CS5285
Information Security for eCommerce

Lecture 7

Dr. Gerhard Hancke
CS Department
City University of Hong Kong

1

Reminder of previous lecture

- Authentication
  - What is needed for entity authentication?
  - Data origin authentication (sure who generated)
    - MAC/Encryption/Signature
    - Stops masquerade attacks
  - Freshness (is the message generated now?)
    - Nonce (Random or counter/sequence)/Timestamp
    - Stops replay
  - We looked at protocol examples…
    - Also remember the reflection attack

Today's Lecture

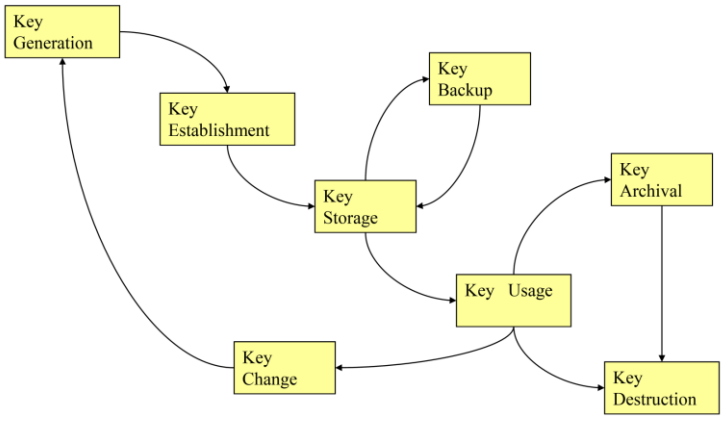
- Key Management
  - For all crypto we need keys (most important)
  - Symmetric key management
  - Asymmetric key management (Certificates)
- CILO1,CILO2, CILO3 CILO4

  (identify requirements, core threat, design and evaluation)

Credit to Keith Martin RHUL (borrowing few slides from his lecture notes – 6, 45-47)

Key Management

What is Key Management?



5

Slide for interest

Keys are needed for all the cryptographic tools we discussed.

We need to create (generate keys), we then need to give the to the people who need to use them (establish) and when there we need to make sure that they are securely stored (and backed-up) so that they do not get stolen or lost.

We use them, we might want to revoke and change the keys. Finally we might need to destroy to keys once we no longer have use for them. We might also need to archive some keys for a while.

Why do we want to archive a key?

Think of when someone has used a key to sign some documents. We cannot just destroy all the key material as we might need to still verify these documents for a while, of maybe keep a symmetric key to make sure we have decrypted all the documents we encrypted

with it in the first place.

Contents in this section from ISO/IEC 11770 IT Security techniques — Key management

Hardware Security Module (HSM)

- HSM is trusted part of system
- Key generation and storage
- Features:
  - Asymmetric Crypto (Signing/Encryption)
  - Symmetric Crypto (Encryption/MAC)
  - Hashes/KDF
  - Random numbers (True random, DRBG)
- Secure networking (TLS), Tamper resistant

DRBG Deterministic random bit generator

Example: Thales Luna Network HSM 7

https://cpl.thalesgroup.com/encryption/hardware-security-modules/network-hsms

•**Cryptography**Full Suite B support

•Asymmetric: RSA, DSA, Diffie-Hellman, Elliptic Curve Cryptography (ECDSA, ECDH, Ed25519, ECIES) with named, user-defined and Brainpool curves, KCDSA, and more

•Symmetric: AES, AES-GCM, DES, Triple DES, ARIA, SEED, RC2, RC4, RC5, CAST, and more

•Hash/Message Digest/HMAC: SHA-1, SHA-2, SM3, and more

•Key Derivation: SP800-108 Counter Mode

•Key Wrapping: SP800-38F

•Random Number Generation: designed to comply with AIS 20/31 to DRG.4 using HW based true noise source alongside NIST 800-90A compliant CTR-DRBG

Symmetric-key protocols

- The use of symmetric-key cryptography to produce a shared symmetric secret key.
- The protocols can be classified as:
  - Directly communicating entities
  - Use of a Key Distribution Centre (KDC)
  - Use of a Key Translation Centre (KTC)

For slide 7-12 you are only expected to know the two main models (KDC/KTC) and why these are preferable to the directly communicating model (more efficient key storage)

We can look at key establishment in different categories. We will start with approaches using symmetric keys.

Remember establishment does not mean we the parties have no relationship – they might already (should already) have some sort of relationship in terms of shared keys.

This is for setting up a new temporary key to be used between them

At this early stage you should already be familiar with the later discussed concept of key control (none of the parties have key control

Directly communicating entities

- The case where two entities directly communicate to establish keys.
- Must take place using a secure channel (e.g. using an existing shared secret key or mutually trusted copies of each others' public keys).

This refers to the case where two entities communicate to establish keys without the direct aid of a Trusted Third Party (e.g. a Certification Authority, Key Distribution Centre or Key Translation Centre).

Such key distribution must take place over a secure channel (typically provided using an already existing secret key or asymmetric keying relationship). There has to be a prior trust relationship – how these initial keys get established is outside the scope of this standard.

How many long term keys must each user store in this system?

Lets get back to our organisation with 1000 people – if we did symmetric encryption how much keys much each entity store?

Each person must have a key for every other person. (See start of Lecture 4)

Distribution within a domain

- Two possible cases:
  - asymmetric techniques used, or
  - symmetric techniques used.
- In first case certificates may need to be distributed. Entities either contact their authority for certificates, or two entities may exchange them directly.
- In second case - use a KDC or a KTC.

We distinguish between the cases where:

- an asymmetric technique is to be used for key establishment,
- a symmetric technique is to be used for key establishment.

In the first case, various certificates may need to be distributed. Both entities may need to contact its authority to get the public key certificate for the other entity. Alternatively they may directly exchange their certificates.

In the second case, we consider the use of either a Key Distribution Centre or a Key Translation Centre.

How many keys must each user store in this system? I just need to share a key with the KDC/KTC.

Distribution within a domain

- *Key Translation Centre* (KTC): An entity trusted to transport keys between entities that share keys with the KTC.
  - Example of a key transport service.
- *Key Distribution Centre* (KDC): An entity trusted to generate and distribute keys to entities that share keys with the KDC.
  - Arguably example of a key agreement service.

As we shall see in more detail in a moment, a key translation centre is a trusted third party (TTP) that facilities communication between users by securely transporting a key generated by one entity to another entity. Both of these entities must share a key with the KTC.

What does service does 'transport' and 'agreement' suggest to you?

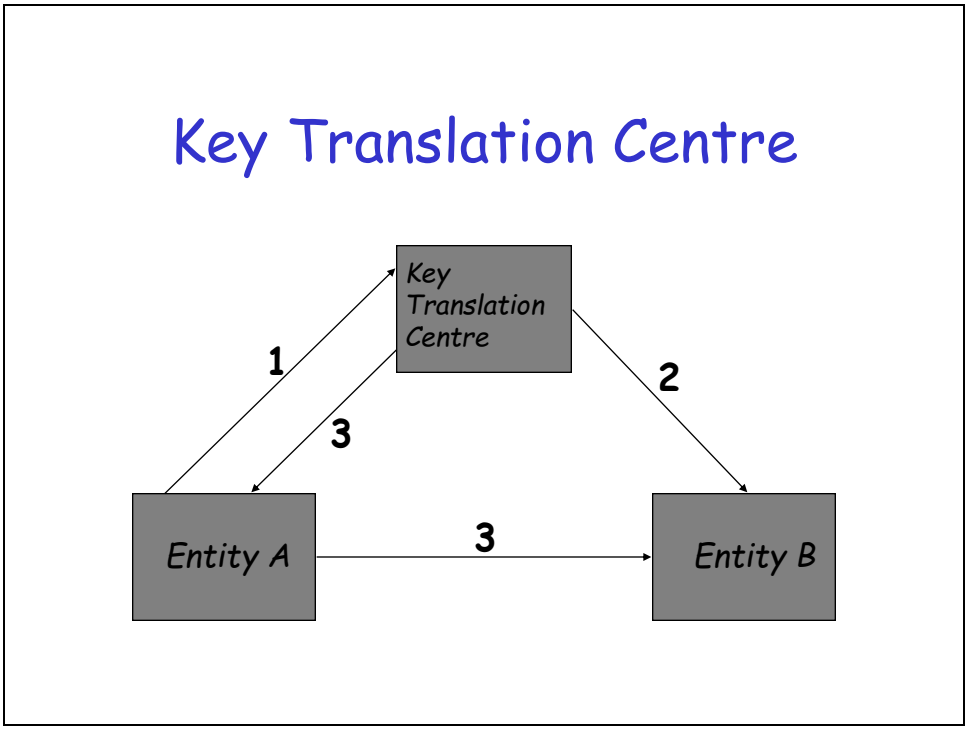
Who has key control? Person sending key.

A KTC service is an example of a key transport service.

The key distribution centre is similar, but also creates the key. In a KDC service, the KDC has complete key control. This means that neither of the user's has key control and so, if we trust the KDC, no-one can apply any undue pressure in choosing the key.

We may argue that a KDC is an example of a key agreement service (because in spirit it is essentially between two parties and neither has key control, technically a single entity, the TTP, has key control

so you could also argue this is not true).



The diagram shows two typical uses of a Key Translation Centre (KTC) within one domain.

We suppose the KTC has a shared secret (symmetric) key with both A and B.

The following messages are passed:

1. The KTC receives a key from A, which has been encrypted using the key shared by A and the KTC. The KTC decrypts it and re-encrypts it using the secret key shared with B.

2.Case 1: The KTC sends the re-encrypted key directly to B.

How much long term keys does Alice need to store - one.

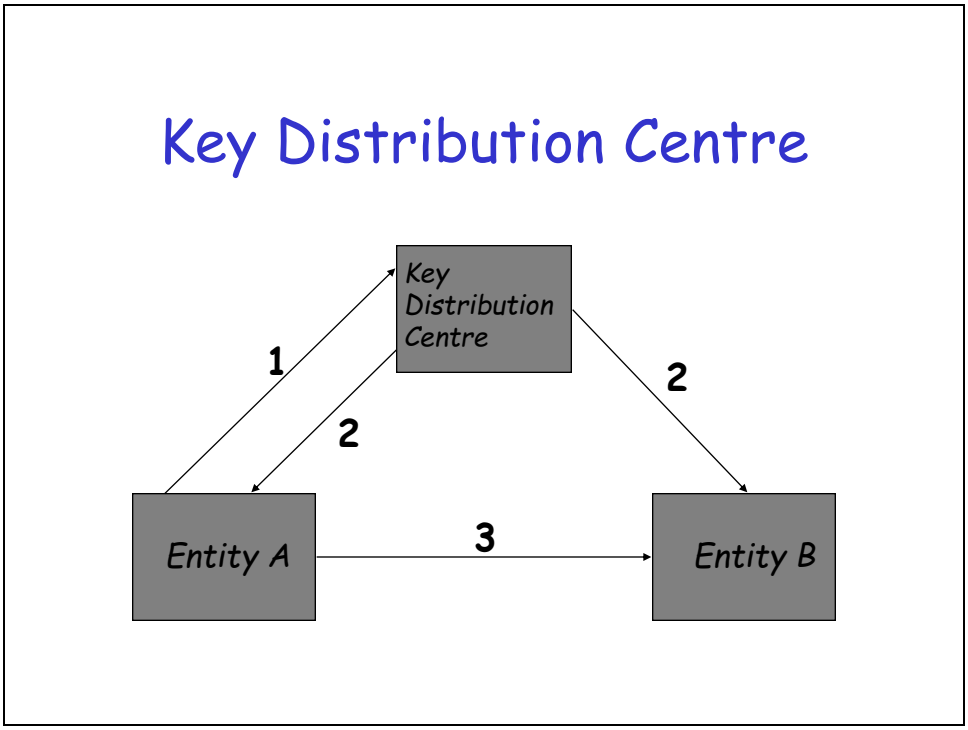
How much long term keys does Bob need to store - one.

3.Case 2: Alternatively, the re-encrypted key is sent to A, who then passes it on to B. This is known as key forwarding.

What is the advantage of key forwarding?

The KTC only communicates with one entity – one session with Alice. Does not need to talk to B.

What if we did not have Alice and Bob, what if we had Alice's smart card reader and Bob smart card – the KTM cannot initiate a session with Bob's smart card….



The diagram shows a typical use of a Key Distribution Centre (KDC) within one domain. We suppose the KDC has a shared secret key with both A and B. The following messages are passed:

1. A requests the KDC to generate and distribute a secret key to be shared by A and B. On receipt of this request the KDC will generate an appropriate secret key.

2. Case 1: The KDC sends the newly generated key to both A and B. The new key is encrypted using the keys that A and B share with the KDC.

3. Case 2: The KDC encrypts the newly generated key using the keys that A and B share with the KDC. It sends both encrypted version of the key back to A. A recovers its own version of the encrypted key and sends the version encrypted under the key B shares with the KDC to B.

Terms for key management

- *Key establishment*: Process of making a secret key available to multiple entities.
- *Key control*: Ability to a choose a key's value.
- *Key agreement*: Process of establishing a key in such a way that neither entity has key control.
- *Key transport*: Process of securely transferring a key from one entity to another.
- *Key confirmation*: Assurance that another entity has a particular key.

Please know these concepts – you must know the difference between key agreement and key transport (key control is the difference) and key confirmation (and how this relates to implicit and explicit key authentication on slide 13)

Once again lets start with a number of definitions, listed within the standard. ISO/IEC 11770 contains a lot of definitions:

• Key establishment is a general term for any procedure that makes a secret key available to more than one party. Note that this includes the case of sending the key to another person (as this means that two people know the key: the sender and the receiver).

• Key control is sometimes a desirable thing and sometimes an undesirable thing. This depends upon the environment and the application. Entity has key control if it has the ability to choose the key.

Key agreement and key transport are examples of key

establishment services – differentiated based on key control

- Key agreement is a type of key establishment service where neither party has key control. So two entities run a process whereby the end up with keys but both parties influence how the key is chosen.

- Key transport is a type of key establishment service where one party has complete key control. Most key transport protocols concentrate on the moving a key from one entity to another, although it could be argued that the complete key establishment service consists of both the service that generates the key and the key transport service.

- Key confirmation is a service that offers an entity assurance that another entity has a particular key, usually the key that has been just established.

So lets think about this for a bit:

Suppose – I want you to get into the AC1 office (I go to the hardware store, randomly pick up a key/lock combo box). Hand the keys to you and then install the lock. What type of key establishment is this? Do I really have key control – in that I can choose the specific key I want you to use? No, I chose a random box. This is key agreement.

Lets say I put a password on the Canvas page, and then handed everyone a note in class with their password. What type of key establishment do we have? Who has key control? I have key control as I decide on the password – so key transport.

Who has the key at the end?

- We can set two requirements of key establishment
  - *Implicit key authentication to A*: The assurance for *A* that only another identified entity can possess a key. This is the basic security requirement.
  - *Explicit key authentication to A*: The assurance for *A* that only another identified entity possesses a key. This is a more stringent security requirement.

See Slide 13 key confirmation

At if the end of the establishment protocol if A believes that it now shares a key with B (and the protocol makes it possible only for B now to also have the key) then we have implicit key authentication.

This means that it is not possible for C,D,E, etc. to have gotten the key.

At if the end of the protocol A believes that it now shares a key with B (and the protocol makes it possible only for B now to also have the key), and during the protocol B also demonstrate he has the key (he uses it to talk to A) then we have explicit key authentication.

So the easiest way to determine whether a key establishment protocol has explicit key authentication is to check whether A and/or B uses the newly established key during the protocol. If not,

it is implicit.

Assurance is a non precise property. It is how sure we are of somethings. The difference here is between 'can possess' and 'posesses'. For implicit A believes that it is only possible for one entity B to possibley possess the key. For explicit A is sure that B definitely possesses the key.

Think of the key as a package. If we send a package to B using courier then once the courier says it is done – we believe that is only possible for B to now have to package (because we trust the courier service). However, we do not know it they actually have it. If our friend now phones us or send us a photo saying package arrived and they describe it to us then we know for sure that they have possession.

Key establishment mech. 6

- Directly communicating entities.
- *A*, *B* must share a secret key *KAB*.
- R_A and R_B are nonces.
- F_A and F_B contain keying material.

Do not memorise the protocols but you must be able to look at a protocol and discuss its security properties.

Mechanism 6 (a nonce-based key distribution mechanism) is derived from the 3-pass authentication protocol in Clause 5.2.2 of ISO/IEC 9798-2. This mechanism is an example of the 'directly communicating entities' model considered in ISO/IEC 11770-1.

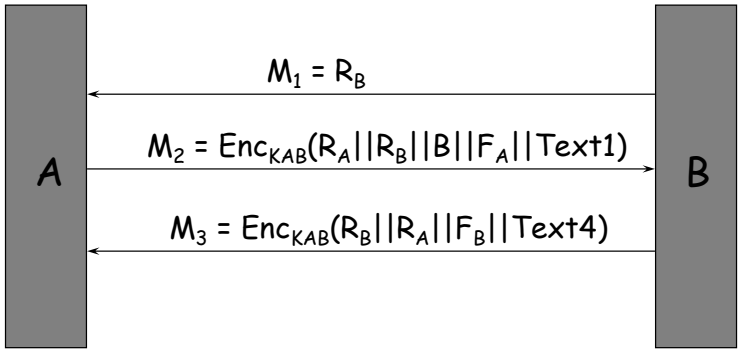
To use this mechanism, *A* and *B* must share a secret key *KAB*.

What type of authentication does this protocol provide?

What type of security does this protocol provide?

Who has key control?

Key establishment mech. 6



- Key K typically a hash of F_A and F_B.

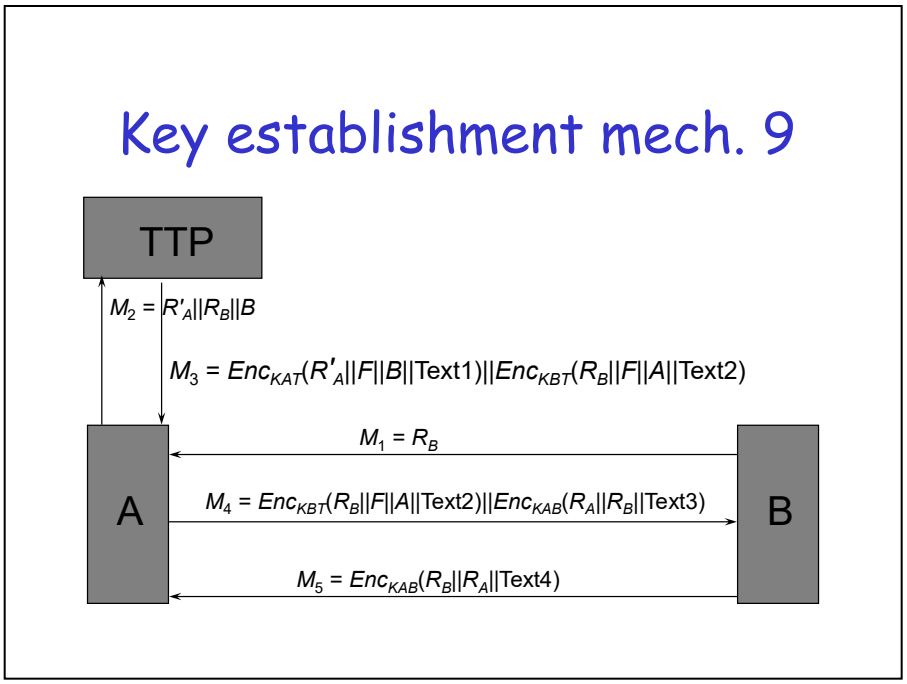
Explain the operation of this protocol in detail.

Is A and B authenticated? Yes.

Who has key control? Neither have control (both contribute), key agreement protocol.

What if either F_A or F_B contains no data? Then one party has key control.

Do we have explicit key authentication? Key confirmation? No, no use of the agreed key.



The goal of this protocol is the establish K_AB

Mechanism 9 (a nonce-based key distribution mechanism) is based on the 5-pass authentication protocol in clause 6.2 of ISO/IEC 9798-2. This mechanism is an example of the 'Key Distribution Centre' model considered in ISO/IEC 11770-1, and where *A* 'forwards' keys to *B*.

Note that, in this protocol, *T* is a third party (a KDC) trusted by both *A* and *B*. Moreover *T* shares the secret keys *KAT* and *KBT* with *A* and *B* respectively.

What are the authentication properties of this protocol?

What freshness? Nonces

Data origin authentication? Encryption, redundancy

TTP is authenticated to *A*

TTP is authenticated to B

A and B mutually authentication

Why are the nonces swopped around in M5?

Else you just send back what was sent to B in M4.

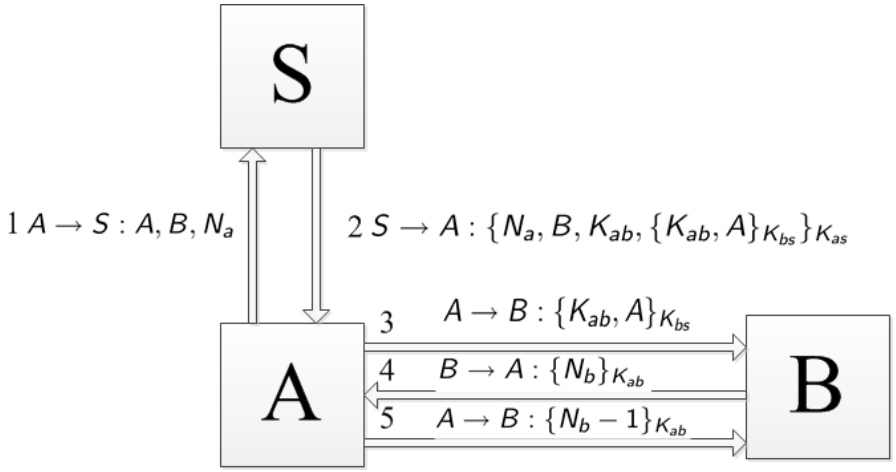
Where is the keying material? F

What key establishment security does the protocol provide? So is this implicit or explicit key authentication?

Key confirmation – B confirms use of key, so does A. Explicit.

Who has key control? TTP

Remember from last lecture….



Anyone know this protocol? This is the Needham Schroeder Protocol. It is the basis of Kerberos. Windows 2000 and later uses Kerberos as its default authentication method.

Remember this from the end of the authentication lecture – self exercise.

If you saw this how would you analyse it?

Assumptions A and B share key with S

Is S authenticated to A? Yes, Freshness (NA), Data origin authentication? Known message encrypted KAS

Is A authenticated to S? No

Is S authenticated to B? No

Is B authenticated to A? Not really, Message 4 (Nb encrypted with KAB), only B could retrieve and use KAB from message 3 but we do not know what Nb is supposed to be. It would have been better for message 3 to be Na,{Kab,A}Kbs and 4 to be {Na,Nb}Kab. Then A and B would be mutually authenticated.

Is A authenticated to B? Yes, message 5. Freshness in nonce, only A can create message with KAB.

Is this key transport or key distribution? Key distribution (not A or B has key control)

Who has key control? S

Implicit of explicit key authentication? Explicit at Kab is used.

Public-key protocols

- The use of public-key cryptography to produce a shared symmetric secret key.
- The protocols can be classified as:
  - Key transport protocols (typically involving public-key encryption and digital signatures)
  - Key agreement protocols (indirectly specified and mostly based on the Diffie-Hellman protocol)

Know the basic approaches for key establishments (transport and agreement) based on public key approaches.

Once again you do not need to know specific protocols but you must be able to look at a given protocol and discuss.

Notation

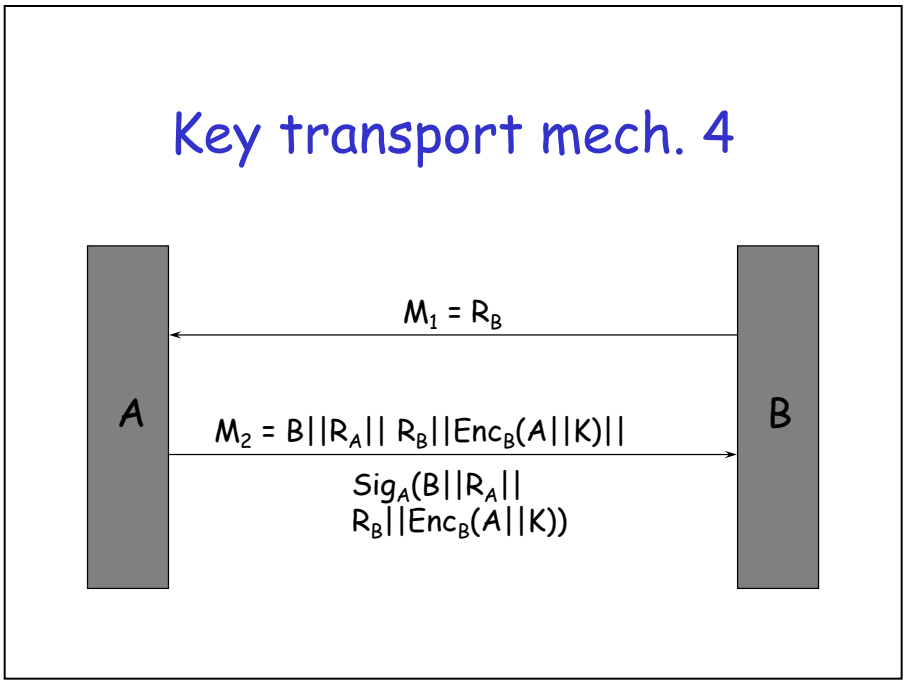
- A and B are (identifiers for) two entities who wish to engage in an authentication protocol.
- R_A is a random nonce generated by A.
- $Enc_A(X)$ denotes the encryption computed with A's public key on the data X.
- $Sig_A(X)$ denotes the signature (with appendix) computed by A on the data X.

Remember that we're dealing with public-key cryptography here. The encryption algorithm is public key – $Enc_A(X)$ is a ciphertext that can be computed by anyone, but which can only be decrypted by A.

Explain why we can't use an MDC to provide origin authentication when using public-key encryption?

Key transport mech. 4

- Key transport protocol.
- *A*, *B* must have authenticated copies of each others public keys.
- R_A and R_B are nonces.
- The notion of keying material has been replaced with simply a key *K*.



Explain in detail the operation of this protocol.

A is authenticated to B

Key transport or key agreement? Key transport, A decides K and then sends to B

Implicit or Explicity Key authentication? Implicit

Last comment: Key hierarchies

- Key are often organised in a hierarchy. Keys in one level used to protect or generate keys in next layer down.
- Only lowest layer keys (*session keys*) used for data security.
- Top layer key is *master key*. Must be protected with care.

You should know the basic concept of key hierarchy

Keys are often organised in *key hierarchies*. Keys in one level of the hierarchy may only be used to protect keys in the next level down in the hierarchy. Only keys in the lowest level of the hierarchy are used directly to provide data security services.

This hierarchical approach allows the use of each key to be limited, thus limiting exposure and making attacks more difficult. For example, the compromise of a single session key (i.e. a key at the lowest level of the hierarchy) only compromises the information protected by that key.

So we have our session keys, keys used to protect and distribute these session keys, another layer to protect these keys and then we go up and up until we get to the very top.

The key at the top level of the hierarchy is referred to as the *master key*. Disclosure of a master key will potentially enable the

possessor to discover or manipulate all other keys protected by it (i.e. all keys in that particular hierarchy). It is therefore desirable to minimise access to this key, perhaps by arranging that no single user has access to its value.

Why use a key hierarchy

- The more we use a key the more likely it is to be compromised….
- Key establishment from nothing is hard…
- Use top layer keys sparingly (long lifetime)
  - These keys used for only for key establishment
- Low level keys used often (short lifetime)
  - Compromise of a session key of limited significance.

24

Key hierarchy has influence on the key lifetime – session keys are used very often, and also therefore changed very often. As you go up the layers the key lifetime increases. The master key is very rarely used – and attacker has very limited access to cases where this key is used.

So if you wish – session keys are easy to attack, but does not help the attacker much because it changes often. Master key stays the same a long time but master key is very difficult to attack.

Keys in different layers are not necessarily the same type or of equal strength – best practice is generally that a key must be 'stronger' than a key it is distributing. So we will not be distributing 128-bit AES keys with 56-bit DES keys.

You can construct a key hierarchy in a number of ways.

For example, there might be a University system whereby I need to compile a report every day on my activities So there is a University

key, there is a department key, there is a lecturer key, and then a daily session key for working with material that day. So regularly during the day use the session key, I upload weekly reports using lecturer key (get issues a new one for next week), department uploads to university once a month (renew key).

Example: Simple payment card

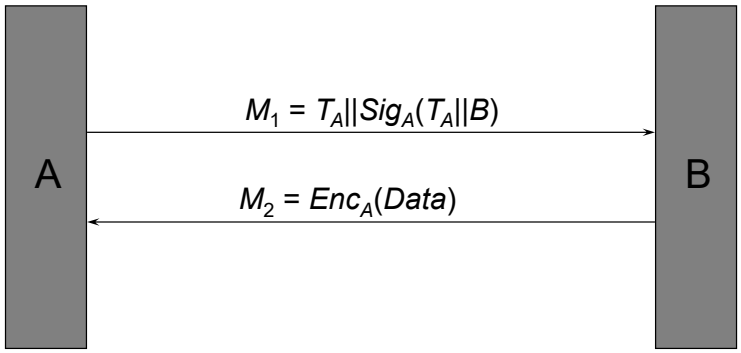
- Closed system
  - Cards/readers all controlled by same entity
- What keys do we need?
  - Card and reader need to share a key
- Problem?
  - Reader cannot store each possible card key!
  - So give all cards the same key?
  - One compromised card = completely broken system!
- The reader contains system master key
  - Card contains card key
  - Reader use master key and UID to derive card key
  - Reader and card communicate….
- If card compromised then only limited damage.

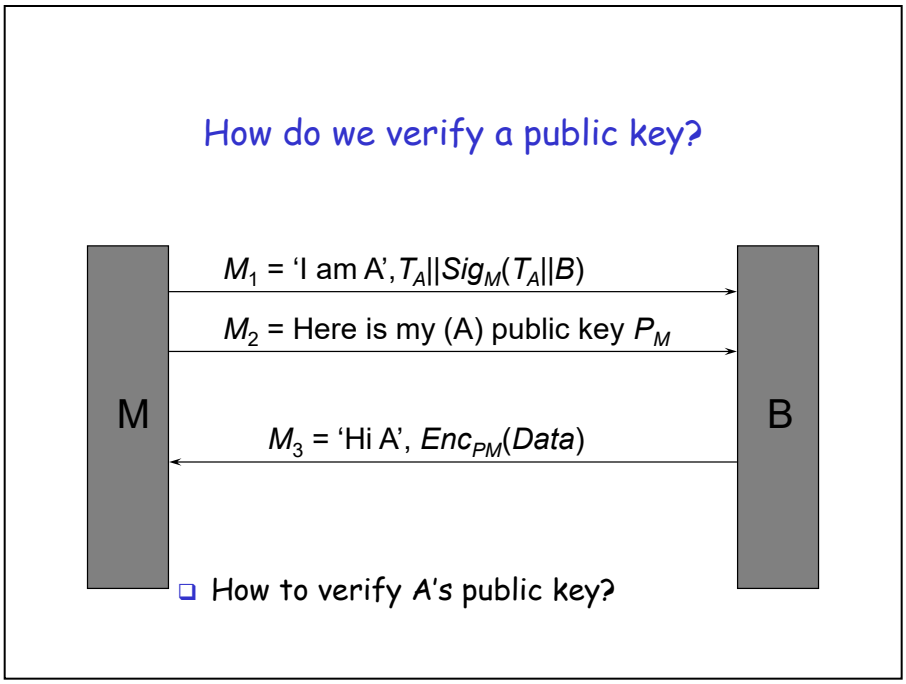
25

For interest

Public Key (Certificate) Management

Remember….we would like to use signatures (and PKE)





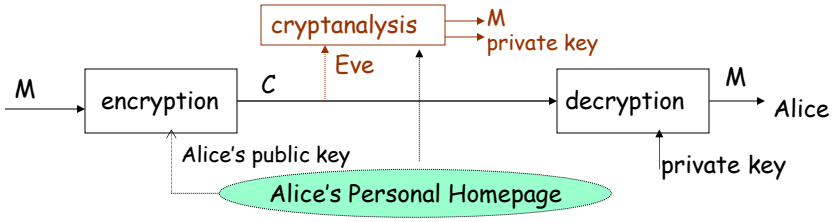
You should be able to explain why we need to properly verify a public key and how a public key certificate helps us in this matter.

In reality the protocol should look more like this – unless B gets the public key from someone else.

What if M tells B she is A and then give him a public key she claims is also As? Then we have a problem, therefore we need to be able to verify that this key does indeed belong to A.

Digital Certificates

- Public key encryption: encrypt using receiver's public key
  - sender **has to be sure** that the public key used for encryption is indeed the receiver's public key
- Digital signature: verify a signature
  - Verifier **has to be sure** that the public key used for signature verification is indeed the signer's public key
- **How can the encryptor / verifier be sure that the public key is authentic?**



- How about posting the public key at a personal homepage?
- How about sending the public key to the encryptor / verifier using email?

This is not easy problem to solve, we cannot simply send public keys by email or put them on websites as they can be changed.

What prevents me from sending you an email say I am A and sending you a random public key that I also claim is A's key? Then you encrypt data with it expecting that only A can decrypt…

Digital Certificates

- How it works:
  - There is an entity called Certification Authority (CA) in the system
  - CA has a public key which is ASSUMED to be well known
    - e.g. built-in, preinstalled into all the web browsers/operating systems
  - CA issues a certificate to each public key owner
  - The certificate bears (1) the public key owner's identity, (2) the public key, (3) a validity period of the certificate and (4) the CA's signature
  - By using the certificate, the CA vouches that the public key in the certificate is owned by the public key owner.
  - The CA publishes a Certification Practice Statement (CPS) that specifies the policies (including liabilities) governing the use of the certificates issued.
- Only the CA can create a legitimate certificate
  - Only the CA can generate the signature in the certificate which requires the knowledge of CA's private key
- Anyone can verify the authenticity of the certificate using CA's public key

  $Cert_A = (ID_A, PK_A, \text{expiry-date}, Sign_{CA}(ID_A, PK_A, \text{expiry-date}))$

Ok so the solution is a public key certificate – the most important bits"

What is the key, who does it belong to and when is it valid? This info is signed by a trusted party.

The Certification Authority

- The "CA" is responsible for:
  - identifying entities before certificate generation
  - Generating user key or verifying user key
  - ensuring the quality of its own key pair,
  - keeping its private key secret.
- The CA, before generating a certificate, ought to check that a user knows the private key corresponding to its claimed public key.

The discussion on proof of possession is for interest – but the CA is very important. You must know what a CA does and its importance in public key management.

The "CA" is trusted by its subscribers for the purposes of certificate generation. The "CA" is responsible for :

- identifying the entities whose public key information is to be incorporated into a certificate,
- ensuring the quality of the CA's own key pair used for generating certificates, and
- securing the certificate generation process and the private key used in the certificate generation process.

One issue of major importance not addressed in ISO/IEC 11770-1 concerns the situation where a user generates his/her own asymmetric key pair, and then requests the CA to generate a certificate for his/her public key. It is generally considered good practice for the CA to ask the user to provide assurance that the user possesses the private key corresponding to the public key

offered for signature (e.g. in the case of a signature key by signing a date-stamped statement to this effect, which the CA can then verify using the offered public key).

The use of these "proof of possession" techniques are mandatory for certificates conforming to IETF RFC 4211. However, the CA is free to enforce the POP in any way it feels is effective.

What can happen is this is not checked?

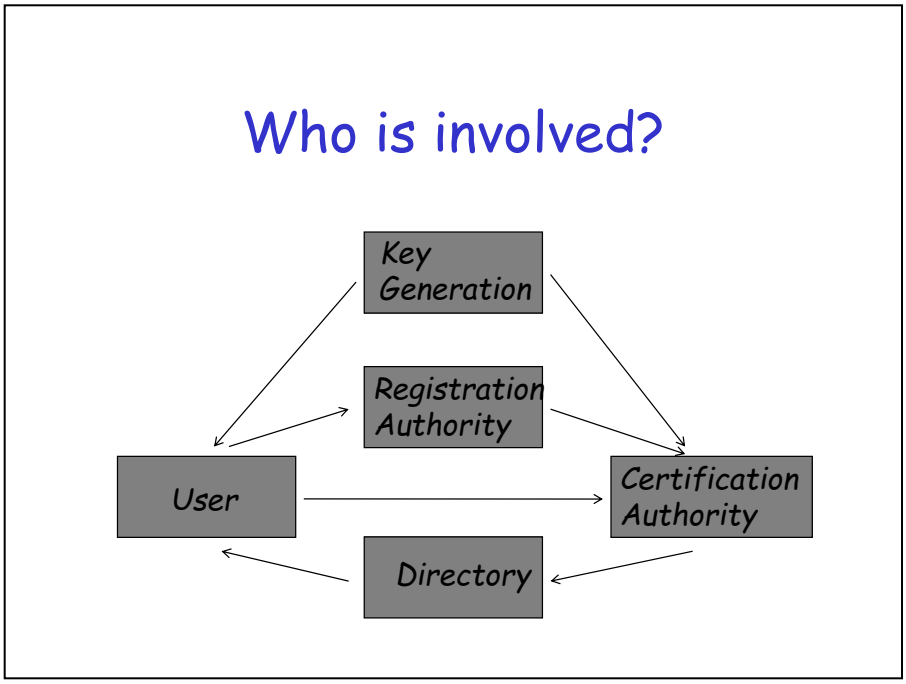
Such a procedure can avoid one user claiming to possess another user's key pair.

This undesirable consequences in certain situations. Can you think of any?

No encryption, signature.

You have a very valuable document, you anonymously publish it online (because you are afraid someone else might publish the same findings). So you sign it with your private key – so you can prove ownership later.. It turns out this is great work - so good that employers are lining up to hire the author.

I go and take you public key, go to the CA and get a certificate that says this is my public key. Now I can prove that this is my document?



You must be able to explain this diagram – who are the entities involves in PKI and what is each of roles in the process of issuing a certificate?

The diagram illustrates the information flows between the main entities who may be involved in the certification process.

Once again - Note that many of these different functional entities may be performed by the same actual entity.

Explain the process involved in certifying a public-key certificate in full.

I user decided that it wants a key and a certificate.

It get the key generation facility to generate it a key-pair. What can act as key generation facility?

User, TTP, CA…

KGF sends key pair to user, and public key to CA.

Meanwhile the user proves its identity to the RA, which then forwards this guarantee to the CA.

The user might communicate with the CA directly, proof of possession.

The CA generates the certificate.

The CA publishes the certificate in a directory – where is can be accessed.

Some Remarks on Digital Certificates

- Certificate authority (CA) is *considered as* a Trusted Third Party (TTP) that issues and signs certificates
  - Verifying CA's signature in a certificate only verifies the **binding validity** between the public key and the identity in the certificate vouched by the CA
  - Verifying CA's signature does **not** verify the identity of the source that the certificate comes from!
    - E.g. Alice may receive Carol's certificate from Bob
  - Certificates are public!
  - Common format for certificates is ITU-T X.509.

Important to note that giving someone a certificate does not proof anything about the person who gave it to you. The certificate only links the ID on it to the key on it.

X.509 certificates

- X.509 is a standard format for public key certificates (and CRLs).
- Standard first published in 1988.
- Latest was published in 2021.
- X.509 is part of the ITU-T Directory series of recommendations (ISO/IEC 9594).
- Certificate format specified in ASN.1.
- Note that the latest edition of X.509 covers 'attribute certificates' as well as public key certificates.

Slide 34 and 35 for interest only.

ITU-T recommendation X.509 contains a standardised format for public key certificates and certificate revocation lists (CRLs). This recommendation is also published as ISO/IEC 9594-8.

The latest edition of the recommendation was published in 2008. The certificate format specified is known as the Version 3 certificate – the same version as was described in the 2000 edition of the standard. This new version of the standard appears to contain only minor revisions.

The X.509 recommendation is part of the ITU-T X.500 Directory series of recommendations (which are also published as the multi-part ISO/IEC standard ISO/IEC 9594). Internet RFC 3280 contains the PKIX 'profile' for X.509 certificates.

The X.509 certificate format is specified in a language called

ASN.1. Abstract Syntax Notation One (bit like XML)

Finally, note that the latest edition of X.509 covers 'attribute certificates', i.e. certificates containing information about a user other than a user's public key, as well as public key certificates. The public key certificate proves the validity of a public key, but an attribute certificate can prove and fact – like that fact that you have a driver's license.

What's Inside a Certificate (X.509)

User Name
Certificate Version
Validity Period
Serial No
User's Public Key
Other user attributes
CA's name
CA's signature (of all the above)

e.g.

User Name (Common Name): www.hsbc.com.hk

Validity Period: Apr 16, 2010 – Apr 17, 2011

User's Public Key: RSA (2048 bits)

Modulus (2048 bits): 30 82 01 0a 02 82 01…

Exponent (24 bits): 01 00 01

CA's name (Issuer): VeriSign Class 3 Extended Validation SSL SGC CA

CA's signature (Certificate Signature Value): Size: 256 Bytes / 2048 Bits

There are many other attributes: Certificate serial no., certificate version number, HSBC public key algorithm, CA's signing algorithm, etc.

$Cert_A = (ID_A, PK_A, \text{expiry-date}, \ldots, Sign_{CA}(ID_A, PK_A, \text{expiry-date}, \ldots))$

Certificate Revocation

- A CA is responsible for the lifetime management of certificates, including renewal, update and revocation.
- Two methods for managing revocation:
  - use of Certification Revocation Lists, i.e. lists of revoked certificates, signed by CA
  - providing an on-line validation service.

Slidess 36-37 is mostly for interest. Know only what a CRL is, and understand why we need to revoke certificates. Why? As the keys might become compromised, user got new keypair, etc.

A CA is also responsible for the lifetime management of certificates, including recertification, update and revocation.

The revocation of a certificate means that it is deemed to be invalid before its expiry date.

There may be many reasons for this, e.g. compromise of the private key, name change, organisation change, etc. There are two commonly discussed methods for managing revocation:

• the use of Certification Revocation Lists (CRLs), i.e. lists of revoked certificates, signed by the CA,

• the provision of an on-line certificate status checking service (such as OCSP). So each time you want to use a certificate you ask the CA if it is still valid.

CRLs

- Each CRL entry contains the serial number of the X.509 certificate being revoked.
- CRLs must be updated at regular defined intervals, enabling CRL user to verify that they are in possession of the 'latest' version.

The X.509 (and the corresponding ISO standards) recommendation includes a standard format for CRLs.

Each CRL entry contains the serial number of the X.509 certificate being revoked.

It is a general requirement of CRLs that they are updated at regular defined intervals. This is the case even if it is exactly the same.

There is a drawback to regular updates – it means there is going to be a time when a revoked certificate is still valid. (until the update is made). Everyone understand that?

So this seems stupid, why not push an update when a certificate is revoked?

Why do CRLs need to be updated at regular intervals, even if there are no new certificates being revoked?

The problem is that the person checking the certificate validity has no idea whether is has the latest update…an attacker might be preventing these pushed updated from coming through. If we had regular updates, the user would realise that is does not have the latest CRL.

By publishing at know periodic intervals the user can be sure that he has the latest copy.

Online certificate status protocol

- Online Certificate Status Protocol (OCSP) enables certificate status to be queried.
- OCSP may provide more timely revocation information than is possible with CRLs.
- Entity issues status request to TTP and suspends acceptance of certificate until TTP gives response.
- The protocol specifies data exchanged between entity checking certificate status and the TTP providing that status.

The Online Certificate Status Protocol (OCSP), defined in both Internet RFC 2560 and ISO/IEC 15945, enables applications to determine the state of an identified certificate.

OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may also be used to obtain additional status information.

An end entity using OCSP services issues a status request to an OCSP TTP and suspends acceptance of the certificate in question until the TTP provides a response.

Name one reason why OCSP might not be as useful as CRL-based methods for revocation?

We create a bottle neck – every single check requires the TTP. There is also now a single point of failure – if you cannot reach the TTP the certificate is not seen as valid.

PKI

- Public Key Infrastructure (PKI) consists of all pieces needed to securely use public key cryptography
  - Key generation and management
  - Certification authorities, digital certificates
  - Certificate revocation lists (CRLs)
- No general standard for PKI
- We consider a few models of PKI

Think back to slide 32 when thinking about PKI.

PKI Trust Models

Monopoly model

- One universally trusted organization is the CA for the known universe
- Big problems if CA is ever compromised
- Big problem if you don't trust the CA!
  - Should country X trust CA in country Y?
  - Where and who would this CA be?

Anarchy model for interest only, you must know how the structured model works.

PKI Trust Models

- Anarchy model
  - Everyone is a CA!
  - Users must decide which “CAs” to trust
  - Used in PGP (Pretty Good Privacy)
    - www.pgpi.org
  - Why do they call it “anarchy”?
    - Suppose cert. is signed by Frank and I don’t know Frank, but I do trust Bob and Bob vouches for Frank. Should I trust Frank?
    - Suppose cert. is signed by Frank and I don’t know Frank, but I do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should I trust Frank?

The fist one is easy, the second is not so clear.

PGP – Anarchy Model

- **Unstructured**
  - Suppose a public key is received and claimed to be Alice's.
  - The public key and Alice's identity are signed by some others (CAs). Each signature is considered as a certificate:

    $Cert_{Bob}(Alice), Cert_{Carol}(Alice), Cert_{Dave}(Alice), Cert_{Eve}(Alice)$

    Example: if my trust in certificates issued by Bob, Carol, and Dave (whose public keys I already have valid copies) are 1/2, 1/3, 1/3, respectively (and I don't have Eve's public key), then the above public key for Alice is considered as trust-worthy as $1/2 + 1/3 + 1/3 \geq 1$

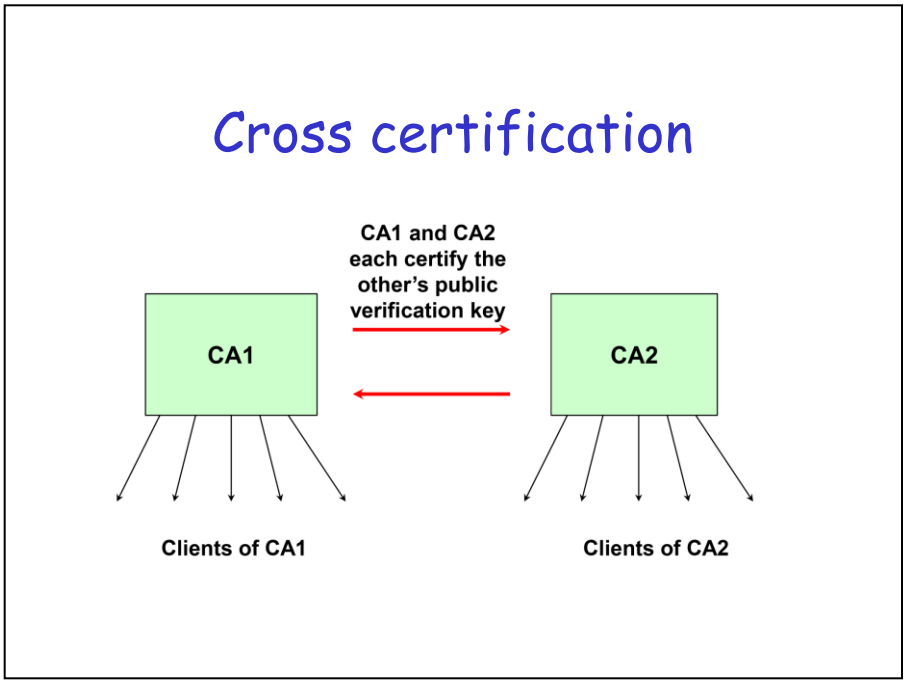
  - Scalability weakness
    - Trust is not transferrable. Alice trusts Bob, Bob trusts Carol, does not necessarily mean that Alice trusts Carol.
    - It may be cumbersome for one to get adequate certificates so that one's public key can be trusted.

PKI Trust Models

- Structured
  - Multiple trusted CAs
  - Used today
  - Browser may have tens of root CAs' public keys build-in
  - User can decide which CAs to trust (by default, you trust what the browser said to trust)

Understand how we need to verify each certificate, used the certificate of the CA who signed it, until we get to the root CA. You must realise how this is achieved through a 'chain' or verification.

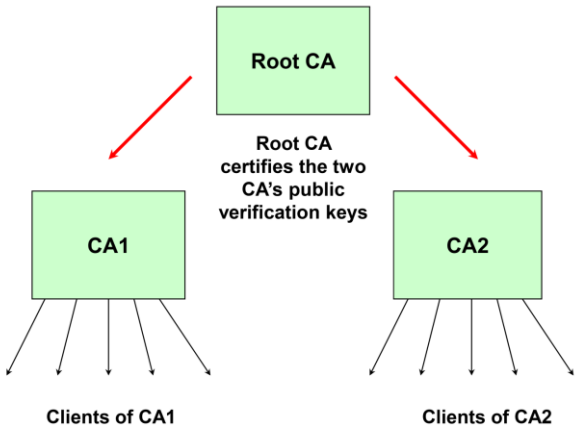
You also need to know who signs a Cas certificate and how the highest level CA's certificates are signed (cross or self certification)



Cross certification is the process where two CAs agree to accept certificates produced by the other.

This is usually done by producing a certificate that certifies the other CA's public key. This allows users to check the other CAs certificates are correct.

Certificate Hierarchy

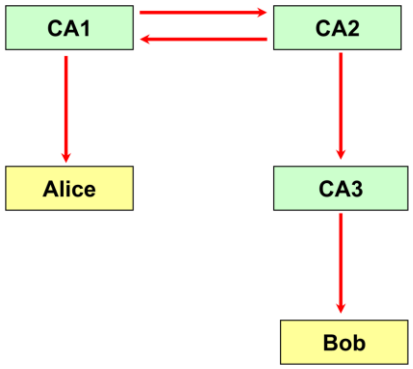


45

Alternatively we have one 'super' CA that only signs Cas certs

In some cases really important CAs can also self sign – how does this work? Well we use the public key on the cert to verify the cert signature (this means the cert if not modified an made by the CA) and we trust that the CA is honest..

Certificate Chains



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

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

Alice gets signed message from Bob and his cert.

She verifies the signature with key on Bob's cert

She verifies Bob's cert with key on CA3 cert

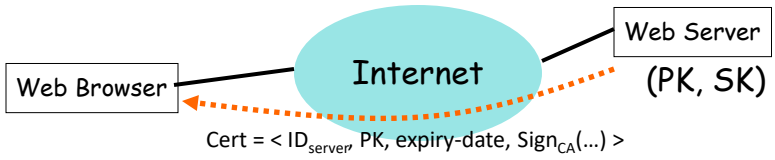
She verifies CA3s cert with key on CA2 cert

She verifies CA2's cert with key on CA1 cert (CA1 and CA2 cross certified)

She trusts CA1, therefore she trusts all the way down to Bob

This is a certificate chain.

How to Use PKI – Secure Web Browsing



- The web browser has the CA's public key built in.
  - **The legitimacy of the web browser software becomes crucial for ensuring the security of digital certificates**
  - A certificate is **NO** more secure than the security of the web browser
- In practice, each browser trusts multiple CAs rather than just one
- Exercise: find out the number of CAs that your browser trusts

In practice we often store the public key of several CAs, this means we can verify public key certificates that are sent to us.

Payment Card Example

- EMV credit cards
  - Open payment (different card vendors and payment terminals different systems)
  - Authentication of card/transaction data
  - Card vendors not giving terminal vendor their master symmetric key!!! So what now?
- How can we authenticate if we do not trust others with our secret keys?
- Use public key crypto – use certificates!
  - Step 1: Card signs transaction data and also sends the card's certificate
    - Card certificate signed by card vendor
  - Step 2: Terminal has card company certificate, so verifies card certificate, then verifies card signature

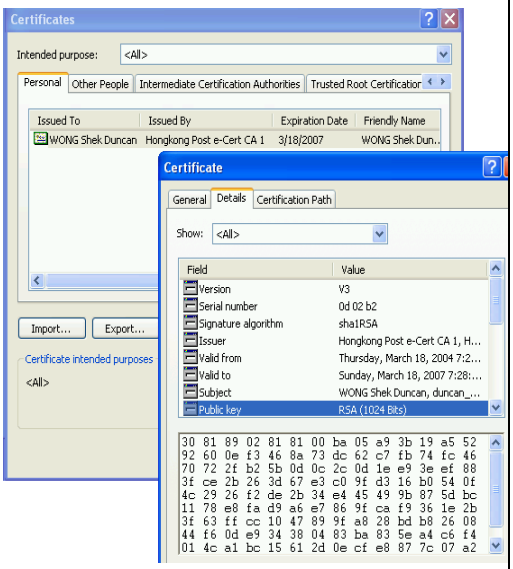
PKI and e-Commerce

- A PKI can be used to ensure secure transactions on the Internet. This is especially important to foster e-commerce development.
- PKI implementation provides a solution for the business/legal aspects of electronic transactions.
- To bring such transactions to equivalent footing as traditional transactions some countries have established legislation governing electronic transactions.
- For example, in Hong Kong, the 'Electronic Transactions Ordinance' (http://www.ogcio.gov.hk/eng/eto/eeto.htm) was established in January 2000.
  - Give legal status to electronic records and transactions, digital signatures.
  - Give recognition to the first public CA, which went into operation in February 2000: Hong Kong Post e-Cert.

Rest of the slides for interest only.

Personal Certificates

- Why want personal certificates?
  - send signed email messages
  - allow your peers to send encrypted emails to you
- Hongkong Post e-cert (Hong Kong's CA)
  - HK$ 50 for a year
  - E-cheque
  - Online (some banks login, tax returns, etc).



You can get one for about 50HKD a year.

PKI Implementation Hurdles

- PKI does have great success, e.g. web security/TLS
- PKI is still not widely used for personal/business transactions. Good security and structural aspects but:
  - Not easy to understand by laymen.
  - Users have little incentive to get certificates as most applications are not PKI-enabled
  - Legal recognition largely untested
    - Very few real court cases (world-wide).
    - Must have certificate from recognised trust service provider
    - Very small number of these
- In Hong Kong, most of the applications support both personal certificate and password-based authentication methods.
  - Only two accepted CAs

For example, EU has similar legal issue to ETO. But very few recognized trusted CAs (some counties only has one).

Key management is boring!

Maybe, but it is very important!

- Without keys we have no security services

52

Rest of the slides only for interest….

Coldboot
https://en.wikipedia.org/wiki/Cold_boot_attack

E-passport

- Sometimes need to design for interesting case
- E-Passports (governed by ICAO)
- Passive Authentication
  - Data signed
  - Who needs to verify?
- Basic access control
  - Verify symmetric key
  - Who needs this key?

53

You travel everywhere – are these guys online? Do they like your government?

Maybe not – so you cannot share symmetric keys. This would also mean the person you are sharing with can recreate the passport (as they have the key).

E-passport c'ntd

- Example of key management things going wrong!
- Basic access control key size
  - Theory: Serial no, birthday, expiry date = 50 bits
  - Reality: Predictable values, so closer to 25 bits
  - Had to fix: Alphanumeric random serial no
- Passive authentication verification
  - A few years after adoption not a lot of people were actually verifying the signature using country certificate

54

Only a few countries sign up to the PK directory. Some apparently manually exchange keys via diplomatic communication – not understanding the notion of 'public key'. So limited checks of the actual signature.

Advanced Threats

- Stuxnet was a multi-stage malware targeting intermediate systems (MS Windows,Siemens PCS7, WinCC) to reach its main objective (Siemens S7 PLCs).
- Stuxnet's delivery mechanism was based on the Microsoft Windows platform, but its primary objective was industrial control systems
- The control systems targeted were highly specific
- A number of zero-day exploits were deployed

55

Zero days

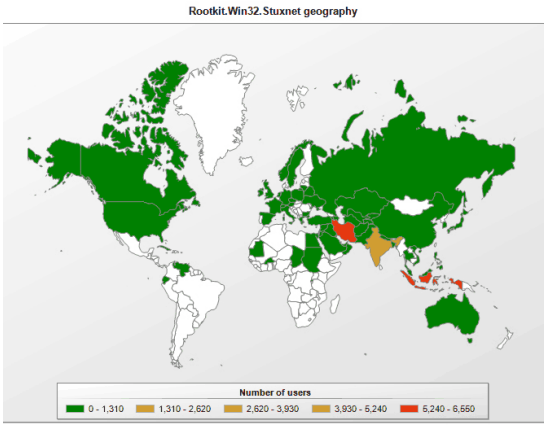
The attack vectors exploit print spoolers remotely, local files (typically via USB pen drives), and network

Shares

Operating system variants from Windows 2000 onwards (including XP, Vista, 7 and server variants 2008 and

2008R2) were catered for explicitly.

STUXNET (1)



- Snapshot of STUXNET infection (Sept 2010)
  by University of Maryland.

56

Stuxnet (2)

- Stuxnet uses different mechanisms, including hiding itself on removable storage media (to get to air-gapped systems)
- A device driver is installed looking for files matching the characteristics of the Stuxnet payload.
  - What do you need to install driver on Windows?
  - Private key of trusted vendor
    - Jan-June 2010(Realtek), Verisign revokes certificate
    - July…(JMicron Tech)
- Core part of this issue: two compromised keys

57

Realtek and Jmicron Tech of Taiwan.

Realtek – audio codecs/Bluetooth chips

Jmicron – USB, SATA, PCIe

The end!

Any questions…