6.857 Computer and Network Security: Lecture Notes (Draft)

Part 4 : Block and Stream Ciphers

Version: September 26, 1998

Conventional conditionally (or computationally) secure cryptography uses a shared secret key of limited length to provide security against an opponent with limited computational resources. These schemes are secure because it is (believed to be) computationally infeasible, rather than information-theoretically impossible, for an adversary to determine the key or message.

Computationally secure ciphers come in two main flavors: *block ciphers* and *stream ciphers*. Block ciphers work on the data one block at a time, where a block may be 64 or 128 bits, say. A stream cipher is modelled after the one-time pad: a pseudo-random stream of bits is generated and xor-ed with the message.

Since we don't know many fundamental facts about computational difficulty (such as whether $P = NP$), we don't yet have the tools to prove unconditionally that cryptosystems are computationally secure. The best that can be done from a theoretical point of view is to prove that a system is secure, based on the *assumption* that some well-defined problem (such as factoring large integers) is hard. Such assumptions may turn out to be false (indeed, it may be that $P = NP$), and so such "proofs" of security are really only reductions; the fundamental proofs are yet to be found.

When one tries to assess the strength of an encryption scheme, one needs to be precise about what sort of attack an adversary is allowed to mount. These can be categorized first of all by the kind of information the adversary is allowed to have about the cryptosystem. For example, it is generally considered appropriate to assume that the adversary knows everything about the design of the system except the secret key. In addition, the adversary is normally allowed to see some input (plaintext), output (ciphertext), or input/output (plaintext/ciphertext) pairs for the cryptosystem being attacked. Such data is normally available in one of the following ways:

- as *known ciphertext* only, such as might be obtained by an eavesdropper listening to an encrypted radio broadcast,
- as *known plaintext/ciphertext pairs*, such as might be obtained by an eavesdropper who later obtains somehow the corresponding plaintext,
- as *chosen plaintext* (with corresponding ciphertext) or as *chosen ciphertext* (with corresponding plaintext), as for example might be obtained by a corrupted insider who can control what input is fed to the cryptographic device containing the unknown key, and also controls whether it is encrypting or decrypting. The insider doesn't know the key; that still has to be inferred from the data obtained.

In some cases there may be additional information known about the way the cryptosystem is used;

for example, that the encryption key is changed regularly in a simple way (such as by adding one), yield *related keys*.

The second way in which attacks can be categorized is by the goal of the adversary; when does he "succeed"? Some natural goals include:

- inferring the secret key,
- being able to decrypt additional ciphertext (perhaps he can do this without knowing the key),
- being able to distinguish the output pad from truly random bits (for a stream cipher),
- being able to distinguish the encryptions of two messages supplied by the adversary himself, with probability better than 1/2.

A proof of security would show that it is computational hard (perhaps as hard as solving some well-known hard problem) for an adversary to achieve his goal, given the information he is allowed to obtain. Such proofs are often hard to come by, particularly for "practical" cryptosystems that must be efficient as well as secure. Thus, we often assess security by "trial by fire." After a cryptosystem has been tested for a while without breaking, we begin to place some trust in it. Sometimes we are still surprised.

1 Block Ciphers

A block cipher is an algorithm that takes as input a fixed-length message block (e.g. 64 or 128 bits) and a key (not necessarily the same length), and produces as output a ciphertext block of the same length as the original message block. The key is reused for many different message blocks, so it needs to be combined with the message in a complex way.

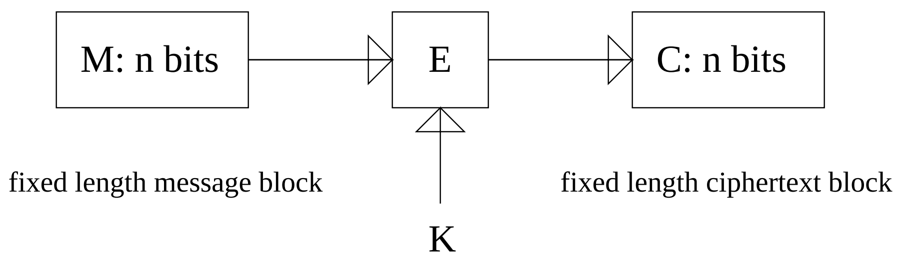

Figure 1: An n-bit block cipher

Feistel ciphers are a class of block ciphers constructed in certain way. They are iterative ciphers; they repeat a given operation several times, in *rounds*. For such a cipher, the key K is used to produce round keys $\{K_1, K_2, K_3, ...K_n\}$. The message M is initially divided into left and right halves L and R. For each of n rounds, the operation

$$L \leftarrow L \oplus f(K_i, R)$$

is executed, followed by an exchange of L and R. After n rounds, the final left and right halves are swapped and concatenated to form the ciphertext.

Decrypting a Feistel cipher is straightforward, as it is possible with the key to recompute the round keys, and then to recompute each $f(R, K_i)$, in reverse order. Undoing the xor is easy, since $L \oplus f \oplus f = L$. Note that the function f does not need to be invertible. The final swap allows the encryption hardware to also be used for decryption, only reversing the order of the round keys.

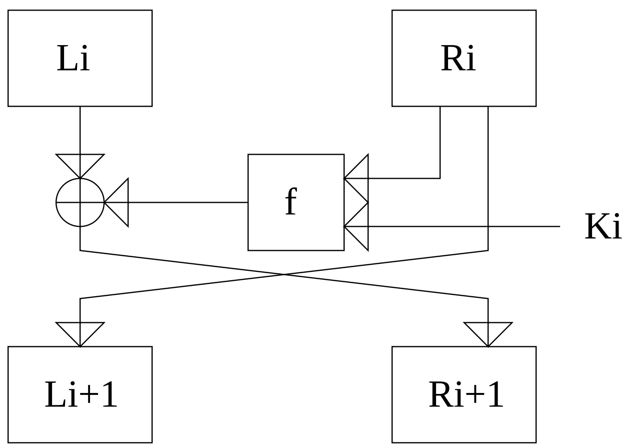

Figure 2: One round of a Feistel cipher

A particular Feistel cipher is specified by saying how the encryption key is turned into round keys, and how the function $f(R, K_i)$ is computed. The design of a good function $f(R, K_i)$ is part art, part science.

2 Modes of Operation of Block Ciphers

A block cipher, per se, only handles a fixed length input block. In this section we study block cipher "modes of operation," which typically means ways to use a block cipher to encrypt messages of varying length, particularly those that are longer than the block size of the cipher.

Electronic Codebook Mode (ECB) is using a block cipher as-is—padding a message to a multiple of block length, breaking it into blocks, and encrypting it. This is rarely used in practice. Two matching blocks of of plaintext would produce matching blocks of ciphertext, and 64 bits is a small enough chunk that building meaningful messages out of known plaintext/ciphertext pairs would be a real danger. A one bit transmission error in this mode spoils a whole message block.

Counter Mode uses a block cipher as a pseudo-random generator. In this mode, we use the block cipher as a primitive to produce a stream cipher. Rather than encrypting the message text, a known string of numbers (possibly just $\{1, 2, 3, ...\}$) is encrypted in ECB mode, producing a string of output blocks $\{R_1, R_2, R_3, ...\}$, that are xor-ed with the message blocks to produce the ciphertext:

$$C_i \leftarrow M_i \oplus E(K, i) \ .$$

Counter mode has the advantages that the ciphertext transmitted can be the same length as the message, no padding needed—extra characters at the end of the last random block are just dropped—and a one bit transmission error spoils only one bit.

Output Feedback Mode, like counter mode, produces a string of "random" blocks that are xor-ed with the message to produce the ciphertext. It also uses the block cipher as a primitive to produce a stream cipher. Instead of feeding the cipher a fixed sequence of numbers, however, each random block is used as input to the block cipher to generate the next. (An initialization block is needed.) This, too, needs no padding and loses only one bit to a one-bit transmission error. A possible disadvantage of output feedback mode is that the expected period is about $2^{n/2}$, where n is the block size, compared to a period of 2^n for counter mode. (This is due to the "birthday paradox.")

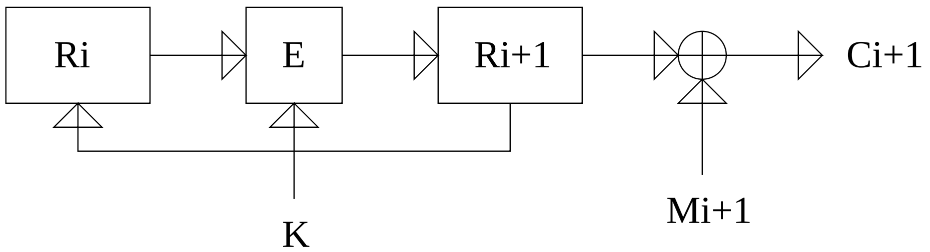

Figure 3: Output Feedback Mode

Cipher Block Chaining (CBC) xor-s each message block with the ciphertext of the previous block before encrypting it; an initialization vector, possibly public, is xor-ed with the first message block before encryption. This does require padding the message to a multiple of the block size, but cipher block stealing (below) can be used to create a ciphertext of the same size as the original message. A one-bit error in transmission will ruin one block and cause a one-bit error in decrypting the next.

"Cipher block stealing" is a CBC variation that efficiently handles messages that are not a multiple of the block size. It alters the last two ciphertext blocks of a message generated with CBC to yield a ciphertext just as long as the message text. The message is padded with zeros and CBC run as usual. Then the last ciphertext block is transmitted as the second to last, followed by only as many bits of the second-to-last (generated) as there are non-padding bits in the last message block. To recover the message, the last full ciphertext block is decrypted, yielding the second-to-last block generated xor-ed with the final message block—but since the message was padded with zeros, the final bits are the unmodified bits that were dropped from the second-to-last block (generated) before it was transmitted. The fractional transmitted block is combined with the decrypted final block to recover the final message block, and to decrypt to recover the second-to-last message block.

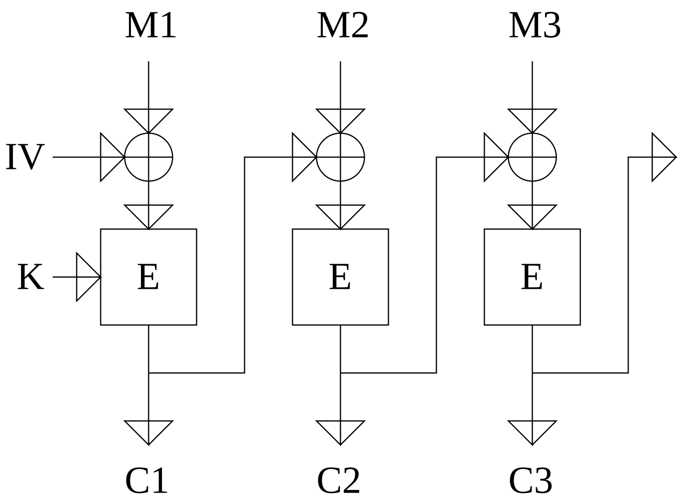

Figure 4: Cipher Block Chaining

3 DES

DES, the national Data Encryption Standard, is a Feistel-type cipher of sixteen rounds, using a 64 bit message block and a 56 bit key. The round keys are generated by selecting 48 bits out of the 56. The function $f(R, K_i)$ expands R from 32 to 48 bits, xor-s R with K_i, breaks the resulting sequence into six bit chunks, looks up each six bit chunk in a table of "S-boxes" to get back four bits, scrambles the resulting 32 bits, and returns the result. The details can be found in any standard text on cryptography, such as Menezes et al. (1997).

One of many functions of the S-boxes is to magnify the effect of changes in the plaintext on the ciphertext.

The NSA was consulted during the design of DES; the motivation and effect of the changes they proposed has been debated. Current differential analysis suggests that their changes to the S-boxes strengthened DES, but the NSA pushed for the 56 bit key length—down from 128 bits that had been suggested.

4 Attacks on DES

There are three major attacks on DES: brute-force search of the key space, a differential attack, and a linear attack.

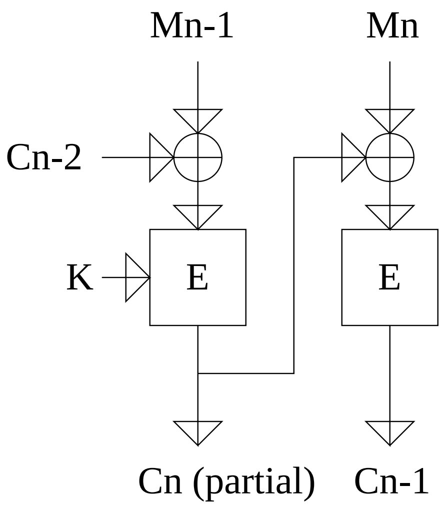

Figure 5: Cipher Block Stealing

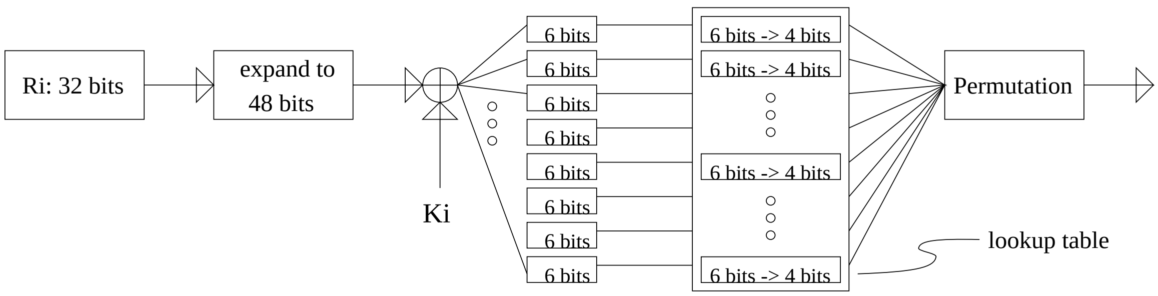

Figure 6: DES: $f(R, K_i)$ for one round

4.1 Brute Force

Here we assume that the attacker has available a matching message-ciphertext pair, and that he is willing to try all 2^{56} possible keys to find the correct one.

How expensive (in time and money) is this?

In 1977, Diffie and Hellman postulated that they could build a machine for $20 million dollars which could find the key in 10 hours. They extrapolated that in 1987, they could build a similar machine for $200,000.

In 1993, Michael Wiener designed a chip that could do 5,000,000 encryptions/second for $10.50/chip. He designed a machine which uses 5760 chips in parallel to break the DES key in 1.5 days. With $1 million dollars, he could build a machine which could break the key in 3 hours. This design was carried out to a sufficient level of detail to be quite convincing, although the machine was (to my knowledge) never built.

In January of 1997, RSA Laboratories announced a series of "DES challenges" to stimulate actual efforts to do a full DES key-search on a challenge message. The first challenge was broken in June 1997, using spare cycles on machines scattered across the net.

Most recently, the Electronic Frontier Foundation, in conjuction with Cryptography Research, built a $250,000 machine that broken the latest challenge in 56 hours, winning a $10,000 RSA Labs prize.

The time to break encryption keys continues to decrease as processing speed increases. Approximately every 18 months, processing time doubles in speed. There have been about 13 doublings since 1977. To stay at the same level of security as 1977, DES needs to have 70 bits, and the number of bits must be able to increase as computing power grows. Unfortunately, the DES 56 bit key is of fixed length. See Blaze et al. for a discussion of appropriate key lengths.

4.2 Differential Cryptanalysis

Differential cryptanalysis is a chosen-plaintext attack. This method may not be too realistic in practice, since it requires that adversary to accumulate about 2^{47} plaintext/ciphertext pairs encrypted under the unknown key. The plaintexts are arranged in pairs, and the difference (xor) between two plaintexts in a pair is a fixed constant. The corresponding output differences are examined and used to infer the unknown key. This method to break the DES key was known when it was designed, so the S–boxes are designed to be amazingly resilient to this form of attack. Differential analysis was developed by Biham and Shamir (1991).

4.3 Linear Cryptanalysis

The linear method of breaking the DES encryption scheme relies on forming a linear approximation to the encryption key. It gathers statistical evidence about the key from many known plaintext/ciphertext pairs (it is a known-plaintext as opposed to a chosen-plaintext attack). It requires the encryption of about 2^{43} randomly generated plaintexts. This method was invented and implemented by Mitsuru Matsui, who found a DES key in about 50 hours.

5 Strengthening DES

DESX: A simple but effective way to strengthen DES is to have two new 64-bit keys, K_I and K_O, in addition to the standard 56-bit key K. Then we encrypt:

$$C = K_O \oplus DES(K, M \oplus K_I)$$

That is, we first xor the input message with K_I, then encrypt as usual with DES, and then xor the resulting output with K_O to get the ciphertext. Kilian and Rogaway (1996) have argued that DESX has an effective key length of $119 - t$, where the adversary is assumed to be able to collect 2^t matching input/output pairs.

Double DES: The Double DES encryption method cascades two DES boxes with different keys. One might think that this method is equivalent to using DES with a 112 bit key, but this is not true. It can be attacked with the "Meet in the Middle" attack which requires $O(2^{56})$ keys to be searched.

Two-key Triple DES: Let $DES(K, M)$ denote the DES–encryption of message M with key K. Let $DES^{-1}(K, M)$ denote the DES–decryption of message M with key K. Two-key Triple DES uses two keys K_1 and K_2 and encrypts a message M as follows:

$$C = DES(K_1, DES^{-1}(K_2, DES(K_1, M)))$$

We note that if $K_1 = K_2$, we get single DES, so Triple DES is backwards compatible. This is why the middle encryption uses DES in decryption mode.

This method can be attacked with 2^{56} chosen plaintexts and a similar amount of storage.

6 New Block Ciphers

There are many other block ciphers that have been proposed, such as IDEA (used in PGP), RC5, and Blowfish.

The U.S. government has begun a process to select a replacement for DES, to be called the AES (the Advanced Encryption Standard). See the NIST page. The AES will have 128-bit blocks and key sizes of 128, 192, or 256 bits. At the time of this writing, fifteen candidate block cipher algorithms were being considered: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, RIJNDAEL, SAFER+, SERPENT, and TWOFISH.

Here we present RC5, as it is particularly simple.

6.1 RC5

Rivest has recently designed another encryption method called RC5. (RC1 and RC3 were never published. RC2 (a block cipher) and RC4 (a stream cipher) are in widespread use, and RC6 (a block cipher, an elaboration of RC5) has been proposed as an AES candidate.)

First, there is a separate extension that expands the key to have as many words as rounds. Then, letting

L, R = halves of message
$S[1\ldots]$ = key information

it does the following operation 16 times: (rounds) RC5 is a Fiestel-type cipher, proposed by Rivest, in which the round function is

$$L \leftarrow ((L \oplus R) \lll R) \oplus K_i \ ,$$

where $x \lll y$ indicates rotation of x by an amount indicated in the least significant bit positions of y.

Note that on older machines, the time to left shift may be proportional to the value being left shifted by. This could leave RC5 vulnerable to a timing attack. However, most modern computers do this operation in a single cycle.

RC5 has the advantage that it is fast and easily parameterizable. It can use variable sized keys and a variable number of iterations. So far, it has held up against both linear and differential attack.

7 Stream Ciphers

A stream cipher is a pseudo-random number generator (occasionally called a pseudo-random bit generator when one wants to emphasize that a particular generator is outputting bits rather than general numbers). It is used to generate a long "pad" from a shorter "seed" (which is the shared secret key). To encrypt a message, the output is xor-ed with the message, just as for a one-time pad. As for a one-time pad, one has to be careful not to re-use the same output to encrypt a different message.

7.1 Blum-Blum-Shub

In 1982 Blum, Blum, and Shub proposed the following simple generator, which is also called the $x^2 \pmod{n}$ generator. Let n by the product of two large distinct prime numbers, both congruent to 3 modulo 4. The seed x_0 can be any non-zero quadratic residue modulo n (that is, $x_0 \equiv y * y \pmod{n}$ for some nonzero y modulo n). Then we can define sequence $\{x_i : i \geq 0\}$ by the recurrence:

$$x_{i+1} = x_i^2 \pmod{n} \ .$$

The output bit sequence $b_0, b_1, \ldots$ is defined by:

$$b_i = \mathrm{lsb}(x_i) \ ,$$

where $\mathrm{lsb}(x)$ is the least-significant bit of x.

This scheme produces an output that provably passes all polynomial-time statistical tests, assuming that factoring n is hard: the output of this generator should be indistinguishable from random bits. It is not terribly efficient, however.

7.2 RC4

A more practical stream ciphr is RC4 (meaning "Ron's Code 4"), a variable key-size stream cipher. It uses a time-varying S-box (an array of bytes): $S[0], S[1], ..., S[255]$. The entries of S are always a permutation of the numbers 0 through 255, and the initial permutation is a function of the variable-length key. RC4 uses two counters, i and j, initialized to zero.

To generate a random byte, do the following (all additions are mod 256):

```
i = i + 1
```

```
        j = j + S[i]
        swap (S[i], S[j])
        Output S[S[i] + S[j]]
```

The output byte is xor-ed with the plaintext to produce ciphertext or xor-ed with the ciphertext to produce plaintext. Encryption with RC4 is fast – about 10 times faster than DES.

Initializing the S-box is also easy. First, fill it linearly: $S[0] = 0, S[1] = 1, ..., S[255] = 255$. Then fill another 256-byte array with the key, repeating the key as necessary to fill the entire array: $K[0], K[1], ..., K[255]$. Set the index j to zero. Then:

```
for i = 0 to 255
    j = (j + S[i] + K[i]) mod 256
    swap (S[i], S[j])
```

RC4 produces a pseudo-random key (pad), which is what characterizes stream ciphers. According to RSA Data Security, Inc. the algorithm is immune to differential and linear cryptanalysis and is highly non-linear. The RC4 algorithm's state, consisting of S, i, and j, can be one of about $256! * 256^2 = 2^{1700}$ possible states. The S-box slowly evolves with use: i ensures that every element changes and j ensures that the elements change randomly.

RC4 is exportable as long as its key is shorter or equal to 40 bits. But given this restriction, it becomes vulnerable to brute force attacks. RC4 is currently being used in commercial products such as Lotus Notes, Apple's AOCE, and Oracle Secure SQL. It is also part of the Cellular Digital Packet Data specification.

8 References

Biham, Eli, and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems," *Journal of Cryptology* **4**,1 (1991), 3–72.

Blaze, Matt, Whitfield Diffie, Ronald L. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thompson, and Michael Wiener. "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security." Available on the web at `http://theory.lcs.mit.edu/~rivest/bsa-final-report.ps`

Blum, Lenore, Manuel Blum, and Michael Shub, "Comparison of two pseudo-random number generators," *Advances in Cryptology—Proceedings of Crypto '82*, (Plenum Press, 1983), 61–78.

Cryptography Research, Inc. "DES Key Search Project Information," Available on the web at `http://www.cryptography.com/des/index.html`.

Feistel, Horst, William A. Notz, and J. Lynn, Smith. "Some Cryptographic Techniques for Machine-to-Machine Data Communications," *Proc. IEEE*, **63**, 11 (Nov. 1975), 1545–1554.

Kilian, Joe, and Phillip Rogaway. "How to Protect DES Against Exhaustive Key Search," *Advances in Cryptology–CRYPTO '96*, (Springer, 1996), 252–267.

Menezes, Alfred, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography* (CRC, 1997).

NIST. *AES: A Crypto Algorithm for the Twenty-First Century...* `http://www.nist.gov/aes/`

Rivest, Ronald L. "The RC5 Encryption Algorithm." *Fast Software Encryption* (Springer Lecture Notes in Computer Science No. 1008, 1995), 86–96. (Or see Rivest's publication page: `http://theory.lcs.mit.edu/~rivest/publications.html`.)