Cryptographic Key Management

Dr Keith Martin

Information Security Group

Royal Holloway, University of London

United Kingdom

keith.martin@rhul.ac.uk

Aims of presentation

- Explain the importance of key management within a cryptographic system
- Explore the various stages of a cryptographic key lifecycle
- Identify many of the challenges involved in providing effective key management
- Demonstrate a number of different techniques for cryptographic key distribution
- Comment on some of the challenges ahead for research on key management

Sections

1. Basic cryptography (revision!!)
2. Importance of key management
3. Management of a cryptographic key
4. Key establishment
5. Public key management
6. Research challenges

1. Basic Cryptography (Revision!!)

Essence of information security

Let’s imagine first an old “computer free” office, where everything is done by telephone and paperwork.

What are the basic security processes in the physical world that help us to make security decisions about information that we receive?

Essence of information security

Now imagine a modern fully networked office environment. Let’s suppose that nobody has implemented any information security controls.

How do you identify the sender of a file?

Can anyone else read an email that you send to a colleague?

How do you sign a contract?

Is this a more secure environment than the old office?

Cryptography

Cryptography is ….

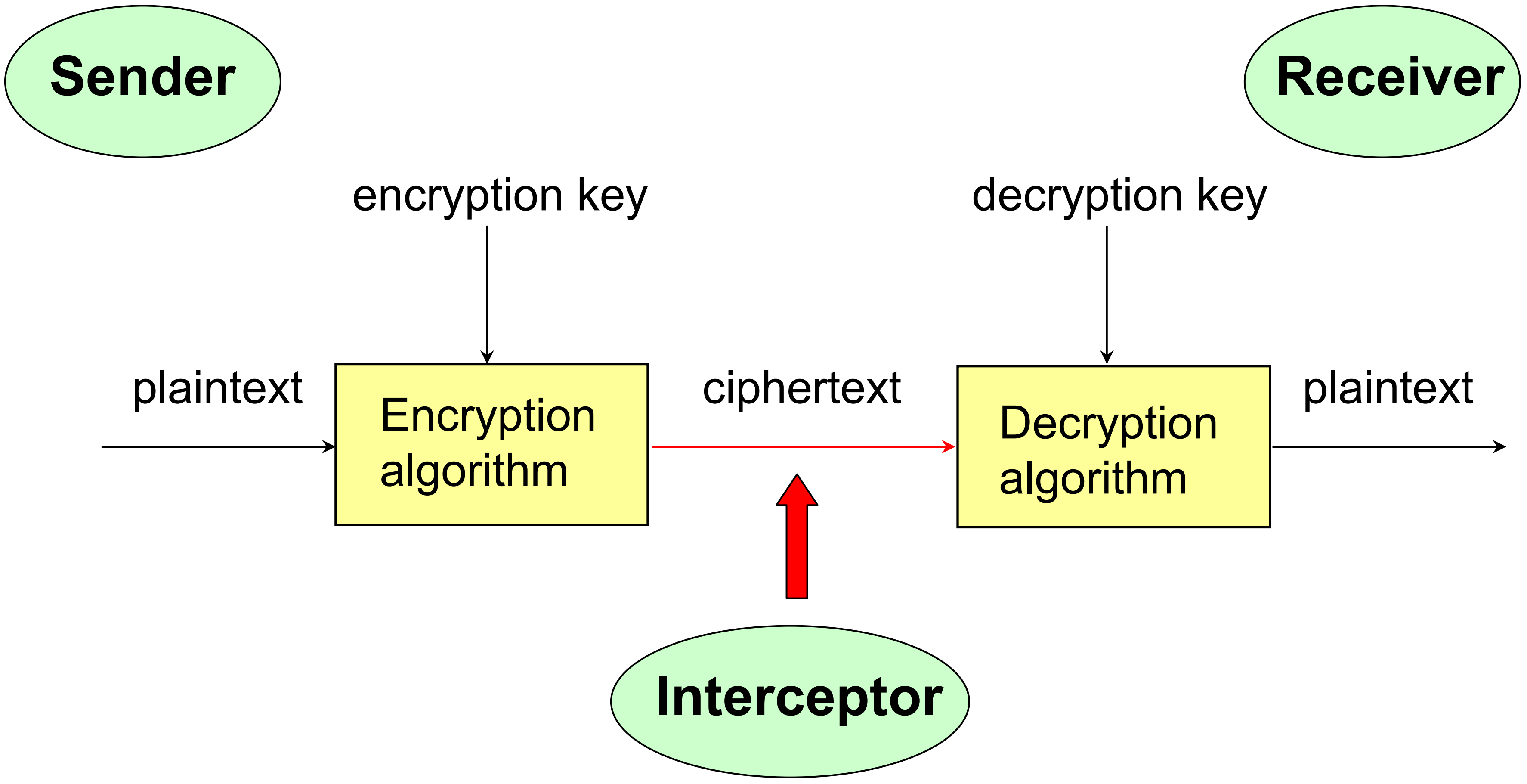
"*the art of secret writing*"

"***the miraculous cure that will solve all computer security problems***"

"*the recognised means of providing integrity, authentication and confidentiality services in an electronic environment* "

"***A toolkit of primitives that can be assembled as essential components of a security system***"

A cipher system



Three important questions

1. Can cryptography prevent a communication from being intercepted?
2. Which of the following need to be kept secret?
   a) Encryption algorithm
   b) Decryption algorithm
   c) Encryption key
   d) Decryption key
3. Does using good encryption guarantee the confidentiality of a message?

Symmetric systems

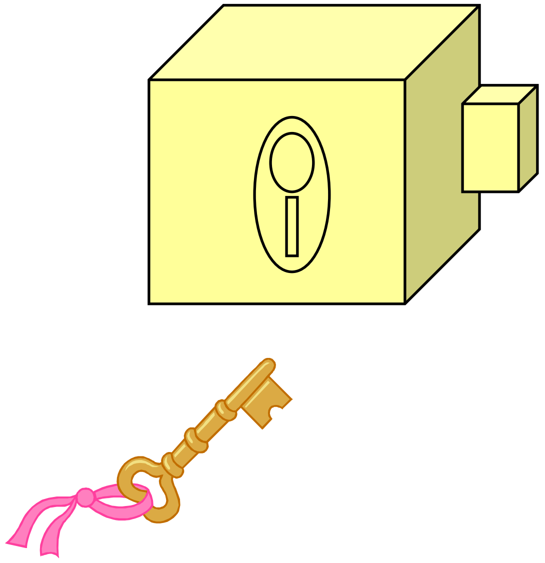
In **symmetric** cipher systems the decryption key is easily obtained from the encryption key.

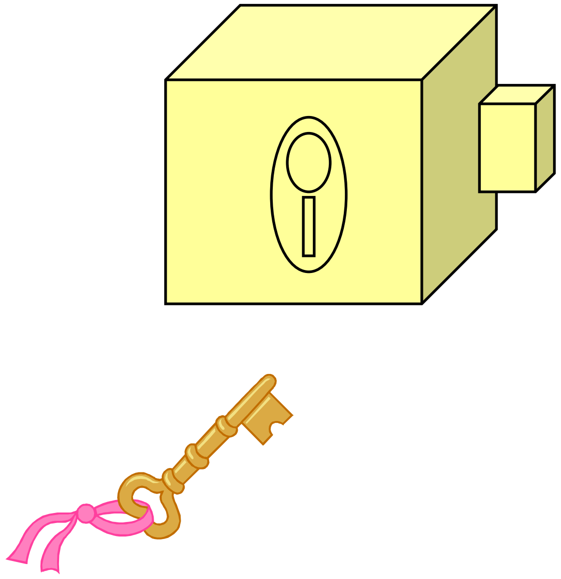
We will thus assume that in a symmetric cipher system the encryption key and the decryption key are exactly the same.

All practical cipher systems prior to the 1980’s were symmetric cipher systems. Indeed symmetric systems are still heavily use today and there is no sign that their popularity is fading.

Symmetric systems

Locking **=** **Unlocking**





Public key systems

In **public key** cipher systems it is computationally infeasible (in other words, practically impossible) to determine the decryption key from the encryption key.

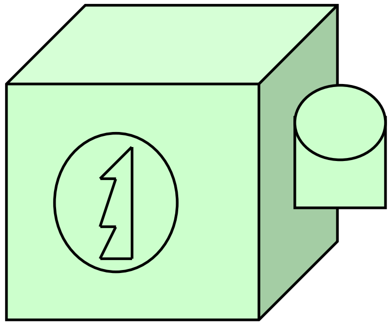
In this case the encryption key and the decryption key must be different.

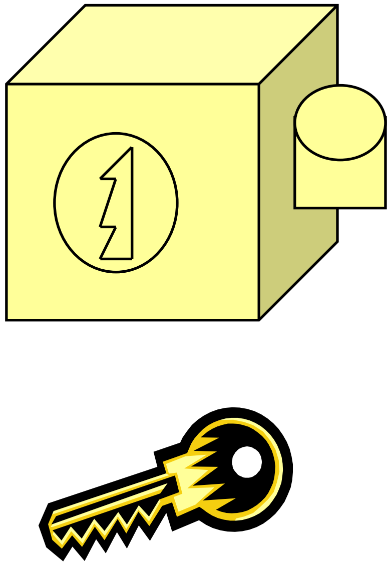
For this reason, public key cipher systems are sometimes referred to as **asymmetric** cipher systems.

Public key systems

Anyone can lock

Only a key holder can unlock





Role of the encryption key

What is the critical implication for the security of the encryption key that differs between symmetric and public key cipher systems?

Which is better?

The ability to make encryption keys public makes the concept of public key cryptography seem extremely attractive for a number of different applications.

However public key cryptography comes with its own set of problems.

Symmetric and public key cipher systems are often both implemented and used together in real systems.

Other types of service

- **Entity authentication**
  - **the assurance that a given entity is involved and currently active in a communication session (sometimes referred to as identification).**
- **Data integrity**
  - **the assurance that data has not been altered in an unauthorised (or accidental) manner since the time that the data was last created, transmitted or stored by an authorised user.**
- **Data origin authentication**
  - **the assurance that a given entity was the original source of some data (sometimes referred to as message authentication).**
- **Non-repudiation**
  - **the assurance that an entity cannot deny any previous commitments or actions (normally with respect to origin of data).**

Public Key Infrastructures

Digital signatures

Block ciphers

Stream ciphers

Message authentication codes

Bit commitment

Hash functions

One-way functions

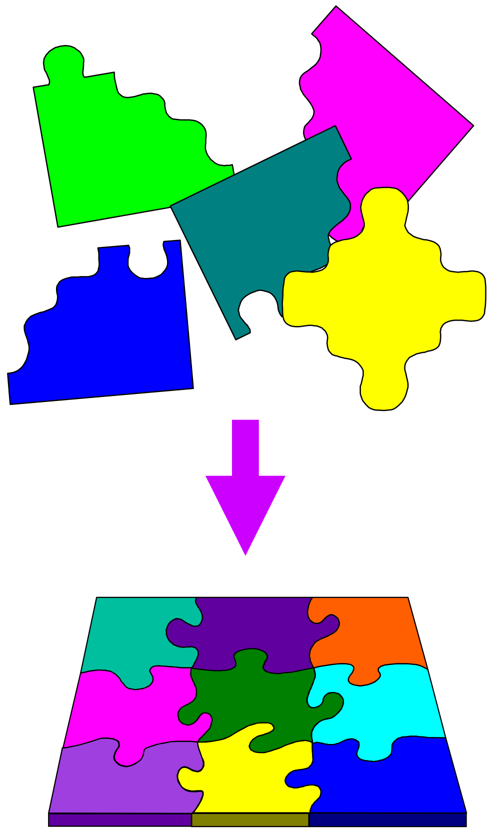
Secret sharing schemes

Zero-knowledge protocols

2. Importance of Key Management

Security is like a jigsaw puzzle

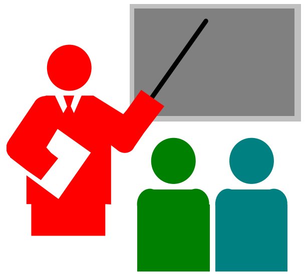
- Security Systems include:
  - *Physical Security*
  - *Access Control*
  - *Auditability*
  - *Accountability*
  - *Network Security*
  - *Security Management*
  - *Policies, Standards and Procedures*
  - *Cryptography*
  - *Disaster Recovery*



Management of Cryptographic Systems

A cryptographic security system is a form of insurance and may cost a considerable amount to purchase and to operate. Part of this cost is the management of the system, which includes:

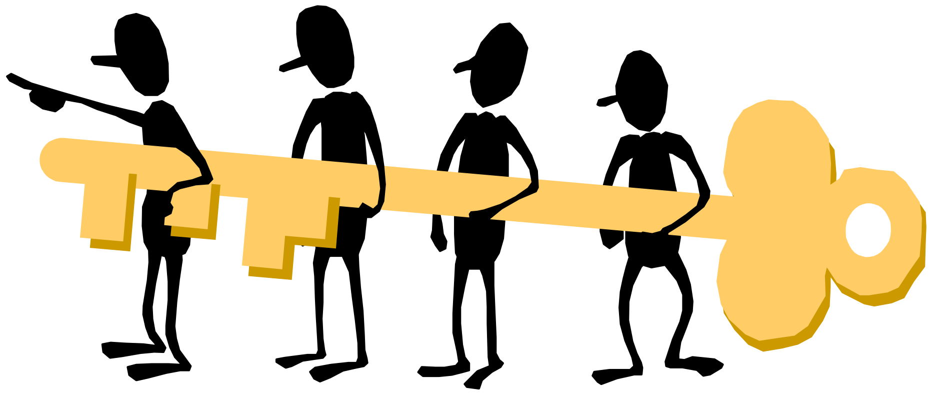
Procedures and Standards
Audit Trail Management
User Management
Token Management (e.g. smart cards)
Key Management
Access Control
Security Violations Investigation
Contingency Planning



Most organisations will have a dedicated Security Department, although all employees must take security seriously.

Key Management

“A chain is only as strong as its weakest link”



The security of the system is dependent on the security of the keys - regardless of algorithms, a security system without strong management procedures and processes has no security

What is Key Management?

ANSI X9.17 (Financial Institutions Key Management – Wholesale, 1985):

"...this standard establishes methods (including the protocol) for the **generation, exchange, use, storage** and **destruction** of these secret keys. This standard not only permits interoperability among financial institutions, but also permits interoperability between financial institutions and their wholesale customers."

Choice of Key Management System

- Usually determined by a combination of:
  - ➢ Network topology (e.g. point-to-point, many-to-many)
  - ➢ Cryptographic services (e.g. confidentiality, non-repudiation)
  - ➢ Cryptographic mechanisms (e.g. encryption, digital signature)
- Government restrictions may need to be taken into account.
- Royalty and license payments may also be relevant.
- There is usually no “right” answer!

Key Management Standards

There are many international and national standards relating to key management. For example:

ANSI X9.17 / ISO 8732
ANSI X9.24
ETEBACS (France)
AS2805.6.xx (Australia)
APACS 40 & APACS 70 (UK)
ISO 11166
ISO 11568

In addition, there are many proprietary key management systems, some closely related to standards, some loosely related to standards and others completely non-standard.

NOTE: adherence to standards does not guarantee security!!!

3. Management of a cryptographic key

Stages in key management

Key generation

Key destruction

Key establishment

Key storage

Key change

Key usage

Key generation

Symmetric keys:

- random or pseudo-random
- functions of passwords and PINs
- standard (ANSI X9.17) way to generate pseudo-random DES keys
- exclude weak and semi-weak keys
- some keys may need to be in component form

Asymmetric keys:

- typically must meet some number-theoretic requirements
- usually met by searching (so may take some time to generate key set)
- may not be practical to generate own key set

Pseudorandom number generators

- should possess the properties:
  - Uncorrelated sequences
  - Long period
  - Uniformity
  - Efficiency
- For example:
  - Blum-Blum-Shub

Generation from passwords/PINs

- PKCS#5 Password Based Cryptography Standard
- Derived key = f(P,S,C,L), where:
  - F = key derivation function
  - P = password or PIN
  - S = salt (64 pseudorandom data bits)
  - C = iteration counter (>1000)
  - L = length of derived key (bytes)

Key length

How often should a key be changed?

- Single length DES key - frequently?
- Double or triple length DES key / AES key - occasionally/never?
- RSA key - ?

RSA modulus (bits)	Exhaustive Key Search (bits)
512	56
1024	80
2048	112
3072	128

The “strength” of a key should be commensurate with the lifetime and importance of the information that is being protected. In practice, this requirement may be impossible to achieve!

Key storage

Secret keys need to be stored securely:

- inside a tamper-resistant hardware security module
- on a smart card or other token
- encrypted with another key and stored on a database

Notes:

- The third method above simply transfers the problem to the encrypting key.
- Storing plaintext keys in software is usually regarded as providing a lower level of security than storing them in tamper-protected hardware.
- Keys may need to be archived for long periods of time (e.g. 7 years in the case of the London Stock Exchange).

Hardware security modules

- Secure key storage usually requires the use of a tamper-resistant hardware security device, such as a host security module or PC security module.
- Usually some form of local master key (LMK) is stored inside the device and other keys, encrypted under the LMK, can be held outside the device, but submitted to the module when required to be used.
- In some cases, all the keys may be held inside the security module.
- The tamper-resistant features mean that all keys held inside the module will be deleted from memory in the event of an attack on the device.
- Back-up procedures for all keys held inside the security module must be in place!

Hardware security modules

- Tamper-resistant features that may be used include:

    Micro-switches
    Electronic mesh
    Potting sensitive components in resin
    Temperature detectors
    Light-sensitive diodes
    Movement / tilt detectors
    Voltage / current detectors
    Secure components ("security chips")

- Note that many security modules are in physically secure environments (such as a computer centre) and so some of the above features may be regarded as unnecessary. However, devices (say) in a retail environment may need a high level of protection.

Hardware security modules

- There are companies (such as TNO in Holland and T-Systems in Germany) that carry out evaluations of the physical protection offered by security modules.
- The ITSEC scheme (a joint initiative to evaluate security products) has not really taken off - it is expensive and time-consuming to get a product evaluated.
- The FIPS 140-2 standard provides four levels of approval for security devices, including physical security - level 4 is extremely hard to achieve.

Remark:

A paper published by Bond and Clayton (Cambridge) in 2002 showed how to extract keys from a FIPS level 4 certified device (an IBM 4758 security module), but this was really an attack on the device API rather than a physical attack on the module.

Local master keys

- Often generated and held in component form
- Components are combined inside the HSM
- Outside the HSM, the components should be stored separately in physically secure locations
- The LMK is usually a strong key (for example double-length DES or AES key)
- All other keys are encrypted with the LMK
- These other keys, encrypted with the LMK, can be stored safely outside the HSM on a database

Key change

- In all cryptographic systems there should be the facility to change keys. For instance:
    - regular updates (planned)
    - key compromise (unplanned)
- Many systems are designed so that it is extremely difficult and expensive to change certain keys. In the case of compromise of such a key, losses may include:
    - cost of distributing new key
    - cost of distributing new cards
    - cost of investigation into the compromise
    - cost of changing system and procedures
    - non-quantifiable costs
        - e.g. damage to reputation, loss of customer confidence

Key destruction

Keys, when no longer needed, must be destroyed in a secure manner. Simply deleting a key file is not sufficient.

ANSI X9.17 (Section 3.6.1):
“Paper-based keying materials shall be destroyed by crosscut, shredding, burning or pulping. Keying material stored on other media shall be destroyed so that it is impossible to recover by physical or electronic means.”

Key usage

Keys must only be used for their intended purpose. Separation of keys is therefore required. Separation is enforced using a hardware security module. For example:

• **Storage**	**-**	**store key under a specified variant of a Master Key**
• **Distribution**	**-**	**use variant of key or variant of key encrypting key for encryption**

Other techniques:

- **IBM Control Vectors**
- **Tagging of DES keys (uses the parity bits)**

Note: No universally accepted standards to achieve key separation.

Example of key misuse

Function 1: **Generate a 4 digit PIN by encrypting the account number with a PIN Key, scanning the output for the PIN and return the resulting PIN in encrypted form.**

Function 2: **Generate an 8-character MAC using a MAC Key and return the resultant value.**

Misuse: **Use Function 2 to generate a MAC over the account number, using the PIN Key. The result is an 8 character MAC, which will, with a probability of about 0.9, yield the PIN.**

Solution: **Prevent a PIN Key from masquerading as a MAC key.**

Example of key masquerade

- In many systems different key types are stored encrypted under different variants of a Storage Master Key (SMK), which (in theory) prevents a key from being misused.
- Such systems also tend to have export and import functions, to permit a key to be exported (encrypted under a Transport Key (TK)) to another system or imported (encrypted under a TK) from another system.
- In order to allow interoperability between different vendors’ solutions, variants are not usually applied to the TK.
- Hence, the “bad guy” can simply export a key of one type from encryption under a variant of the SMK to encryption under the TK and then import the same key from encryption under the TK to encryption under a different SMK variant.
- This situation is permitted by the ANSI X9.17 standard!

Example revisited

$$\begin{tikzcd}
E_{SMK(v1)}(\text{PIN Key}) \arrow[dr, "\text{Export PIN Key}"] & \\
& E_{TK}(\text{PIN Key}) \arrow[dl, "\text{Import PIN Key}"] \\
\cancel{E_{SMK(v2)}}(\text{PIN Key}) &
\end{tikzcd}$$

$E_{SMK(v2)}(\text{MAC Key})$

PIN Key now masquerading as a MAC key

TR-31 Key block

- An ANSI sub-committee is currently defining a new key block, to ensure that a key can only be used for its intended purpose.
- The key block should be usable for either key storage or key distribution.

Header (clear)	Optional Header (clear)	Key (encrypted)	Authenticator (MAC)

- Header includes key usage, mode of use, exportability, algorithm.
- Key encrypted using a variant of the storage or distribution key, in CBC mode.
- Authenticator calculated using a different variant of the storage/distribution key.
- Currently, only 3-DES supported (but extensions planned).

4. Key establishment

Key establishment

- Key predistribution
  - All keying material issued in advance on initialisation of the system
- Key distribution
  - "Trusted" entity involved in establishment of keys
- Key agreement
  - Communicating parties jointly establish keys

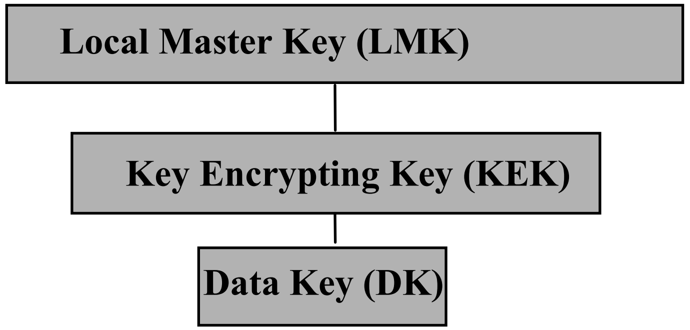
Manual key establishment

In many situations top level keys need to be handled manually. Such keys only exist outside HSMs in the form of at least two, usually three, components.

Rules:

- Each bit of the key should depend on each component
- No person is ever in possession of more than one component
- Components should be stored in separate locations

Master/Session key scheme (I)



LMK:	**double length DES key** **manual exchange, in component form** **infrequent change** **used to encrypt KEK(s) or DK(s) (but not both)**
KEK:	**optional key** **electronic distribution** **double length DES key** **used to encrypt DK(s)**
DK:	**single or double length DES key** **“working key” - e.g. Encryption, MACing, etc.** **frequent change** **electronic distribution**

Master/Session key scheme (2)

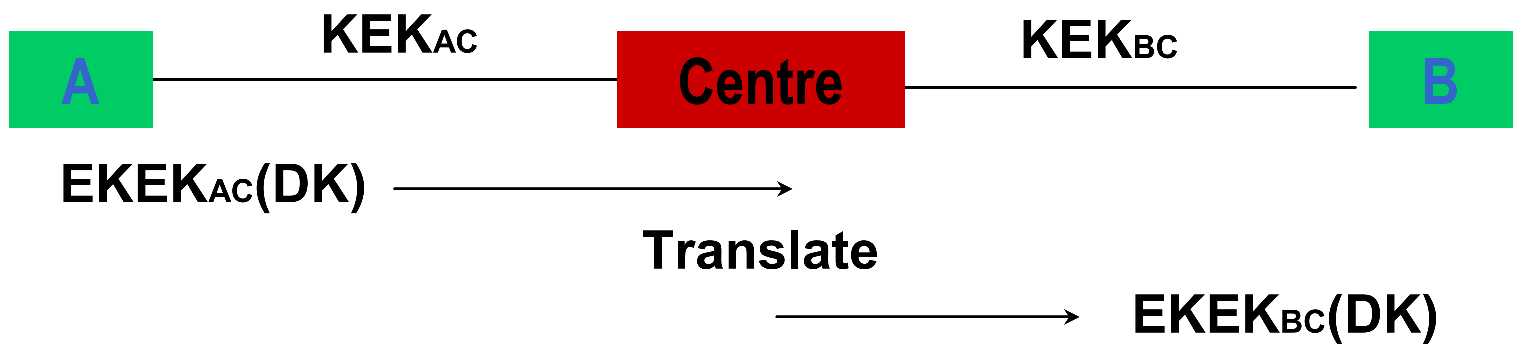
- For a simple point-to-point system, the Master/Session key scheme is fine, but becomes unmanageable for large many-to-many systems.

- In such cases a Key Distribution Centre or Key Translation Centre may be used, so that each party only has a permanent keying relation with the Centre and yet can still communicate with other parties.

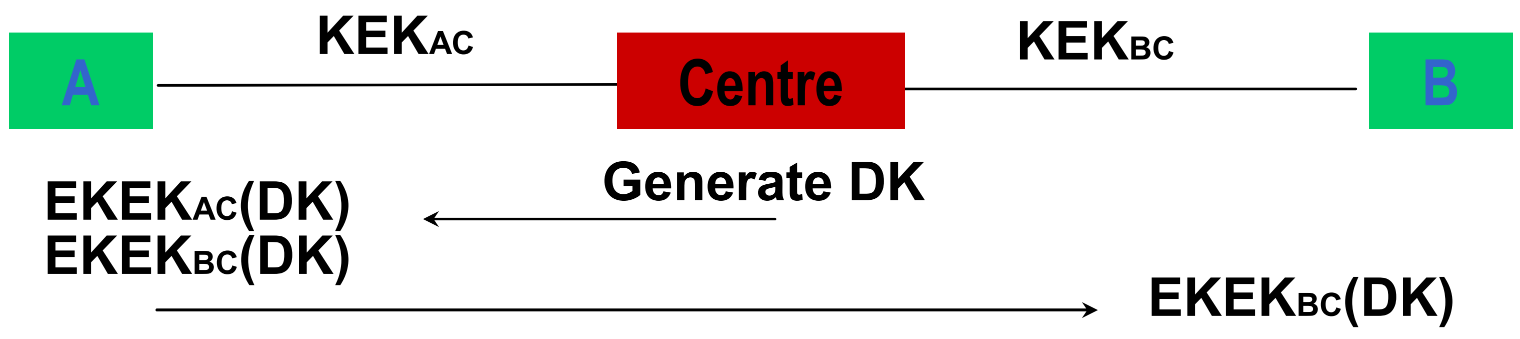
The Centre must be trusted!

Master/Session key scheme (3)

Key Translation:



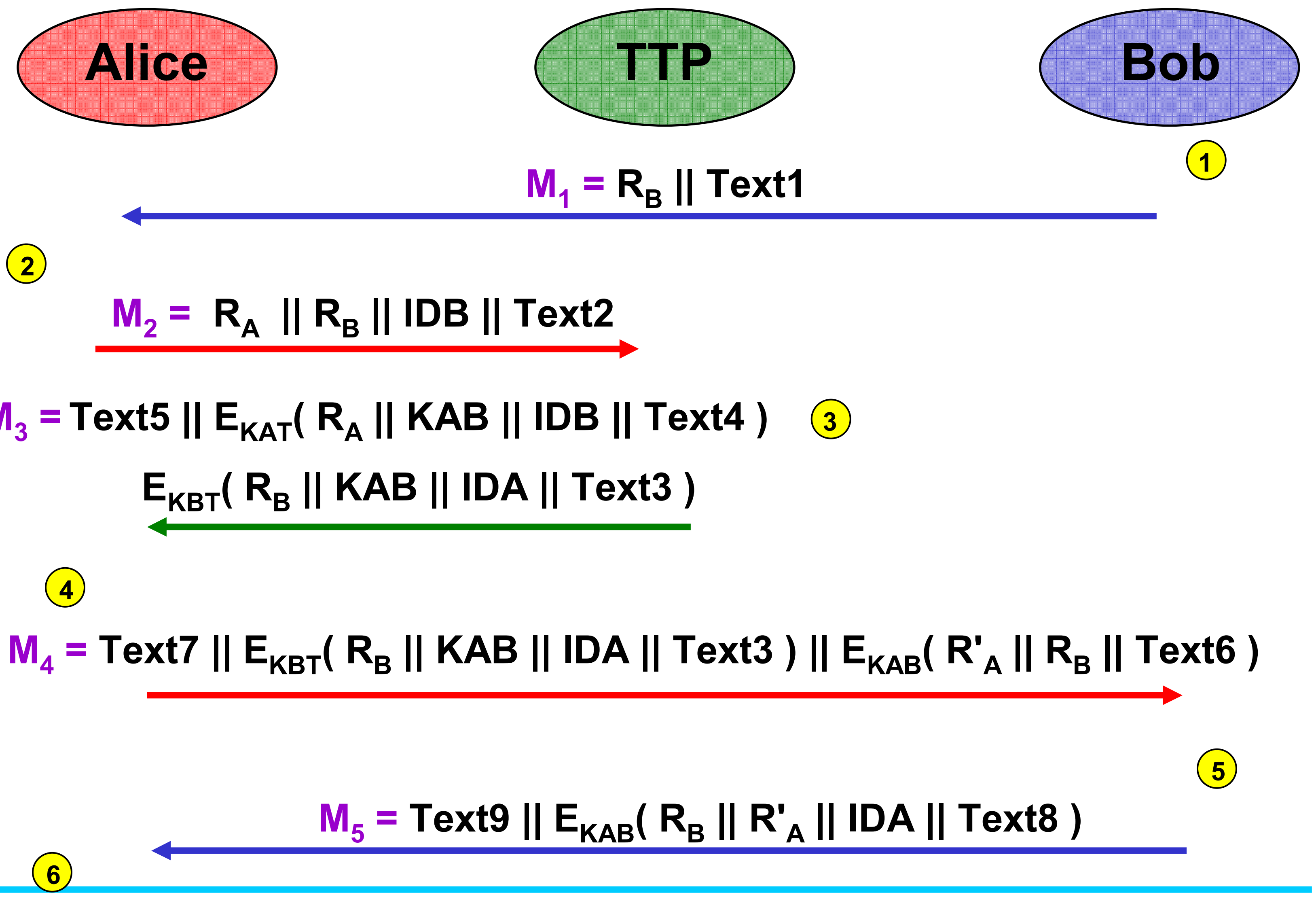
Key Distribution:



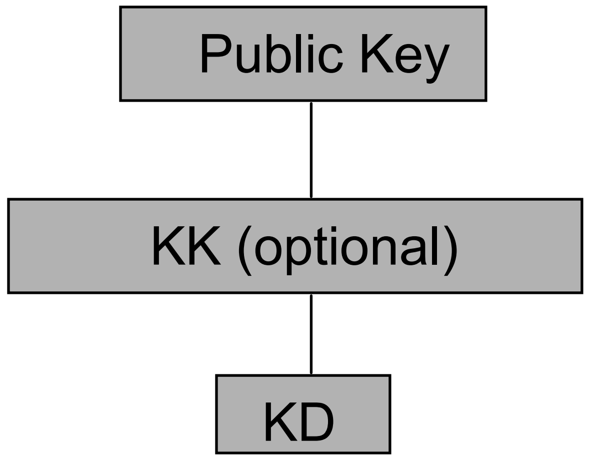
Key establishment requirements

Security requirement	Explanation
Mutual entity authentication	During the key agreement process Alice and Bob are able to verify each other's identity to make sure that they knew who they were agreeing a key with
Mutual data origin authentication	At all times during the process Alice and Bob are able to be sure that information being exchanges has come from the other party and not an attacker
Mutual key agreement	At the end of the process Alice and Bob should have agreed upon a symmetric key
Key confidentiality	The symmetric key that is finally agreed upon should at no time have been accessible to any other party than Alice and Bob
Key freshness	At the end of the process Alice and Bob should be happy that the key that they have agreed upon is a fresh one, and not one used before
Mutual key confirmation	At the end of the process Alice and Bob should have some evidence that they have both ended up with the same key and that there have been no mistakes made at either end
Joint key control	At the end of the process Alice and Bob should be happy that they both had approximately equal involvement in the choice of key, and that neither of them could have deliberately chosen a particular key

ISO 9798-2 Example 8


Alice
TTP
Bob
1
$M_1 = R_B \| Text1$
2
$M_2 = R_A \| R_B \| IDB \| Text2$
$M_3 = Text5 \| E_{KAT}(R_A \| KAB \| IDB \| Text4)$
3
$E_{KBT}(R_B \| KAB \| IDA \| Text3)$
4
$M_4 = Text7 \| E_{KBT}(R_B \| KAB \| IDA \| Text3) \| E_{KAB}(R'_A \| R_B \| Text6)$
5
$M_5 = Text9 \| E_{KAB}(R_B \| R'_A \| IDA \| Text8)$
6

Hybrid scheme

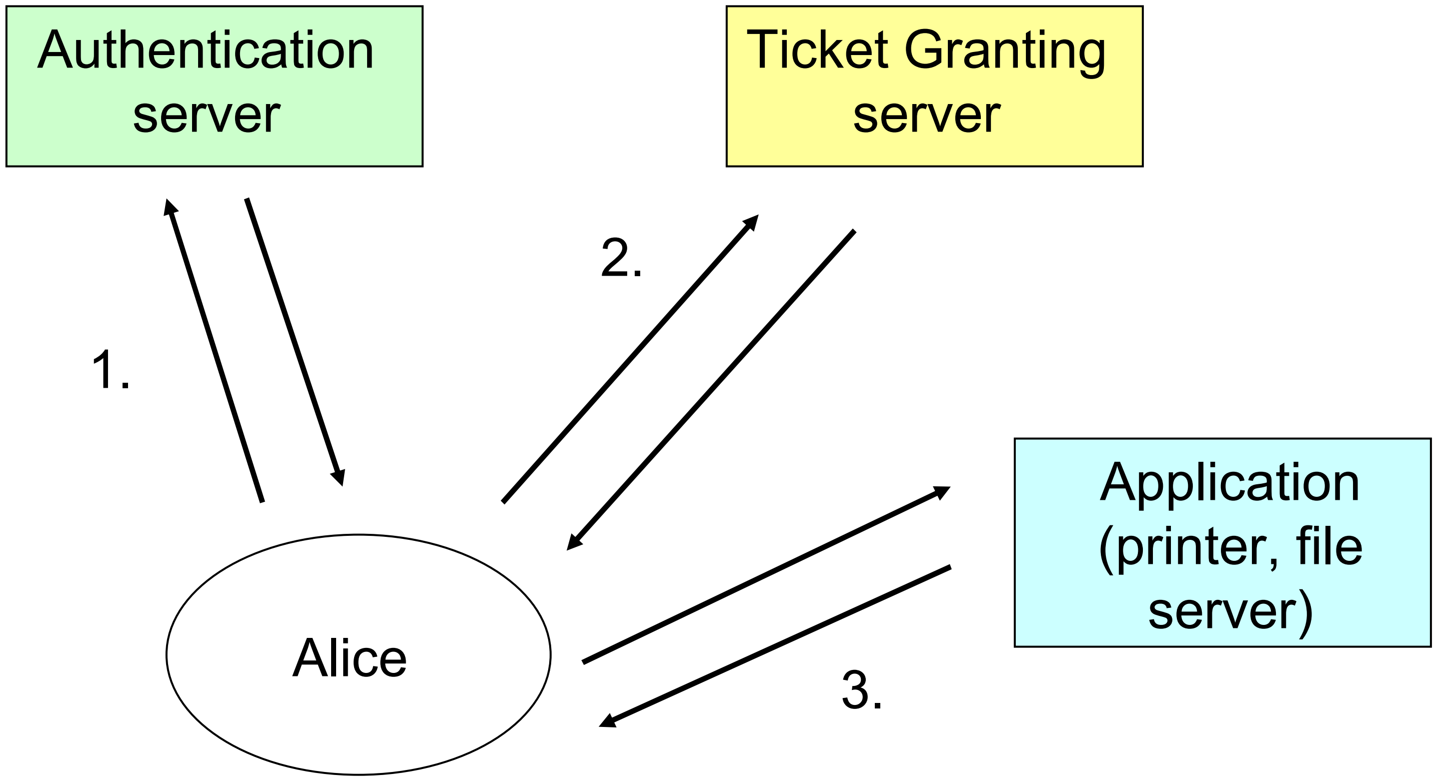


Use Public key for encrypting a symmetric KK or KD

- Particularly useful for many-to-many systems (e.g. SSL).
- Only RSA Public Keys need to be distributed - no need for secrecy, but integrity is required. This is provided via the use of a Public Key Certification Authority (CA).

Kerberos / Single sign on (1)

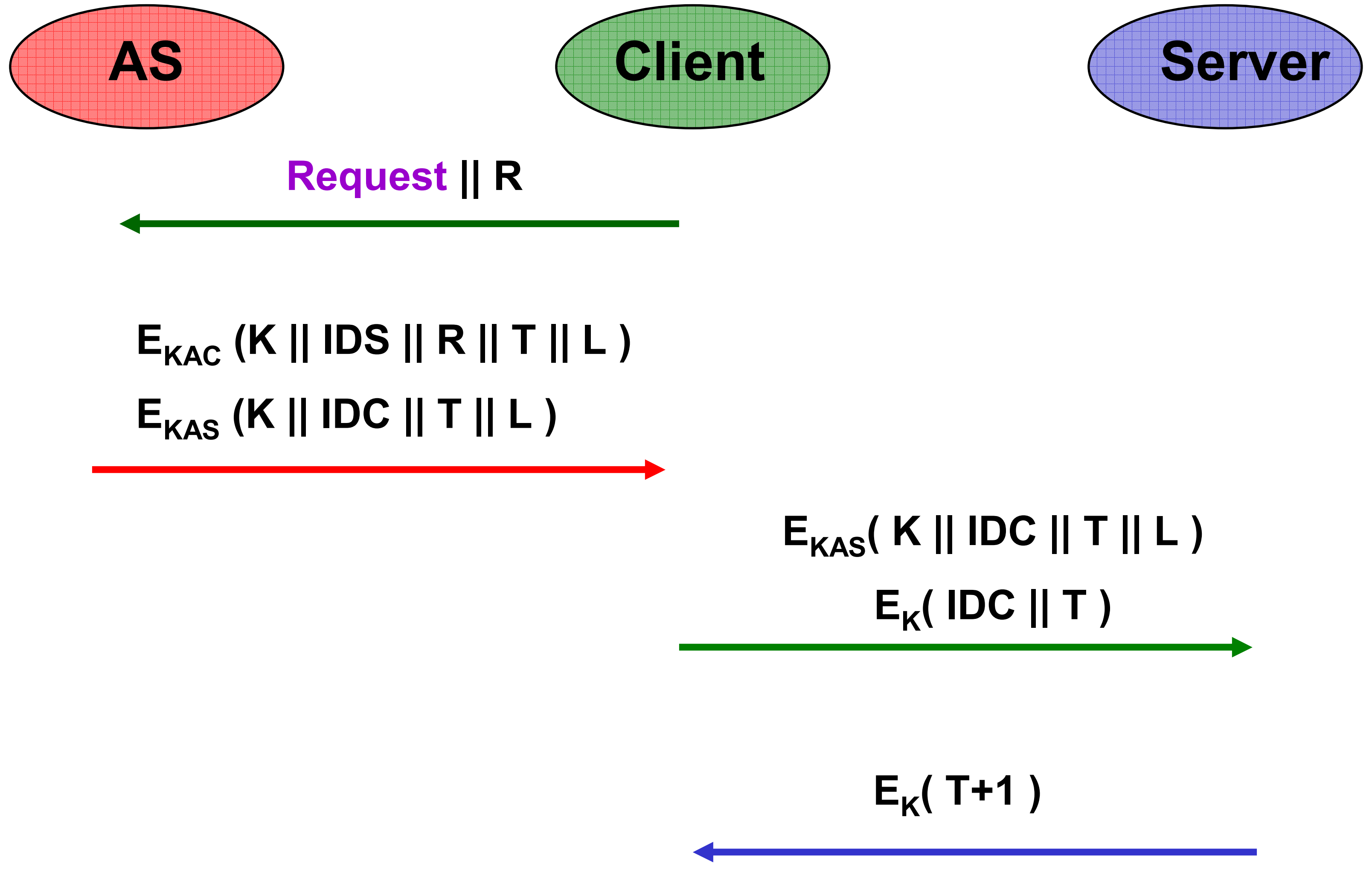
Principle: Alice uses her password to sign on once a day



Kerberos / Single sign on (2)

1. Alice gets a “daily key” KA from the authentication server
   - Based on Alice’s long term secret (password)
   - KA is stored on Alice’s machine and deleted at the end of the day
2. Alice uses KA to get application key K from the ticket granting server
3. Alice establishes a secure link with the application using K

Kerberos / Single sign on (3)


AS
Client
Server
Request || R
E_{KAC} (K || IDS || R || T || L)
E_{KAS} (K || IDC || T || L)
E_{KAS}(K || IDC || T || L)
E_K(IDC || T)
E_K(T+1)

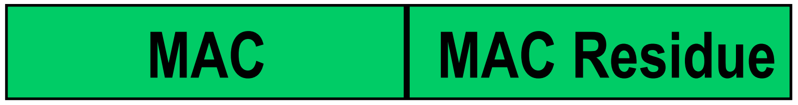
Unique Key Per Transaction (I)

- A number of schemes exist which provide automatic update of keys with every transaction.
- Thus, there are no static keys in the system.
- Particularly useful in the retail environment, where insecure terminals may be used.
- Possible recovery or resynchronisation problems.

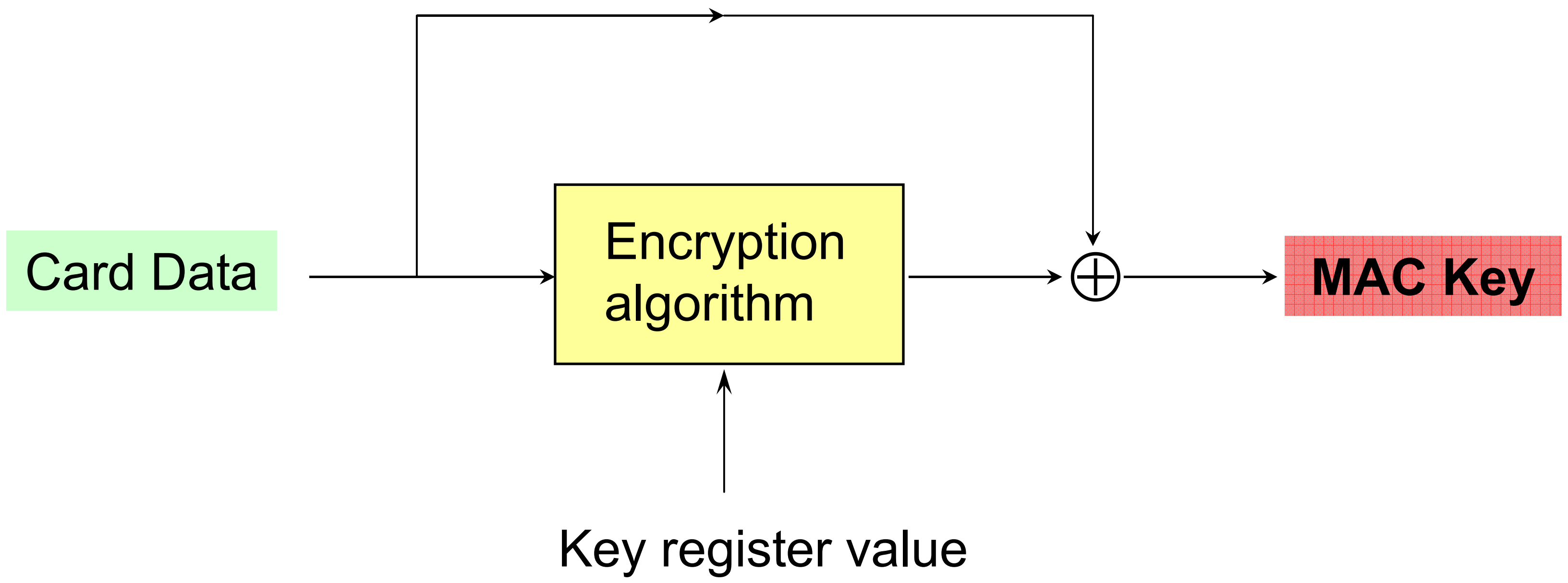
Unique Key Per Transaction (2)

Example: Racal Transaction Key Scheme (APACS 40/70)

- Based on single length DES (APACS 40) or double length DES (APACS 70).
- Terminal and Host have a key register, which is updated with each transaction.
- The transaction key(s) are derived from key register value and card data.
- The new register values are a function of old register values.
- Uses MAC Residues

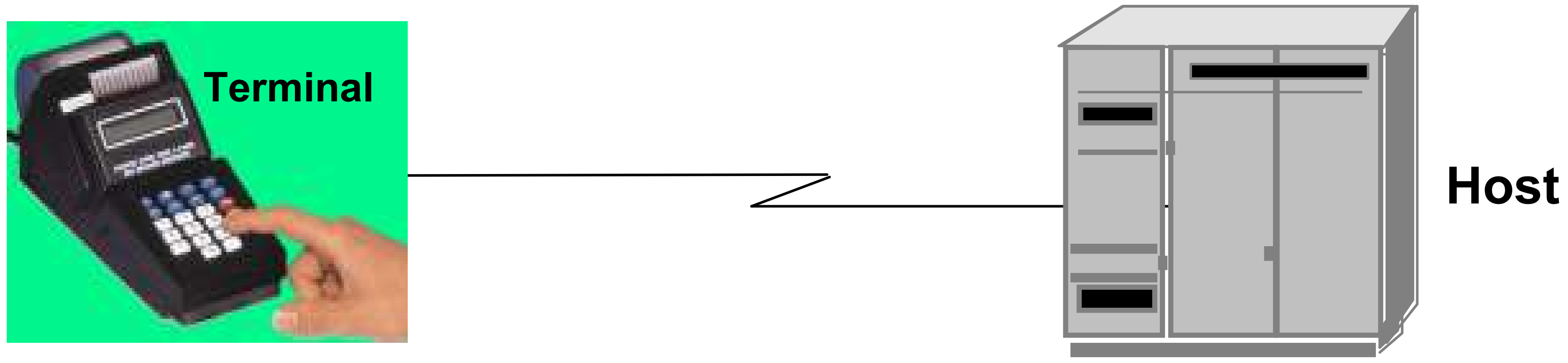


Transaction key derivation



Unique Key Per Transaction (3)

Racal Transaction Key Scheme



1. Generate transaction keys from register and card data
2. MAC Request Message and retain MAC Residue

Request Message + MAC

$$\xrightarrow{\hspace{10cm}}$$

3. Generate transaction keys from register and card data

Unique Key Per Transaction (4)

Racal Transaction Key Scheme (continued)

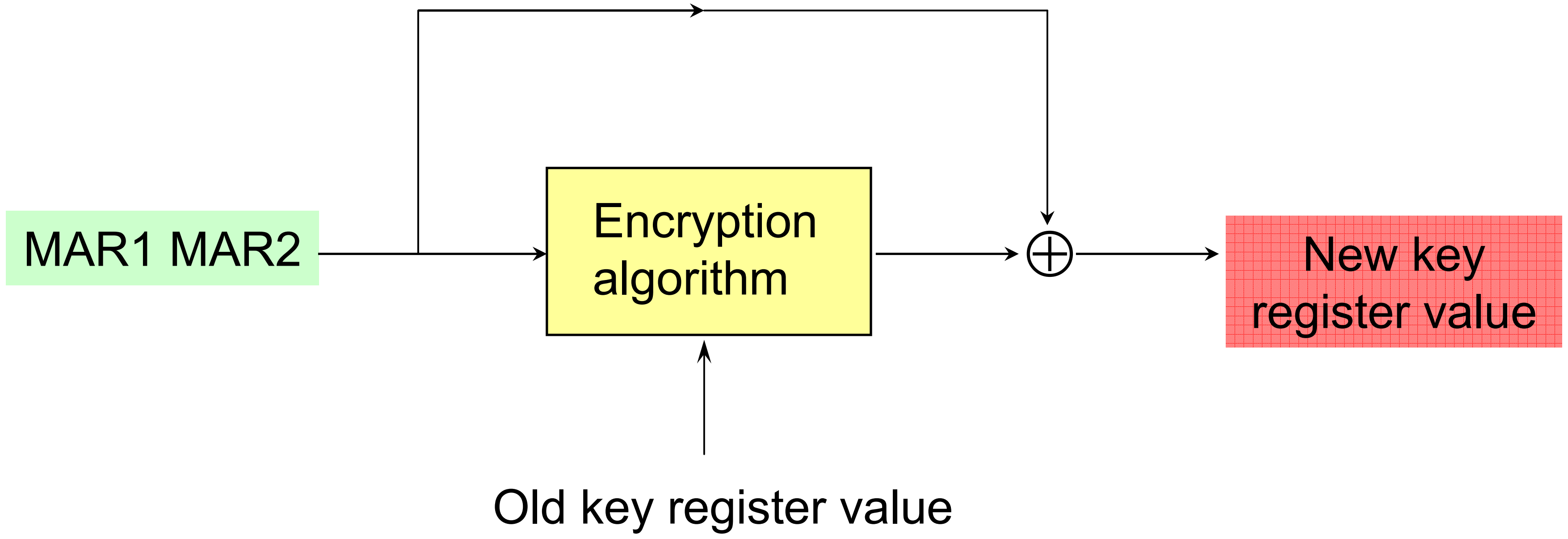
4. Validate Request Message MAC and retain MAC Residue
5. Generate Response Message MAC and retain MAC Residue
6. Update register using MAC Residues

Request Message + MAC

$$\xleftarrow{\hspace{10cm}}$$

7. Validate Response Message MAC and retain MAC Residue
8. Update register using MAC Residues

Key register update

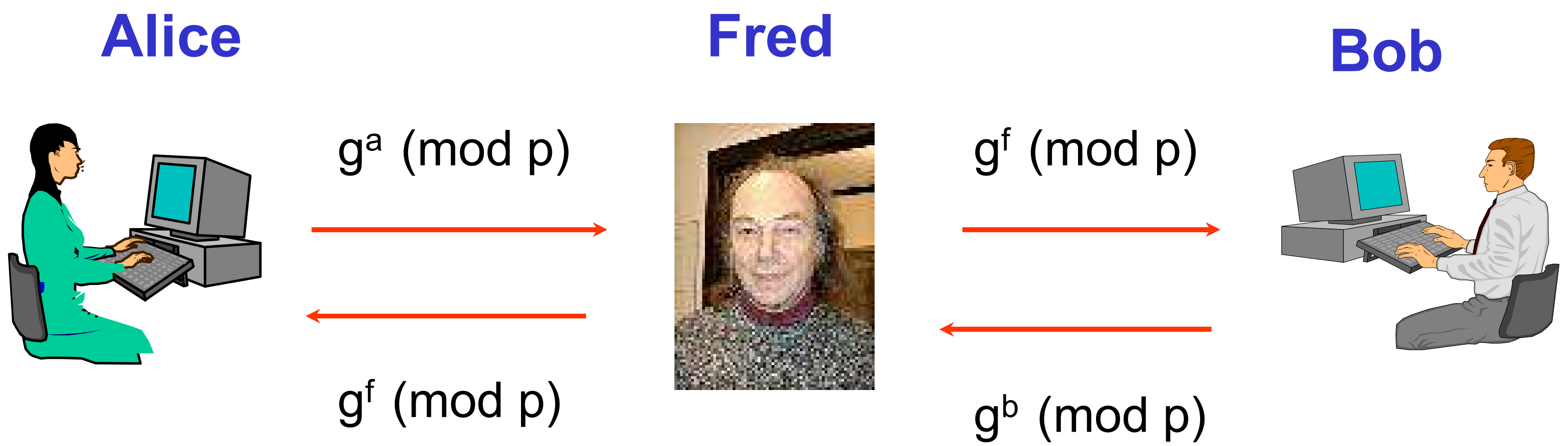


Unique Key Per Transaction (5)

Derived Unique Key per Transaction (DUKPT) Scheme

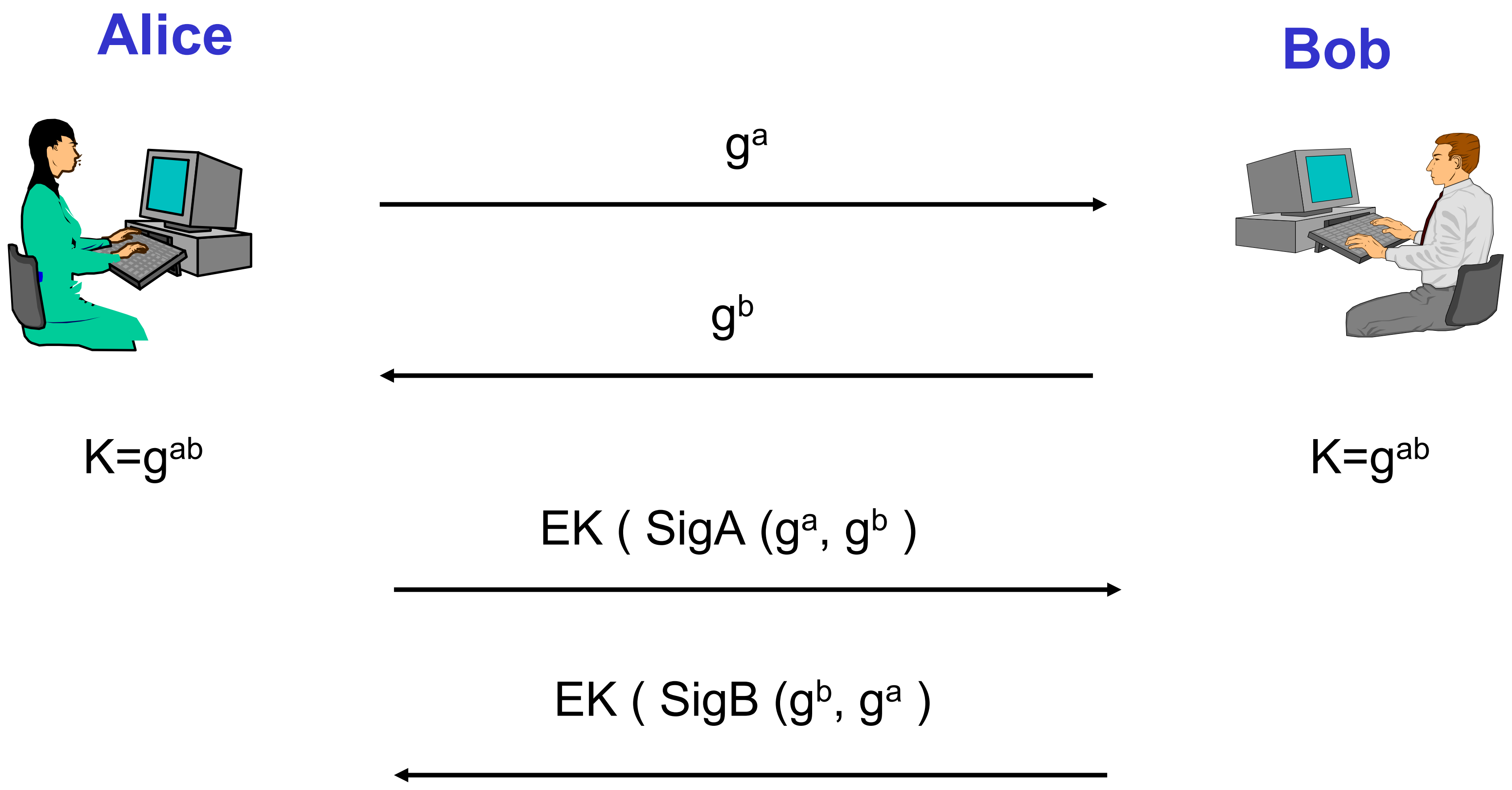
- This is a scheme used in the retail environment, supported by Visa, amongst others.
- The fundamental idea is that there is a base derivation key, from which transaction keys are generated (in an irreversible way) using the terminal ID and a transaction counter.
- The host only needs to maintain a copy of the base derivation key - the clever bit is for the host to be able to calculate quickly the transaction key for a particular terminal.
- The terminal need only keep a copy of its last transaction key, from which it can easily derive the next key.

Diffie-Hellman key agreement



Alice and Bob both believe that they have agreed a common key, but in fact they have both actually agreed different keys with Fred.

Station-to-station key agreement



63

Quantum Key Distribution (1)

- **The laws of quantum physics can be used to create an unbreakable key distribution system (Quantum Key Distribution – QKD). It is based on the polarisation of light photons (effectively 4 states) and polarisation filters.**
- **A separate insecure channel is used convey information about the states of the photons that were sent. Incorrect "guesses" about the state are discarded.**
- **An eavesdropper can only guess at each state, but an incorrect guess cannot be later corrected.**
- **Eavesdropping can be detected with a high degree of probability (the act of eavesdropping may alter the state – Heisenberg's Uncertainty Principle).**
- **The result is a "random" bit sequence – used as a one-time pad.**

Quantum Key Distribution (2)

1. **Alice sends Bob a series of photons and Bob measures them.**
2. **Alice tells Bob on which occasions he measured them the correct way (i.e. using a rectilinear or diagonal filter), but not the actual value sent.**
3. **Alice and Bob discard the incorrect measurements and concentrate only on the correct measurements to form the one-time pad.**
4. **Alice and Bob check the integrity of their one-time pad by checking a few of the bits (which are then discarded). This process will detect whether eavesdropping has occurred (with high probability).**

Neils Bohr: Anyone who can contemplate quantum mechanics without getting dizzy hasn't understood it.

Reality check

- QKD is a reality, but is not yet a practical proposition except in highly controlled environments.
  - 1988 – first demonstration (over distance of 30 cm).
  - 1995 – using optic fibre over 23km.
  - 2002 – using optic fibre over 67km
  - 2002 – free transmission, 2km.
  - 20?? – plans for 150km trial in the U.S.
  - 20?? – plans for a 1000km exchange via satellite
- A number of QKD products have appeared on the market.

4. Public key management

The need for a PKI

Suppose that you have received a digitally signed message that claims to have been signed by Alice and that you want to verify this signature.

This requires you to possess a public verification key. By some means (we won't specify how) you are presented with a key and told "this key is Alice's public key". You use it to verify the signature, and it seems to work.

It may well be valid – but you should always be suspicious!

Write down as many things as you can think of that could mean that in fact you do **not** have a valid signature by Alice on this message at all.

Public key certificates

A **public key certificate** is a set of data that binds an identity to a particular public key value. The four core pieces of information that are contained in a public key certificate are as follows:

- **Name of owner**
  - could be a person, device, or even role.
  - Should uniquely identify the owner within the environment in which the public key will be employed.
- **Public key value**
- **Validity time period**
  - identifies date and time from which the public key is valid, and more importantly the date and time of its expiry.
- **Signature**
  - Creator of certificate digitally signs all data that forms the public key certificate. This binds the data and acts as a guarantee that the creator of the certificate believes the data is correct.

X509 v3 public key certificates

Version	**This specifies the X.509 version being used (in this case v3).**
Serial Number	**A unique identifier for the certificate.**
Signature	**The digital signature algorithm used to sign the certificate.**
Issuer	**The name of the creator of the digital certificate.**
Validity	**The dates and times between which the digital certificate is valid.**
Subject	**The name of the owner of the digital certificate.**
Subject Public Key Info	**The actual public key and the identifier of the public key algorithm associated with it.**
Issuer Unique ID	**An optional identifier for the creator of the digital certificate.**
Subject Unique ID	**An optional identifier for the owner of the digital certificate.**
Extensions	**A range of optional fields that include:** **• a key identifier (in case owner owns more than one public key)** **• key usage information that specifies valid uses of key** **• the location of revocation information** **• identifier of the certificate policy** **• alternative names for the owner**

Certificate authorities

It should be clear that the “creator” of a public key certificate plays an extremely important role.

A creator of a public key certificate is normally referred to as a **Certificate Authority** (or **CA**).

1. **The CA takes responsibility for ensuring that the information on a certificate is correct. The CA creates (or issues) the public key certificate to the owner.**
2. **Whenever anyone has need of the owner’s public key they request a copy of the public key certificate. The certificate might be made available on a central server, or the owner or even the CA might send the certificate to whoever requires it.**
3. **The recipient of the public key certificate checks that the certificate is in order, and if they are happy with it then they are free to use the public key contained in the certificate.**

Trusting a digital certificate

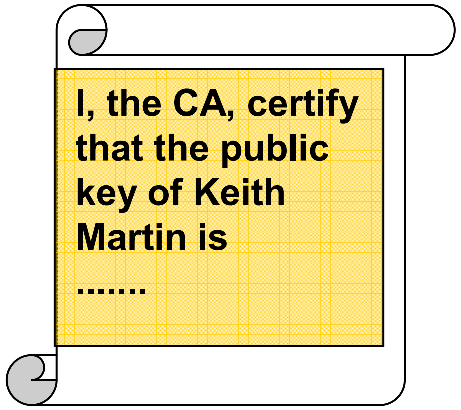
There are three things that the recipient "needs to be able to do" in order to be satisfied that the public key certificate is in order:

1. **The recipient needs to be able to trust (directly or indirectly) the CA to have done their job correctly and to have gone through some process to verify all the fields of the certificate.**
2. **The recipient needs to have access to the public verification key of the CA in order to verify the CA's digital signature on the certificate.**
3. **The recipient needs to check all the fields in the certificate. In particular they must check that the certificate is valid, it applies to the correct owner and that the other fields are all satisfactory.**

For each of the above three recipient checks, what are the precise implications of them not being done.

Meaning of certificates

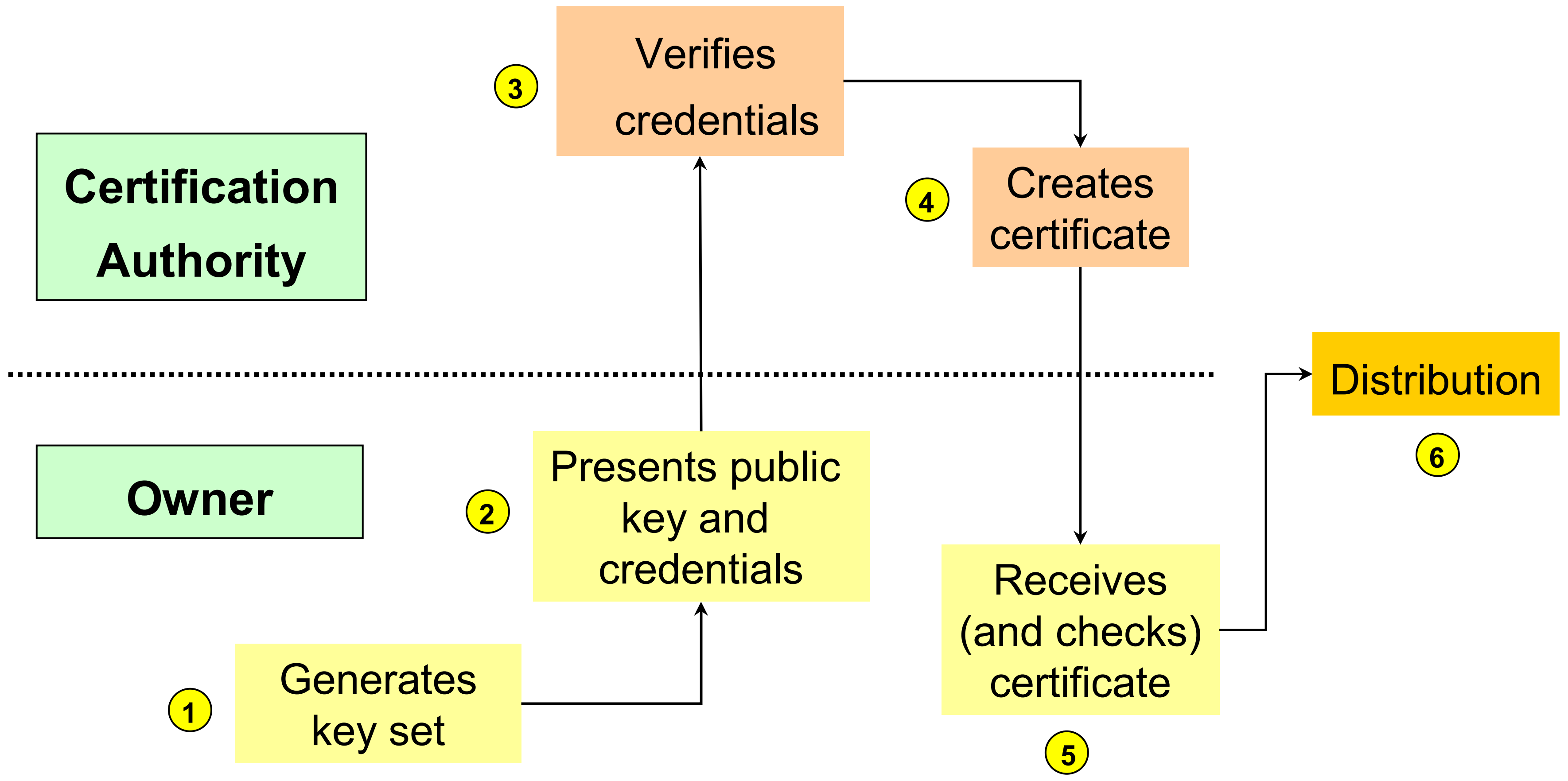
By digitally signing the information in the public key certificate, the Certification Authority is effectively making the statement:



I, the CA, certify that the public key of Keith Martin is ....

1. Can you use a public key certificate directly to encrypt messages or verify digital signatures?
2. If someone presents you with their public key certificate, is this proof of their identity?

Example certificate issuing process



Generating the public key pair

	Pros	Cons
Owner	The owner is placed in full control of their own key material.	The owner might not have the capability or skill to perform this operation in a secure fashion. The owner needs to prove to the CA that they actually know the private key before the certificate can be issued.
Issuer (CA)	It may be easier and more secure to manage the generation of key material centrally. The certification process might appear more seamless to the owner.	The owner must trust the CA to securely deliver the private key to the owner and to dispose of it afterwards.

Registration

The stage of the certification process at which the owner presents their credentials to the CA for checking is arguably the most vital stage in the entire certification process.

In many application environments a separate entity known as a Registration Authority (RA) performs this operation.

There are two arguments for keeping the roles of CA and RA at least slightly separate:

1. **Most of the functionality of a CA can be essentially performed by a computer, whereas for many applications the role of the RA requires human intervention.**
2. **Checking the credentials of a certificate applicant is often the most complex part of the certification process. There is thus a strong argument for distributing the registration activities across a number of "local" RAs.**

Proof of possession

However registration is done, there is one very important check that must be performed before proceeding with the issuing of a public key certificate – that the owner actually knows the private key corresponding to the public key in the certificate.

If the CA does the key generation then this problem does not arise, but if the owner generates the key pair then this check is essential.

This process is referred to as demonstrating **Proof of Possession**.

1. Why should the CA check ownership of the private key?
2. How can a CA check ownership of a private key without the owner revealing the private key?

Certificate distribution

- **Pushing**
  - the owner of the certificate automatically provides the certificate when it is required
  - the problem with pushing is that the receiver of the certificate needs to check that the certificate that they have just received is still valid.
- **Pulling**
  - users must request copies of certificates when they need them.
  - the problem will pulling is that this requires the relevant CAs to be online to distribute the certificates when required to do so.
  - an advantage is that the receiver is more likely to get the latest valid certificate, although it may still be prudent that the receiver performs checks to ensure that the certificate has not been revoked.

Certifying the certifiers

Someone, somewhere, must generate the CA's public key pair, and someone, somewhere, must certify this public key. However it is done, it must be done securely. If someone gets hold of the private key of the CA then they can generate certificates themselves, and the whole system falls apart.

1. Who generates the public verification key of a CA?
2. How does a CA arrange to certify its own public verification key?
3. How is the public verification key of a CA distributed to those entities who need to rely on it?

Revocation

We must consider how to handle certificates that need to be "withdrawn" before their expiry date. This process is often referred to as **certificate revocation**.

- **Certificate Revocation List** (or **CRL**s)
  - A lists of certificates that have been revoked.
  - CRLs need to be maintained carefully, with clear indications of how often they are updated.
  - CRLs need to be signed by the CA and be made available to users as easily as possible.
- **Online Certificate Status Protocol** (**OCSP**)
  - An online database containing the status of certificates issued by the CA .

Establishing a PKI

We need to consider how trust can be established and managed within a PKI. For example:

- How is trust in a CA established?
- Who are the candidates for CAs?
- How do you choose a CA?
- How do CAs recognise one another?
- How is liability managed?

Joining CA domains

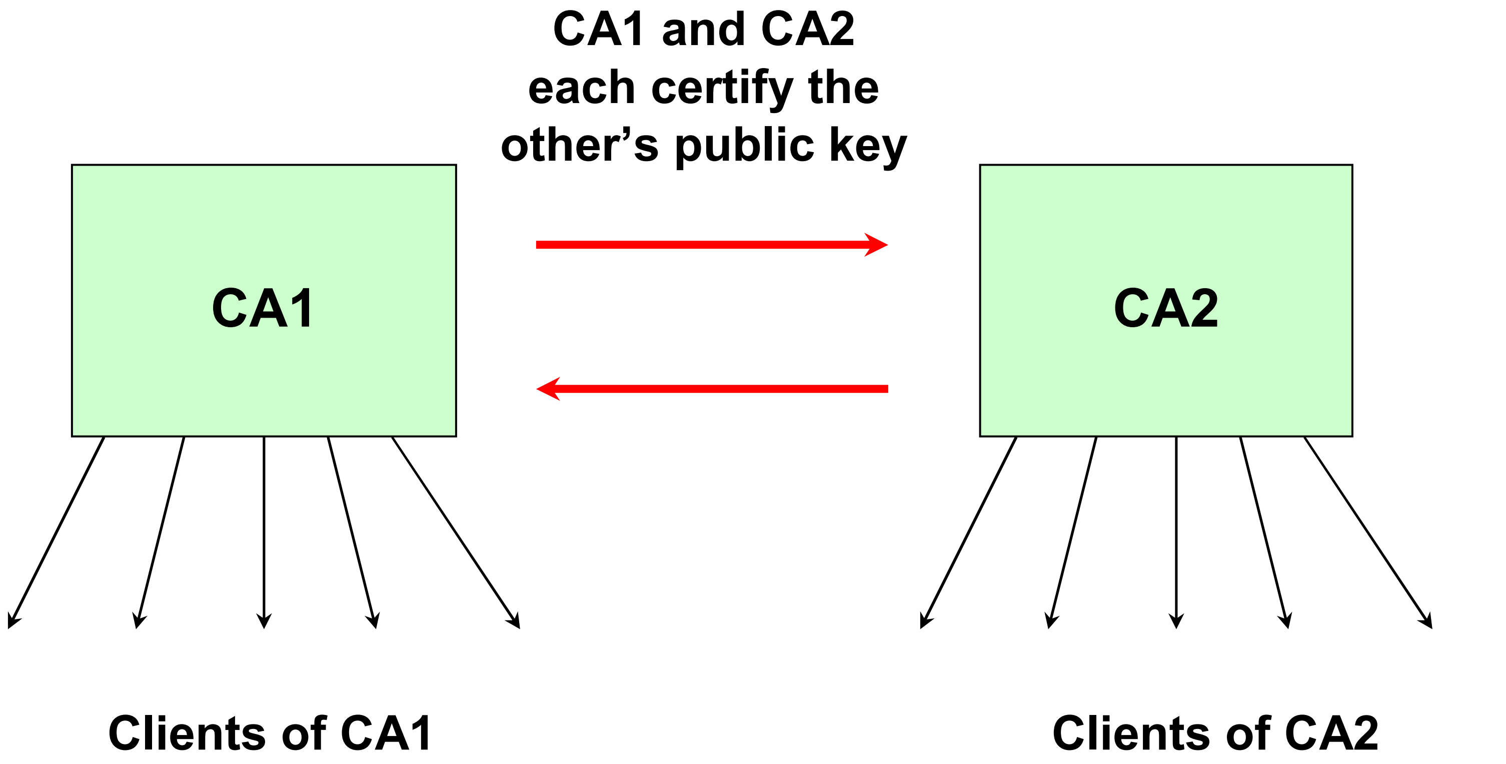
An owner of a public key certificate has by necessity placed some trust in the CA who has issued this certificate.

However, for larger and more open PKIs, it is likely to be the case that the owner of a certificate will:

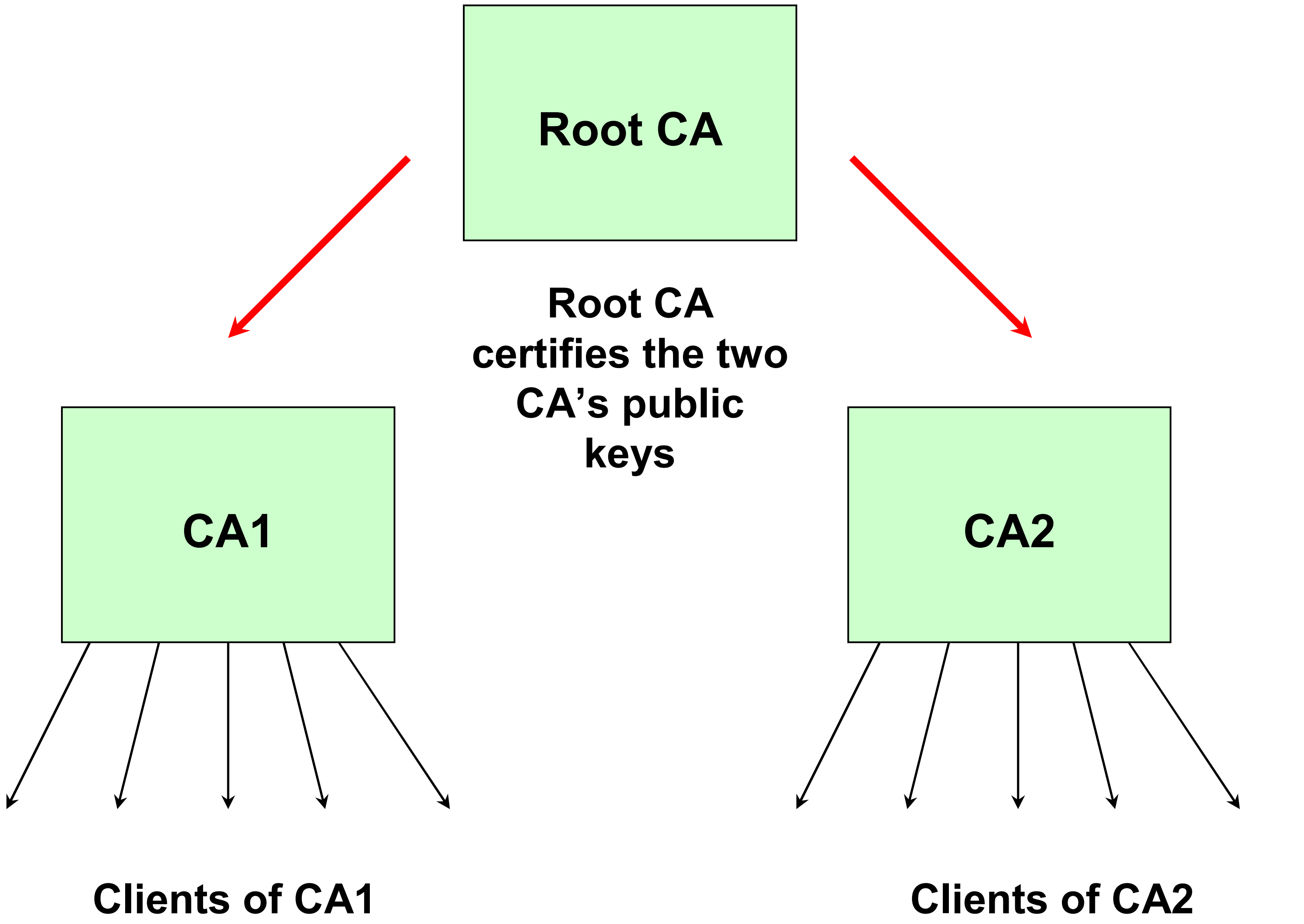
- want users who do not have a business relationship with the owner’s CA to be able to rely on the owner’s certificate
- want to rely on certificates that were not issued by the owner’s CA.

There is thus a need for techniques that somehow “join” different certification domains and allow certificates issued by one CA to be recognised by another CA.

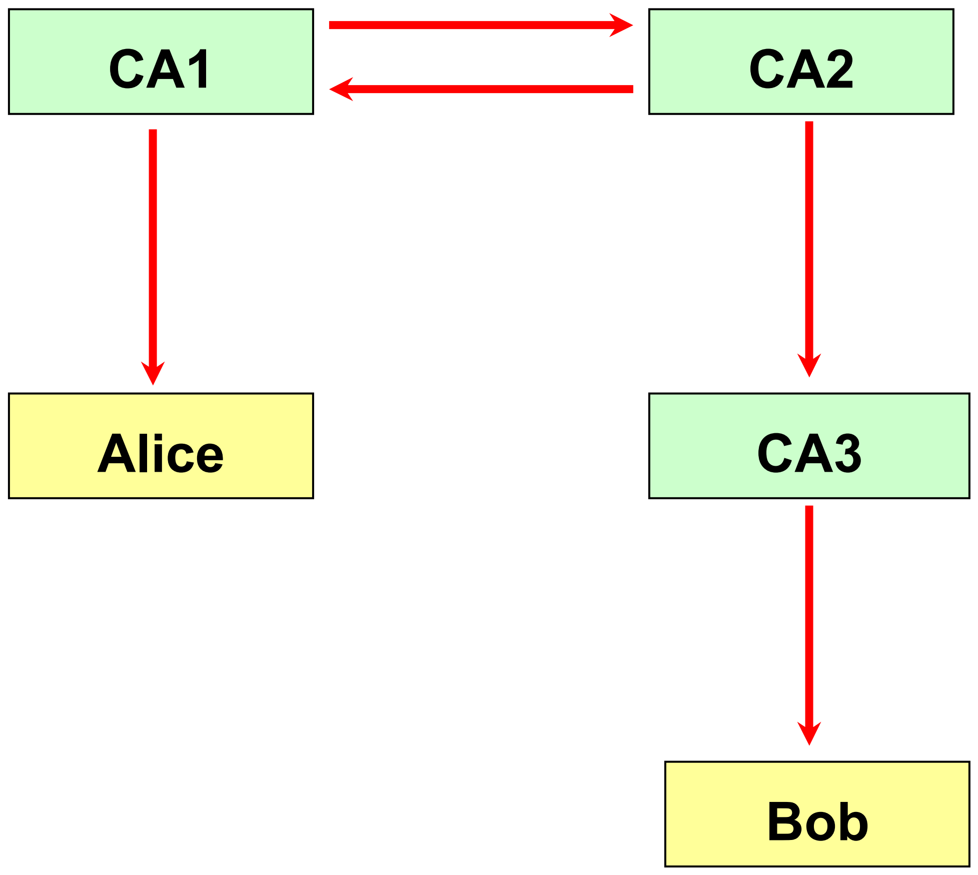
Cross certification



Certificate hierarchies



Certificate chains



When Alice wants to check the authenticity of Bob's public key she must verify each link in the chain:

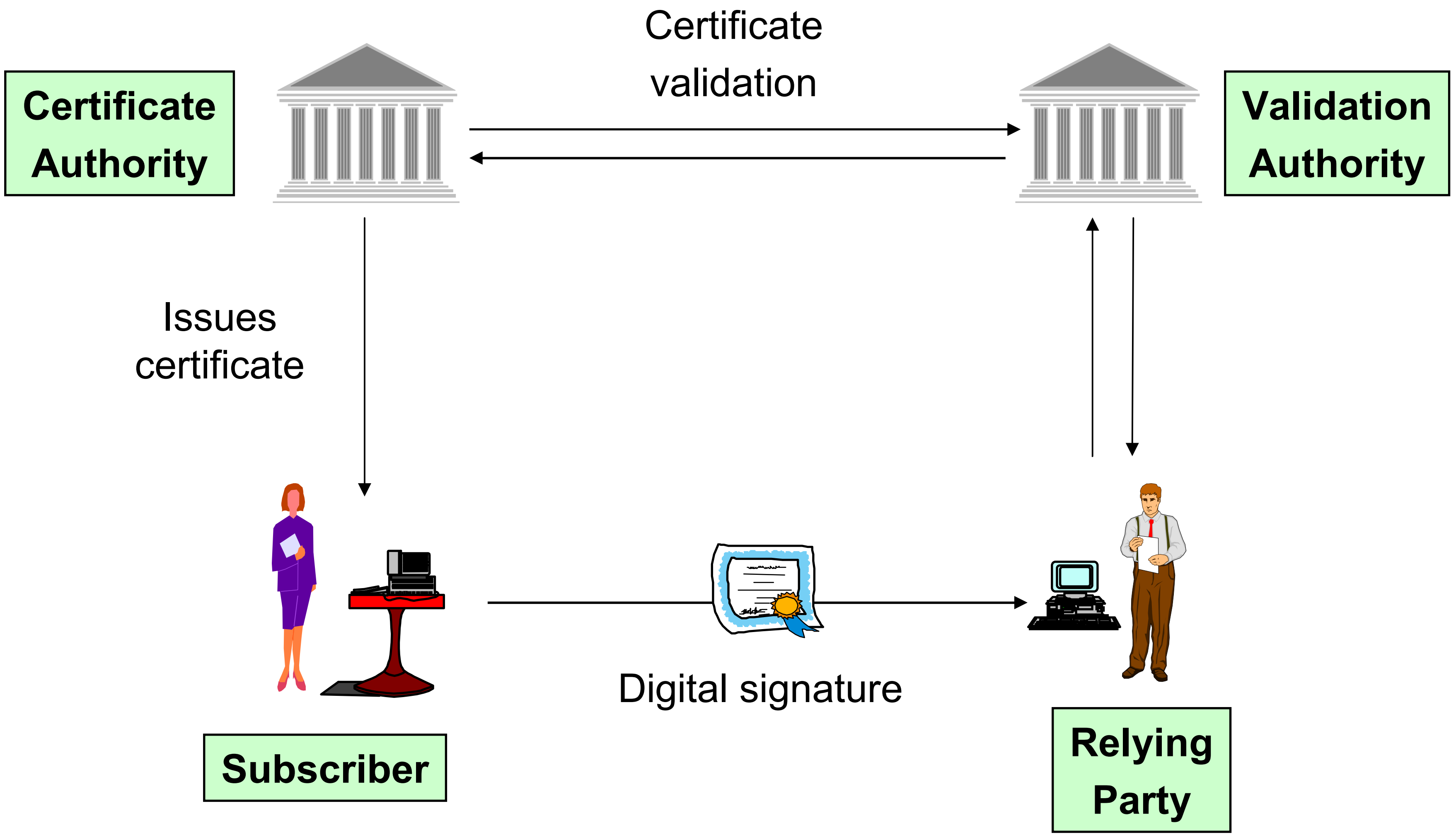
Public key of CA2	CA1
Public key of CA3	CA2
Public key of Bob	CA3

Liability issues

You trust your London based CA, and it has cross-certified with another CA in Luxembourg, who acts as the root CA for a small CA in Belgium, and one of the Belgian CA’s clients has sent you a certificate chain that connects back to its public verification key. You want to use this public verification key to verify the signature on a transaction that the Belgian customer has promised you.

- What happens if it all goes wrong?
- Where does the liability lie?
- Who is responsible to whom?
- And how much protection is offered?

Four corner model



The state of PKI

In the mid 1990's it was widely forecast that PKIs would be implemented on a broad scale to provide security services.

Every year it was widely forecast that PKI adoption was just round the corner. The next year the same thing happened...

We are still waiting for this massive adoption of PKI to take place. Progress has been substantial, but it has been slow and very unsteady. There are any competent business organisations offering CA services, and some efforts are underway to establish PKI standards.

We are still awaiting what David Lacey has referred to as “the golden age of PKI”. For what reasons do you think that this is the case?

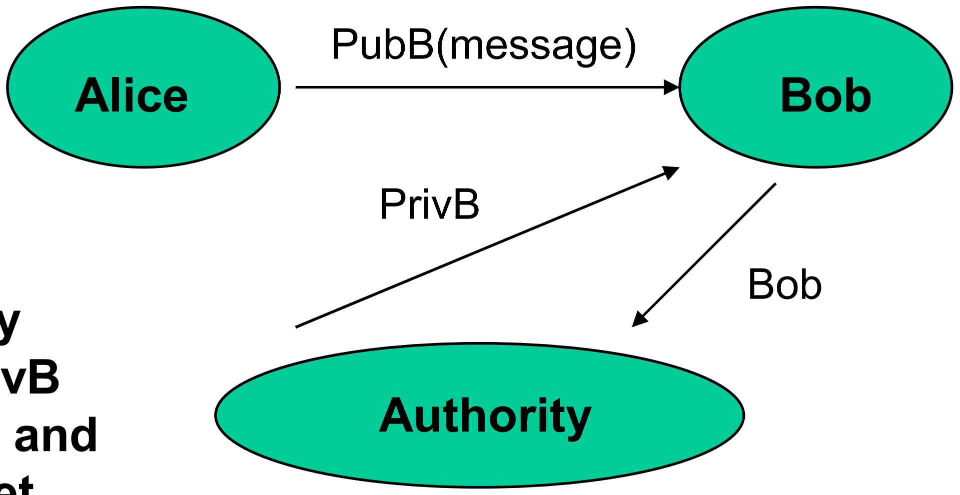
5. Research challenges

Challenge 1

Making public key cryptography really work

IDPKC

1. Alice derives Bob's public key PubB from some public information and sends message encrypted under PubB to Bob.



3. Authority derives PrivB from PubB and some secret value, and returns this to Bob.

2. Bob identifies himself to an authority and requests the private key PrivB corresponding to PubB.

Challenge 2

Managing cryptographic keys in complex environments

Challenge 3

Making key management as invisible (yet effective) as possible

Summary

- A cryptographic system is only as secure as the method by which its keys are managed
- There are many different issues involved in managing keys, all of which need to be addressed to make a system secure
- There are many different techniques for implementing the different stages of a key lifecycle
- The technical components are only one very small part of a key management solution

Never forget Kerckhoff!

The security of a cryptographic system must not depend on keeping secret the cryptographic algorithm. The security depends on keeping secret the key.