6.857 Computer and Network Security September 19, 2002

Lecture Notes 5 : Block Ciphers

Lecturer: Ron Rivest *Scribe: Boufounos/Castagnola/Michalakis*

[These notes come from Fall 2001. Check with students' notes for new topics brought up in 2002.]

Outline

1. Block Ciphers: DES, RC5, AES
2. Modes of Operation: ECB, Counter, CBC, OCB

1 Block Ciphers

Historically cryptography evolved differently from number theory. That is why many ciphers do not use number theory heavily. The goal of block ciphers is to encrypt a message block of fixed length. One issue that arises is handling variable length blocks. This can be dealt with using different modes of operation, which we will discuss in the next section. In this lecture we will talk about three kinds of block ciphers:

- DES—Created in 1976 to be a government standard.
- RC5—Professor Rivest invented that. He likes it because it is simple.
- AES—The new government standard. Created last year (200).

1.1 DES—*(Data Encryption Standard, 1976)*

DES was adopted by the U.S. government in 1976. It predates the number theory research and results we have seen in the course. Its publication was an interesting event: several people thought is should be a government secret for national security reasons. However, the government predicted that by the time it would be widely adopted, it would have been weak enough for them to be able to break it with strong enough computers, if there was a national security need.

This is why the key length of 56-bits was the most controversial decision. The key length of the standard was probably a result of compromise between the National Bureau of Standards (NBS) and the National Security Agency (NSA). The former wanted stronger key lengths, while the latter wanted weaker ones.

DES was developed by IBM, the only company with cryptographic expertise at the time. Their proposal was the only serious one in the competition. Apparently, NSA also played a role in the development of the standard, helping IBM with some confidential research the NSA had done previously.

DES is based on *Feistel* structures. These are simple, easy to compute, weak cryptographic modules, which are iterated to make them stronger. Specifically, DES is using 16 rounds of the structure shown in Figure 1. The message is initially broken to two halves, L_0 and R_0. It is then passed through the structure to get the next round of left and right values L_1 and R_1.

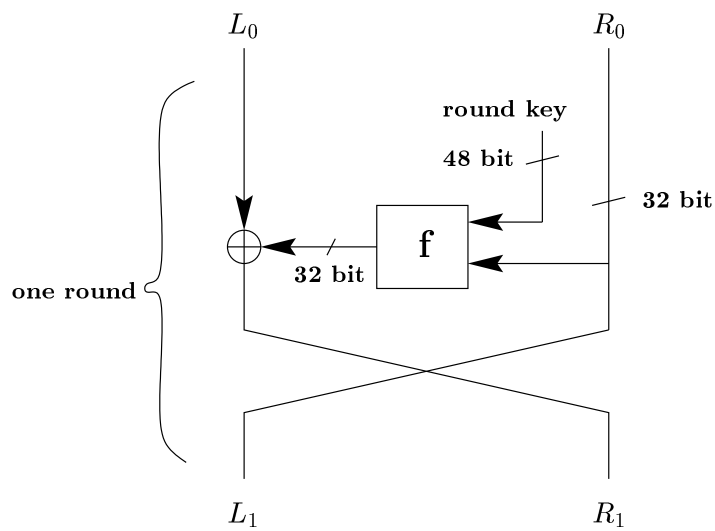

Figure 1: The Feistel block for DES.

The 48-bit round key changes at each round, according to a predetermined scheme, shown in Figure 2. The original key is broken to two 28-bit pieces, which are inserted into two cyclic shift registers. The registers shifts their values on every round, and 48 bits are selected to make the round key.

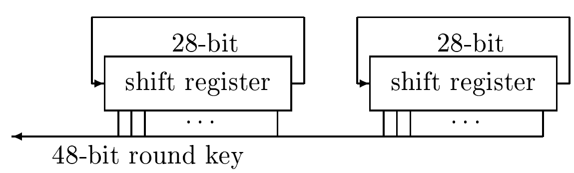

Figure 2: Key generation for DES

Finally, the function f is shown in Figure 3. The input R undergoes an expansion to 48-bits and is XOR-ed with the round key. The resulting number is then broken to 8 sets of 8-bit values. These are passed through 8 *S-boxes*, the main sources of nonlinearity of the algorithm. The S-boxes are just lookup tables that produce a 4-bit output for each 6-bit input. Each of the 8 S-boxes are different. The resulting output is then combined and permuted to produce a 32-bit final output.

In the original specification the S-boxes were given without any further explanation. In retrospect, it seems that the S-boxes were very well designed. The resulting cipher is as strong a cipher one

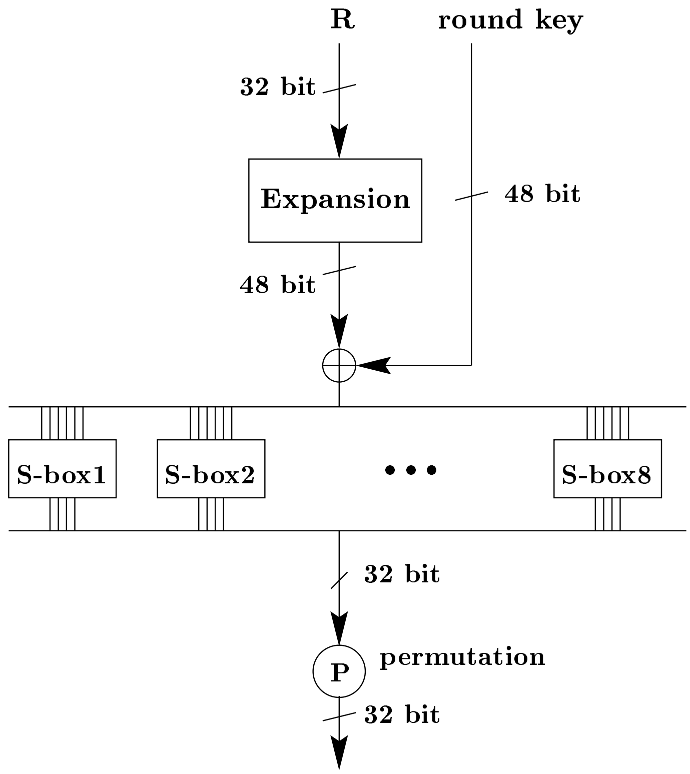

Figure 3: The f block for DES.

can get with 56-bit keys. It seems that NSA knew about the attacks we will be discussing next and helped to IBM to design the S-boxes to be resistant to these attacks.

1.2 Attacks to the DES

Differential Attack

This attack was proposed by Ali Shamir and Eli Biham. It is a chosen plaintext attack, i.e. it assumes the opponent has black box access to the encoding machine. This is a large assumption to make since it means that the opponent can feed any plaintext to the encoder and obtain the ciphertext.

The attack works by choosing many random pairs p_1, p_1' such that $p_1 \oplus p_1' =$ constant. If one performs many trials, one can deduce the last round key. From that, it is easy to deduce the remaining key. The number of trials depends on the S-boxes. For the choice in DES, the algorithm needs 2^{47} trials, which is better than brute force, in principle.

Question: Is the choice of the constant important?

Answer: Yes. The constant affects how changes propagate through the Feistel structures. In retrospect, the choice of S-boxes was good. Research showed that random S-boxes had worse performance than the ones chosen.

Question: Does it seem that the S-boxes were deliberately picked?

Answer: Most probably yes. It seems that the NSA had some knowledge of the attack and IBM had some hints from the NSA to make DES stronger.

Question: When was the attack published?

Answer: Their paper appeared in the Proceedings of the CRYPTO '90 conference.

Linear Attack

This is due to Matsui. It is a known plaintext attack, which makes much more appealing than the previous attack. It relies on statistics to see how the bits of each components interact with each other (e.g. $L_0[5] \oplus R_0[7] \oplus R_0[11] = K[11]$ with probability $\frac{1}{2} + p$, where K is the round key, $L[i]$ is the i'th bit of L, and p is a very small but positive number.) Using these relations we can guess the round key with high probability. This attack needs only 2^{43} known plaintext/ciphertext pairs.

Question: How is DES compared to 3DES.

Answer: There was a need to enhance DES by taking DES three times, with 3 different keys (as in Figure 4) which combined result to a 168-bit key algorithm. This seems perfectly secure under linear and differential attacks, but it is too slow compared to AES.

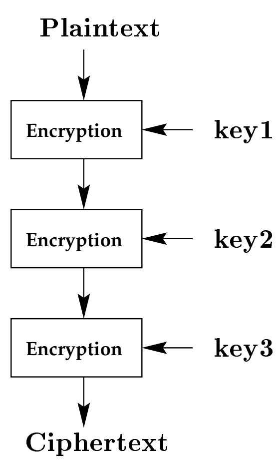

Figure 4: 3DES.

Question: Why not 2 boxes (instead of three) for 3DES?

Answer: It makes it susceptible to meet in the middle attacks (as in Problem set 1).

Question: How are the storage requirements compared to the computational requirements?

Answer: The storage requirements are very small. They basically involve keeping counters.

Question: Is it illegal to work on that due to DMCA?

Answer: It is a good question. The answer is not very clear. For something to be punishable under DMCA, it should be associated with copyrighted work. However, the act has had a chilling effect in research.

1.3 RC5—*(Ron's Code 5, 1994)*

RC5 takes as an input a 64 bit message and a 32 bit key. It breaks the message in two halves L and R, each of 32 bits. At first L is XORed with R. Then the outcome is rotated left by a value between 0 and 31 which is chosen by picking the lower 5 bits of R. The outcome of this operation is added (mod 2^{32}) with the key. Finally the output of this operation becomes the right half for the next iteration and R becomes the left half. For strong encryption we typically use 24 or 32 iterations. Figure 5 illustrates a single iteration of the algorithm.

1.4 AES—*(Advanced Encryption Standard, 1997)*

Historical Note: In 1997 the National Institute of Standards and Technology (NIST) announced a call for algorithms for AES. It was an international competition; there was even a conference in Rome. There were about 15 interesting proposals and the Rijndacl scheme, proposed by two Belgians, was the winner.

Specifications:

128 bits of input and output and 128, 192, 256 bit key lengths supported.

Scheme:

1. State:

   The 128 bits are represented by a 4x4 matrix of bytes.

2. Operations:

   - *Substitution:* This is a byte-wise substitution of all 16 bytes of the state matrix.
   - *Row Shift:*
     First row is shifted to the right by 0 bytes.
     Second row is shifted to the right by 1 byte.
     Third row is shifted to the right by 2 bytes.
     Fourth row is shifted to the right by 3 bytes.

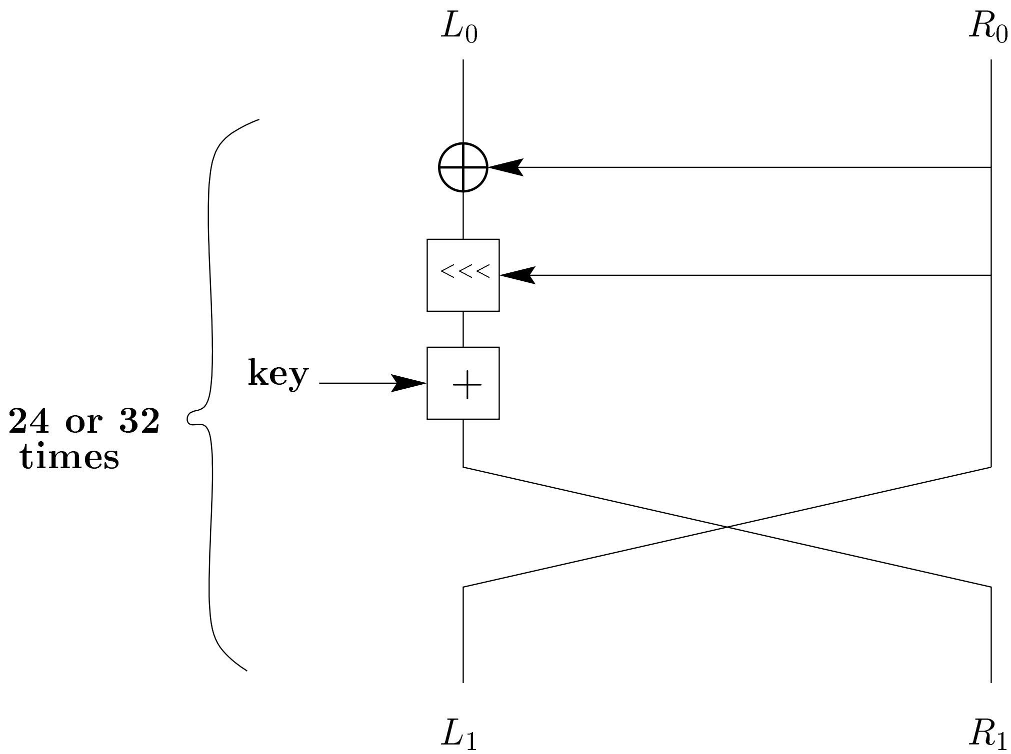

Figure 5: A single iteration of the RC5 algorithm.

The resulting matrix has the elements in the first column of the state matrix moved on the diagonal.

- *Column Mix:* Every column is multiplied by an invertible matrix over a finite field that is a power of 2. So $C_i^{'} = A \times C_i$.
- *Key Addition:* The state matrix is XORed with the key.

Question: Is the key a matrix?

Answer: Yes, the key is a matrix and if it is bigger than the state matrix we pick the columns with a scheduling scheme.

Question: How do we decrypt a cipher?

Answer: We can decrypt it, because all operations are invertible.

Question: Why isn't AES vulnerable to 2-time pad attacks?

Answer: Because the substitution operation is not linear.

Question: How long will it take to break the scheme, assuming Moore's law?

Answer: Now we can break 80 bit keys and given Moore's law we advance at a rate of 1 bit per 18 months, so it will take about 60 years. Most probably though Moore's law will die in the next 10 years.

2 Modes Of Operation

The question we want to answer now is how do we deal with arbitrary sizes of messages that are larger than our encryption/decryption algorithm can handle?

2.1 ECB—*Electronic Code Book (the dumb way)*

The message is broken down to pieces and each piece is encrypted separately.

Not recommended: Repetitions of input are passed to the output. Makes attacks easier. Figure 6 illustrates this scheme.

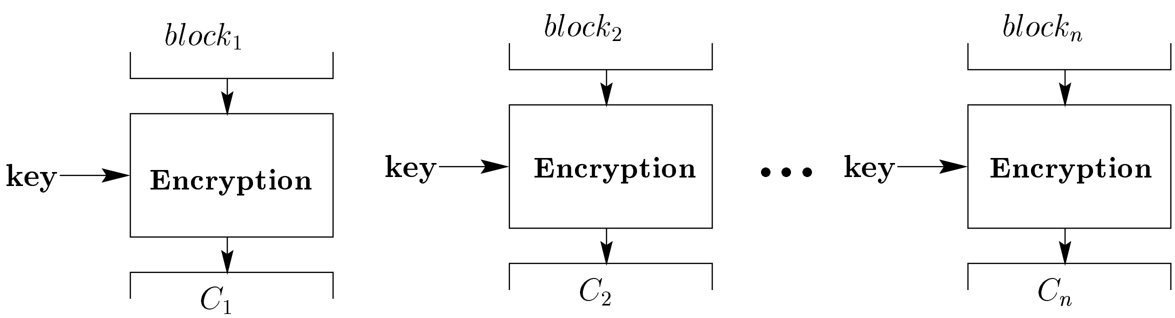

Figure 6: ECB mode block encryption.

2.2 CBC—*Cipher Block Chaining*

The algorithm takes as an input an initial vector IV, and each ciphertext block is used as an input to encrypt the next message block.

For the last block we use *Ciphertext Stealing*. This technique involves the following: Assume we have n blocks and we pad our last message block with zeros until we reach the full block size. If we were to use a direct encryption using C_{n-1} as input to the $n-th$ block, then an adversary knows that $|Block_n| - |P_n| = \phi$ bits are zeros and knowing C_{n-1} and C_n could have an easier time performing an attack.

Instead what we do is to hide what would otherwise be C_{n-1} (call it X_{n-1}) except the first $|P_n|$ bits, which we publish as our C_n. We use X_{n-1} as input to the $n-th$ block just like we did in the previous steps and we take the output of the $n-th$ encryption and we publish it as our $(n-1)-th$ cipher block C_{n-1}.

Figure 7 illustrates the chaining steps of the algorithm.

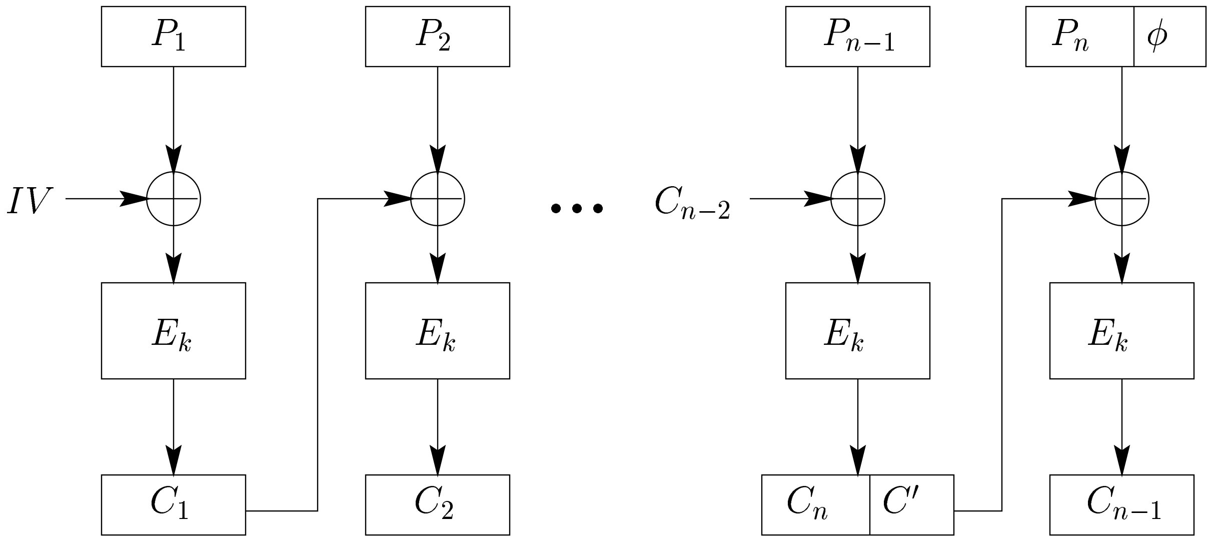

Figure 7: CBC mode with Ciphertext Stealing at the end.

2.3 Counter Encrypt

We simply encrypt the series 1,2,3,4,... with the key and then we XOR each output E(1), E(2),... with the corresponding message blocks M_1,M_2,... to get the ciphertext.

3 OCB—*Offset Codebook mode*

Two objectives: Achieve Confidentiality and Authentication with the same algorithm. We typically use 2 different techniques: we encrypt for Confidentiality and we use a MAC for authentication, but with OCB we can do both together.

OCB continued in the next lecture.