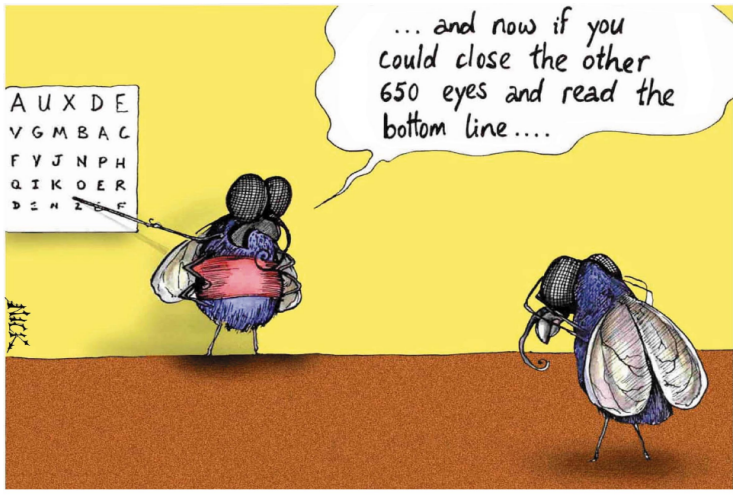



Chapter 8

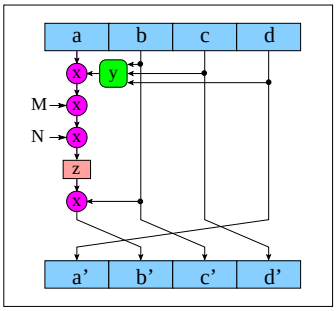
Lecture 8 - Encryption


Fly hell...



Fly hell

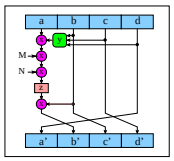
Tut6, Q1: (RFC1321)



One round of 64, *performed for each 512 bits of the input.* $\langle a, b, c, d \rangle$ *are four 32-bit registers. At the end they contain the 128 bit value of the checksum.*

The x *operations are XOR operations on 32 bit values. Note* b' *is one-way...*

Tut6, Q1: (RFC1321)



y *is one of four functions of three input 32 bit values:*

$$
\begin{aligned}
y_1 &= (b \wedge c) \vee (\neg b \wedge d) \\
y_2 &= (b \wedge d) \vee (\neg d \wedge c) \\
y_3 &= b \oplus c \oplus d \\
y_4 &= c \oplus (b \vee \neg d)
\end{aligned}
$$

z *rotates its input value.* M *is a 32-bit chunk of the input stream.* N *is a changing 32 bit constant.*

Tut6, Q2:

Show that the composite string 101101011.01000 is divided exactly by 100101.

Answer: *This is the division:*

```
                 101001000
      100101  )10110101101000
               100101
                 100001
                 100101
                    100101
                    100101
```


Tut6, Q2:

Answer: *By adding the bits 110 we get the remainder. This is the division:*

```
                 1011011.01000
   100101  )1010010111110.00000
            100101
              110001
              100101
               101001
               100101
                 110011
                 100101
                  101100
                  100101
                    1001.00
                    1001.01
                         01000
```

Tut6, Q3: MD5SUM

No matter how you change the file, approximately 50% of the bits in the hash should change - In tests I got 48%, 47% and 54%.

So randomly trying strings will give almost random hashes. The hash-space is 2^{128}, as the hash is a 128-bit value. To get a 50% chance, if each attempt is independent at 1ms/try that's about $10,790,283,070,806,014,188,970,529,155$ years. (I think)

http://www.cits.rub.de/MD5Collisions/

Project

1. For your project, Sandeep and I will email you if we are concerned about your progress report.

2. Note that the related-work, regurgitation, repeating, web-brain-dump part of your project should not be more than 25-45% of both your time spent and your final submission.

3. The intent is that you do/create/compare something that is uniquely your own groups.

Project - what is unique?

- This could be write-your-own-software, compare-two-or-more-things, local-case-study, results-of-measurements, investigate-some-problem, and so on.
- 60-70% of what you present is YOUR work, or justification/confirmation of YOUR idea.
- Not at the level of original research, but more than just regurgitation.

Project

- The project is to be presented as a 10-20 page formal paper in LNCS format. Extra appendices can take the total to (say) 50 pages.
- You may also do demonstrations and provide a CD with results/code if appropriate.

A sample formal paper:

http://www.comp.nus.edu.sg/~hugh/CS3235/typeinst.pdf

Format of a formal paper

- Title, authors, abstract
- Introduction, background knowledge, what you did, related work and summary/conclusion
- References (and then any appendices)
- Document (reference) your sources - any unreferenced copied text will result in an extraordinarily low mark.

Format of a formal paper

- LaTeX/LyX (miktex/latex2e), with this class file:

  http://www.comp.nus.edu.sg/~hugh/CS3235/typeinst.pdf

- Word - with these files:

  http://www.comp.nus.edu.sg/~hugh/CS3235/typeinst.pdf

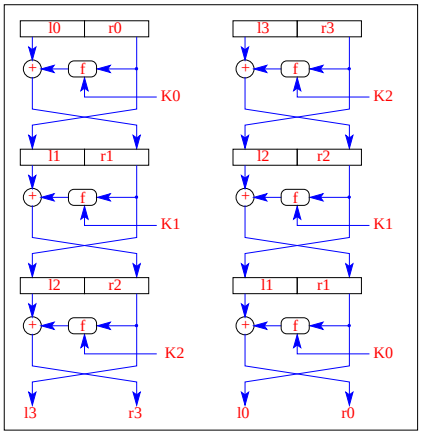
DES - Data Encryption Standard

- ✔ DES was first proposed by IBM using 128 bit keys, but its security was reduced by NSA (the National Security Agency) to a 56 bit key.
- ✔ At 1ms/GUESS. It would take 10^{80} years to solve 128 bit key encryption.
- ✔ The DES Standard gave a *business* level of safety, and is a product cipher.

DES - Data Encryption Standard

- ✔ The (shared) 56 bit key is used to generate 16 subkeys, which each control a sequenced P-box or S-box stage.
- ✔ DES works on 64 bit messages called *blocks*.
- ✔ If you intercept the key, you can decode the message.
- ✔ However, there are about 10^{17} keys.

Feistel

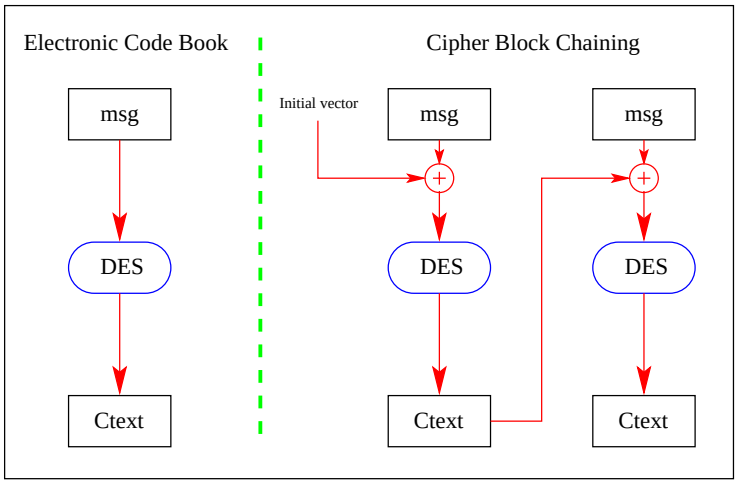


Each of the 16 stages (rounds) of DES uses a Feistel structure which encrypts a 64 bit value into another 64 bit value using a 48 bit key derived from the original 56 bit key.

DES modes of operation

- ✔ The US government specifically recommends not using the weakest simplest mode for messages, the Electronic Codebook (ECB) mode.
- ✔ They recommend the stronger and more complex Cipher Feedback (CFB) or Cipher Block Chaining (CBC) modes.
- ✔ The CBC mode XORs the next 64-bit block with the result of the previous 64-bit encryption, and is more difficult to attack.

DES modes of operation



DES software

DES is available as a library on both UNIX and Microsoft-based systems. The Java library included DES. For C, there is typically a *des.h* file, which must be *included* in any C source using the DES library:

```
#include “des.h”
//
// - Your calls
```

DES software

After initialization of the DES engine, the library provides a system call which can both encrypt and decrypt:

```
int des_cbc_encrypt(clear, cipher, schedule, encrypt)
```

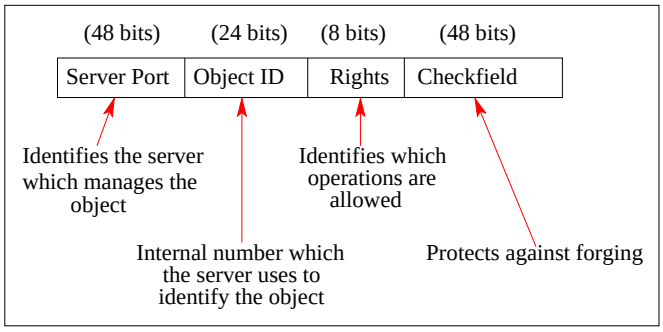
where the *encrypt* parameter determines if we are to encipher or decipher.

The *schedule* contains the secret DES key.

Case study: Amoeba capabilities

- ✔ All Amoeba objects are identified by a *capability* string which is encrypted using DES encryption. A *capability* is long enough so that you can't just make them up.
- ✔ If you have the string, you have whatever the capability allows you. If you want to give someone some access to a file, you can give them the capability string. They place this in their directory, and can *see* the file.

Case study: Amoeba capabilities



To further prevent tampering, the capability is DES encrypted. The resultant bit stream may be used directly, or converted to and from an ASCII string with the ***a2c*** and ***c2a*** commands.

AES

- Symmetric block-based FAST data encryption standard.
- Adopted by US Govt in 2000
- Algorithm specified in code form
- Uses substitution, shifts, mixing

AES algorithm (Tutorial)

```
Cipher(byte in[4*Nb],byte out[4*Nb],word w[Nb*(Nr+1)])
begin
   byte state[4,Nb]
   state = in
   AddRoundKey(state, w[0, Nb-1])
   for round = 1 step 1 to Nr-1
     SubBytes(state)
     ShiftRows(state)
     MixColumns(state)
     AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for
   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
   out = state
end
```


Public key systems

- ✔ In 1976 Diffie and Hellman published the paper "*New Directions in Cryptography*", which first introduced the idea of *public* key cryptography.
- ✔ Public key cryptography relies on the use of enciphering functions which are not *realistically* invertible unless you have a deciphering key

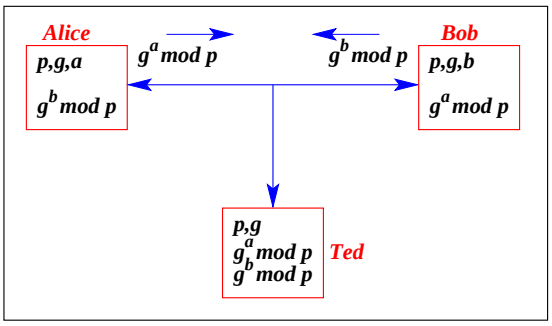
Easy to do one way - hard to do the other way.

Not realistically invertible...

- ✔ The discrete logarithm problem:
  - ✔ easy to calculate $n = g^k \bmod p$ given g, k and p,
  - ✔ hard to calculate k in the same equation, given g, n and p.

Diffie-Hellman key agreement

Two separated users *create* and *share* a secret key. A third party is not realistically able to calculate the shared key.



Knowledge different

- All participants know two system parameters p, and g
- Alice and Bob each have a secret value (Alice has a and Bob has b)
- Alice and Bob each calculate and exchange a public key (g^a **mod** p for Alice and g^b **mod** p for Bob).
- Ted knows g, p, g^a **mod** p and g^b **mod** p, but not a or b.

Diffie-Hellman key agreement

Both Alice and Bob can now calculate the value g^{ab} mod p.

1. Alice calculates $(g^b$ **mod** $p)^a$ **mod** $p = (g^b)^a$ **mod** p.
2. Bob calculates $(g^a$ **mod** $p)^b$ **mod** $p = (g^a)^b$ **mod** p**.**

And of course $(g^b)^a$ **mod** $p = (g^a)^b$ **mod** $p = g^{ab}$ **mod** p which is the shared key.

Diffie-Hellman key agreement

Ted has a much more difficult problem. It is difficult to calculate g^{ab} **mod** p without knowing either a or b. The algorithmic run-time of the (so-far best) algorithm for doing this is in

$$O(e^{c\sqrt{r \log r}})$$

where c is small, but ≥ 1, and r is the number of bits in the number.

Diffie-Hellman key agreement

By contrast, the enciphering and deciphering process may be done in $O(r)$:

Bit size	Enciphering	Discrete logarithm solution
10	10	23
100	100	1,386,282
1,000	1,000	612,700,000,000,000,000,000,000

Use of Diffie-Hellman key agreement

✔ To share a DES key...