Computer Science Tripos part 1b — Security

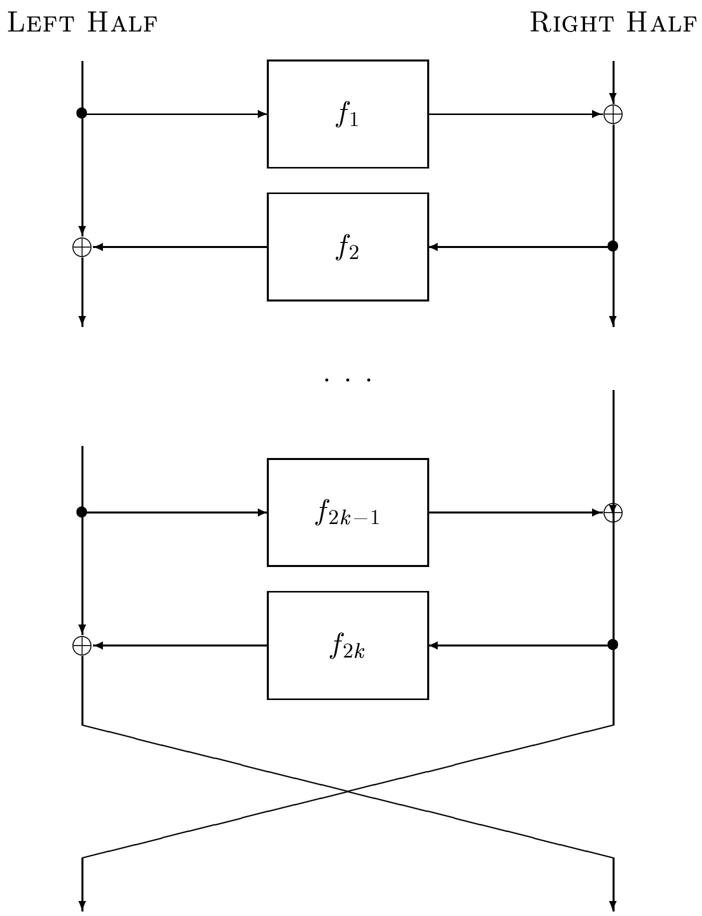
Lecture 3 — Basic Cryptology

Ross J. Anderson

1 Block ciphers

In the first lecture we mentioned two types of ciphers — stream ciphers and block ciphers. The former came first historically, and are still used in applications such as generating sequences for frequency-hop radio. However commercial applications tend to use block ciphers, and they are in some sense the more fundamental construction; so in this introduction to cryptology we will look at block ciphers and how they can be used for various purposes.

The most common basic design was invented by Horst Feistel and is named after him. A Feistel cipher has the following 'ladder' structure:



A notation which you may see for the Feistel cipher is $\psi(f_1, f_2, ..., f_{2k})$ where

f_1, f_2, ... are the successive round functions. Under this notation, the above cipher is. 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 and speed.

1.1 TEA

The Tiny Encryption Algorithm, or TEA, is a Feistel cipher invented by David Wheeler at the laboratory. It is intended for use in applications where code size is at a premium, or where it is necessary for someone to remember the algorithm and code it on an arbitrary machine at a later time. It is also useful for pedagogic purposes as it is simple to describe.

The round function is based on a shift and add operation, with the carry bit providing the nonlinearity. The golden ratio $\frac{\sqrt{5}-1}{2}$ is added in every second round to prevent chosen plaintext attacks (can you see why?). The official specification is given in C and is:

```
void code(long* v, long* k) {
unsigned long y=v[0],z=v[1],sum=0,            /* set up */
 delta=0x9e3779b9, n=32 ;                     /* key schedule constant*/
while (n-->0) {                               /* basic cycle start*/
  sum += delta ;
   y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
   z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;   /* end cycle */
              }
v[0]=y ; v[1]=z ;            }
```

Since its round function is relatively weak, with nonlinearity coming only from the carry propagation, TEA has 64 rounds. However, its simplicity means that it runs more quickly in software than many other algorithms with fewer, more complex, rounds.

1.2 DES

The Data Encryption Standard, or DES, algorithm is also a Feistel cipher. Its round function consists of an expansion, followed by key addition, followed by a pass through a row of eight S-boxes (tables of substitutions, in this case from 6-bit inputs to 4-bit outputs). This design was conditioned by the kind of digital logic available when it was designed by IBM during the early to mid 1970's. For a complete description, see Schneier or any other standard textbook.

When it was published in 1977, it aroused fierce controversy. Its critics attacked it on three general grounds.

1. The key was too short. As it is only 56 bits, an opponent who has a block of plaintext can find the key in an average of 2^{55} trials. If he could construct a parallel machine with a million (2^{20}) special processors, each of which would test a million candidate keys a second, then a key could be found in an average of 2^{15} seconds. The cost of such a machine was estimated at $1m by DES's critics, and at $200m by IBM.

2. The design principles of the algorithm were kept secret at the request of the US National Security Agency, and this led to allegations that the agency had somehow induced IBM to place a 'trapdoor' in the algorithm which would give them easy access to communications encrypted with it.

3. There should not be a standard at all, as its adoption would give an enormous incentive for opponents to develop theoretical or computational tools to attack it.

The first two allegations have now been answered. A keysearch machine built by EFF for $250,000 can find a DES key in a few days; for details see `http://www.eff.org/pub/Privacy/Crypto_misc/DESCracker/`. Previously, DES keys had been found by distributed keysearch efforts involving thousands of PCs over a period of months. It is also believed that the intelligence agencies of the USA, Britain, France and Russia (at least) have dedicated hardware machines that can find DES keys in hours or less. Thus the first criticism is justified: DES's 56 bit key is simply too short for applications requiring a high level of security, and a search for a replacement is underway.

The second criticism, that there was a trapdoor in the algorithm, has been definitively refuted. Over the last ten years, a theory of block ciphers has been developed, and this theory has shown that the DES algorithm is a well optimised design, and changing almost anything about it — the key schedule, the S-boxes, even the order of the S-boxes — makes it significantly weaker. This theory, which will be described in more detail in part 2, basically looks for upper bounds for linear and other approximations to each round and thus enables designers to compute the number of rounds required for a given resistance against attack.

The jury is still out on the third criticism. Some people argue that it is foolish for firms to 'put all their eggs in one basket' and point to the disruption which replacing DES is bound to cause to the banking industry. Others point out that the vast majority of 'home-brew' algorithms are very weak, and that in the absence of a standard most systems will end up much more vulnerable than they would with DES.

2 Example Application

An example of a block cipher application is the one-time password calculator. There are a number of these on the market: they protect against the kind of attack in which an opponent monitors a network and collects users' logon passwords. One design actually looks like, and works like, a pocket calculator. However it has an extra button, and when this is pressed it prompts for the entry of the user's personal identification number (PIN), just like an ATM.

The user now initiates a logon sequence, but rather than asking him for a password the system displays a seven figure random challenge. She types this in to her calculator too, and it then displays a seven figure response. She enters this as her password, and if it is correct, she is logged on.

Writing $\{X\}_K$ for the text X encrypted under the key K, and $|$ for concatenation, a simple way to do this calculation is:

$$\text{response} = \{\text{PIN} \mid \text{challenge}\}_{KID} \pmod{10^8}$$

Here KID is the unique key stored in the device, which might be calculated by the security server from a system master key K and the user name ID:

$$KID = \{ID\}_K$$

This is still not foolproof — an attacker who can manipulate the network might take over a session once it had been established — but devices like this can stop the common attacks on password based systems.

3 Modes of Operation

In the above example, only a single encryption was needed to calculate a response from a challenge — since DES and TEA act on blocks of 64 bits, or 16 hex digits, we could get both the PIN and the challenge into one block. However, we will not always be so lucky; we will usually find that we have to encrypt a lot more than 16 digits. How should we go about doing this?

There are five 'official' modes of operation for encrypting multiple blocks, and choosing the right one is one of the most important factors in using a block cipher securely.

3.1 Electronic Code Book

Electronic code book, or ECB, mode is the most straightforward — we just encrypt each succeeding eight byte block of plaintext with our block cipher to get ciphertext. Although straightforward, it is not very secure; much computer data is highly formatted, and these patterns will show through the encryption, letting an opponent deduce information about the plaintext.

In one popular electronic mail system, the encryption used is DES ECB with the key derived from an eight character password. When we look at a ciphertext generated by this system, we see that a certain block is far more common than the others, and this corresponds to a plaintext of nulls. Thus if you DES encrypt a null block with each password in a dictionary and sort the answers, you can break at sight any ciphertext whose password was one of those in your dictionary.

In addition, using ECB mode to encrypt messages of more than 16 digits which had an authenticity requirement — such as bank payment messages — would be a rather bad idea, as messages could be cut and spliced together along the block boundaries.

3.2 Output Feedback

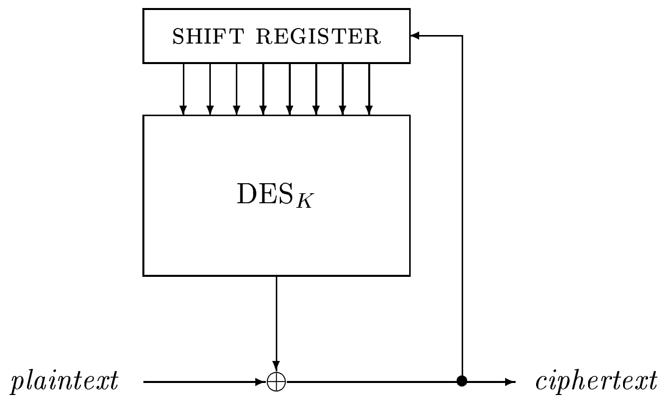
Output feedback, or OFB, mode consists of repeatedly encrypting an initial value and using this as a keystream in a stream cipher of the kind discussed in the last lecture. The i-th block of keystream will be given by

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

This also fails to provide message integrity, as attacks in depth can be carried out just as on any other stream cipher. Also, using a block cipher in OFB mode will not be as efficient as a proper stream cipher design; but it may be convenient where you have equipment with block ciphering hardware already installed and you want to generate pseudorandom numbers with it.

3.3 Cipher Feedback

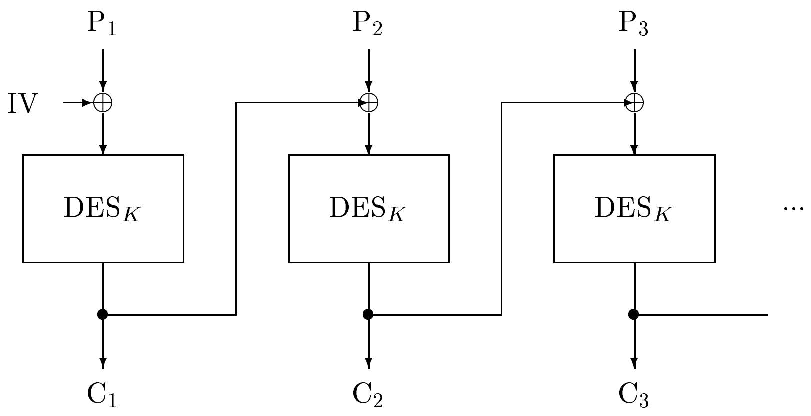
Cipher feedback, or CFB, mode is a rather inefficient use of our encryption algorithm: we use our block cipher to encrypt the last 64 bits of ciphertext, and then exclusive-or the next plaintext bit with a single bit of the output. Thus each round of the block cipher encrypts 1 bit rather than 64:



However, the resulting cipher is self-synchronising — even if we get a burst error and drop a few bits, the system will resynchronise after 64 bits. There are some applications (especially military ones) where this property is invaluable.

3.4 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.

3.5 Message Authentication Code

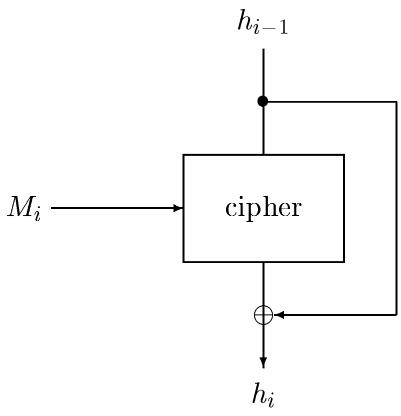
A block cipher can be used to calculate a message authentication code, or MAC. The MAC is simply the last ciphertext block returned when the message is encrypted using CBC mode. However, in this mode the intermediate results are kept secret in order to prevent splicing attacks.

Like CBC, the result 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 where both integrity and privacy are needed, one might 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.

3.6 Hash Function

Finally, 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:



However, we have to take account of the famous 'birthday problem' in probability theory. If we have 23 people in a room, it is just more than equally likely that two of them will share the same birthday — a result which seems highly counterintuitive at first. In fact, if you sample with replacement from a population of N objects, then you expect to get your first 'double' (object sampled for the second time) after drawing a little more than $\sqrt{N}$ samples.

This result had traditionally been used to count the fish in a lake (catch them, ring them and let them go). However, in 1978 Rabin proposed a system which used a 32-bit hash function, and in a paper the following year entitled 'How to swindle Rabin', Yuval pointed out the following attack.

Prepare 2^{17} messages N_i to the effect of 'Pay Yuval $100, Signed Rabin' and 2^{17} messages M_i which say 'Pay Yuval $100,000, Signed Rabin'. These messages can be made different by varying the spacing used in line justification, for example - we only need 17 points of difference in each case. How work out $h(N_i)$ and $h(M_i)$, sort the two lists, and look for the match. Say $h(N_j) = h(M_k)$. Yuval now gets Rabin to sign $h(N_j)$, and then takes $h(M_k)$ round to the court to get a possession order on Rabin's house!

Attacks of this kind are known as birthday attacks; there is a large literature on the subject, and many cryptographic schemes have been broken by them. For this reason, a cipher such as DES is too small to use in the above hashing scheme: with 64 bit hash values, we could find collisions with an effort of about 2^{33} computations.

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 cipher's round function is a complicated mixture of the register operations available on 32 bit processors.

4 Further Applications

The first large scale commercial application of block ciphers was in managing the personal identification numbers (PINs) used in automatic teller machines. This is described in some detail in the attached article 'Why cryptosystems fail'. The basic idea is that a block cipher (DES) is used to derive the PIN from the customer account number, and a hierarchy of keys (bank–ATM, bank–VISA, and so on) ensures that customer PINs are encrypted at all times in the system except when they are inside tamper-resistant cryptographic hardware, either in the bank's computer room or in the ATM itself.

There have been many successful attacks on ATM systems, but none of them (to my knowledge) have involved breaking a DES key. Most are due to procedural failures (such as theft of cards and PINs in the post), implementation blunders (such as 'test dispense' facilities on cash machines that could be abused) protocol design errors (such as putting encrypted PINs on customer cards without including the account number in the encryption) and system design errors (such as printing all the information needed to forge a card on the ATM ticket).

4.1 Authentication protocols

Another application is in authentication protocols. Where two people — conventionally called Alice and Bob — want to communicate, they may use a trusted third party, Sam, to effect an introduction. Both Alice and Bob share a key with Sam when they join the system; when they wish to communicate, they ask him for a key encrypted under the key he shares with Alice and also under the key he shares with Bob. Writing $\{P\}_K$ for the result of encrypting the plaintext P with the key K, we could describe this protocol as

$$\begin{array}{ll} A \rightarrow S: & A, B \\ S \rightarrow A: & \{A, B, K_{AB}, T\}_{K_{AS}}, \{A, B, K_{AB}, T\}_{K_{BS}} \\ A \rightarrow B: & \{A, B, K_{AB}, T\}_{K_{BS}}, \{M\}_{K_{AB}} \end{array}$$

Designing such protocols is not as easy as it looks as one must take care that old keys, and keys from sessions with other principals, are not reused; and that parts of different runs of the protocol cannot be spliced together to spoof users. Observe that the curly bracket notation $\{P\}_K$ is assumed to preserve integrity as well as confidentiality; simple cipher-block-chaining is not enough as an opponent might cut and paste fragments of different ciphertexts. We might use a variable-length block cipher to implement $\{\}$, or a combination of MAC and CBC encryption using different keys.

4.2 Kerberos

A practical example of this technology may be found in Kerberos, a distributed access control system that originated at MIT but is now installed in NT v 5. Instead of a single trusted third party, Kerberos has two kinds: an authentication server to which users log on, and a ticket granting server which gives them tickets allowing access to various resources such as files. This enables more scalable access management (in a university, for example, one might manage students by colleges but file servers by departments).

Firstly, the user A logs on to the authentication server using a password. The mechanism is that client software fetches a ticket from the server which is encrypted under the user's password and which contains a session key K_{AS}. If the user now wants access to a resource B controlled by the ticket granting server S, the following protocol takes place in order to set up a key K_{AB} with timestamp T_S and lifetime L:

$$
\begin{array}{ll}
A \rightarrow S: & A, B \\
S \rightarrow A: & \{T_S, L, K_{AB}, B, \{T_S, L, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
A \rightarrow B: & \{T_S, L, K_{AB}, A\}_{K_{BS}}, \{A, T_A\}_{K_{AB}} \\
B \rightarrow A: & \{T_A + 1\}
\end{array}
$$

Thus the user asks the ticket granting server for access to B. If this is permissible, the ticket $\{T_S, L, K_{AB}, A\}_{K_{BS}}$ is created containing a suitable key K_{AB} and given to A to use. He also gets a copy of the key in a form readable by him, namely encrypted under K_{AS}. He now verifies the ticket by sending a timestamp T_A to the resource, which confirms its aliveness by sending back the timestamp incremented by one (this is a convention to indicate that the resource was able to decrypt the ticket correctly and extract the key K_{AB}).

5 Problems

1. Write the decryption code for TEA in a language other than C and test that it deciphers data encrypted using the C code in these notes.

2. Sketch a design for an IFF system. Describe your threat model, security policy, and choice of cryptographic mechanism.

3. How would you attack DES reduced to one round? To two rounds? To three rounds?

4. A single bit error is introduced by the communications channel into CBC encrypted ciphertext. When this is decrypted, how many plaintext bits will have been corrupted?

5. DES in ECB mode is used to encrypt ASCII encoded English text. How much known plaintext is needed before an opponent can build a large enough dictionary to read ciphertext? State and justify your assumptions. What is the corresponding figure for CBC mode?

6. Your opponent has invented a new stream cipher mode of operation for DES. He thinks that OFB could be improved by feeding back into the key port rather than the data port of the DES chip. He therefore sets his secret key equal to K_0, and generates a keystream by

$$K_{i+1} = \{K_0\}_{K_i}$$

   Is this better or worse than OFB?

7. How strong is MAC if neither the IV nor the message length are fixed?

8. Someone is trying to build a system using smartcards as principals and does not want to trust the terminals completely. He therefore sets out to redesign Kerberos using random challenges instead of timestamps. Using the notation N_A for a random challenge generated by A, his protocol runs:

$$\begin{array}{ll} A \to S: & A, B, N_A \\ S \to A: & \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\ A \to B: & \{K_{AB}, A\}_{K_{BS}} \\ B \to A: & \{N_B\}_{K_{AB}} \\ A \to B: & \{N_B - 1\}_{K_{AB}} \end{array}$$

   What's wrong with this?