6.915 Computer and Network Security **Fall Term, 1995**

Lecture 4 : September 19, 1995

Lecturer: Ron Rivest *Scribe: Rena Yang*

Current Events:

Two Berkeley computer science students found a security flaw in Netscape. They found that the "random" numbers used to form Netscape's encryption keys were based on the time and date of a transaction and several other bits of information that an electronic intruder could gather. This greatly reduced the search space, allowing them to break Netscape's encryption within a minute

Moral: Make sure your random bits are random.

Topics covered today: Authentication, Computational Security, DES, RC5.

1 Unconditional Authentication

In the last lecture, we introduced one time pads, an information theoretically secure form of encryption. We now explore the use of one time keys to authenticate the sender, because perfect secrecy does not imply perfect authenticity. The only indication of a sender's authenticity is the intelligibility of the received message. This is no guarantee, however.

An example where authentication is particularly important is in bank transactions. If intruders knows the approximate format of a bank transaction, they can alter bits to record that a user has deposited or withdrawn $10,000 instead of $1,000. The bank needs a way to authenticate an ATM machine.

We cannot simply append an encrypted "This is Alice" string to the message because this does not authenticate the front part of the message. Instead, can we use one time keys for authentication by creating a Message Authentication Code (MAC), a function of the key and of the message.

Suppose we want to send a message M and want to authenticate it, we will send M with a MAC(M, K') appendend to it. K' is the key for authentication shared by the two parties (K is the key they use for encryption instead.)

The receiver of the message now computes the MAC with M and K'. If the received

MAC equals the computed MAC, the message is authenticated as coming from the correct sender.

The MAC function needs to have some special properties in order to be secure. In particular we want for an adversary to be hard to compute MACs of messages without knowing the key K'. We will present a proposal that is information-theoretically secure, but relies on the fact that the key is used only once.

Let K' be composed of two secret parts: a and b. Also assume that there exists a non-secret large prime p. Assume that $0 \leq M, a, b < p$. The MAC is computed as follows:

$$\mathrm{MAC}(M, K') = aM + b \bmod p$$

The adversary can now see M and $MAC(M, K')$ without gaining any further information. This is because, since K' is secret, any number modulo p could be the rigth MAC for a message M. If the adversary intercepts the message M with the rigth MAC Y and tries to change the message to M' and guesses

$$Y' = aM' + b \bmod p$$

for each Y', there is a unique a-b pair such that

$$Y = \mathrm{MAC}(M, (a, b)) \text{ and } Y' = \mathrm{MAC}(M', (a, b))$$

So we can argue that the adversary has a $1/p$ chance of success, giving the adversary no advantage over guessing.

We can never reuse K', though. If, for some reason, Alice and Bob get amnesia and use K' twice, the adversary now has two equations and two unknowns, and can now solve the equations for a and b.

In this implementation, the MAC is as long as the message. If we use fewer bits of the message, we can use up fewer bits of the MAC, but this implementation still consumes bits of the one time pad for every message sent or received. Long keys are somewhat difficult to distribute, although DAT tapes and CD ROMs can now store several gigabytes of data.

How can we get secrecy and authentication without using up keys? In order to achieve this we have to give up information-theoretic security and settle for computational security which will be the next topic.

2 Computational Security

Information theoretic encryptions are based on the impossibility of breaking an encryption.

Computational security is based on the computational difficulty of breaking an encryption. It is based on a key that is not consumed each time a message is encrypted or decrypted.

2.1 Data Encryption Standard (DES)

Data Encryption Standard (DES) is a computational security method that was developed in the 1970's by IBM and the NSA. It is a block cipher: that is it encrypts only blocks of messages 64-bits long. We will discuss later how to use DES (or any other block cipher) to encrypt longer messages. DES uses a 56-bits long keys.

If an adversary sees a message M and a ciphertext $C = \mathrm{DES}_K(M)$, we assume that it will be difficult to compute K.

DES is an iterative scheme which mixes linear and nonlinear functions to encrypt. The message M is split in two 32-bits parts L and R. These two parts are fed to a single iteration process which is described in Figure 1. The new L and R are then fed back to an identical iteration. The 56-bits key is trasnformed in 16 48-bits long keys which are used at each round by means of a key scheduling algorithm that we will not describe here.

The boxes marked by a 6 in the figure are called S–boxes. Those are basically tables that map 6-bits inputs into 4-bits output. The tables are constant for each encryption, but they are composed of nonlinear functions with the desirable property that small changes to the input to a table will have large effects on the output. This will create an avalanche effect if an adversary tries to probe the key with bits of message which are slight alterations of each other.

DES is easy to decode, however. We basically run the 16 iterations backward. We know R(n-1) from L(n). We can just run the encryption forward with R(n-1) and the key, then reverse engineer L(n-1) by adding the encrypted R(n-1) to R(n). See Figure 2.

DES is not easy to implement efficiently in software, but is quite efficient in hardware.

DES was adopted in 1977 as an encryption standard by the government and is still around. Every 5 years, it has been reevaluated. In 1987, the NSA did not want to bless it, but businessmen twisted the NSA's arm to continue to use DES as an encryption standard because they had invested money in DES encryption hardware. In general, though, the government does not like to bless cryptography schemes because they can be used to hide secrets from the government.

2.2 Methods of finding DES keys

Brute Force: must try all 2^{56} possible keys to match a given message-cyphertext pair.

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.

More recently, a VLSI designer named Wiener designed a chip which 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. The paper describing this design is availiable from the class Web page.

Differential Cryptanalysis: A more clever (non-brute) method of breaking the DES key requires Alice to encrypt whatever the adversary wants (chosen plaintext). This method may not be realistic in most cases. It reduces the number of keys to 2^{47} by accumulating statistical evidence about the key. This method to break the DES key was known when it was designed, so the S–boxes are designed to be amazingly resilient under this form of attack.

Linear Cryptanalysis: The linear method of breaking the DES encryption scheme relies on forming a linear approximation to the encryption key. It also gathers statistical evidence about the key from observation (as opposed to requiring encryption of chosen plaintexts), but only needs to search 2^{43} randomly generated keys. This method has actually been used.

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.

2.3 Strenghtening DES

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. See Figure 3.

Triple DES: Let $\mathrm{DES}_K(M)$ denote the DES–encryption of message M with key K. Let $\mathrm{DES}_K^{-1}(M)$ denote the DES–decryption of message M with key K.

Triple DES uses two keys K_1 and K_2 and encrypts a message M as follows:

$$C = \mathrm{DES}_{K_1}(\mathrm{DES}_{K_2}^{-1}(\mathrm{DES}_{K_1}(M)))$$

This method has an effective key of 90 bits.

An observation about this method is that if $K_1 = K_2$, we can get single DES, so Triple DES is backwards compatible. This is why the middle encryption uses DES in decryption mode.

2.4 New Encryption Methods

Professor Rivest has recently designed another encryption method called RC5.

It does the following operation 12 times:

It breaks a message block up into a left L and right R side.

$$L = ((L \oplus R) \ll R) + K$$

$$R = ((R \oplus L) \gg L) + K$$

where $\ll X$ is bit rotation by X bits. This rotation is dependent on the data. So far, it has held up against both linear and differential attack.

RC5 is easy to decrypt and can use variable sized keys and a variable number of iterations. A paper describing RC5 in more details is available from the class Web page.

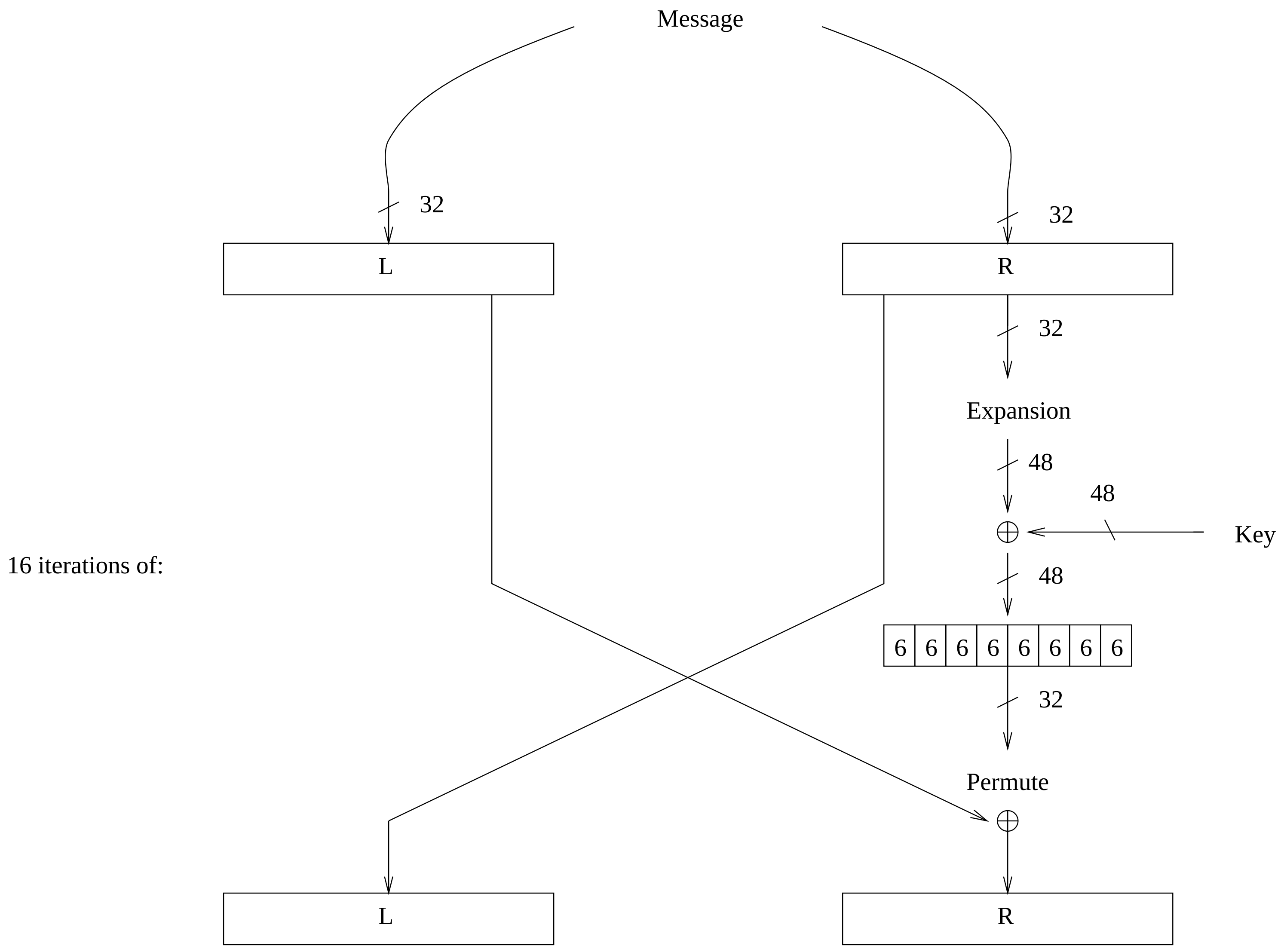

Figure 1: More Detailed DES Structure

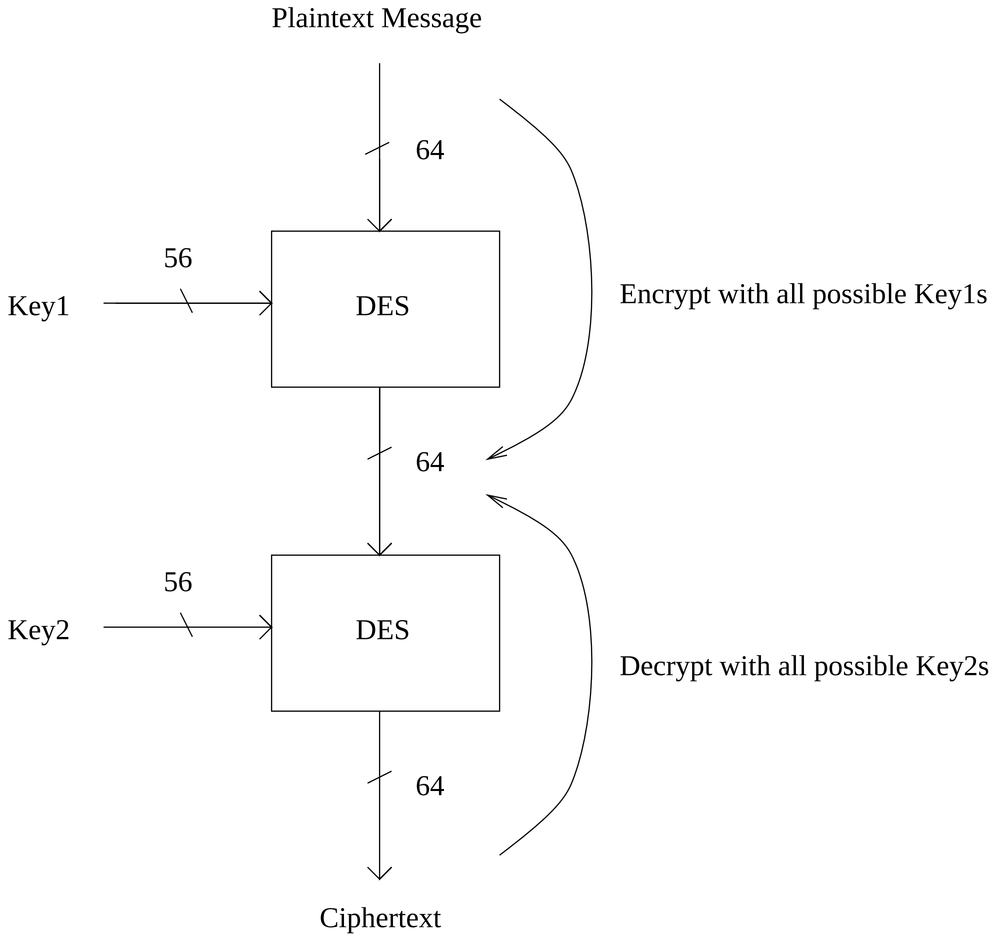

Figure 2: Decryption with DES

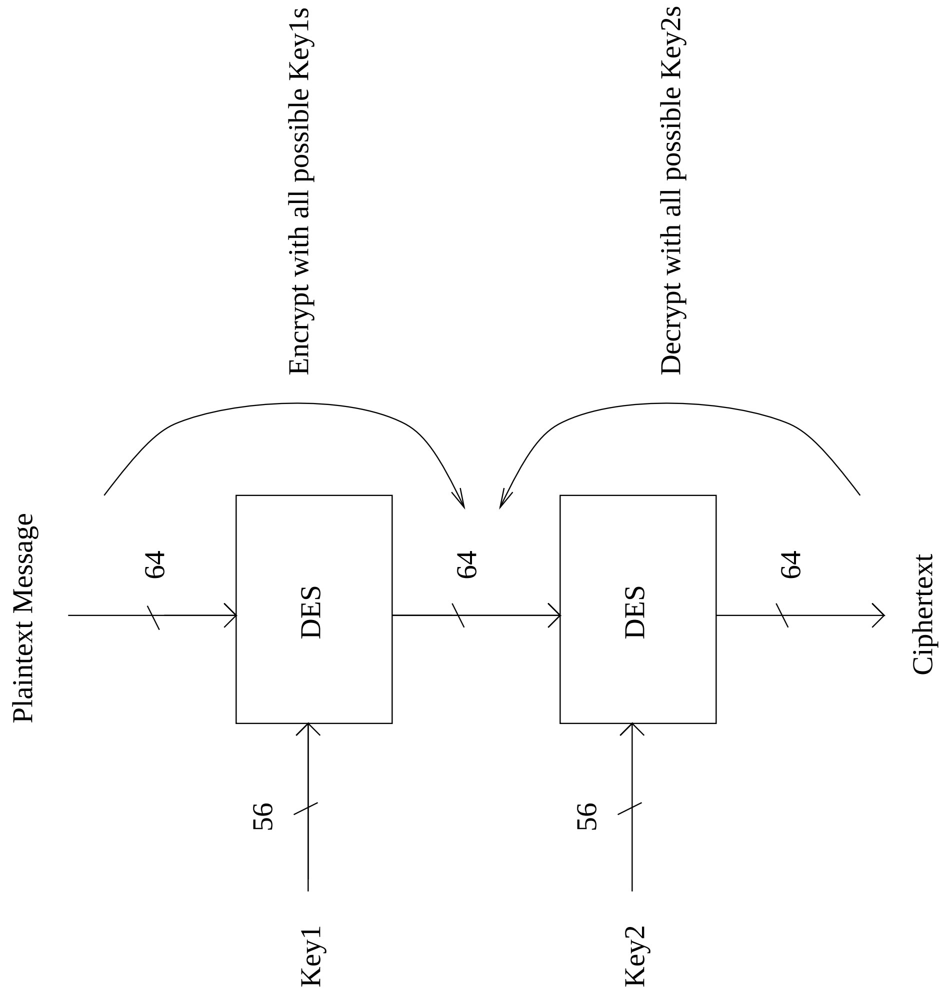

Figure 3: Double DES Meet in the Middle Attack