**6.857 Computer and Network Security** **Fall Term, 1997**

Lecture 4 : 16 September 1997

*Lecturer: Ron Rivest* *Scribe: Michelle Goldberg*

# 1 Conditionally Secure Cryptography

Conditionally (or computationally) secure cryptography uses a shared secret key of limited length to provide security against an oponent with limited computational resources by making it computationally infeasable extract the key or message.

## 1.1 Standards of Proof

Rigorous proofs of conditional cryptography are in very limited supply. It would be nice to be able to prove that a keyspace search or comparable effort (order of $2^n$, for key length n) would be required to find a key or decode a message, but for the most part it's not possible. One way of showing a given system to be computationally secure is to show that breaking it (reliably) would be comparable in difficulty to solving an NP-hard problem – but given that whether $P \neq NP$ (whether the class of problems solveable in polynomial time is the same as the class of problems solveable in non-deterministic polynomial time) is not yet known, even showing a system to be NP-hard to break isn't entirely conclusive. Were it to be shown that $P = NP$, much of computationally secure cryptography as it presently exists would collapse. Even the "proveably secure" systems are based on the *assumption* that factoring (or another such problem) is sufficiently hard.

So what are the resources of the limited adversary, and what test(s) might a cryptographics system be asked to pass to demonstrate reasonable security? One proposed "turing test" for a cryptographic system is that an adversary, given $M_x$ and $M_y$, should not be able to determine (with greater than 50% probability) which of two ciphertexts is $C(M_x)$ and which is $C(M_y)$ with less than a keyspace search. Another proposed test is that an adversary should not be able to determine the encryption key even with many examples of

- known ciphertext (an eavesdropper), or

- known ciphertext/plaintext pairs (an eavesdropper able to correlate to plaintexts stolen or made public), or
- chosen ciphertext/plaintext pairs (someone able to submit items for encryption, possibly an insider[1]), or
- related keys used.

There are no proofs that any of these practical proposals are sufficient tests.

## 1.2 Block Ciphers

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

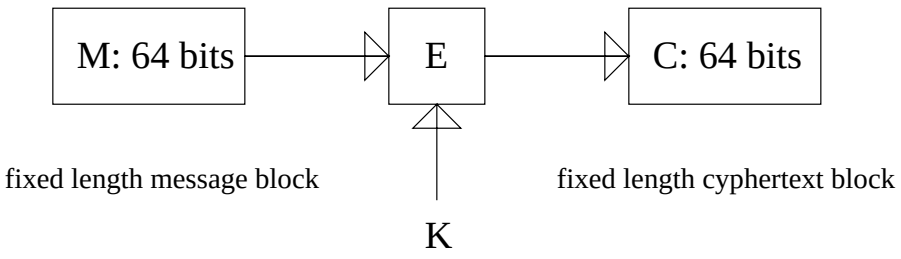


Figure 1: General Block Cipher

**Feistel ciphers** are based on a certain (public domain) style of design. For such a cipher, the key $K$ is broken into round keys $\{K_1, K_2, K_3, ...K_n\}$, and the message $M$ broken 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 return as the ciphertext. Decrypting is straightforward, as it is possible with the key to recompute the the round keys and each $f(R, K_i)$ in turn, and the XOR function is its own inverse. The final swap of the two halves of the ciphertext allows decryption to be done with the same hardware as encryption, with only reversing the order of the round keys necessary.

[1]The field of differential cryptanalysis is based on analyzing CT/PT pairs with slight differences.

How the key is turned into round keys and the selection of the function $f(R, K_i)$ are the distinguishing features of specific Feistel ciphers. The design of a good function $f(R, K_i)$ is an art form.

In 1993, Wiener proposed to build for one million dollars a machine that would crack a Feistel cipher in seven hours. Matsui did undertake to break one, and found a key in about 50 hours after searching $2^{42}$ plaintext/ciphertext pairs[2]. The recent Internet Challenge this year did so as well, applying a full keyspace search run in spare cycles on machines scattered across the net.

**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, XORs $R$ 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.[3] One of many functions of the S-boxes is to magnify the effect of changes in the plaintext on the ciphertext.

The NSA was involved in 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.

Variations on DES in use now as possibly stronger include DESX, wherein $C(M) = K_2 \oplus DES(K, M \oplus K_1)$, and triple DES, for which $C(M) = DES_{K_1}(DES^{-1}_{K_2}(DES_{K_1}(M)))$[4].

DES is expected to be replaced as a national standard by AES, a cipher to be determined by the NIST. A call for proposals for a 128 bit block cipher with a 128 bit key is expected soon.

Other block ciphers include Blowfish, IDEA, CAST, and RC5. RC5 is a Fiestel-type cipher, possibly to be proposed by Rivest for AES, in which the round function is $L \leftarrow ((L \oplus R) <<< R) \oplus K_i)$, where $<<<$ indicates rotation.

[2]This using linear cryptanalysis, which looks at statistical variations in the ciphertext in relation to the plaintext.

[3]See the applied cryptography book for details.

[4]Double DES is vulnerable to bidirectional attacks, working forward from the plaintext and backwards from the ciphertext.

## 1.3 Modes of Operation of Block Ciphers

**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. 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 $\{R_1, R_2, R_3, ...\}$, which are XORed with the message blocks to produce the ciphertext: $C_i \leftarrow M_i \oplus c(K, i)$. This has the advantage 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 which are XORed with the message to produce the ciphertext, but instead of feeding the cipher a fixed sequence of numbers, 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.

**Cipher Block Chaining** (CBC) XORs each message block with the ciphertext of the previous block before encrypting it; an initialization vector, possibly public, is XORed 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 alters the last two ciphertext blocks of a message generated with CBC to avoid a ciphertext longer than 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 XORed 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.

## 1.4 Message Authentification Codes with Block Ciphers

Cipher Block Chaining can be used to generate a MAC using a block cipher. CBC is run as usual over the whole message (or ciphertext), and the last block, or a fraction of it, is appended as the MAC. This generates a fixed-length MAC calculated from the entire text of the message. If the message is also being encrypted, a separate key is used to generate the MAC. (This is necessary if the message rather than the ciphertext is being authentificated, but is a separate key needed if the ciphertext is being authentification?)

This CBC-based MAC is somewhat vulnerable for messages of non-fixed length: given one-block message $M_1$ with MAC $A_1$, and message $M_2$ with MAC $A_2$, a second message $(M_1, M_2 \oplus A_1)$ can be constructed which also has MAC $A_2$. MACs of this sort are therefore usually encrypted with another key.

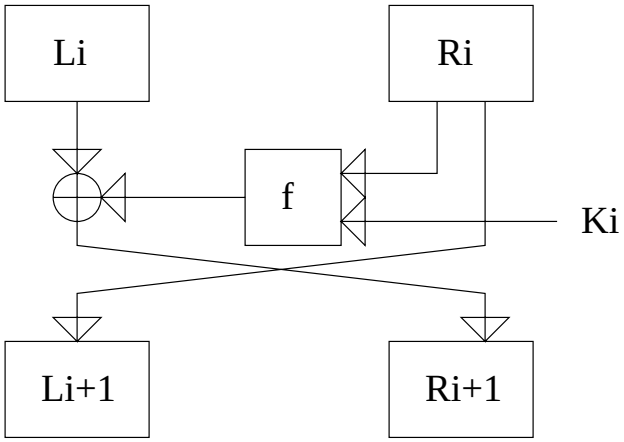


Figure 2: Feistel Cipher

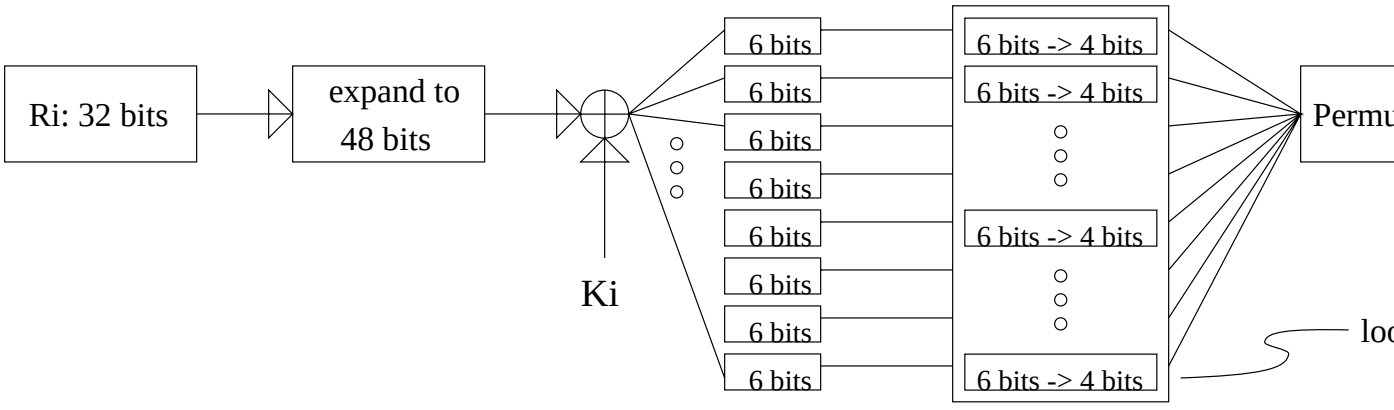


Figure 3: DES: $f(R, K_i)$ for one round

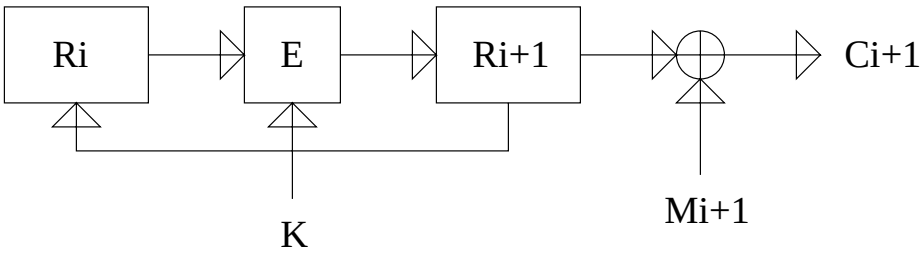


Figure 4: Output Feedback Mode

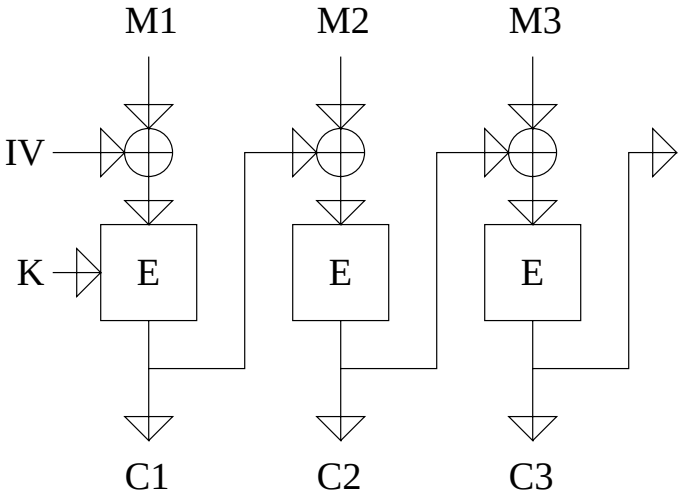


Figure 5: Cipher Block Chaining

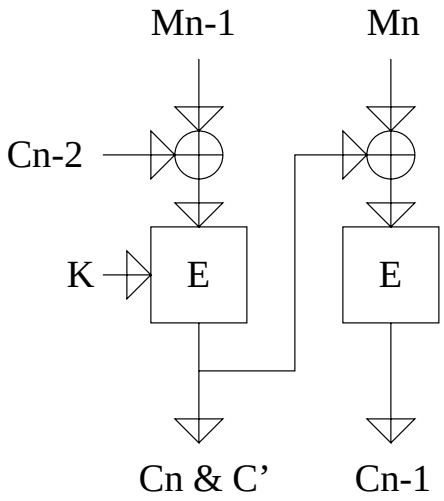


Figure 6: Cipher Block Stealing