INF3510 Information Security
University of Oslo
Spring 2016

Lecture 6

Key Management and PKI

Audun Jøsang

Key Management

- The strength of cryptographic security depends on:
  1. The size of the keys
  2. The robustness of cryptographic algorithms/protocols
  3. The protection and management afforded to the keys
- Key management provides the foundation for the secure generation, storage, distribution, and destruction of keys.
- Key management is essential for cryptographic security.
- Poor key management may easily lead to compromise of systems where the security is based on cryptography.

Key Usage

- A single key should be used for **only one** purpose
  – e.g., encryption, authentication, key wrapping, random number generation, or digital signature generation
- Using the same key for two different purposes may weaken the security of one or both purposes.
- Limiting the use of a key limits the damage that could be done if the key is compromised.
- Some uses of keys interfere with each other
  – e.g. an asymmetric key pair should only be used for either encryption or digital signatures, not both.

Types of Cryptographic Keys

- Crypto keys are classified according to:
  – Whether they're public, private or symmetric
  – Their intended use
  – For asymmetric keys, also whether they're static (long life) or ephemeral (short life)
- How many types of keys are there?
- 19 different types of cryptographic keys defined in: NIST Special Publication 800-57, Part 1, "Recommendation for Key Management"

  http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

Crypto Period

- The crypto period is the time span during which a specific key is authorized for use
- The crypto period is important because it:
  - Limits the amount of information, protected by a given key, that is available for cryptanalysis.
  - Limits the amount of exposure and damage, should a single key be compromised.
  - Limits the use of a particular algorithm to its estimated effective lifetime.

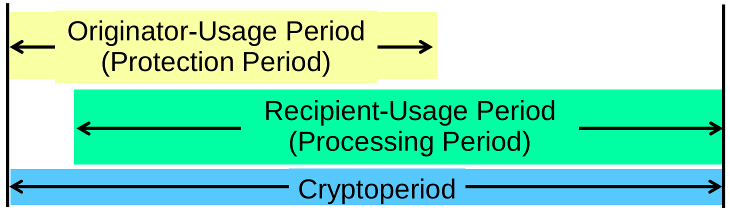
Factors Affecting Crypto-Periods

- In general, as the sensitivity of the information or the criticality of the processes increases, the crypto-period should decrease in order to limit the damage resulting from compromise.
- Short crypto-periods may be counter-productive, particularly where denial of service is the paramount concern, and there is a significant overhead and potential for error in the re-keying, key update or key derivation process.
- The crypto-period is therefore a **trade-off**

Key Usage Periods

- A key can be used for <u>protection</u> and/or <u>processing</u>.
  - Protection: Key is e.g. used to encrypt or to generate DigSig
  - Processing: Key is e.g. used to decrypt or to validate DigSig
- The **crypto-period** lasts from the beginning of the protection period to the end of the processing period.
- A key **shall not** be used **outside** of its specified period.
- The processing period can continue after the protection period.

<u>Cryptoperiod for symmetric keys</u>



Recommended Crypto Periods

Ref: NIST SP 800-57 Part 1

Key Type	Cryptoperiod	
	Originator-Usage Period OUP (Protection Period)	Recipient-Usage Period (Processing Period)
1. Private Signature Key	1-3 years	—
2. Public Signature Key	Several years (depends on key size)	
3. Symmetric Authentication Key	<= 2 years	<= OUP + 3 years
4. Private Authentication Key	1-2 years	
5. Public Authentication Key	1-2 years	
6. Symmetric Data Encryption Keys	<= 2 years	<= OUP + 3 years
7. Symmetric Key Wrapping Key	<= 2 years	<= OUP + 3 years
8. Symmetric RBG Key (Random Bit Generator)	(See SP800-90)	

Recommended Crypto Periods (cont.)

Ref: NIST SP 800-57

Key Type	Cryptoperiod: Originator-Usage Period OUP (Protection Period)	Cryptoperiod: Recipient-Usage Period (Processing Period)
9. Symmetric Master Key	About 1 year	
10. Private Key-Transport Key	<= 2 years	
11. Public Key-Transport Key	1-2 years	
12. Symmetric Key-Agreement Key	1-2 years	
13. Private Static Key-Agreement Key	1-2 years	
14. Public Static Key-Agreement Key	1-2 years	

Recommended Crypto Periods (cont.)

Ref: NIST SP 800-57

Key Type	Cryptoperiod: Originator-Usage Period OUP (Protection Period)	Cryptoperiod: Recipient-Usage Period (Processing Period)
15. Private Ephemeral Key Agreement Key	One key-agreement transaction	
16. Public Ephemeral Key Agreement Key	One key-agreement transaction	
17. Symmetric Authorization (Access Control) Key	<= 2 years	
18. Private Authorization (Access Control) Key	<= 2 years	
19. Public Authorization (Access Control) Key	<= 2 years	

Key strength comparison

Ref: NIST SP 800-57

Security Strength	Symmetric key algorithms	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
≤ 80	2TDEA[21]	$L = 1024$, $N = 160$	$k = 1024$	$f = 160\text{-}223$
112	3TDEA	$L = 2048$, $N = 224$	$k = 2048$	$f = 224\text{-}255$
128	AES-128	$L = 3072$, $N = 256$	$k = 3072$	$f = 256\text{-}383$
192	AES-192	$L = 7680$, $N = 384$	$k = 7680$	$f = 384\text{-}511$
256	AES-256	$L = 15360$, $N = 512$	$k = 15360$	$f = 512+$

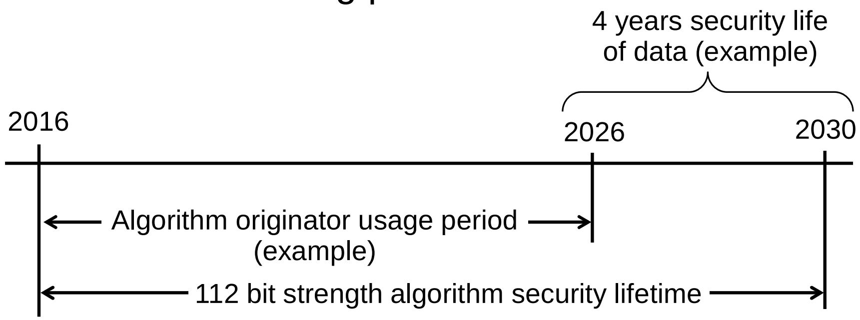
Security-strength time frame

Ref: NIST SP 800-57

Security Strength		Through 2030	2031 and Beyond
< 112	Applying	Disallowed	
	Processing	Legacy-use	
112	Applying	Acceptable	Disallowed
	Processing		Legacy use
128	Applying/Processing	Acceptable	Acceptable
192		Acceptable	Acceptable
256		Acceptable	Acceptable

Algorithm security life with 112 bit strength

- The algorithm strength determines how long the data will remain secure
- Imortant when encrypting data that must remain confidential for long periods.



Key Generation

- Most sensitive of all cryptographic functions.
- Need to prevent unauthorized disclosure, insertion, and deletion of keys.
- Automated devices that generate keys and initialisation vectors (IVs) should be physically protected to prevent:
  – disclosure, modification, and replacement of keys,
  – modification or replacement of IVs.
- Keys should be randomly chosen from the full range of the key space
  – e.g. 128 bit keys give a key space of 2^{128} different keys

When keys are not random

- Revealed by Edward Snowden 2013, NSA paid RSA (prominent security company) US$ 10 Million to implement a flawed method for generating random numbers in their BSAFE security products.
- NSA could predict the random numbers and regenerate the same secret keys as those used by RSA's customers.
- With the secret keys, NSA could read all data encrypted with RSA's BSAFE security product.

Random Number Generator Seeds

- RNG keys are used to initialise the generation of random symmetric and asymmetric keys
- Knowing the seed may determine the key uniquely
- Requires confidentiality and integrity protection
  – Periods of protection for seeds, e.g.:
    a. Used once and destroyed
    b. Used for multiple keys, destroyed after last key generation
    c. Kept and destroyed at the end of the protection period

Key Generation Examples

- Stream cipher keys
  - Long true random key stream (One-Time Pad), or
  - Short random key (e.g. 128 bits) input to keystream generator to generate pseudorandom key stream
- AES symmetric block cipher keys
  - Select adequate key length, 128, 192 or 256 bits
  - Ensure that any key is as probable as any other
- RSA asymmetric cipher
  - Make sure modulus $n = p \cdot q$ is sufficiently large to prevent factoring, e.g. $| n |= 4096$ bit
  - Randomness in seeds to generate primes p and q must by twice the security required. If e.g. 128 bit security is required then use 256 bit randomness

Compromise of keys and keying material

- Key compromise occurs when it is known or suspected that an unautorized entity has obtained a secret/private key.
- When a key is compromised, immediately stop using the secret/public key for **protection**, and revoke the compromised key (pair).
- The continued use of a compromised key must be limited to processing of protected information.
  - In this case, the entity that uses the information must be made fully aware of the risks involved.
  - Continued key usage for processing depends on the risks, and on the organization's Key Management Policy.

Key Compromise Recovery Plan

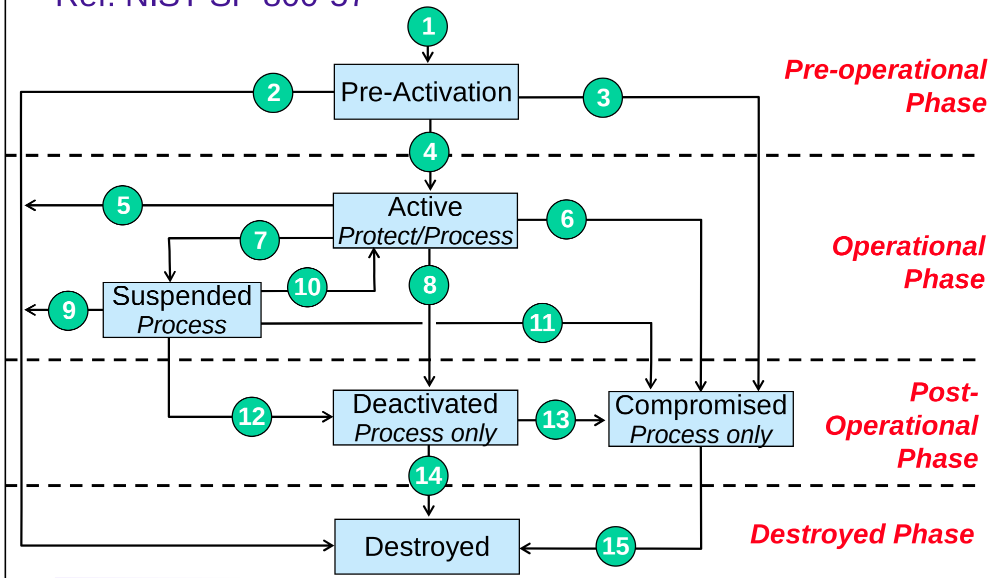
- A compromise recovery plan **should** contain:
  - The identification of the parties to notify.
  - The identification of the personnel to perform the recovery actions.
  - The re-key method.
  - Any other recovery procedures, such as:
    - Physical inspection of equipment.
    - Identification of all information that may be compromised.
    - Identification of all signatures that may be invalid due to the compromise of a signing key.
    - Distribution of new keying material, if required.

Undetected Key Compromise

- The worst form of key compromise is when a key is compromised without detection.
  - Nevertheless, certain protective measures can be taken.
- Key management systems (KMS) **should** be designed:
  - to mitigate the negative effects of (unknown) key compromise.
  - so that the compromise of a single key has limited consequences,
  - e.g., a single key should be used to protect only a single user or a limited number of users, rather than a large number of users.
- Often, systems have alternative methods for security
  - e.g. to authenticate systems and data through other means that only based on cryptographic keys.
- Avoid building a system with catastrophic weaknesses.

Key States, Transitions and Phases

Ref: NIST SP 800-57



Key Protection

- Active keys should be
  - accessible for authorised users,
  - protected from unauthorised users
- Deactivated keys must be kept as long as there is data protected by keys. Policy must specify:
  - Where keys shall be kept
  - How keys shall be kept securely
  - How to access keys when required

Key Protection Examples

- Symmetric ciphers
  - Never stored or transmitted 'in the clear'
  - May use hierarchy: session keys encrypted with master
  - Master key protection:
    - Locks and guards
    - Tamper proof devices
    - Passwords/passphrases
    - Biometrics
- Asymmetric ciphers
  - Private keys need confidentiality protection
  - Public keys need integrity/authenticity protection

Key destruction

- No key material should reside in volatile memory or on permanent storage media after destruction
- Key destruction methods, e.g.
  - Simple delete operation on computer
    - may leave undeleted key e.g. in recycle bin or on disk sectors
  - Special delete operation on computer
    - that leaves no residual data, e.g. by overwriting
  - Magnetic media degaussing
  - Destruction of physical device e.g high temperature
  - Master key destruction which logically destructs subordinate keys

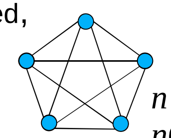
Why the interest in PKI ?

Cryptography solves security problems in open networks,
… but creates key management complexity.

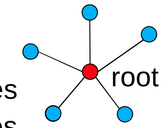
Public-key cryptography simplifies the key management,
… but creates trust management challenges.

Key distribution: The challenge

- Network with n nodes
- We want every pair of nodes to be able to communicate securely under cryptographic protection
- How many secure key **distributions** are needed ?
  - Symmetric secret keys: Confidentiality required,
    - $n(n-1)/2$ distributions, quadratic growth
    - Impractical in open networks
  - Asymmetric public keys: Authenticity required,
    - $n(n-1)/2$ distributions, quadratic growth
    - Impractical in open networks
  - Asymmetric public keys with PKI: Authenticity required,
    - 1 root public key distributed to n parties
    - linear growth
    - … more difficult than you might think



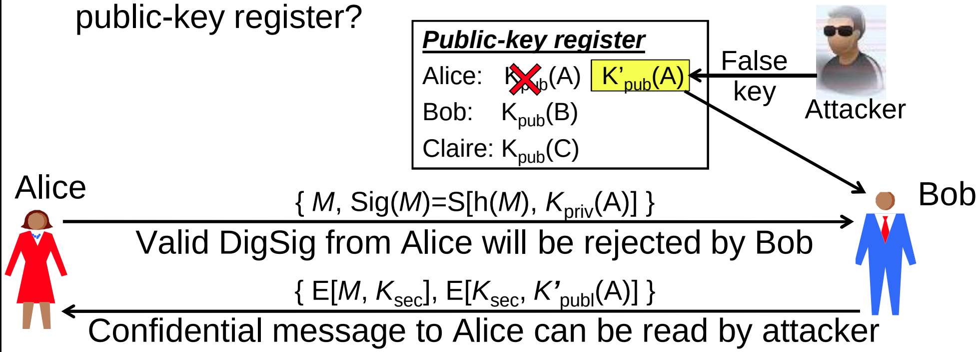
n nodes
$n(n-1)/2$ edges



n nodes
n edges

Problem of ensuring authentic public keys

- Assume that public keys are stored in public register
- Consequence of attacker inserting false key for Alice in the public-key register?



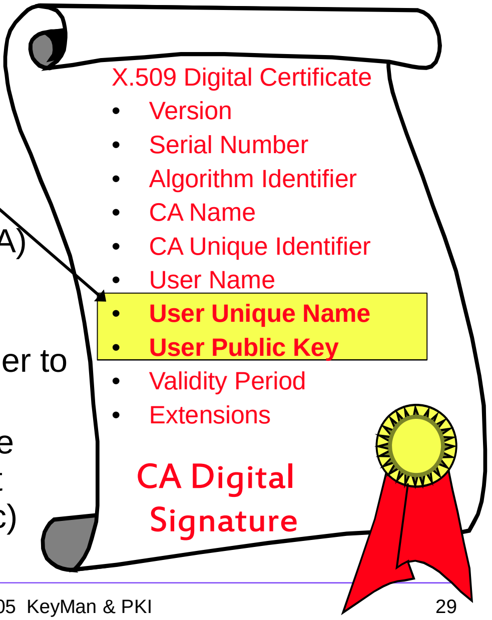
- Broken authenticity breaks security assumptions

Public-key infrastructure

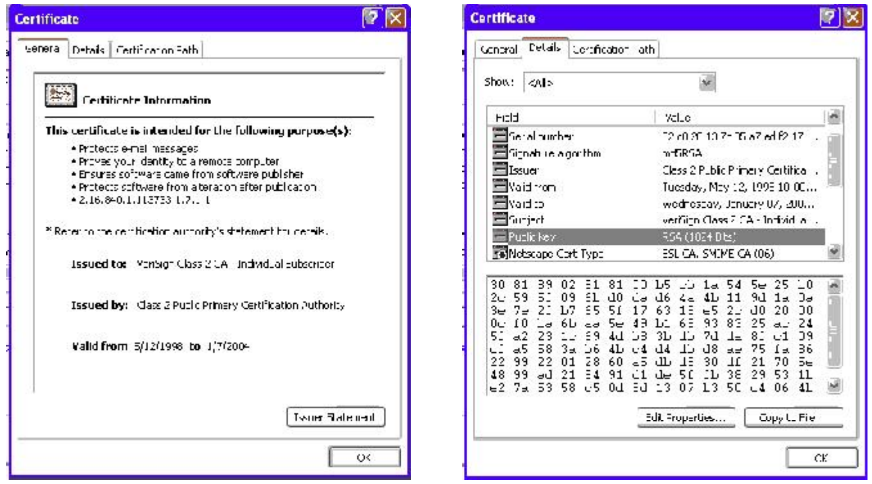
- Due to spoofing problem, public keys must be digitally signed before distribution.
- The main purpose of a PKI is to ensure authenticity of public keys.
- PKI consists of:
  - **Policies** (to define the rules for managing certificates)
  - **Technologies** (to implement the policies and generate, store and manage certificates)
  - **Procedures** (related to key management)
  - **Structure of public key certificates** (public keys with digital signatures)

Public-Key Certificates

- A public-key certificate is simply a public key with a digital signature
- Binds name to public key
- Certification Authorities (CA) sign public keys.
- An authentic copy of CA's public key is needed in order to validate certificate
- **Relying party** validates the certificate (i.e. verifies that user public key is authentic)

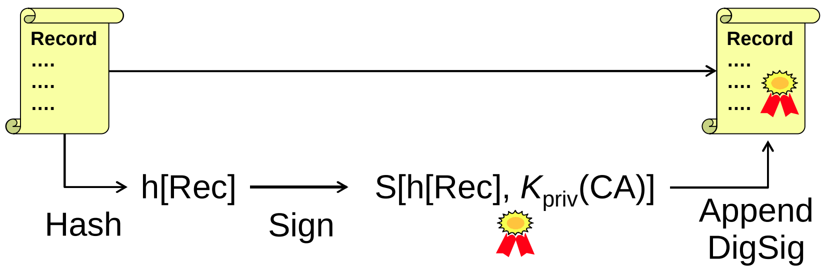


Example of X.509 certificate

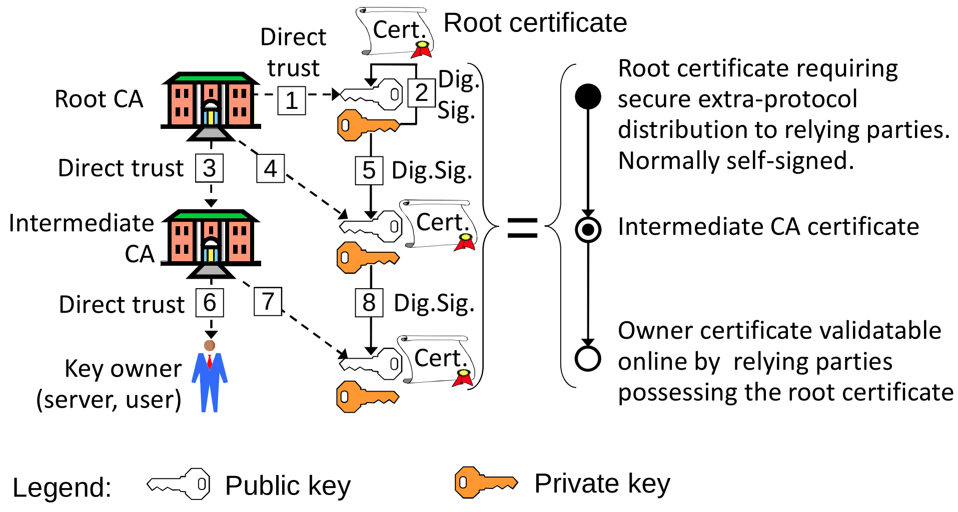


How to generate a digital certificate?

1. Assemble the information (name and public key) in single record Rec
2. Hash the record
3. Sign the hashed record
4. Append the digital signature to the record



PKI certificate generation

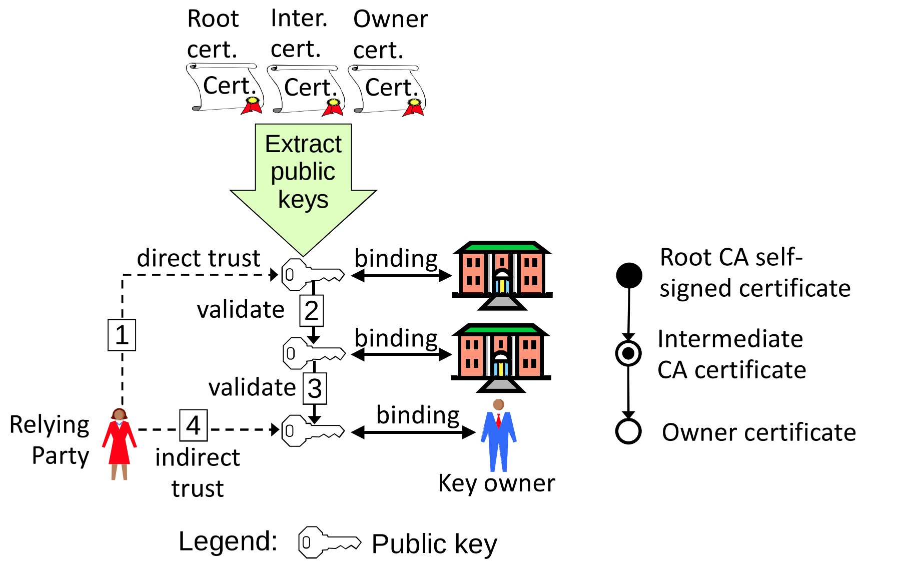


Self-signed root keys: Why?

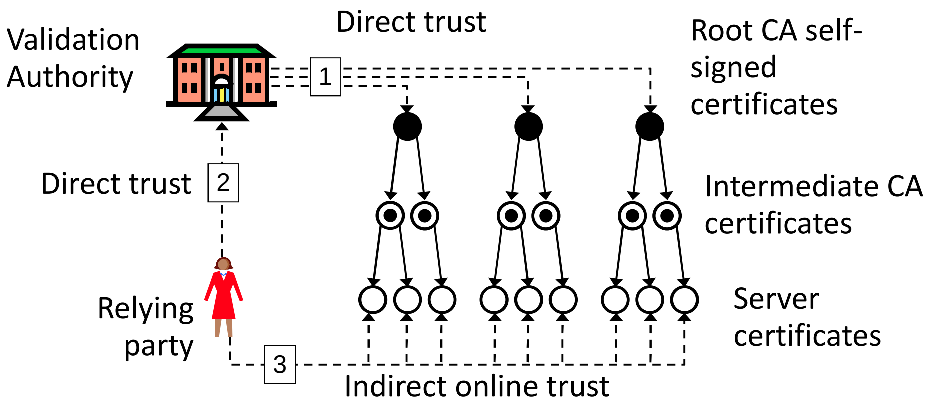
- Many people think a root public key is authentic just because it is self-signed
- This is deceptive
  - Gives impression of assurance
  - Disguises insecure distribution of root key
  - Gives false trust

- Self-signing provides <u>absolutely no security</u>
- Only useful purposes of self-signing:
  - X.509 certificates have a field for digital signature, so an empty field might cause applications to malfunction. A self-signature is a way to fill the empty field
  - Self-signature can be used to specify a cert as a root

Certificate and public key validation

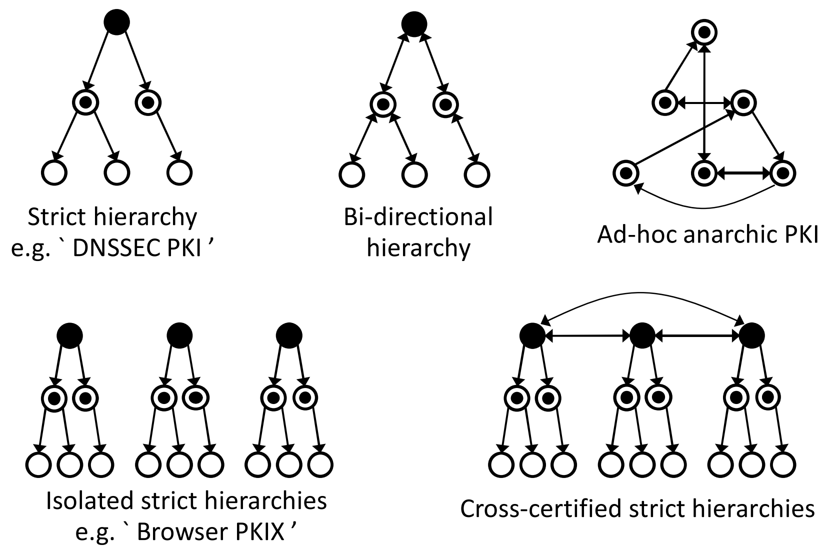


Validation Authorities



- A validation authority can assist relying parties to validate certificates

PKI Trust Models

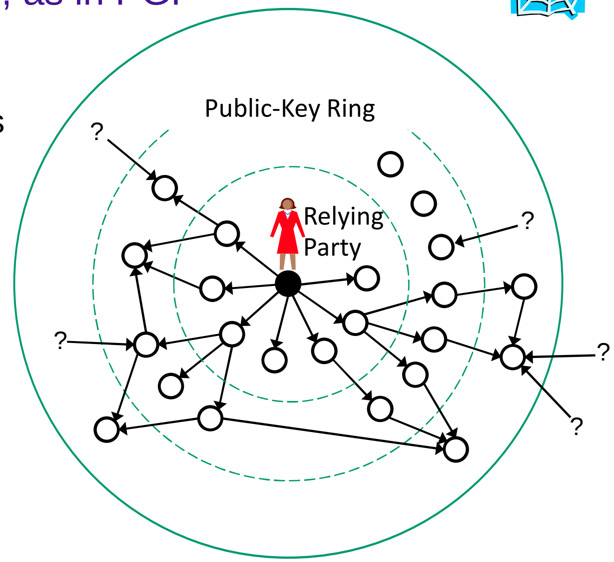


PKI trust models
Strict hierarchical model

- Advantages:
  - works well in highly-structured setting such as military and government
  - unique certification path between two entities (so finding certification paths is trivial)
  - scales well to larger systems
- Disadvantages:
  - need a trusted third party (root CA)
  - ‘single point-of-failure’ target
  - If any node is compromised, trust impact on all entities stemming from that node
  - Does not work well for global implementation (who is root TTP?)

Web of trust PKI model
User-centric model, as in PGP

- Each party signs public keys of others whose keys have been verified to be authentic.
- Public keys signed by trusted people can be considered authentic too.



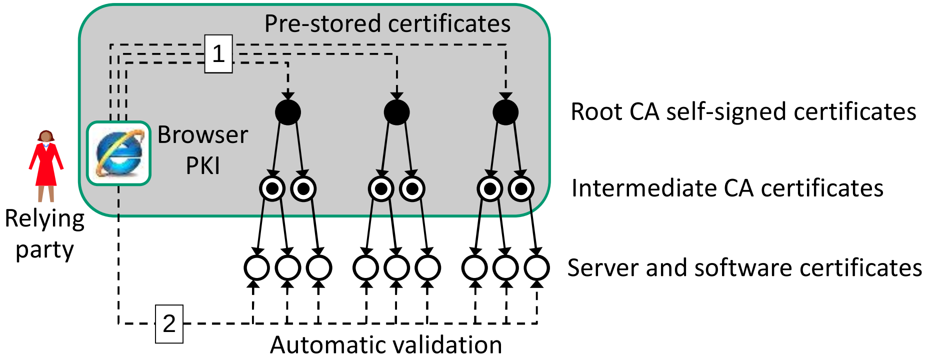
PKI trust models
User-centric model

- Each user is **completely responsible** for deciding which public keys to trust
- Example: *Pretty Good Privacy (PGP)*
  - ‘Web of Trust’
  - Each user may act as a CA, signing public keys that they will trust
  - Public keys can be distributed by key servers and verified by fingerprints
  - OpenPGP Public Key Server: http://pgpkeys.mit.edu:11371/
- GPG (Gnu Privacy Guard)
  - Open-Source version of PGP

PKI trust models
User-centric model

- Advantages:
  - Simple and free
  - Works well for a small number of users
  - Does not require expensive infrastructure to operate
  - User-driven grass-root operation
- Disadvantages:
  - More effort, and relies on human judgment
    - Works well with technology savvy users who are aware of the issues. Does not work well with the general public
  - Not appropriate for more sensitive and high risk areas such as finance and government

The Browser PKI
(PKI based on the X.509 certificates)



The browser PKI model consists of isolated strict hierarchies where the (root) CA certificates are installed as part of the web browser. New roots and trusted certificates can be imported after installation

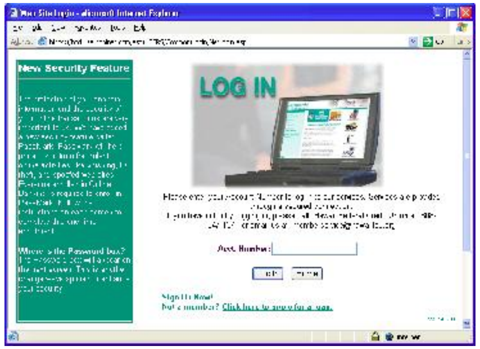
Browser PKI and malicious certificates

- The browser automatically validates certificates by checking: certificate name = domain name
- Criminals buy legitimate certificates which are automatically validated by browsers
  - Legitimate certificates can be used for malicious phishing attacks, e.g. to masquerade as a bank
  - **Malicious certificates are legitimate certificates !!!**
- Server certificate validation is not authentication
  - Users who don't know the server domain name cannot distinguish between right and wrong server certificates

Browser PKI root certificate installation

- Distribution of root certificates which should happen securely out-of-band, is often done through online downloading of browser SW
- Users are in fact trusting the browser vendor who supplied the installed certificates, rather than a root CA
- Example: used by *Mozilla Firefox* and *Microsoft Internet Explorer*
- Browser vendors decide which CA certs to distribute with browsers
  - This is an important political issue

Phishing and fake certificates Hawaii Federal Credit Union



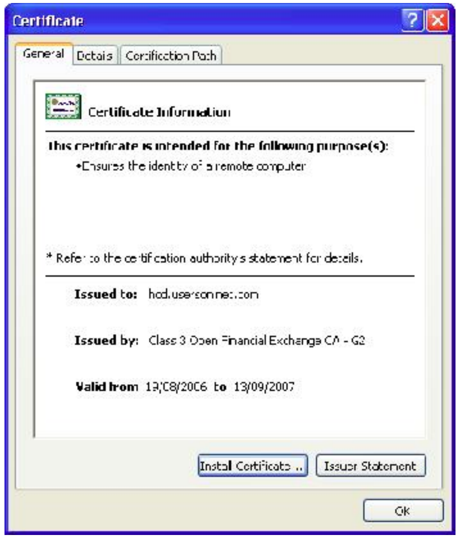
Authentic bank login

https://hcd.usersonlnet.com/asp/USERS/Common/Login/NettLogin.asp

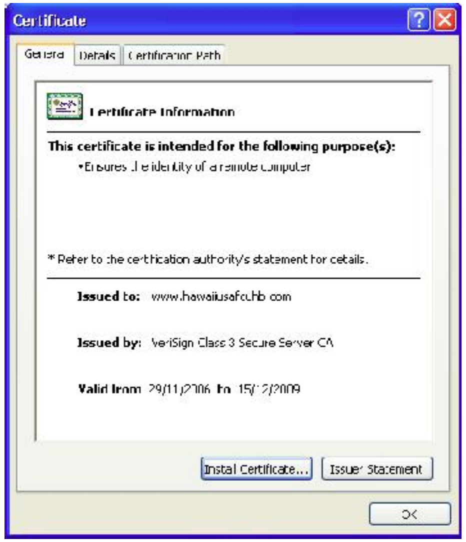
Fake bank login

https://hawaiiusafcuhb.com/cgi-bin/mcw00.cgi?MCWSTART

Authentic and Fake Certificates

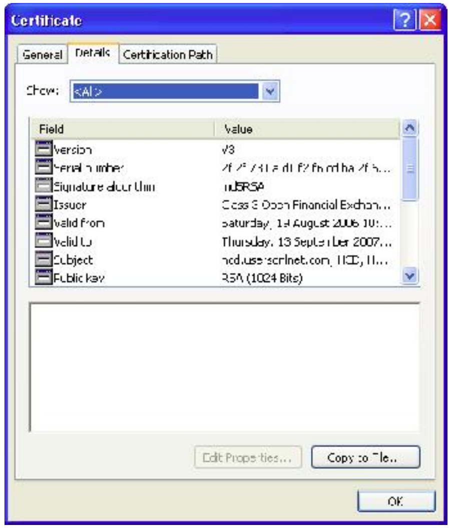


Authentic certificate

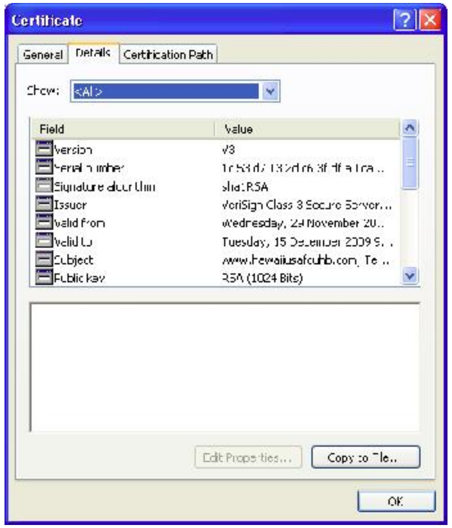


Fake certificate

Certificate comparison 2

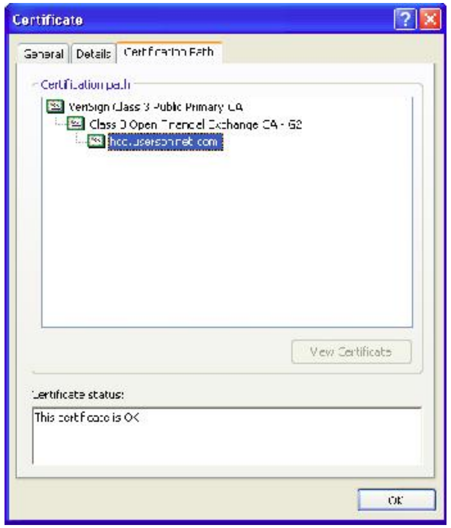


Genuine certificate

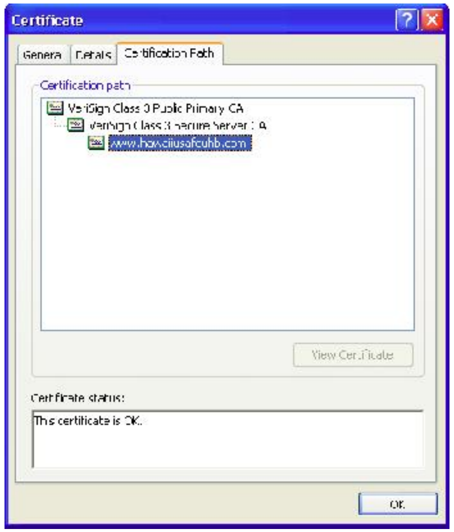


Fake certificate

Certificate comparison 3

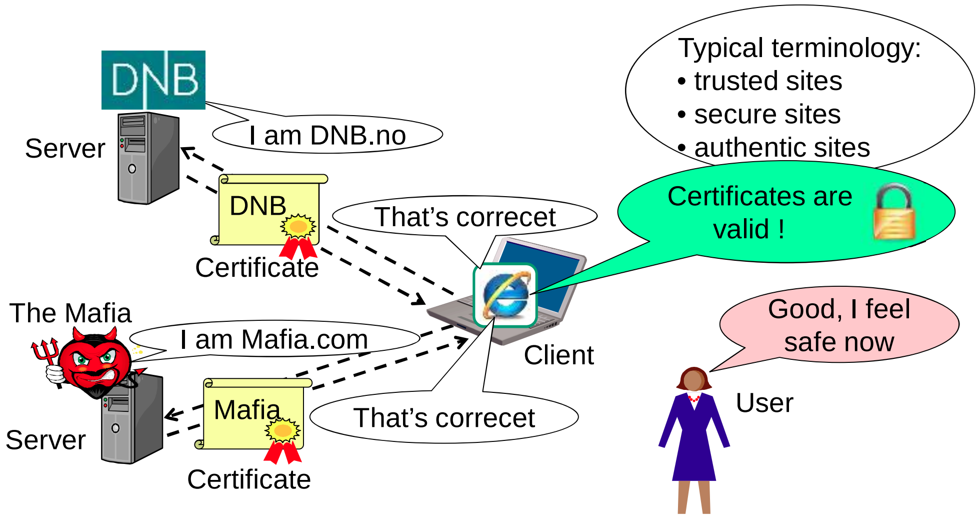


Genuine certificate



Fake certificate

Meaningless Server Authentication



Extended validation certificates

- Problem with simple certificates:
  - Can be bought by anonymous entities
- EV (Extended Validation) certificates require registration of legal name of certificate owner.
- Provides increased assurance in website identity.
- However, EV certificates are only about identity, not about honesty, reliability or anything normally associate with trust.
- Even the Mafia can buy EV certificates through legal businesses that they own.

Extended validation certificates

a) Normal website without encryption

Protocol changes from http to https when connection is encrypted with TLS/SSL.

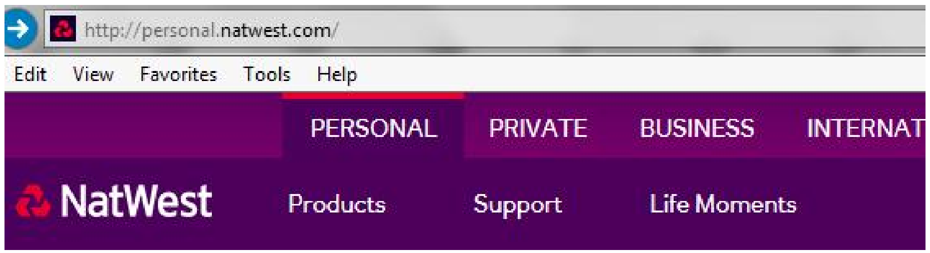
Domain name changes when going to secure website with encrypted traffic

The address bar turns from white to green, indicating to visitors that the website is using an Extended Validation Certificate.

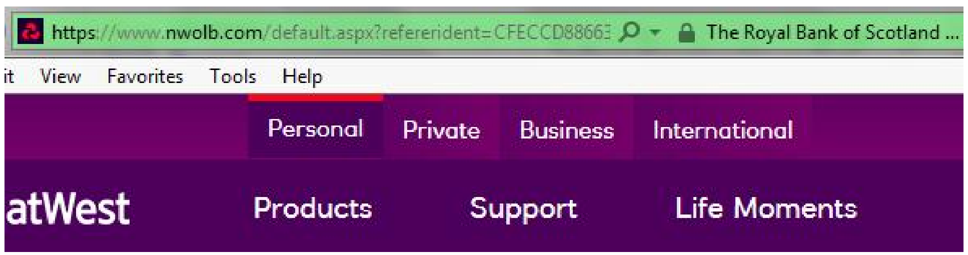
Legal name of website owner is displayed on the address bar when using Extended Validation Certificates.

b) Secure website with EV certifiate and encryption

Problem of interpreting EV Certificates



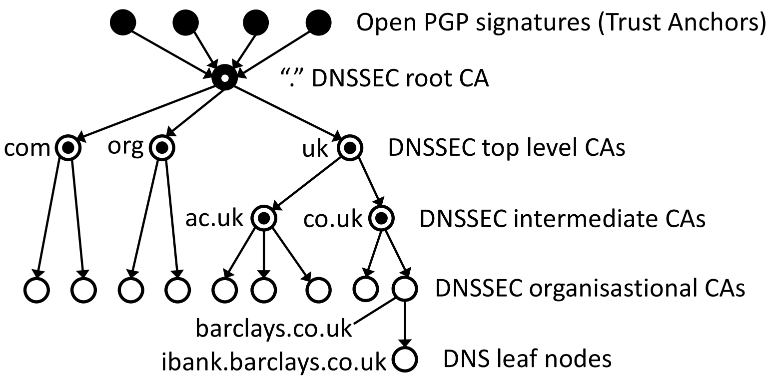
- Domain name and owner name not always equal
  - E.g. NatWest Bank is owned by Royal Bank of Scotland



Stuxnet with valid SW signature

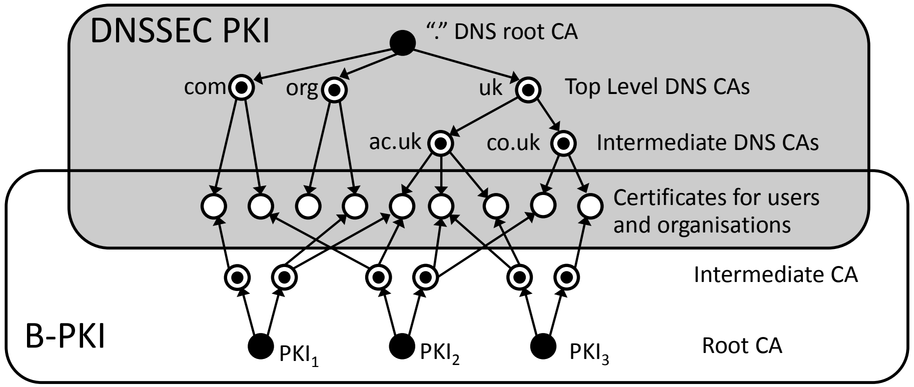
- Stuxnet worm is described as the most advanced malware attack ever, because
  - It used multiple zero-day exploits
  - It targeted a specific industrial control system
  - It was signed under a valid software (SW) certificate
- Stuxnet worm could be automatically validated by every browser in the whole world
- Anybody can buy SW certificates and sign whatever they want, even the Mafia !!!
- SW certificates only give evidence about who signed the SW, not that the SW is trustworthy.

DNSSEC PKI



- The DNS (Domain Name System) is vulnerable to e.g. cache poisoning attacks resulting in wrong IP addresses being returned.
- DNSSEC designed to provide digital signature on every DNS reply
- Based on PKI with a single root.

DNSSEC PKI vs. Browser PKI



- In B-PKI, any CA can issue certs for any domain → problematic
- CAs under the DNSSEC PKI can only issue certificates for own domain
- The DNSSEC PKI and the B-PKI both target the same user/org nodes
- DANE: DNSSEC-based Authentication of Named Entities
  - Alternative to B-PKI, standards exist, not deployed, complex

CRL: Certificate Revocation Lists

- Certificate Revocation
  - **Q: When might a certificate need to be revoked ?**
  - **A:** When certificate becomes outdated **before** it expires, due to:
    - private key being stolen or disclosed by accident
    - subscriber name change
    - change in authorisations, etc
- Revocation may be checked online against a certificate revocation list (CRL)
- Checking the CRL creates a huge overhead which threatens to make PKI impractical

PKI services

- Several organisations operate PKI services
  - Private sector
  - Public sector
  - Military sector
- Mutual recognition and cross certification between PKIs is difficult
- Expensive to operate a robust PKI
- The Browser PKI is the most widely deployed PKI thanks to piggy-backing on browsers and the lax security requirements
- DNSSEC PKI might replace the browser PKI

PKI Summary

- Public key cryptography needs a PKI to work
  - Reduces number of key distributions from quadratic to linear.
  - Digital certificates used to provide authenticity and integrity for public keys.
  - Acceptance of certificates requires trust.
  - Trust relationships between entities in a PKI can be modelled in different ways.
  - Establishing trust has a cost, e.g. because secure out-of-band channels are expensive.

End of lecture