Distributed Systems:
- **Security** -

Anca Jurcut
E-mail: anca.jurcut@ucd.ie

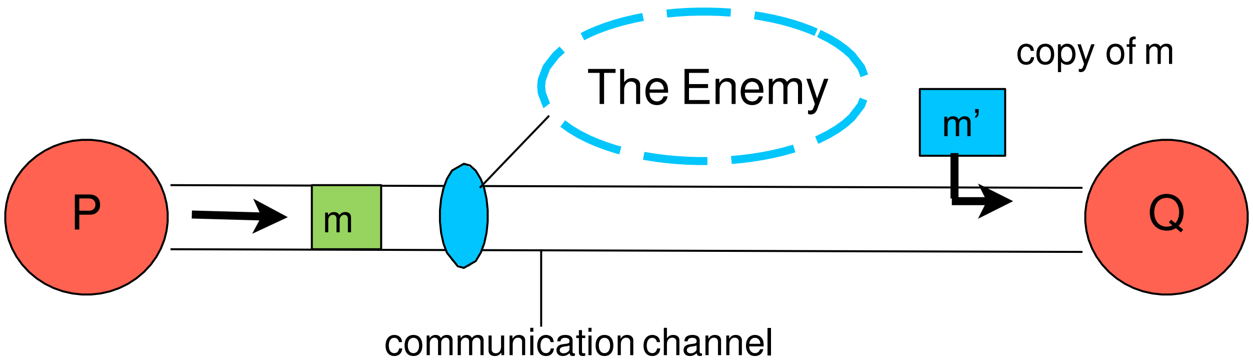
School of Computer Science and Informatics
University College Dublin
Ireland

Security in Distributed Systems

- Why are distributed systems vulnerable to security attacks?
  - promote the sharing of resources - open to external access
  - Exposed interfaces to services offered by the distributed system
  - Insecure networks
  - Hackers likely to have knowledge of the algorithms used to deploy services in distributed systems

Protecting Resources

- Access to shared resources managed by processes
- processes outline how you interact with the resource
- need to protect processes that
  - execute shared objects
  - communicate with shared processes



Security Policy v's Mechanisms

- Security Mechanisms: techniques used to protect a shared resource
  - e.g. lock used to lock a door
- Security Policies: rules which govern the use of security mechanisms
  - e.g. rule which says the door must be locked when it is not guarded
- policies are independent of the mechanism used but are just as vital
  - i.e. provision of a lock does not ensure a door is secure unless there is a policy for it's use

Security Mechanisms

- Goal of security mechanisms: to protect shared resources from:
  - Unauthorized access (hackers)
  - Malicious attacks (viruses)
  - Incorrect Usage (mistakes by valid users)
- Today we examine mechanisms for the protection of data and other resources in a distributed system
  - whilst allowing interactions between computers implied by security policies
- A key technique that underpins security is **cryptography**
  - *"The art of encoding information in a format that only the intended recipients can access."*

Today's Topics

- Security threats and Methods of attack
- Designing Secure Systems
- Cryptographic Algorithms
- Uses of Cryptography
  - secrecy and integrity
  - authentication
  - digital signatures
- Applications of Cryptography
  - digital certificates
  - access control
  - credentials
- Case Study: Kerberos

3 Security Threats

- **Leakage** - the acquisition of information by unauthorized recipients.
  - *Choicepoint*, the leading US provider of identification and credential verification services with a turnover of $1.1bn, leaked 163,000 private records in 2004/05 resulting in costs of over $55m (to date).
- **Tampering** – the unauthorized alteration of information.
  - E-Trade, an online stock-broker, lost $18m in 3 months due to hackers who snagged banking credentials which they used to transfer money to personal accounts.
- **Vandalism** – interference with the proper operation of a system without gain to the perpetrator.
  - Pakistani hackers recently vandalized the website of Mitnick Security Consulting, the company formed by Kevin Mitnick – perhaps the most famous hacker of them all:
    - http://en.wikipedia.org/wiki/Kevin_Mitnick

Methods of Attack

- In order to attack a any system, attackers need to either
  - access an existing communication channel OR
  - establish a new channel that *looks like* an authorized one

- Methods of attack can be further classified by the way in which the channel is misused...

channel = communication mechanism between processes

5 Methods of Attack

- **1. Eavesdropping** – obtaining copies of messages without authority.
  - October 2007, a German security expert presented a SMS-based Trojan that copied all SMS messages on a mobile phone and created conference calls to allow monitoring of all phone calls.
- **2. Masquerading** – sending or receiving messages using the identity of another principal without their authority.
  - E.g. e-mails claiming to be from banks that contain links to fake login pages.
- **3. Message Tampering** – intercepting messages and altering their contents before passing them on to the intended recipient.
  - Man-in-the middle attacks, such as fake web sites that mimic bank web-sites, where users to interact as normal with the bank website, but do so via an intermediary website that records all transmitted information.

5 Methods of Attack

- **4. Replaying** – storing intercepted messages and sending them at a later date.
  - Type of *man-in-the-middle* attack that is often used to maintain credentials.
  - e.g. a customer accesses their online bank account, without realizing that the HTTP requests are being recorded. At a later date, the hacker can use the record to log in to the customers bank account.

- **5. Denial of Service** – flooding a channel or other resource with messages in order to deny access to others.
  - On its launch March 2006, *Sun Grid*, which offered a sample text-to-speech service, suffered a denial of service attack.
  - In February 2000, Yahoo, Amazon, and eBay were hit by repeated distributed denial of service attacks that repeatedly made the sites inaccessible over a two day period.

Designing Secure Systems

- Worst case assumptions:
  - Exposed interfaces
  - Insecure networks.
    - fake messages, spoofed host addresses, etc.
  - Algorithms and program code available to attackers.
    - Best practice: publish, scrutinize, and rely on the keys
  - Attackers may have access to large resources.
    - Hardware is cheaper so design for the future.
- Guidelines:
  - Limit Lifetime and Scope of Secrets.
    - Limit life of passwords and secret keys
  - Minimize the *Trusted Base*.
    - Keep the number of trusted components to a minimum.
    - Try to separate applications from data and protect the data.

trusted base = portion of the system that is responsible for the implementation of it's security (including all hardware and software components that they rely on)

Basics

- *Encryption* is the process of encoding a message in a way that hides its contents.
- *Cryptography* is the study of techniques for encrypting and decrypting data.
- All modern cryptography algorithms are based on the use of *secrets* called ***keys***

key = parameter used in an encryption algorithm in such a way that the encryption cannot be reversed without knowledge of the key

- Two main approaches currently exist:
  - **Shared Keys**: both the sender and receiver know what the key is.
  - **Public/Private Keys**: the sender uses a public key to encrypt the message, whereas the receiver uses a private key to decrypt the message.

Cryptographic Algorithms

- Cryptography is about the definition of algorithms that encrypt/decrypt content.

- These algorithms can be viewed as mathematical transformations that take the form:

$$E(K1, M) = \{M\}_{K1}$$
$$D(K2, \{M\}_{K1}) = M$$

- **Symmetric algorithms = shared key algorithms**
  - since they assume that $K1 = K2$.
- **Asymmetric algorithms= public/private key algorithms**
  - since they assume that $K1 \neq K2$.

Symmetric Algorithms

- For a given key, K we define the encryption function as:

$$\boxed{E(K1,\ M)\ =\ F_K([M])}$$

- Key Design Objective:
  - $F_K([M])$ should be easy to compute
  - $F_K^{-1}([M])$ should be hard (hopefully infeasible) to compute

- Such functions are known as **one-way functions** and are essential to protect the content of $\{M\}_K$

- In general, the strength of the encryption depends on the size of K.
  - If K has N bits, then a brute-force attack requires on average 2^{N-1} iterations, and in the worst-case 2^N iterations to find K.

Asymmetric Algorithms

- Public/private key scheme (Diffie & Hellman, 1976) eliminates the need for *trust* between the communicating parties
- Exploit a specific class of one-way functions known as **trap-door functions**:
  - One way function with a secret exit
  - Easy to compute in one direction, but infeasible to compute in the other direction without a second secret.
- The pair of keys needed for an asymmetric algorithm is derived from a common root.
  - E.g. the keys are a pair of very large prime numbers, and the root is generated by multiplying those numbers together.
    - primes are multplied together - easy to compute
    - determination of original multiplicands - is infeasible

- **Basic idea:** identification of the pair of keys from the common root is infeasible without knowledge of at least one of the keys.

Block Ciphers

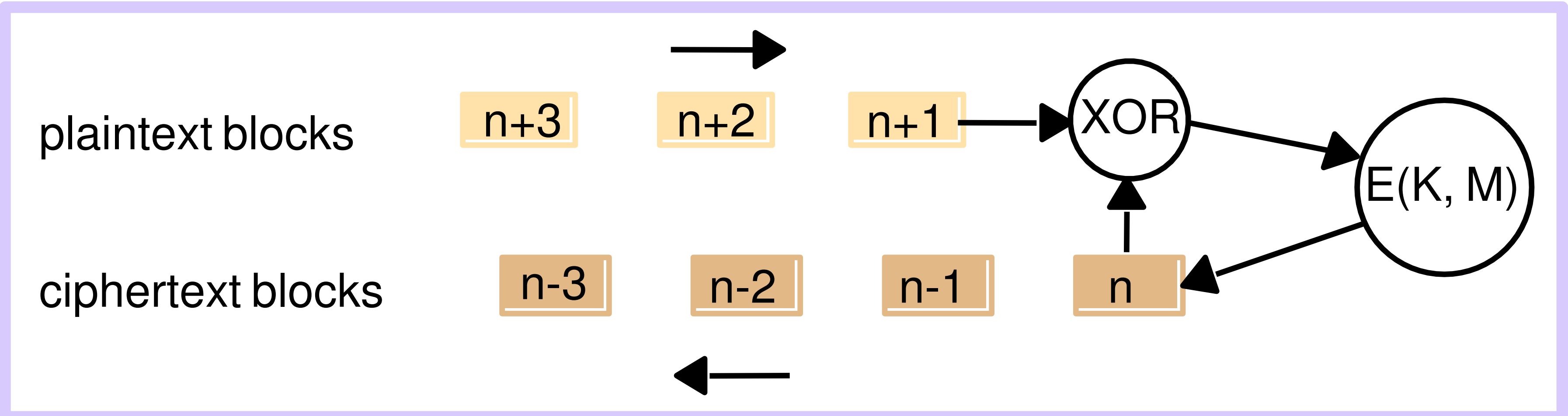
- Most encryption algorithms work on fixed size blocks of data
  - 64 bits is most common
- Each message is divided into blocks, with the last block being padded to the standard length if necessary.
- In simple block ciphers, each block is encrypted independently, and transmitted immediately after encryption.

Simple block ciphers

- Two major limitations:
  - Value of each block does not depend on the preceding blocks. hackers can easily recognise regular patterns in the cipher text and infer relationships to plaintext
  - Integrity of data in not guaranteed unless a checksum is used to validate contents of plaintext
- In order to ensure data **integrity** and improve **security** of the encrypted method, **Cipher Block Chaining (CBC)** is normally used.

Cipher Block Chaining

- Each cipher text block is combined with the preceding cipher text block using an XOR operation before it is encrypted.
  - Remember: XOR is its own inverse: two operations of it produce the original value
- On decryption, the block is first decrypted and then XOR'd with the previous encrypted block to obtain the original text.



Designing Cryptographic Algorithms

- Based on two principles outlined in Shannon's Information Theory:

- **Confusion:**
  - Non-destructive operations such as XOR and circular shifting are used to combine each block of plaintext with the key.

- **Diffusion:**
  - Dissipation of the repetition and redundancy of plain text by transposing sections of each plaintext block.

Symmetric Encrypting with TEA

- TEA: Tiny Encryption Algorithm - *(see fig 1 in next slide for algorithm)*
  - Developed in C by Wheeler and Needham, 1994
- Uses rounds of integer addition, XOR, and bitwise logical shifts to achieve confusion and diffusion of the bit-patterns in the plaintext.
  - Each plaintext block is 64 bits long (2x 32 bit integers).
  - The key is 128 bits long (4x 32 bit integers).
- Encryption consists of 32 rounds.
- In each round, the two halves of the text are repeatedly combined with shifted portions of the key and each other.
  - use of XOR and bitwise logical shifts - confusion
  - shifting and swapping - diffusion
- A non-repeating constant (*delta*) is also used to obscure the key in cases where a section of the text does not vary.

TEA Encryption

```
void encrypt(unsigned long key[], unsigned long text[]) {
   unsigned long y = text[0], z = text[1];          1
   unsigned long delta = 0x9e3779b9, sum = 0; int n;    2
   for (n= 0; n < 32;n++) {                3
      sum += delta;                    4
      y += ((z << 4) + key[0])^(z+sum)^((z >> 5) + k[1]);   5
      z += ((y << 4) + key[2])^(y+sum)^((y >> 5) + k[3]);   6
   }
  text[0] = y;      7
  text[1] = z;      8
}
```

TEA Decryption

- The decryption function is the inverse of the encrypt function.

```
void decrypt(unsigned long k[], unsigned long text[]) {
   unsigned long y = text[0], z = text[1];    1
   unsigned long delta = 0x9e3779b9, sum = delta<<5; int n;  2
   for (n= 0; n < 32; n++) {     3
      z -= ((y << 4) + k[2]) ^ (y + sum)  ^ ((y>> 5) + k[3]); 4
      y -= ((z << 4) + k[0]) ^ (z + sum)  ^ ((z>> 5) + k[1]); 5
      sum -= delta;    5
   }
  text[0] = y;     7
  text[1] = z;     8
}
```

TEA in use

```c
void tea(char mode, FILE *infile, FILE *outfile, unsigned long k[]){
   /* mode is 'e' for encrypt, 'd' for decrypt, k[] is the key.*/
   char  ch,Text[8];  int i;
   while(!feof(infile)) {
     /* read 8 bytes from infile into Text */
     i = fread(Text, 1, 8, infile);
     if (i <= 0) break;
     /*  padlast block with spaces */
      while (i < 8) { Text[i++] = ' ';}

     switch (mode) {
     case 'e':
        encrypt(k, (unsigned long*) Text); break;
     case 'd':
        decrypt(k, (unsigned long*) Text); break;
     }
    /* write 8 bytes from Text to outfile */
    fwrite(Text, 1, 8, outfile);
   }
}
```

Advantages

- Fasters than alternative symmetric algorithms such as *DES*
  - types of bitwise operations used (bitwise XOR, logical shifting) makes it easy to optimize on current hardware implementations
- 128-bit key is secure against brute force attacks

- **Aside:** DES was the U.S national standard for many years --with advances in hardware and computation power, its 56-bit key is now too small to resist brute force attacks

Asymmetric Encryption Algorithms

- depend on trap door functions of large numbers to produce keys: K_e, K_d
- encryption function performs an operation (such as exponentiation, multiplication) on M using K_e as follows:

$$E(K_e, M) = \{M\}_{Ke}$$

- decryption uses a similar function using K_d as follows:

$$D(K_d, \{M\}_{Ke}) = M$$

Asymmetric Encryption Algorithms

- General Approach:
  - Principle p generates keys K_e (made public), K_d (kept secret)
  - K_d is the piece of secret knowledge that enables p to reverse the encryption
  - Any holder of K_e can encrypt messages M to generate $\{M\}K_e$
  - **ONLY** the principals with the secret K_d can operate the trap door

Asymmetric Encrypting with RSA

- RSA: Rivest, Shamir, Adelman Algorithm
  - Developed in 1978
  - Patents held by the RSA Corporation
- Public/Private Key Cipher based on the product of two very large (> 10^{100}) prime numbers P, Q.
- Security strength relies on the fact that determining P and Q (from the resulting product N) is so difficult it is next to impossible to compute.
  - Currently requires keys > 768 bits.
  - For longer term security keys should be > 2048 bits.
- No flaws found, Widely used today

RSA Encryption

To find a key pair *e*, *d*:

1. Choose two large prime numbers, *P* and *Q* (each greater than 10^{100}), and form:
   $N = P \times Q$
   $Z = (P-1) \times (Q-1)$

2. For *d* choose any number that is relatively prime with *Z* (that is, such that *d* has no common factors with *Z*).

   Computations involved illustrated using small integer values for *P* and *Q:*
   Let $P = 13$, $Q = 17$.
   $N = 221$, $Z = 192$
   Choose $d = 5$

3. To find *e* solve the equation:

   $$e \times d = 1 \bmod Z$$

   That is, $e \times d$ is the smallest element divisible by *d* in the series:

   $$0Z + 1, 1Z+1, 2Z+1, 3Z+1, \ldots$$

   $e \times d = 1 \bmod 192 = 1, 193, 385, \ldots$
   385 is divisible by *d*
   $e = 385/5 = 77$

RSA Encryption

- To encrypt text using the RSA method, the plaintext is divided into equal blocks of length k bits where $2^k < N$
  - That is, such that the numerical value of a block is always less than N; in practical applications, k is usually in the range 512 to 1024.

k = 7, since $2^7 = 128$ (and $2^8 = 256$ which is > N)

- The function for encrypting a single block of plaintext M is:

  $\boldsymbol{E'(e,N,M) = M^e \bmod N}$

for a message M, the ciphertext is $M^{77} \bmod 221$

- The function for decrypting a block of encrypted text c to produce the original plaintext block is:

  $\boldsymbol{D'(d,N,c) = c^d \bmod N}$

RSA Encryption

- Rivest, Shamir and Adelman prove*d* that *E′* and *D′* are mutual inverses of each other:
  - That is, for all values:

$$E'(D'(x)) = D'(E'(x)) = x$$

- The two parameters *e,N* can be regarded as a key for the encryption function, and similarly *d,N* represent a key for the decryption function.

- So, we can write $K_e = \langle e,N \rangle$ and $K_d = \langle d,N \rangle$, and we get the encryption and decryption functions:

$$E(K_e, M) = \{M\}_{Ke}$$

$$D(K_d, \{M\}_{Ke}) = M$$

Uses of Cryptography

- Secrecy and Integrity
- Authentication
- Digital Signatures

Secrecy and Integrity

- Ensuring the safety and correctness of information of transmitted over networks.
  - Relies on the fact that an encrypted message can only be decrypted by someone that has the corresponding decryption key.
  - Secrecy is maintained so long as the decryption key is not compromised.
  - Encryption maintains data integrity so long as some form of checksum is also provided.
- Example:
  - Sending messages to u-boats during the second world war
- Issues:
  - How do we transmit the keys securely?
  - How do we know that the message isn’t a copy of an earlier message?

Authentication

- Supporting communication between pairs of principals:
  - The receipt of secure message implies that the sender must have the corresponding encryption key - hence deduce identity of sender (if key is only known to two the parties)
  - If the key is known to only one recipient, then that recipient is uniquely identified by the decryption key
- Example 1: Authenticated Communication with a Server
  - Let A and B be two principles, S is a third party server
  - A wishes to access file located on file server B
  - S is authenticating server that is securely managed
    - issues passwords and holds secret key for all principles in the system
  - Ticket: is an encrypted item issues by authentication server containing the identity of a principle to who it is issued and a shared key that has been generated for a new communication session

```
E.g. Ticket = {KAB, Alice}KA
```

Cryptography notations

K_A - Alice's secret key

K_B - Bob's secret key

K_{AB} - Secret key shared between Alice and Bob

K_{Apriv} - Alice's private key

K_{Bpriv} - Bob's private key

$\{M\}_K$ = Message M encrypted with Key K

$[M]_K$ = Message M signed with Key K

Authenticated communication with a server

Step 1: A contacts the server stating identity and requesting a ticket to send a message to B:

$$A \xrightarrow{\text{“Hi, I would like a ticket to contact B please?”}} S$$

Step 2: A receives a response encrypted in K_A, consisting of a ticket, encrypted in K_B, and a new shared secret key, K_{AB}:

$$S \xrightarrow{\{\{\mathtt{Ticket}\}_{KB}\ ,\ K_{AB}\}_{KA}} A$$

Step 3: A decrypts the response using K_A and sends a message to B that includes the ticket, her identity and a request R to access a file and whose content is encrypted using K_{AB} :

$$A \xrightarrow{\{\{\mathtt{Ticket}\}_{KB}\ ,\ \mathtt{Alice},\ \mathtt{R}\}_{KAB}} B$$

Step 4: B receives the message and decrypts the ticket, which includes the id of the sender and the shared key

Step 5: B uses the shared key to decrypt the message.

Authenticated communication with public keys

Step 1: A accesses a key distribution service (*KDS*) to obtain a public key for B - K_{Bpub} (called a certificate).

$$A \xrightarrow{\text{“Hi, I would like a public key for B please?”}} KDS$$

Step 2: A creates a new shared key K_{AB} and encrypts it using B's public key

$$\{K_{AB}\}_{KBpub}$$

Step 3: A sends a message to B containing the encrypted shared key and some content that is encrypted using the shared key, K_{Bpub} :

$$A \xrightarrow{(\{K_{AB}\}_{KBpub},\ \texttt{Keyname})} B$$

Step 4: B receives the message and decrypts the shared key using its private key K_{BPriv}

Step 5: B decrypts the content using the decrypted shared key K_{AB}.

Authenticated communication with public keys

- Problem: Key exchange is vulnerable to middle man attacks
  - Enemy may intercept Alice’s initial request to the KDS for B’s public certificate and send a response containing his own public key
  - He can then intercept all messages
  - We can guard against this if A ensures that B’s public key certificate is signed with a public key that she has received in a secure manner

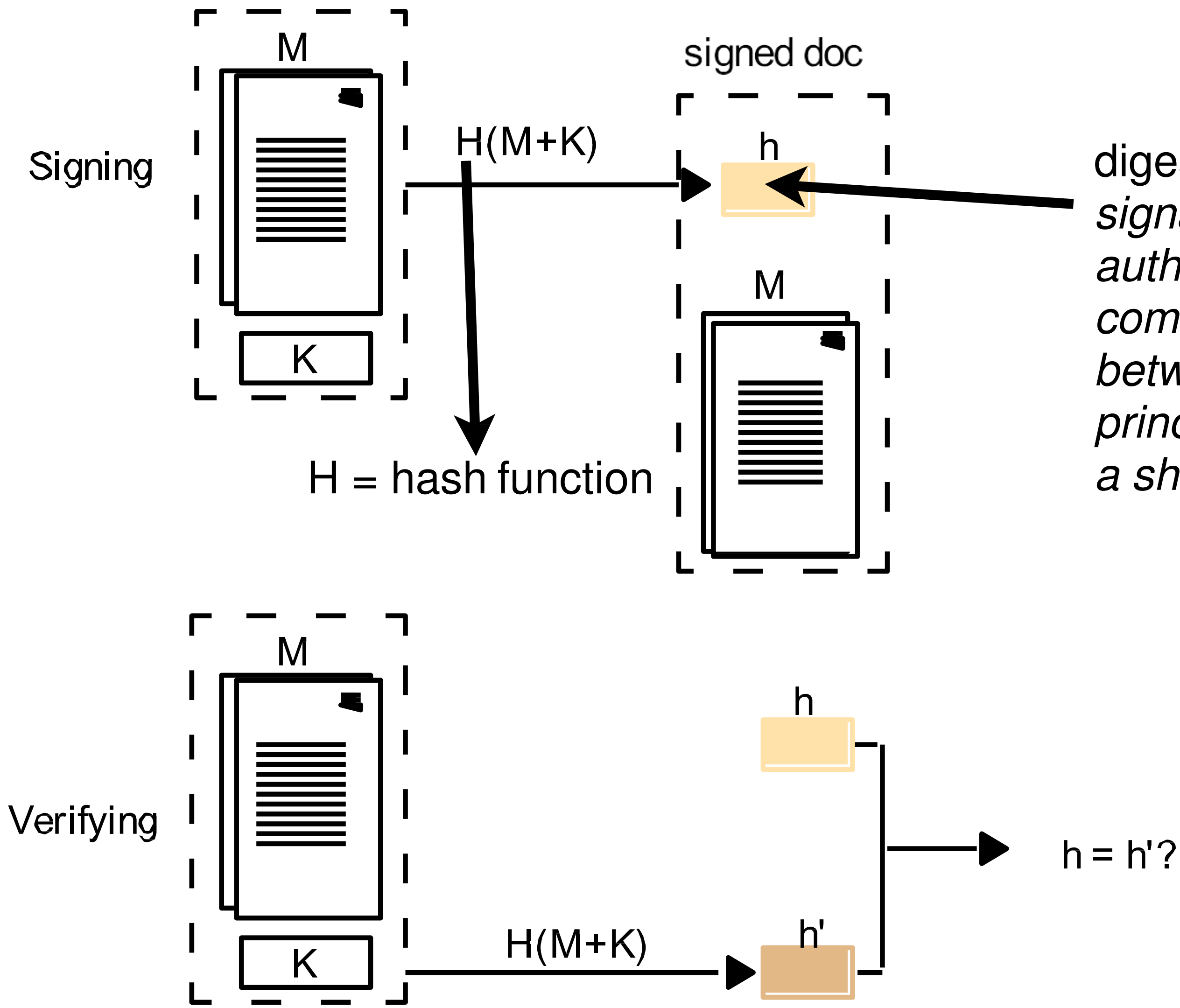
Digital Signatures

- Analogous to conventional signatures.
  - Used to verify that a document is an unaltered copy of the one produced by the signer.
- Works by encrypting the message using the signers key
  - In some cases, a compressed form of the message, called a digest, is encrypted instead. More secure
  - encrypted digest = signature
  - The signature is then appended to the message.
  - The receiver applies the same encryption technique to the message to try to recreate the digest.
  - This recreated digest is compared against a decrypted version of the digest that was sent with the message.
- This can be implemented using Public/Private keys:
  - The sender encodes the message using a private key
  - the receiver decodes the message using the senders public key.

Digest Functions

- Used to produce a fixed sized bit pattern that characterizes an arbitrary length message/document
- Digest Functions are also known as **Secure Hash Functions** written as `digest function h = H(M)`
  - These are the same as those used in P2P Systems.
- For any given Digest Function, H, it is vital that H(M) is different to H(M') for all likely pairs of messages
  - operations may not be information preserving - not meant to be reversible
  - h may not be unique - **information reducing transformation**
  - Require H(M) to be easy to compute, but difficult to reverse
  - Such functions are called **one-way hash functions**
- <u>Notice</u>: If such a function allow a scenario where H(M) = H(M'), then it would be possible for a user to send message M' but claim that they sent M.

Shared KeyDigital Signatures

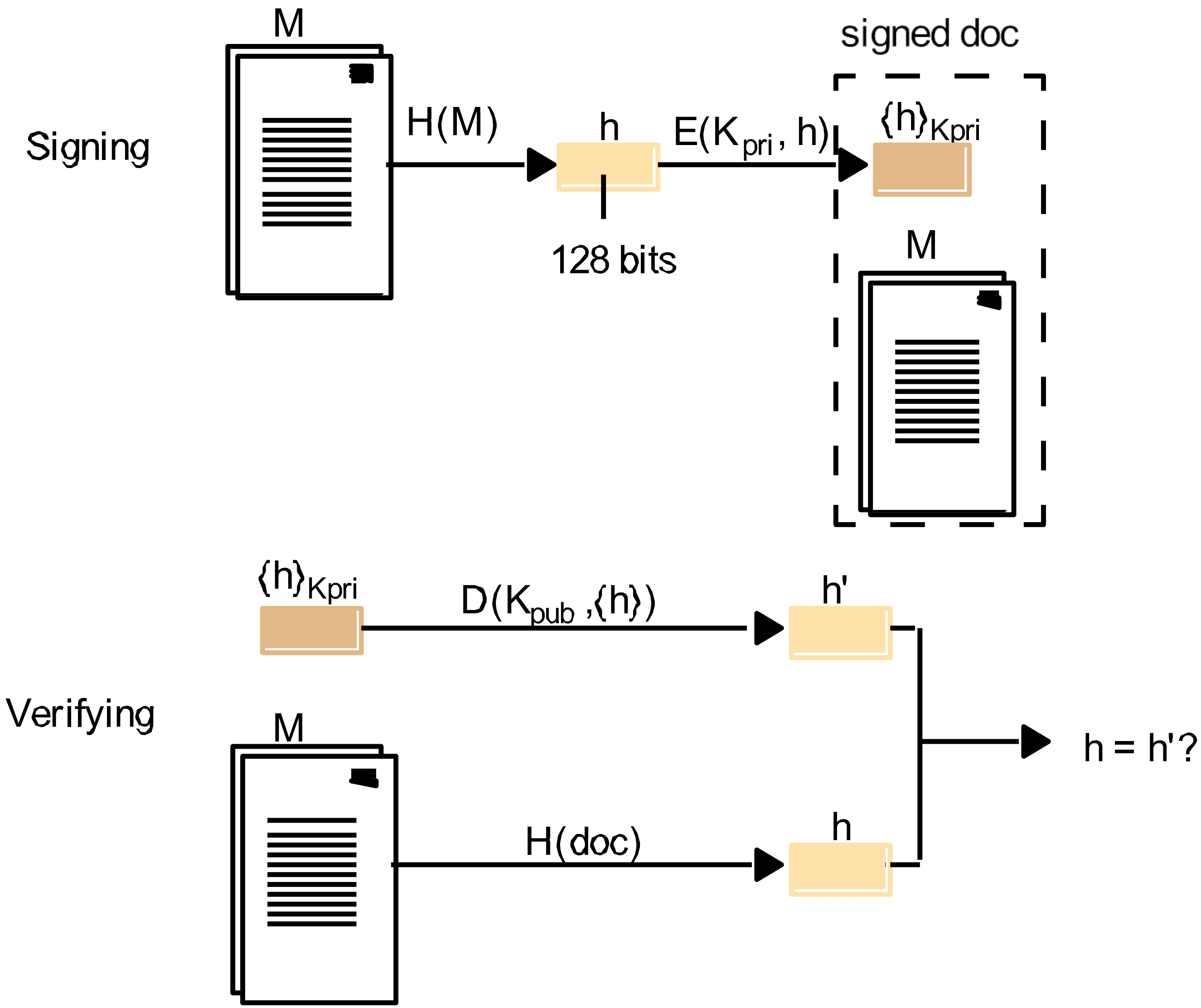


digest h is a **MAC**:
signature used to authenticate communication between pairs of principles based on a shared secret

Shared KeyDigital Signatures

- Benefits:
  - Low computational cost
- Drawbacks:
  - The signer must arrange for the verifier to receive the shared key.
  - Disclosure of the shared key reduces the security of the key.
- Improvements:
  - Delegation of the verification to a trusted third party.
    - Adds significant complexity and cost to the model.
  - Transmission of the key via a secure channel
    - In such cases the keys are known as **Message Authentication Codes (MAC's)**.
- Examples:
  - Secure Transmission of files between trusted parties
  - Secure Socket Layer (SSL) Communication.

Public KeyDigital Signatures



Public KeyDigital Signatures

- Overcomes Drawbacks of Shared Keys:
  - Public Key is used for validation, so the private key does not need to be shared.
  - Masquerading is more difficult so security can be maintained in less trusted environments.
- Example:
  - Digitally Signed Music Files
    - Signed file created from unsigned source upon purchase
    - Public key distributed to various machines for playback
    - Illegally downloaded music cannot be signed as private key is not known.
    - Can secure distribution of the public key to limit distribution.

Secure Digest Functions

- MD5: Message Digest Function 5
  - Developed by Ron Rivest in 1992
  - Uses 4 rounds where one of four non-linear functions are applied to each of sixteen 32-bit segments of a 512-bit block of source text.
  - The result is a 128-bit digest.

- SHA-1: Secure Hash Algorithm #1
  - Developed byUS National Institute for Standards and Technology (NIST)
  - Produces a 160 bit digest
  - More costly to compute that MD5.
  - Considered vulnerable since 2004.

Applications of Cryptography

- Digital Certificates
- Access Control
- Credentials

Certificates

- Digital certificates can be viewed as an attachment to an electronic message that is used to verify that a user is who they claim to be

- Issues regarding certificate management.
  - What information should a certificate hold?
  - How is a certificate created?
  - How is a certificate validated?
  - What happens when a certificate needs to be revoked?

- In general, certificates may only be created by trusted authorities (e.g. a bank, a well-known company).
  - Often they must themselves be authorized by a higher authority in order to become a trusted authority.
  - This leads to the idea of certification chains - where should it start?

X.509

- The most widely used standard for certificates.
  - Binds the public key to a named entity called the subject.
  - Also includes a digitally signed issuer

- Certificate validation consists of:
  - Obtaining the public key of the issuer (Certifying Authority)
  - Validating the their signature

1. Certificate type:	Public Key
2. Name:	Bob's Bank
3. Public Key:	K_{Bpub}
4. Certifying Authority:	Fred - The Bankers Federation
5. Signature:	`{Digest(field2+field3)}`K_{Fpriv}

Certificates

- The main problem with digital certificates is revocation.
  - To revoke a certificate, every copy of that certificate would have to be destroyed.
  - This is difficult because certificates are stored in files and files can be copied…
- Often the easy solution is to place a time limit on the certificate.
  - Once it expires, a new certificate must be obtained.
- When this is not enough, the only alternative is to inform all recipients potential that the certificate is now invalid.
  - This is a lot more complex to implement…

Access Control

- Controlling access to resources / services.
  - Remote Services are typically accessed via messages of the form:

    <op, principal, resource>
  - Where:
    - op = the operation that is to be performed
    - principal = the identity or credentials of the requestor
    - resource = the resource that the operation is to be performed on
  - E.g. <GET, anonymous, /index.htm>
  - Upon receipt of such a message, the service must first authenticate the principal.
  - Next, the service must check that the principal is allowed perform the operation…

Access Control

- While the access control is often application specific, one of two basic techniques are commonly employed:
  - Capabilities
  - Access Control Lists

- Both of these techniques build on the notion of a **protection domain**:
  - An execution environment that is shared by a collection of processes.
  - Contains a set of `<resource, rights>` pairs that outlines that `rights` of all processes to a given `resource`.
  - Examples of rights include: read, write, execute, …

Capabilities

- Each process holds a set of capabilities that identifies the access rights of that process.
- Capabilities are implemented as digital certificates that contain:
  - **Resource Identifier:** the target resource
  - **Operations:** a list of valid operations
  - **Authentication Code:** the digital signature
- Services only supply capabilities to clients when they have authenticated them as belonging to the claimed protection domain
- When a client wishes to access a resource, it sends a message of the form:

```
<op, userId, capability>
```

- Upon receipt of this message, service validates the capability and check that the capability includes the specified operation

Capabilities

- The main problems with capabilities are:
- Key Theft:
  - if a malicious user obtains a valid capability then there is nothing to stop that user accessing the resource.
- The Revocation Problem:
  - Capabilities are digital certificates and as such, once granted, are difficult to revoke.
- Solutions to these problems have been proposed that require the inclusion of:
  - Information about the holder of each capability
  - Distribution of lists of revoked capabilities.

Access Control Lists

- Each resource stores a list of `<domain, operations>` pairs that identify the `operations` that may be performed by processes from a given `domain`.
- Domains may be specified for groups of processes or individual processes as appropriate.

- When a client wishes to access a resource, it sends a message of the form:

  `<op, principal, resource>`
- For each request, the service authenticates the principal and checks to see if the operation is specified in the principals access control list.
- scheme adopted by most file systems - Unix, Windows NT - set of access permission bits associated with each file

Credentials

- A set of evidence provided by a principal when requesting access to a resource.
  - The evidence includes the trusted authority that issued the credentials.
- In its simplest form, a credential is a digital certificate that states the principals identity.
- In more complex forms, credentials can be a combination of the principals identity + a backers credentials.
  - Here the backers credential lend more weight to the principals credentials.

Summary

- Essential to protect communication channels and interfaces of systems with shared resources - hold information that might be subject to attack
  - E.g. e-mail, financial transactions
- Security protocols, policies and mechanisms are designed to protect such resources
- Two kinds of Security mechanisms:
  - Shared key/Secret key cryptography
  - Public key cryptography

Summary

- Secret key cryptography - symmetric - same key used for encryption and decryption
  - A and B share same key - can exchange encrypted information without risk
  - problem: how to exchange keys?
- Public key cryptography - asymmetric - different keys used for encryption and decryption - knowledge of one does not reveal the other
  - one key made public, anyone can send messages to the holder of corresponding private key - holder of private key can sign messages and certificates

Summary

- RSA most widely used asymmetric encryption algorithm
  - should be used with 768-bit keys or greater
- secret key encryption (symmetric) algorithms **out-perform** public key encryption (asymmetric) algorithms by several orders of magnitude
  - asymmetric algorithms only used in hybrid protocols to establish a secure channels that use shared keys for subsequent exchanges
- *Kerberos* is a well designed scheme for authenticating users and the protection of services within an organisation
  - we will now take a closer look at Kerberos...