Chapter 8

Lecture 8 - Protocols

Mid semester Test

- ✔ 9th October 2003
- ✔ LT27, 14:30
- ✔ MCQ, closed book
- ✔ Covers everything up to and including today...

Last session

- Finish on error correction
- Encryption
  - Symmetric keys
    - DES
  - Public keys
    - RSA

This session

- Kerberos
- Voting
- Contract signing

Summary

- ✔ Substitution, Vigenère, index of coincidence
- ✔ DES, Feistel, modes of operation
- ✔ Public key, Diffie Hellman, RSA

Vigenère

If our keyword was BAD, then encoding HAD A FEED would result in

Key	B	A	D	B	A	D	B	A
Text	H	A	D	A	F	E	E	D
Cipher	I	A	G	B	F	H	F	D

If we can discover the length of the repeated key (in this case 3), and the text is long enough, we can just consider the cipher text to be a group of interleaved monoalphabetic substitution ciphers and solve accordingly.

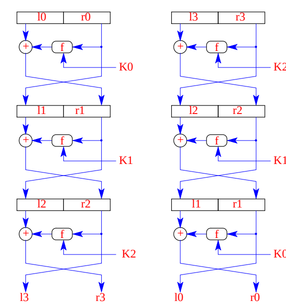
Analysis

The index of coincidence is the probability that two randomly chosen letters from the cipher will be the same, and it can help us discover the length of a key

$$IC = \frac{1}{N(N-1)} \sum_{i=0}^{25} F_i(F_i - 1)$$

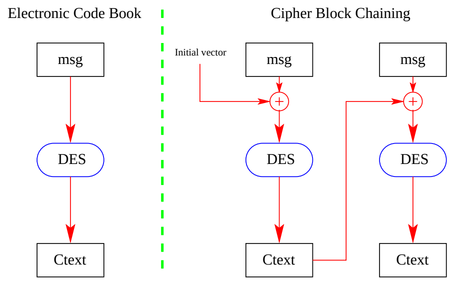
where F_i is the frequency of the occurences of symbol i and N is the length of the cipher.

DES - Feistel



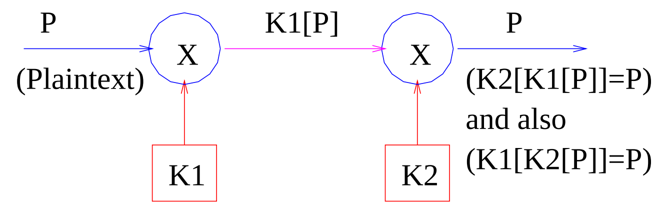
Each of the 16 stages (rounds) of DES uses a Feistel structure which encrypts a 64 bit value into another 64 bit value using a 48 bit key derived from the original 56 bit key.

DES modes of operation

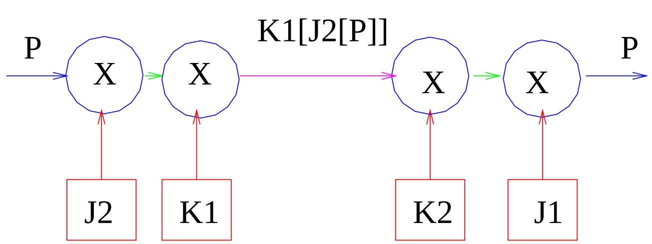


Public key systems

✔ Public key cryptography relies on the use of enciphering functions which are not *realistically* invertible unless you have a deciphering key.

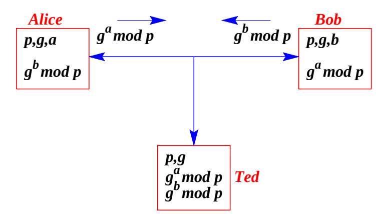


Authentication



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.



RSA coding algorithms

The four processes needed for RSA encryption:

1. Creating a public key
2. Creating a secret key
3. Encrypting messages
4. Decoding messages

Uses of encryption

1. Generating encrypted passwords with 1-way functions
2. Checking integrity by appending digital signature
3. Checking the authenticity of a message.
4. Encrypting timestamps with messages to prevent replay attacks.
5. Exchanging a key.

Protocols

Systems in which the protocol plays a large part:

1. Kerberos protocol for distributing keys
2. Voting protocols
3. Contract signing protocols

These three protocols are by no means the only ones.

Other examples

- ✔ Key distribution
- ✔ Clipper
- ✔ Oblivious transfer, in which two parties can complete a joint computation, without either party revealing any unnecessary data.

Kerberos/Cerberus

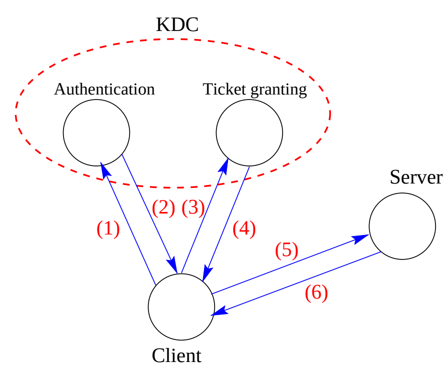
Kerberos

- ✔ Network ***authentication*** protocol.
- ✔ Strong authentication for client/server applications using public key cryptography.
- ✔ Kerberos is freely available in source form
- ✔ Kerberos is also available in commercial products.
- ✔ Client can prove its identity to a server (and vice versa) across an insecure network connection.

Kerberos

- ✔ After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.
- ✔ Must have a Key Distribution Center (KDC)
- ✔ Kerberos uses Needham-Schroeder protocol.

Kerberos



Kerberos

When a client first authenticates to Kerberos, she:

1. Talks to KDC, to get a *Ticket Granting Ticket*
2. Uses that to talk to the *Ticket Granting Service*
3. Uses the ticket, to interact with the server.

This way a user doesn't have to reenter passwords every time they wish to connect to a Kerberized service. If the Ticket Granting Ticket is compromised, an attacker can only masquerade as a user until the ticket expires.

Kerberos protocol

✔ Two sorts of credentials used, ***tickets*** and ***authenticators***.

✔ A *ticket* $T_{c,s}$ contains the client's name and network address, the server's name, a timestamp and a session key. This is encrypted with the server's secret key (so that the client is unable to modify it).

✔ An *authenticator* $A_{c,s}$ contains the client's name, a timestamp and an optional extra session key. This is encrypted with the session key shared between the client and the server.

Kerberos protocol

✔ A *key* $K_{x,y}$ is a session key shared by both x and y.

✔ When we encrypt a message M using the key $K_{x,y}$ we write it as $\{M\}K_{x,y}$.

Kerberos protocol

Alice wants session key for communication with Bob:

- Alice sends message to Ted containing her identity, Ted's TGS identity, and one-time value (n) : $\{a, tgs, n\}$.
- Ted responds with a key encrypted with Alice's secret key (which Ted knows), and a ticket encrypted with the TGS secret key: $\{K_{a,tgs}, n\}K_a \ \{T_{a,tgs}\}K_{tgs}$.
  Alice now has ticket and session key: $\{T_{a,tgs}\}K_{tgs}$, $K_{a,tgs}$
- Alice can prove her identity to the TGS, as she has session key $K_{a,tgs}$, and *Ticket Granting Ticket*: $\{T_{a,tgs}\}K_{tgs}$.

Kerberos protocol

Later, Alice can ask the TGS for a specific service ticket:

- When Alice wants a ticket for a specific service (say with Bob), she sends an *authenticator* along with the *Ticket Granting Ticket* to the TGS: $\{A_{a,b}\}K_{a,tgs}\ \{T_{a,tgs}\}K_{tgs}\ , b, n$.
- The TGS responds with a suitable key and a ticket: $\{K_{a,b}, n\}K_{a,tgs}\ \{T_{a,b}\}K_b$.
- Alice can now use an authenticator and ticket directly with Bob: $\{A_{a,b}\}K_{a,b}\ \{T_{a,b}\}K_b$.

Weaknesses

Host security: Kerberos makes no provisions for host security; it assumes that it is running on *trusted* hosts with an *untrusted* network.

KDC compromises: Kerberos uses a principal's password (encryption key) as the fundamental proof of identity.

Salt: This is an additional input to the one-way hash algorithm.

Voting protocols

A voting protocol is one in which

- independent systems vote in a kind of election, and
- afterwards we can check that the vote was correct.
- Each voter is only allowed a single vote, and
- the system should be corruption-proof.

Voting protocols

Example with Alice, Bob and Charles (!), who vote and then encrypt and sign a series of messages using public-key encryption. For example, if Alice votes v_A, then she will broadcast to all other voters the message

$$R_A(R_B(R_C(E_A(E_B(E_C(v_A))))))$$

where R_A is a random encoding function which adds a random string to a message before encrypting it with A's public key, and E_A is public key encryption with A's public key.

Voting protocols

- ✔ Each voter then signs the message and decrypts one level of the encryption.
- ✔ At the end of the protocol, each voter has a complete signed audit trail and is ensured of the validity of the vote.

Tossing a coin

- ✔ Alice and Bob want to toss a coin
- ✔ Alice calculates two primes p, q and calculates $N = pq$, sends N to Bob. $N = 35 = 5 * 7$
- ✔ If Bob can factorize the number, then Bob wins a coin toss.
- ✔ Bob selects random x, and sends $x^2 \text{ mod } N = y$ to Alice. $y = 31^2 \text{ mod } 35 = 16$

Tossing a coin

Alice calculates the four square roots of 16:

- $4^2 \text{ mbox } 35 = 16$
- $31^2 \text{ mbox } 35 = 16$
- $24^2 \text{ mbox } 35 = 16$
- $11^2 \text{ mbox } 35 = 16$

This is easy for Alice, as she knows the prime factors of N. She then sends one of these back to Bob.

Tossing a coin

- ✔ If Bob receives x or $-x$, then he learns nothing, but
- ✔ if Bob receives either of the other values, he can add this to x, and then find the GCD of the result with N:

$$\begin{aligned} \text{GCD}(24+31, 35) &= \text{GCD}(55, 35) \\ &= 5 \end{aligned}$$

- ✔ Alice is unable to tell she has divulged the factor

Oblivious transfer

✔ In an oblivious transfer, randomness is used to convince participants of the fairness of some transaction

✔ In a coin-tossing example, Alice knows the prime factors of a large number, and if Bob can factorize the number, then Bob wins a coin toss.

✔ A protocol allows Alice to either divulge one of the prime factors to Bob, or not, with equal probability.

✔ Alice is unable to tell if she has divulged the factor, and so the coin toss is fair.

Contract signing

✔ Signing contracts can be difficult.

✔ If one party signs the contract, the other may not. We have one party bound by the contract, and the other not.

✔ In addition, both may sign, and then one may say "*I didn't sign any contract!*" afterwards.

Contract signing

Oblivious transfer used for contract-signing where

- Up to a certain point neither party is bound
- After that point both parties are bound
- Either party can prove that the other party signed

Alice and Bob exchange signed messages, agreeing to be bound by a contract with ever-increasing probability

Contract signing

✔ In the event of early termination of the contract, either party can take the messages they have to an adjudicator, who chooses a random probability value (42% say) before looking at the messages.

✔ If both messages are over 42% then both parties are bound.

✔ If less then both parties are free.