1010
1000	1101
1001	0010
1010	0101
1011	0110
1100	0100
1101	1000
1110	1011
1111	1110

Table 1: *Block encryption function for the block cipher in 2(a,b), with respect to some fixed key K.*

- The record for factoring a product of two primes is a number of 768 decimal digits. The calculation, distributed over hundreds of processors, took two years.
- There are about 3×10^7 (30 million) seconds in a year.

Language statistics There are about 100 printable ASCII characters. In ordinary ASCII text in English (including punctuation and capitalization) the most common character by far is a space, accounting for about 18% of all characters, distantly followed by lower-case 'e', which accounts for about 9% of all characters. Note that this is different from the letter-frequency statistics we used earlier, in which we considered only the 26 letters and ignored punctuation and capitalization.

2.1 Insecure Encryption in a very old version of Microsoft Word

The earliest version of encryption in Microsoft Word (present in Windows 95) was to have the user select a password of printable characters, and then simply XOR repeated copies of the password with the plaintext. For instance, if the password was

```
hEre_15a*p@5SwD
```

which has 16 characters, the the first byte of ciphertext will be the first character of plaintext XORed with 'h', the second byte of ciphertext will be the second character of plaintext XOR'd with 'E', *etc.* Once the seventeenth charcacter of plaintext is reached, we return to the first character of the password and XOR with 'h', etc. In general, if the three strings are viewed as lists of bytes, we have, in Python:

```
ciphertext[i]=plaintext[i]^password[i % 16]
```

Assume all passwords have exactly 16 characters.

(a) How many different passwords are there? Is it feasible to carry out an exhaustive-search attack that recovers the plaintext from ciphertext by trying out all possible keys? There are three possible answers here: (i) it can be done on my laptop by a program that runs in under an hour; (ii) it can be done with specialized hardware and/or a widely distributed network of collaborating computers in under a year; (iii) it cannot be done with available computing power in under a century. Pick the best answer and justify with a rough calculation.

(b) Assume that the original plaintext is at least several thousand characters long, in ASCII characters, consisting of ordinary English sentences with spacing, punctuation, and the like. You possess the encrypted version of the file. Describe an efficient procedure for recovering the *first character of the password.* (The same procedure, applied fifteen additional times, should recover the entire password, which can then be used to decrypt the file.)

(c) Now suppose that documents are *not* ordinary English, but rather random, uniformly distributed sequences of characters. You have a brief plaintext document `a.doc`, its encryption `a_enc.doc`, and the encryption `b_enc.doc` of a long second document. Describe an efficient procedure for recovering the plaintext `b.doc`. (Assume the user, like most users, employed the same password for each document he encrypted.)

2.2 Block Cipher in Cipher Feedback Mode

The accompanying diagram shows a block cipher mode of operation called *cipher feedback mode* (CFB). Like most of the other modes we have seen, there is an initial *IV* block that is generated at random and sent in the clear. Like CTR mode, this mode uses a block cipher as a stream cipher, XORing the plaintext with the keystream. Unlike CTR mode, the keystream cannot be generated completely from the IV and the key, since each keystream block depends on the preceding plaintext blocks.

Here are equations and a diagram describing the encryption algorithm. The inputs are the plaintext blocks $P_1, P_2, \ldots,$ along with the IV block and the key K. E is the block encryption function. The outputs are $C_0, C_1, C_2, \ldots,$ where $C_0 = IV$.

$$C_0 = IV,$$
$$C_i = P_i \oplus E(K, C_{i-1}),$$

if $i > 0$.

(a) Write equations for the decryption function, giving the plaintext blocks $P_1, P_2, \ldots$ in terms of $C_0, C_1, \ldots$.

(b) Suppose this encryption mode is used with a block cipher with a four-bit block size. The shared secret key is K, and the block encryption function E_K is given in Table 1. You receive a message

$$0000 \quad 1111 \quad 0000.$$

Determine the plaintext message. (Remember that the first block of the received message is the IV, so that there will be two plaintext blocks, not 3.)

(c) This problem concerns what happens when you re-use the initialization vector in CFB mode. You receive two new messages that were encrypted with a different key K' (so that Table 1 is no longer relevant). The messages are

$$0000 \quad 1111 \quad 0000$$

$$0000 \quad 0000 \quad 1111.$$

Let P_1P_2 and $P_1'P_2'$ denote the corresponding plaintext messages. Describe *all* the information you can obtain about the blocks P_i and P_i' from this description.

M	**E(K,M)**
0000	0011
0001	0001
0010	0111
0011	1100
0100	1001
0101	1111
0110	0000
0111	1010
1000	1101
1001	0010
1010	0101
1011	0110
1100	0100
1101	1000
1110	1011
1111	1110

Table 2: *Block encryption function for the block cipher in Problem 2, with respect to some fixed key K.*