Computer Science Tripos part 2 – Security

Lecture 7 – Block Ciphers

Ross J. Anderson

1 Introduction

In this lecture we will look at block ciphers. Just as stream ciphers are the mainstay of military applications, so most commercial applications use block ciphers. We will look at four designs – SAFER, DES, Skipjack and IDEA – so that we can discuss the design principles and pitfalls in a reasonably general setting, and see the complexity tradeoffs involved.

We will then revise the modes of operation of block ciphers, which we first touched on in 1b, and go on to consider hash functions. These are not the same as the hashing used in hash tables, but cryptographically strong one-way functions which are easy to compute but extremely hard to invert.

Although Shannon mentioned the possibility of block ciphers in the 40's, the application that motivated the development of the modern block cipher was the identify friend or foe (IFF) systems used in air combat. It could be difficult to implement an IFF system with a stream cipher – we would get code replay attacks and synchronisation problems. So we have an environment in which an attacker can observe a lot of valid challenge-response pairs (a known plaintext attack) and may occasionally be able to get responses to bogus challenges of his own (a chosen plaintext attack).

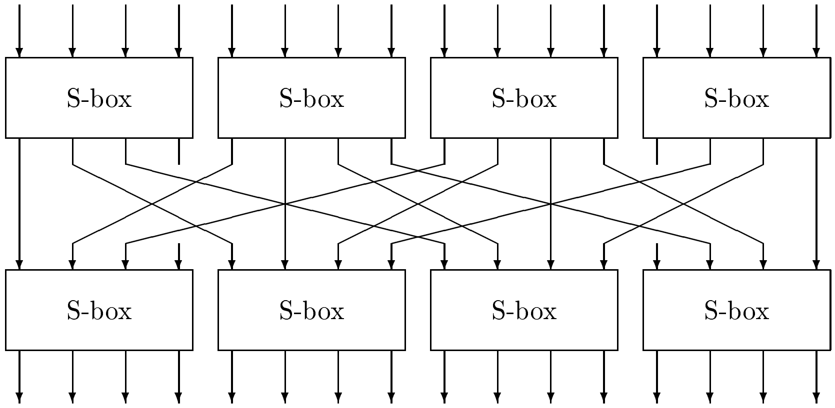
We are looking for a cipher which will withstand both these types of attack, in the sense that the attacker should not be able to recover the key or otherwise predict the response to a new challenge.

2 SP-Networks

Shannon suggested that strong ciphers could be built by combining substitution with transposition. For example, one might do a Playfair followed by a Vigenère, then move the text one character to the left, do another Playfair and so on. He described the properties of a cipher as being confusion and diffusion – a substitution or stream cipher confuses an attacker about the value of a plaintext symbol, while diffusion means spreading the plaintext information through the ciphertext.

The earliest block ciphers were networks which combined substitution and permutation circuits, and so were called SP-networks. This diagram shows an

SP-network with sixteen inputs and two layers of four-bit invertible substitution boxes (or *S-boxes*), each of which can be visualised as a lookup table containing some permutation of the numbers 0 to 15:



The point in this arrangement is that if we were to implement an arbitrary 16 bit to 16 bit function we would need 16.2^{16} bits of memory, or hundreds of thousands of gates, while a a four bit to four bit function takes only 4.2^4 or 64 bits of memory. One might hope to implement a block cipher in which the key consisted of some choice of a number of four-bit S-boxes.

However, this arrangement is not secure. An attacker who can mount a chosen plaintext attack – such as for example an enemy radar network which sends a chosen pattern of IFF challenges to our aircraft – can quickly reconstruct all the S-boxes. Can you see how?

One way to strengthen a system of this type is to use many more rounds. There will still be some information leakage; for example, an opponent could try to find a linear approximation to the S-box, or to check guesses about the key by doing statistical tests on large numbers of corresponding plaintext and ciphertext blocks. However, with a well designed cipher, the signal to noise ratio will become lower as the number of rounds increases, and given enough rounds we might get a secure system. We will discuss later how this leakage can be quantified and the necessary number of rounds estimated.

For now, we will merely state that we can reduce the per-round information leakage, and thus the required number of rounds, by designing the rounds carefully. An example of a modern block cipher of this type is SAFER, which was designed by James Massey to fit into smartcards – this means using byte-oriented operations and not too much memory for tables. This algorithm has been placed in the public domain; it is used by the government of Singapore and by a number of other organisations.

In a typical configuration, there are twelve rounds, one of which is depicted in the following diagram.

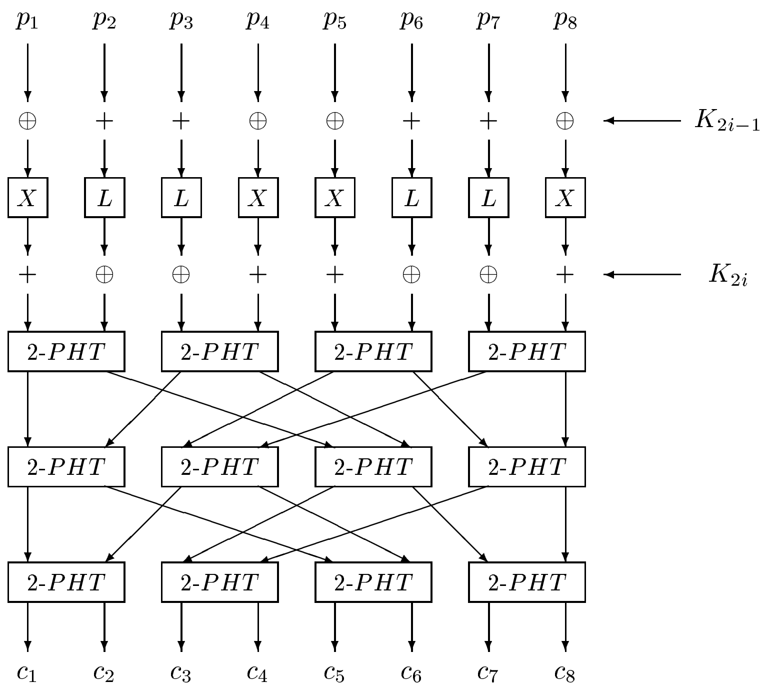


Figure 1: One round of SAFER K

The cipher acts on 64-bit blocks. Each input block is first combined with key material, and is then put through byte-wide S-boxes. These S-boxes are implemented as lookup tables of 256 bytes: one implements the function $x \leftarrow 45^x \pmod{257}$ and its inverse implements $x \leftarrow \log_{45} x \pmod{257}$. These substitutions are very nonlinear and provide good confusion. More key material is then added, and a diffusion stage then mixes up the bytes using 2-PHT, a 'pseudo-Hadamard transform'. If the inputs to the 2-PHT are (a_1, a_2) then its output is (b_1, b_2) where

$$b_1 = 2a_1 + a_2$$

$$b_2 = a_1 + a_2$$

with the arithmetic being carried out modulo 256 in each case (there is nothing magical about this — it is just an easy to implement transform of determinant 1). Finally, there is a routine for expanding the user supplied key (which can be of 64 or 128 bits) to give the 1536 bits of key material used by the twelve rounds. This has the feature that there are no patterns in the key bit distribution which an attacker is likely to be able to use.

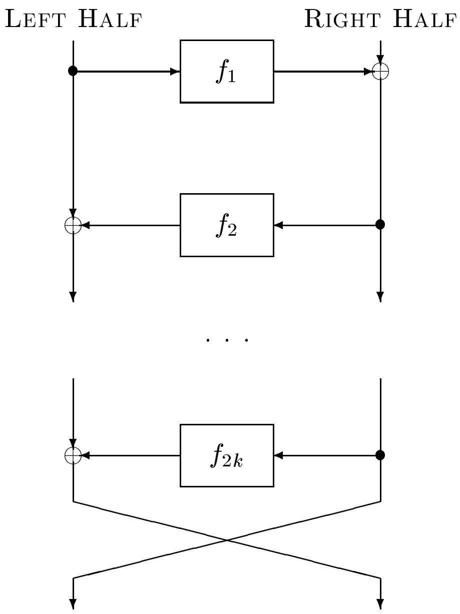


Figure 2: The Feistel cipher structure

Another example of a block cipher based on an SP-network is Serpent, which is our candidate submission for the Advanced Encryption Standard (a competition being run by the US government to find a successor to DES). Details can be found on the Serpent home page, `www.cl.cam.ac.uk/~rja14/serpent.html`

3 Feistel Ciphers

Most fielded block ciphers use a more complex structure, which was designed by Horst Feistel and is named after him. We touched on this in the 1b course where we gave TEA as a teaching example. A Feistel cipher has the 'ladder' structure of figure 2.

A notation which you may see for the Feistel cipher is $\psi(f, g, h, ...)$ where f, g, h, ... are the successive round functions. Under this notation, the above cipher is $\psi(f_1, f_2, ...f_{2k-1}, f_{2k})$. The basic result which enables us to decrypt a Feistel cipher – and indeed the whole point of his design – is that:

$$\psi^{-1}(f_1, f_2, ..., f_{2k-1}, f_{2k}) = \psi(f_{2k}, f_{2k-1}, ..., f_2, f_1)$$

In other words, to decrypt we just use the round functions in the reverse order. Thus the round functions f_i do not have to be invertible, and the Feistel structure lets us turn any one-way function into a block cipher. This means that we are less constrained in choosing a round function with good diffusion and confusion properties, and which also satisfies any other design constraints such as code size, table size, speed, etc.

3.1 The Luby-Rackoff result

The seminal theoretical result on Feistel ciphers was proved by Mike Luby and Charlie Rackoff in 1988.

Firstly we need to introduce the idea of random and pseudorandom functions and permutations. The strongest requirement we can make of a k-bit block cipher is that it be a random permutation – a randomly chosen element of the symmetric group on 2^k elements – and we often model this conceptually using a random oracle.

In the random oracle model, we envisage a block cipher, or random permutation, as a box containing a demon with some dice and a file store. When presented with a k-bit input he has already seen, he gives us the same answer as before; when presented with a new input, he rolls dice until he gets a k-bit output he has never used before and gives us that. With a random function, the demon takes an arbitrary input and, if he has already seen it, gives us the same answer as before; if not, he rolls the dice once to get a k-bit value and outputs that, regardless of whether he has output it before or not. Pseudorandom permutations (respectively functions) are functions which cannot be distinguished from random permutations (resp. functions) in polynomial time.

The Luby-Rackoff result is that a three-round Feistel cipher $\psi(f_1, f_2, f_3)$ is a pseudorandom permutation generator provided that the three round functions f_i are pseudorandom functions. There are a number of technicalities we omit; in engineering terms, the effect is that given a really good round function, three rounds of Feistel are enough for resistance against chosen-plaintext attack while four rounds suffice to guard against chosen-ciphertext attacks.

3.2 DES

The specification of the Data Encryption Standard, or DES, algorithm is contained in the textbooks (particularly Schneier, who has source code as well). Briefly, it is a Feistel cipher, with a 64-bit block and 56-bit key. Its round function operates on 32-bit half blocks and consists of an expansion from 32 bits to 48, after which 48 bits of round key are mixed in using exclusive-or; the result is passed through a row of eight S-boxes, each of which takes a six-bit input and provides a four-bit output. The round keys are derived from the user supplied key by using each user key bit in twelve different rounds according to a slightly irregular pattern. The appendix contains a copy of the full specification; you can get source code to play with from many sources including Schneier.

As mentioned in 1b, DES caused some controversy. Some people said that there should not be a standard at all, as it would provide too tempting a target; but many proprietary cipher designs have turned out to be weak, and industry needs standards for interoperability.

DES was also widely criticised for having too short a key, and this has now been substantiated. The Electronic Frontier Foundation built a DES keysearch machine in 1998 for under $250,000 which broke a DES challenge in 3 days. It contained 1,536 chips run at 40MHz, each containing 24 search units which each need 16 cycles to do a test decrypt. The search rate is thus 2.5 million test decryptions per second per search unit, or 60 million keys per second per chip. The design of the cracker can be found at `http://www.jya.com/cracking-des.htm`.

Previously, DES had been broken in a distributed software effort on the net. This is much less efficient, and took 14,000 Pentium-level computers over four months. It is also believed that the intelligence agencies of the USA Britain, France and Russia (at least) have keysearch machines too. So DES's 56 bit key is now too short for applications requiring a high level of security.

The third criticism of DES was that, since IBM kept its design principles secret at the request of the NSA, perhaps there was a 'trapdoor' which would give them easy access. However, the design principles were published in 1992 after researchers had independently rediscovered the cryptanalytic techniques which DES was designed to forestall and which the NSA wanted kept secret. We will now look at these techniques.

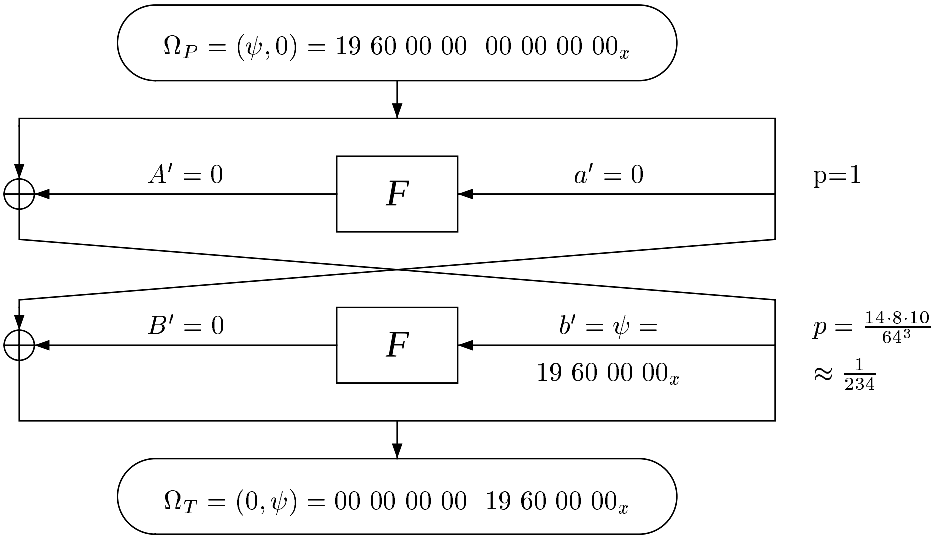
4 Differential and Linear Cryptanalysis

At Eurocrypt 87, NTT proposed a new block cipher called FEAL, and offered a reward for anyone who could break it. Two Israeli researchers, Eli Biham and Adi Shamir, took up the challenge. They noticed that when one input to FEAL two plaintext blocks B_1 and B_2 whose bitwise difference $B_1 \oplus B_2$ had a particular value, then the bitwise difference of the corresponding ciphertexts would also have a certain value with a known probability. By pushing a 'differential' through the cipher and detecting the target output difference when it appeared, they could deduce the inputs to the last rounds of the cipher and thus deduce some round key bits. The effect was that with 128 chosen plaintexts, one could break FEAL.

Biham and Shamir then turned their attention to DES, and showed that the same technique could be used to provide a faster attack on DES than keysearch. Their attack proceeds by tabulating input/output difference tables for the eight S-boxes, and then searching for differential characteristics which propagate through a number of rounds of the algorithm.

As an example, if two 32-bit blocks B_1 and B_2 have $B_1 \oplus B_2 = 1960\ 0000$, and are passed through the DES round function, then with probability $p = \frac{1}{234}$

we will have $F(B_1) = F(B_2)$. This enables us to construct a 64-bit difference pattern, or 'characteristic', which will pass through two rounds of DES with this probability.



This characteristic obviously allows an attack on DES that is faster than keysearch. With some optimisation, one gets an attack with 2^{47} chosen plaintexts – in fact, each time we encrypt an attack structure containing 2^{11} carefully chosen texts, we have a 2^{-36} probability of pushing a differential through the cipher and whenever this succeeds we get 22 bits of key (the other keybits are recovered by brute force search).

The publication of this attack led IBM to announce in March 1992 that it had known about differential cryptanalysis all along, but had kept quiet about it because its disclosure would have weakened America's competitive advantage over other countries in cryptanalysis. See the exercise for the S-box design criteria which they then revealed.

The next advance came in 1993 when Mitsuru Matsui unveiled linear cryptanalysis, which enables DES to be attacked with 2^{43} known plaintexts. This was actually done; in the first break of DES reported in the open literature, Matsui solved a key using 12 workstations for 50 days.

His technique is even simpler than differential cryptanalysis: it simply looks for linear approximations to S-boxes. For example, the least nonlinear of the DES S-boxes is S5, where if we write the inputs as x_i and the outputs as y_j, the relation $y_1 + y_2 + y_3 + y_4 = 1 + x_1$ holds with probability 5/16. From such relations we build up linear approximations to multiple rounds of the cipher, finding for example that the best linear relation in 16 round DES holds with probability 1/2 - 1.19 x 2^{-22}. The upshot is that 14 key bits of DES can be recovered with about 2^{43} known plaintexts.

The key fact about differential and linear cryptanalysis is that they give us a set of quantitative tools with which we can estimate the number of rounds a block cipher needs in order to attain a given level of security.

5 Skipjack

Skipjack is the block cipher used by US and NATO forces for encrypting data up to Secret. It is a 32-round block cipher with a 64-bit block and an 80-bit key, and was designed originally in the mid-80's for microcontroller applications. It is thus very parsimonious in its use of memory and has a very compact implementation.

Skipjack does not have the Feistel structure, but acts on four words $w_1, \ldots, w_4$ of 16 bits using a nonlinear function G according to two rules. In rule A we have

$$\begin{aligned} w_1^{k+1} &= G^k(w_1^k) \oplus w_4^k \oplus k \\ w_2^{k+1} &= G^k(w_1^k) \\ w_3^{k+1} &= w_2^k \\ w_4^{k+1} &= w_3^k \end{aligned}$$

While under rule B:

$$\begin{aligned} w_1^{k+1} &= w_4^k \\ w_2^{k+1} &= G^k(w_1^k) \\ w_3^{k+1} &= w_1^k \oplus w_2^k \oplus k \\ w_4^{k+1} &= w_3^k \end{aligned}$$

The cipher proceeds with eight round of rule A, eight of rule B, then eight more of rule A and finally eight of rule B. The G function is defined as a four-round Feistel structure whose round function consists of xor'ing the input with round key followed by a 256-byte nonlinear lookup table as an S-box.

Another interesting design criterion was that, as the equipment for which it was designed could fall into enemy hands, it should not be capable of being 'improved' by people who did not fully understand what they were doing (for example, it should not be straightforward to increase the key size). Since Skipjack was declassified in June 98, an interesting attack has been found which works against reduced round versions of the cipher. In effect, the round A / round B structure prevents certain types of mixing happening until right at the end of the encryption process. The paper describing this attack is included.

6 IDEA

Our final example of a block cipher is IDEA, which is used in the popular email encryption program PGP. This is based on mixing a number of incompatible algebraic operations in order to make attacks based on approximations to any of these algebraic structures unlikely. Eight rounds are used: the round structure, plus the final transformation, are depicted in figure 3 overleaf.

7 Revision – Modes of Operation

As we discussed in 1b, there are five 'official' modes of operation for using a block cipher on multiple blocks, and choosing the right one is one of the most important factors in using a block cipher securely.

7.1 Electronic Code Book

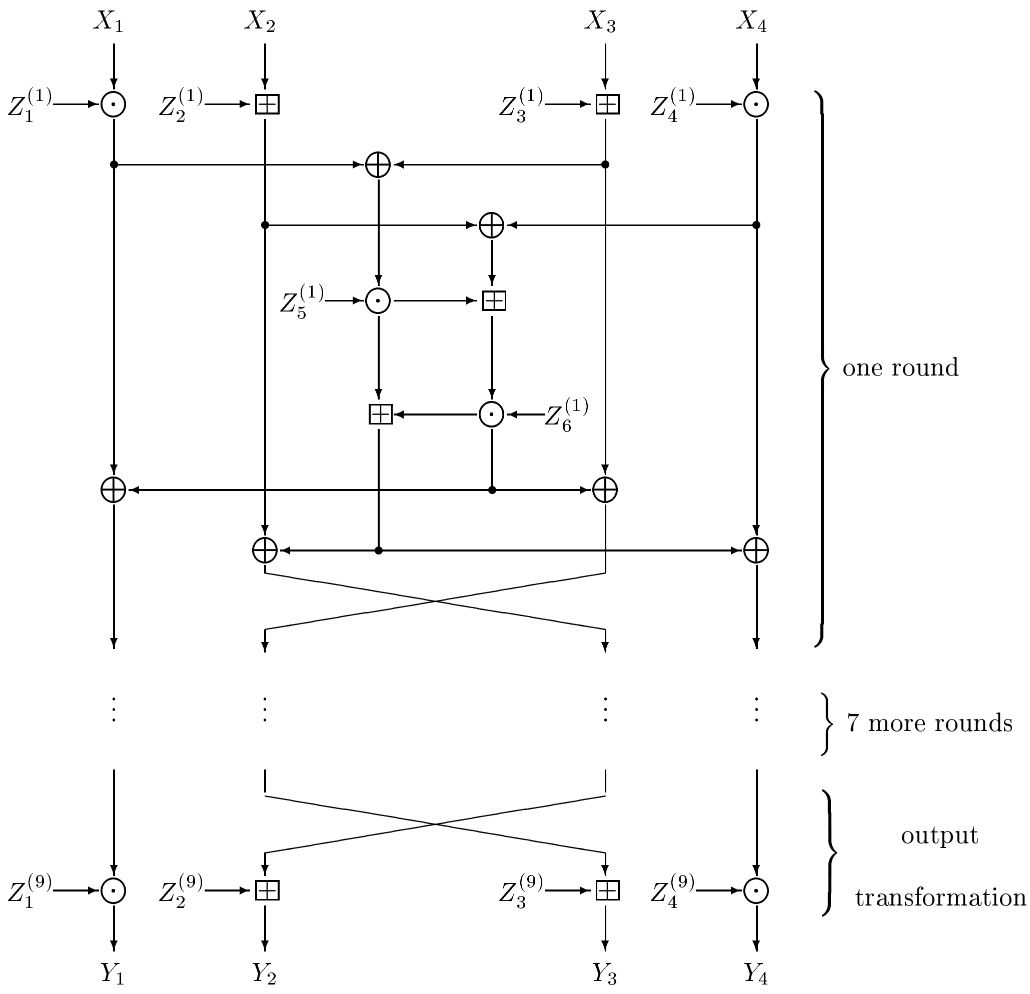
In Electronic code book (ECB) we just encrypt each succeeding eight byte block of plaintext with our block cipher to get ciphertext. This is adequate for many simple operations such as challenge response and some key management tasks, but if we use it to encryp[t redundant data the patterns will show through. Recall the example of the email system which used DES ECB with the key derived from an eight character password: the most common ciphertext block by far corresponded to a plaintext of nulls, giving a trivial precomputation attack based on a password dictionary.

7.2 Output Feedback

Output feedback, or OFB, mode consists of repeatedly encrypting an initial value and using this as a keystream in a stream cipher of the kind discussed in the last lecture. The i-th block of keystream will be given by

$$K_i = \{...\{\{IV\}_K\}_K ...\textit{total of i times}\}$$

This also fails to provide message integrity, as attacks in depth can be carried out just as on any other stream cipher.



X_i : 16-bit plaintext subblock
Y_i : 16-bit ciphertext subblock
$Z_i^{(r)}$: 16-bit key subblock
$\bigoplus$: bit-by-bit exclusive-OR of 16-bit subblocks
$\boxplus$: addition modulo 2^{16} of 16-bit integers
$\odot$: multiplication modulo $2^{16}+1$ of 16-bit integers
with the zero subblock corresponding to 2^{16}

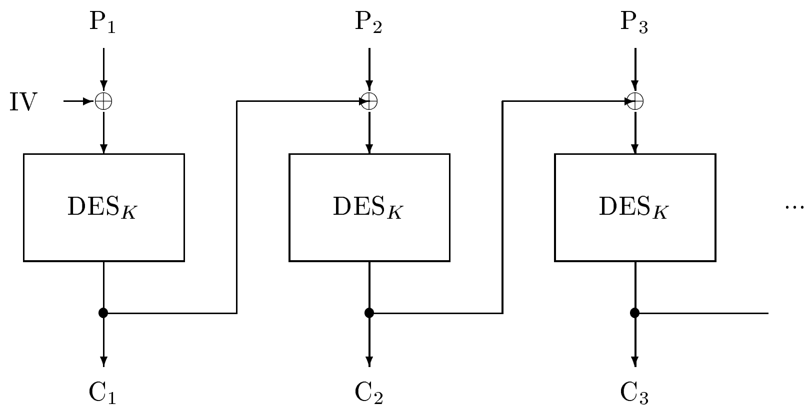
Figure 3: Computational graph for the encryption process of the IDEA cipher.

7.3 Cipher Feedback

Cipher feedback, or CFB, mode is a rather inefficient use of our encryption algorithm: we use our block cipher to encrypt the last 64 bits of ciphertext, and then exclusive-or the next plaintext bit with a single bit of the output. Its advantage is that it is self-synchronising – even if we get a burst error and drop a few bits, the system will resynchronise after one block length. There are some applications (especially military ones using HF radio modems) where this property is invaluable.

7.4 Cipher Block Chaining

Most commercial applications which encrypt more than one block use cipher block chaining, or CBC, mode. In it, we exclusive-or the previous block of ciphertext to the current block of plaintext before encryption:



This mode is effective at disguising any patterns in the plaintext: the encryption of each block depends on all the previous blocks. However, an opponent who knows some of the plaintext may be able to cut and splice a message (or parts of several messages encrypted under the same key), so the integrity protection is not total.

7.5 Message Authentication Code

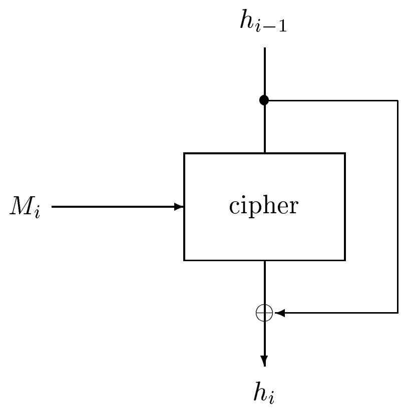
A block cipher can be used to calculate a message authentication code, or MAC. The MAC is simply the last ciphertext block returned when the message is encrypted using CBC mode. However, in this mode the intermediate results are kept secret in order to prevent splicing attacks.

Like CBC, this depends on all the plaintext blocks as well as on the key. It is secure provided the message length is fixed; it can be shown that any attack on a MAC under these circumstances would give an attack on the underlying block cipher.

In applications needing both integrity and privacy, the procedure is to first calculate a MAC on the message using one key, and then CBC encrypt it using a different key. If the same key is used for both encryption and authentication, then the security of the latter is no longer guaranteed.

8 Hash Functions

Recall that a block cipher can be used to calculate a hash of a message, without any secret key input. The basic idea is that we feed the message blocks one at a time to the key input of our block cipher, and use it to update a hash value (which starts off at say $H_0 = 0$). In order to make this operation non-invertible, we add feedforward: the $i-1$st hash value is exclusive or'ed with the ith at the end of round i:



Recall also that because of the 'birthday problem' in probability theory, a 64 bit block cipher is not enough here (the opponent could hash 2^{32} messages and with good probability find two of them with the same hash).

The hash functions most commonly used in applications are all related, and are based on variants of a block cipher with a 512 bit key and a block size of either 128 or 160 bits. MD4 has three rounds and a 128 bit hash value, and collisions have recently been found for it; MD5 has four rounds and a 128 bit hash value; while the US Secure Hash Standard has five rounds and a 160 bit hash value. The block ciphers underlying these hash functions are similar; their round function is a complicated mixture of the register operations available on 32 bit processors.

Hash functions are commonly thought of as pseudorandom functions, though for many applications a weaker condition may suffice. For example, a hash function is called 'collision-free' if it is not practical to find two inputs $M_1 \neq M_2$ with $h(M_1) = h(M_2)$. This is sufficient to generate one-time digital signatures which we discussed in the 1b lectures. (Recall that to sign a k-bit message, we find $2k$ random numbers R_i and publish their hashes $h(R_i)$ as out public signature verification key. To sign the message we now publish R_j if bit j of the message is equal to 0 and R_{k+j} if it is equal to 1.)

Another property that may be required of hash functions and which is also weaker than pseudorandomness is one-wayness; that while it is easy to compute $h(M)$ given M it is impractical to compute M if only $h(M)$ is given.

An example of an application where one-wayness is required is in micropayments. One vulnerability of present GSM mobile phone systems is that the bill is only calculated when the receiver goes on-hook, and international billing takes several days to clear. This has enabled fraudsters operating call-sell operations to make calls that cost thousands of pounds; a single call is used to set up multiple conferences in succession (this is why prepaid mobile phones are not allowed to make international calls). In the next generation system, UMTS, it is proposed that each phone will compute a number of phone ticks by repeated hashing: $t_1 = h(t_0)$, $t_2 = h(t_1)$, and so on, with t_k for some credit limit k being signed by the phone company. The phone will then release a tick per second in order to 'pay' for the call.

9 Books

If you want to read further about the internals of block ciphers, the definitive book is *'Differential Cryptanalysis of the Data Encryption Standard'*, by Eli Biham and Adi Shamir. IBM's explanation of the DES design is in a technical report *'The Data Encryption Standard (DES) and its strength against attacks'*; Matsui's papers appear in the proceedings of Eurocrypt 93 and Crypto 94; and the design of IDEA is explained in Xuejia Lai's *'On the Design and Security of Block Ciphers'*. These books are outside the undergraduate syllabus, but are accessible without too much mathematics.

Block cipher modes of operation should be explained in any good textbook on cryptography, but not all are satisfactory in this respect. Both Schneier and Beker and Piper give an overview; Meier and Matyas *'Cryptography – a New Dimension in Computer Data Security'* has more material than most (it dates from the time when block cipher modes of operation were first being explored).

10 Problems

1. What is the unicity distance of DES encrypted ASCII text?

2. How would you attack DES reduced to three rounds? To six rounds?

3. Why is a MAC weak if neither the IV nor the message length are fixed? Show how to attack a system which uses the same key to compute a MAC on messages and then CBC encrypt them.

4. Here are five of the criteria for the design of the DES S-boxes as revealed by IBM. Estimate how many possible S-boxes meet them.

   - No output bit of an S-box should be too close to a linear function of the input bits;
   - Fixing the two outer inputs and varying the middle four causes the S-box to attain each possible output exactly once;
   - If two inputs to an S-box differ in exactly one bit, then their outputs must differ in at least two bits;
   - If two inputs to an S-box differ in exactly the two middle bits, then the outputs must differ in at least two bits;
   - If two inputs to an S-box differ in the pattern 11xy00, where x and y are arbitrary, then the two outputs must not be the same.

5. Show that collision freedom, one-wayness and pseudorandomness are not the same thing by constructing hash functions with a proper subset of these properties. In what applications might the use of such functions be dangerous?