Symmetric Cryptography

by Daniel Halperin (and with Tadayoshi Kohno)

1. Key Exchange (adapted from Ferguson, Schneier, and Kohno, Problem 2.3). Consider a group of 30 people in a room who wish to be able to establish pair-wise secure communications in the future. How many keys need to be exchanged in total:

Using symmetric cryptography?

Using public key cryptography?

2. Signatures (FSK, 2.4). Suppose Bob receives a message signed using a digital signature scheme with Alice's secret signing key. Does this prove that Alice saw the message in question and chose to sign it?

3. Key Strength and Brute Force Attacks (FSK, 3.8). Suppose you have a processor that can perform a single DES encryption or decryption operation in 2^{-26} seconds, and a plaintext-ciphertext pair encrypted under an unknown key. How many hours would it take, on average, to find that DES key, using an exhaustive approach:

With a single processor?

With a collection of 2^{14} processors?

4. Misusing Stream Ciphers, Part 1 (FSK, 4.3). Suppose you, as an attacker, observe the following 32-byte ciphertext C_1 (in hex)

```
46 64 DC 06 97 BB FE 69 33 07 15 07 9B A6 C2 3D
2B 84 DE 4F 90 8D 7D 34 AA CE 96 8B 64 F3 DF 75
```

and the following 32-byte ciphertext C_2 (also in hex)

```
51 7E CC 05 C3 BD EA 3B 33 57 0E 1B D8 97 D5 30
7B D0 91 6B 8D 82 6B 35 B7 8B BB 8D 74 E2 C7 3B.
```

Suppose you know these ciphertexts were generated using CTR mode with the same nonce. (The nonce is implicit, so it is not included in the ciphertext.) You also know that the plaintext P_1 corresponding to C_1 is

```
43 72 79 70 74 6F 67 72 61 70 68 79 20 43 72 79
70 74 6F 67 72 61 70 68 79 20 43 72 79 70 74 6F.
```

What information, if any, can you infer about the plaintext P_2 corresponding to C_2?

5. Misusing Stream Ciphers, Part 2. In class, we mentioned that you might not even need a single plaintext-ciphertext pair in order to break a code in which keystream was reused. In this problem, you will demonstrate such an attack.

Suppose that you intercept the following four 32-byte ciphertexts (in hex):

```
C1: 53 D1 DD F4 6B 8B 8B ED 8B 0D 51 FD 01 C9 0D 49
    09 3B 1E A6 35 18 79 F2 09 78 1D 54 09 09 27 67

C2: 0C 05 DF 99 7D DB 04 64 90 3E 49 F7 93 60 C3 83
    95 0C FE 18 89 5D 22 C7 D4 C4 6E 16 46 E1 8B A9

C3: 70 F1 F5 D3 74 C6 B7 F1 8D 4C 45 F7 01 C4 17 0D
    7A 00 21 97 47 5F 78 F8 0A 2B 4E 45 02 12 31 32

C4: 21 02 CF D7 2A D8 0F 30 D9 24 06 B4 93 64 D1 82
    DB 43 E4 57 CC 41 2E D2 DA 8A 22 1F 4A FF 8A B7
```

Suppose further that your spies give you reason to believe that the plaintexts are simply ASCII-encoded sentences using standard language (e.g., words found in the CSE484 `/usr/share/dict/words` file, although not necessarily lowercase) and punctuation (i.e., space, comma, period, exclamation point, and question mark).

(a) It turns out that the four ciphertexts above contain two pairs of messages each encrypted with the same reused keystream. Which ciphertexts were encrypted with the same keystream, and how can you detect this?

(b) Recover the four plaintext messages. You'll probably want to write a program to do so; please submit your code along with this assignment. (To reduce the search space, you may assume that the longest word in a message is at most 7 letters long.)

6. CBC Collisions (FSK, 4.6). Let P_1,P_2 be a message that is two blocks long, and let Q_1 be a message that is one block long. Let C_0,C_1,C_2 be the encryption of the first plaintext using CBC mode with a random IV and a random key, and let D_0,D_1 be the encryption of Q_1 using CBC mode with a different, random IV and the same key. Suppose an attacker knows the first message P_1,P_2 and has intercepted both ciphertexts. Further suppose that, by random chance, $D_1=C_2$. Show that the attacker can compute Q_1.

7. Reduced-Round AES. In class, we discussed the rationale behind using multiple rounds in a block cipher encryption scheme such as AES or DES. Your assignment in this problem is to crack a simplified, 1-round version of 128-bit AES.

Recall that AES-128 has a block size of 128 bits, or 16 bytes, and that it treats its plaintext input, keys, intermediate state, and ciphertext output as 4x4 arrays of bytes. For instance, if the plaintext bytes were labeled *ABCDEFGHIJKLMNOP*, then the plaintext would be represented as:

A	*E*	*I*	*M*
B	*F*	*J*	*N*
C	*G*	*K*	*O*
D	*H*	*L*	*P*

Recall that the pseudocode for AES-128 is as follows (the components of this process are also fairly well explained on Wikipedia at http://en.wikipedia.org/wiki/Advanced_Encryption_Standard):

```
Expand the key into 11 "round keys" $Key_1$, $Key_2$, ..., $Key_{11}$
Set the current state to the plaintext
for n = 1 to 10
    XOR the current state with $Key_n$
    SubBytes substitutes every byte of the state using the AES S-box
    ShiftRows rotates each row to the right by 0, 1, 2, or 3 cells respectively
    if n < 10
        MixColumns mixes the bytes in each column using an affine linear
            transformation, i.e., given input column (A,N,K,H) it outputs
            a new column whose four bytes are:
            (2A+3N+K+H, A+2N+3K+H, A+N+2K+3H, 3A+N+K+2H)
end for
XOR the state with $Key_{11}$
```

A few notes about the above:

***Key*$_1$** in fact equals the original secret key.

Second, the math in the ***MixColumn*** operation is performed in a particular field such that every byte has a multiplicative inverse. We implement this as follows:

2A is implemented as **A << 1**.

A+B is implemented as **A⊕B**.

3A is implemented as **2A+A**, or **(A<<1)⊕A**.

Finally, when a multiplication (i.e., a shift) results in a sum larger than 0xFF (255), we XOR the result by 0x11B to get back to a single byte.

Your task is to crack a simplified, single-round version of AES-128 with the following pseudocode:

Generate Key2 = OnesComplement(ShiftRows(Key))
Set the current state to the Plaintext
XOR the current state with the Key
SubBytes on the current state
ShiftRows on the current state
MixColumns on the current state
Set the ciphertext to the current state XOR Key2

We provide sample C/C++ code (**aes1.c** and **aes.h**) that will encrypt a plaintext with a given key. You should write a program that can find the key given the following plaintext-ciphertext pair:

```
  Plaintext: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70
 Ciphertext: 25 9d d5 5f af 12 77 a9 ca 2e 52 b5 6a 07 6d 01
```

Note: In this simplified 1-round version of AES-128, we have tweaked the AES key schedule (the algorithm that generates the *Key_1,Key_2,...,Key_{11}*) to generate Key2 in a way that makes the problem easier to solve. However, with a few hours of searching, we were (*Dan was*) unable to generate a tweak that did not lead to ***multiple solutions***. That is, there are several different (but closely related) keys that will give you the same ciphertext-plaintext pair. This may come up in your testing.

(a) What is the computational complexity of your technique to recover a working key?
(b) How long did it take to find a single solution?
(c) How many solutions are there? Can you tell which is the key we chose?

Hint: *during testing, you should be able to do some back-of-the-envelope calculations of your code's runtime. It should definitely take less than a few hours to find all solutions. Our solution takes around 40 minutes to find all keys on a single core, 64-bit Intel CPU.*

Hint 2: *note that you can generate your own Plaintext/Ciphertext/Key pairs using our provides aes1.c. To create your own ptext.bin and ctext.bin, we suggest using a binary file editor such as* **`bvi`** *or* **`hexedit`**.

Submit your code to Catalyst along with this homework assignment.