INF3510 Information Security
University of Oslo
Spring 2018

Lecture 4

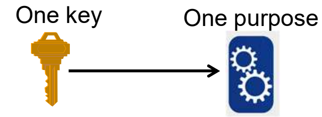
Key Management and PKI

Audun Jøsang

Key Management

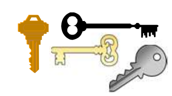
- The strength of cryptographic security depends on:
  1. The size of the keys
  2. The robustness of cryptographic algorithms/protocols
  3. The protection and management afforded to the keys
- Key management provides the foundation for the secure generation, storage, distribution, and destruction of 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)
- 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

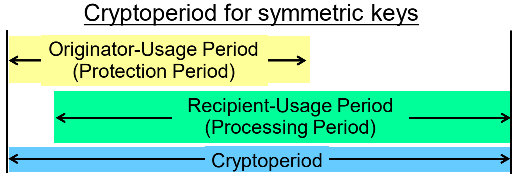
Crypto Period

- 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.

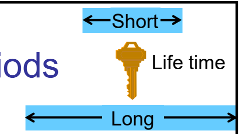
Crypto Periods

- 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>



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**

Security-strength time frame (ignoring QC)

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

Key strength comparison (ignoring QC)

Ref: NIST SP 800-57

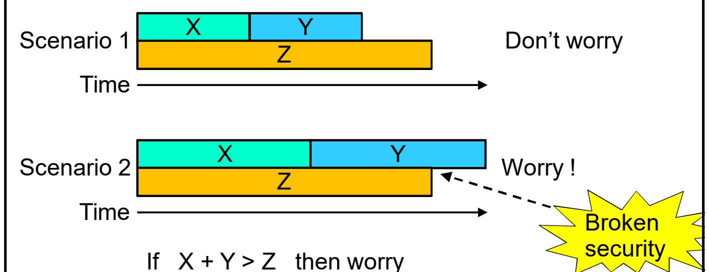
Security Strength	Symmetric key algorithms	FFC (e.g., DSA, D-H) Finite Field Cryptography	IFC (e.g., RSA) Integer Factorization Cryptography	ECC (e.g., ECDSA) Elliptic Curve Cryptography
≤ 80	2TDEA[21]	L = 1024 N = 160	k = 1024	f = 160-223
112	3TDEA	L = 2048 N = 224	k = 2048	f = 224-255
128	AES-128	L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

Towards a Catastrophic Crypto Collapse

- NIST (US National Institute of Standards and Technology) expects practical quantum computers to be built in the 2020s
- Impact on public-key crypto:
  - RSA
  - Elliptic Curve Cryptography (ECDSA)
  - Finite Field Cryptography (DSA)
  - Diffie-Hellman key exchange
- Impact on symmetric key crypto:
  - AES ➢ Need larger keys
  - Triple DES ➢ Need larger keys
- Impact on hash functions:
- SHA-1, SHA-2 and SHA-3 ➢ Use longer output

Should we worry about quantum computing?

X: Time it takes to develop post-quantum crypto
Y: Time period traditional crypto must remain secure
Z: Time it takes to develop praktical quantum computers



If X + Y > Z then worry

Key Generation

- Most sensitive of all cryptographic functions.
- Need to ensure quality, 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.

Compromise of keys and keying material

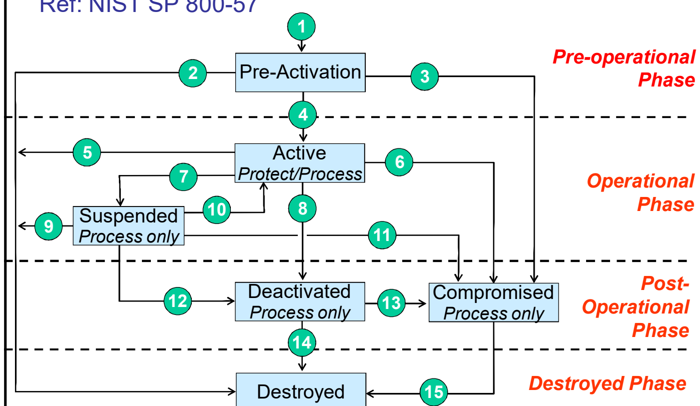
- Key compromise occurs when it is known or suspected that an unautorized entity has obtained a secret/private key.
- When a key is compromised, immediately stop using the secret/public key for **protection**, and revoke the compromised key (pair).
- The continued use of a compromised key must be limited 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.

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 exists 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

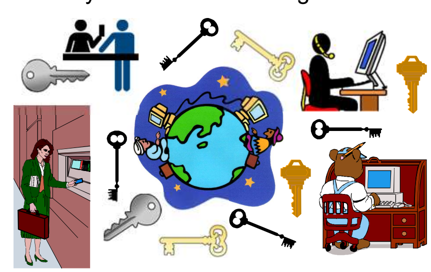
- Symmetric ciphers
  - Never stored or transmitted ‘in the clear’
  - May use hierarchy: session keys encrypted with master key
  - Master key protection:
    - Locks and guards
    - Tamper proof devices
    - Passwords/passphrases
    - Biometrics
- Asymmetric ciphers
  - Private keys need confidentiality 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 (several times)
  - Magnetic media degaussing
  - Destruction of physical device e.g high temperature
  - Master key destruction which logically destructs subordinate keys

Why the interest in PKI ?

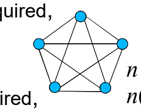
Cryptography solves security problems in open networks, … but creates key distribution challenges.

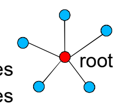


Public-key cryptography simplifies the key distribution, … but requires a PKI which creates trust management challenges.

Key distribution: The challenge

- Network with n nodes
- We want every pair of nodes to be able to communicate securely with 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 of public keys, quadratic growth
    - Impractical in open networks
  - Asymmetric public keys with PKI: **Authenticity** required,
    - 1 root public key distributed to n parties
    - linear growth
    - … easier, but still relatively challenging





Problem of non-authentic public keys

- Assume that public keys are stored in a public register
- What is the consequence if attacker replaces Alice's public key the register?

Public-key register

Alice: ~~$K_{pub}(A)$~~ $K'_{pub}(A)$ ← False key — Attacker

Bob: $K_{pub}(B)$

Claire: $K_{pub}(C)$

Alice → Bob: { M, Sig(M)=S[h(M), K_{priv}(A)] }

Valid DigSig from Alice will be rejected by Bob

Bob → Alice: { E[M, K_{sec}], E[K_{sec}, K'_{pub}(A)] }

Confidential message to Alice can not be read by Alice, but can be read by the attacker

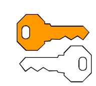
- Broken public-key authenticity breaks security assumptions

Public-key infrastructure

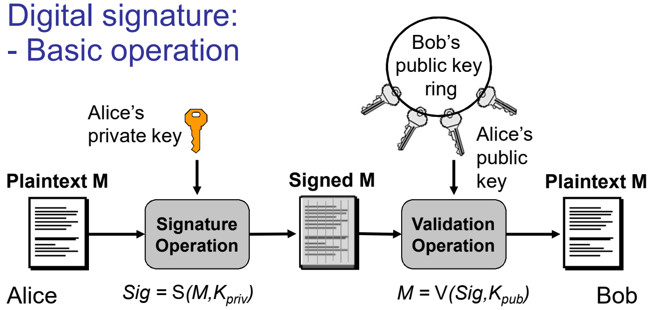
- Due to spoofing problem, public keys must be digitally signed before distribution.
- The main purpose of a PKI is to ensure authenticity of public keys.
- PKI consists of:
  - **Policies** (to define the rules for managing certificates)
  - **Technologies** (to implement the policies and generate,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

Digital Signature Mechanisms

- A MAC (Message Authentication Code) cannot be used as evidence to be verified by a 3rd party.
- Digital signatures can be verified by 3rd party.
  - Used for non-repudiation,
  - data origin authentication and
  - data integrity
- Digital signature procedures have three steps:
  - key generation (public-private key pair)
  - signing procedure (with private key)
  - verification procedure (with public key)

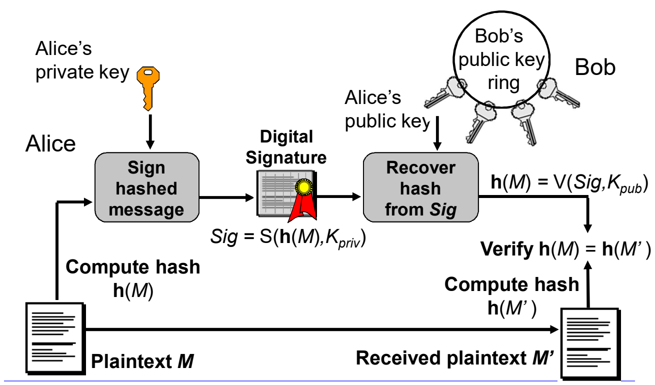


Digital signature: - Basic operation



- S: Signature operation (equivalent to decryption)
- V: Validation operation (equivalent to encryption)
- In practical applications, message *M* is not signed directly, only a hash value h(M) is signed.

Practical digital signature based on hash value

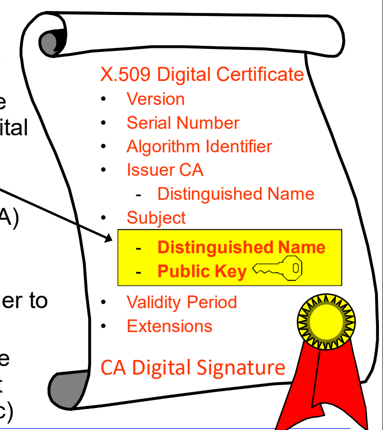


Problems for digital signatures

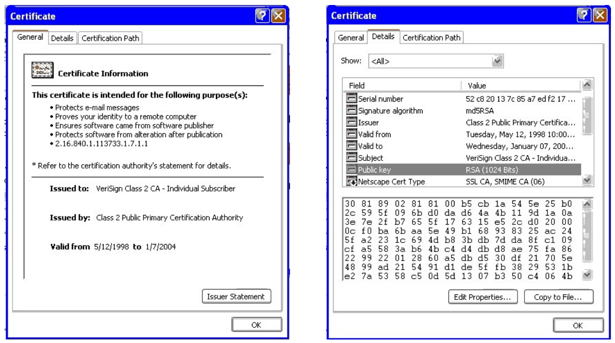
- Digital signatures depend totally on PKIs.
  - Reliable PKIs are hard to set up and operate.
- WYSIWYS (*What You See Is What You Sign*) means that the semantic content of signed messages can not be changed by accident or intent.
  - WYSIWYS is essential but very difficult to guarantee.
- Revoking certificates invalidates digital signatures.
  - Repudiate a signature by claiming theft of private key
- Key decay and algorithm erosion limits life time of digital signatures.
  - Future computers can falsify old signatures

Public-Key Certificates

- A public-key certificate is a data record containing a subject distinguished name and a public key with a digital signature by the CA
- 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)

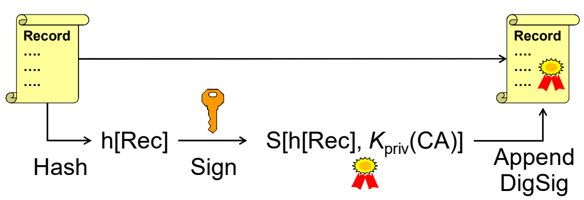


Example of X.509 certificate

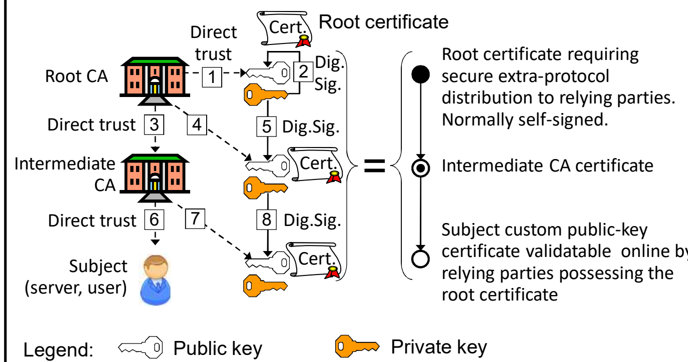


How to generate a digital certificate?

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



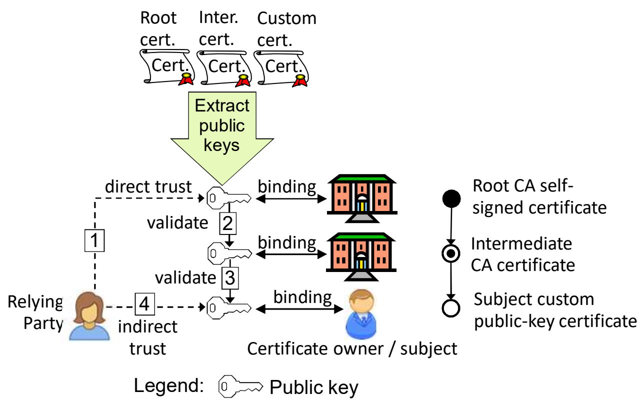
PKI certificate generation



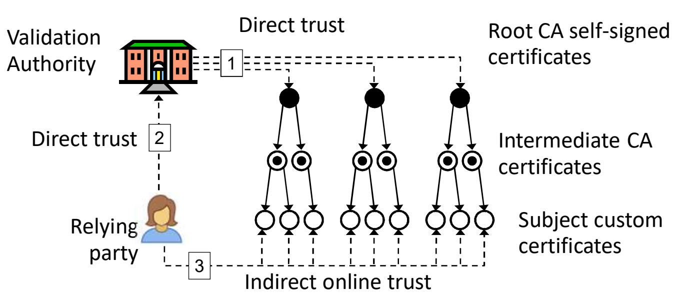
Self-signed root keys: Why?

- Many people think a root public key is authentic just because it is self-signed
- This is deceptive
  - Can give a false impression of assurance
  - Can be used to cover-up certificate falsification
  - Is used to spoof server certificates for TLS inspection
- Self-signing provides absolutely no security
- Only useful purposes of self-signing:
  - Provides a check-sum to detect accidental corruption
  - 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

Certificate and public key validation

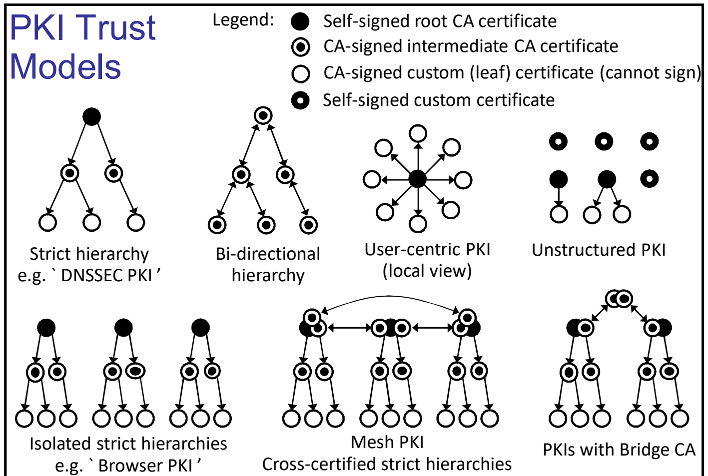


Validation Authorities



- A validation authority can assist relying parties to validate certificates

PKI Trust Models



Meaning of Trust for PKI

- **Trustworthy**: When it is objectively secure and reliable
- **Trusted**: When we decide to depend on it

- A root certificate is **trustworthy** when it has been received securely out-of-band from a reliable CA.
- A root certificate is **trusted** when it is being used to validate other certificates.

- Ideally, only trustworthy root certificates should be trusted
- In reality, many untrustworthy certificates are trusted.

Trust and Certification

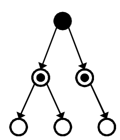
CA business relationships and policy define the PKI trust model

↕

The PKI trust model defines possible certification paths

PKI trust models

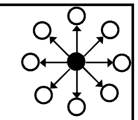
Strict hierarchical model



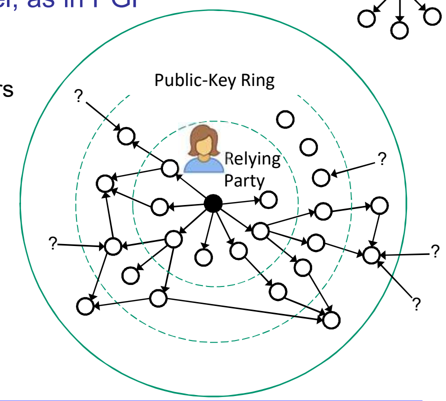
- Advantages:
  - works well in highly-structured setting such as military and government
  - unique certification path between two entities (so finding certification paths is trivial)
  - scales well to larger systems
- Disadvantages:
  - need a trusted third party (root CA)
  - 'single point-of-failure' 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

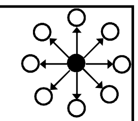


- Each party signs public keys of others whose keys have been verified to be authentic.
- Public keys signed by trusted people can be considered authentic too.



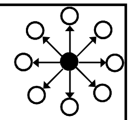
PKI trust models

User-centric model



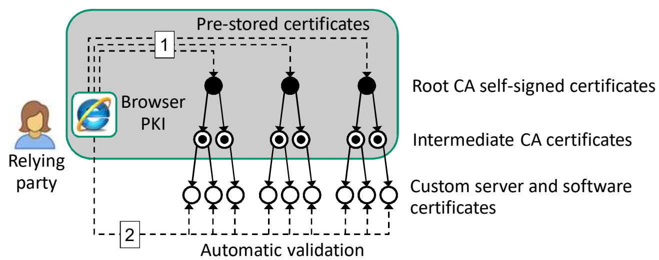
- Each user is **completely responsible** for deciding which public keys to trust
- Example: *Pretty Good Privacy (PGP)*
  - 'Web of Trust'
  - Each user may act as a CA, signing public keys that they will trust
  - Public keys can be distributed by key servers and verified by fingerprints
  - OpenPGP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

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

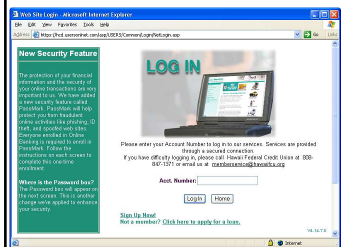
Browser PKI and malicious certificates

- 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 can be legit. certificates !!!**
- Server certificate validation is only syntactic authentication, **not** semantic authentication
  - Users who don't know the server domain name can *a priori* not know if it's a 'good' domain

Browser PKI root certificate installation

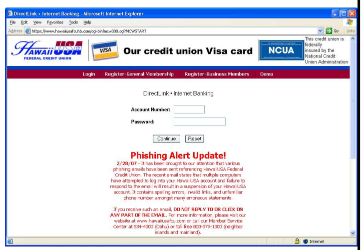
- Distribution of root certificates should happen securely out-of-band (not online)
  - But root certificate distribution is typically done by downloading browser SW
  - Is this secure ?
- Users must in fact trust the browser vendor who install the root certificates,
  - Example: *Chrome, Mozilla Firefox* and *Microsoft Edge*
  - Trust in the root CAs is only implicit
- Browser vendors decide which CA root certs to install
  - This is an important consideration for security
  - How do we know that a browser only contains trustworthy certificates ?

Phishing and fake certificates
Hawaii Federal Credit Union



Genuine bank login

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

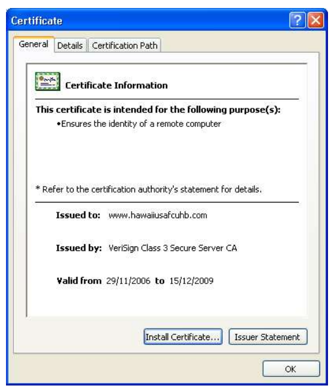


Fake bank login

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

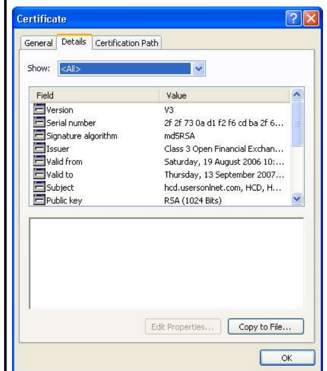
Authentic and Fake Certificates

Genuine certificate

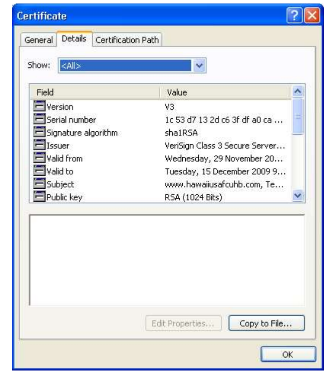


Fake certificate

Certificate comparison 2

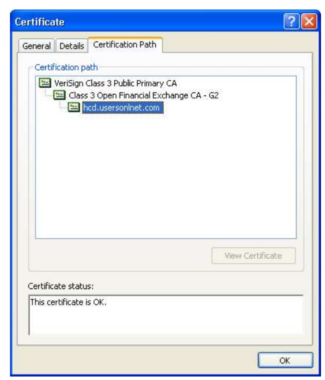


Genuine certificate

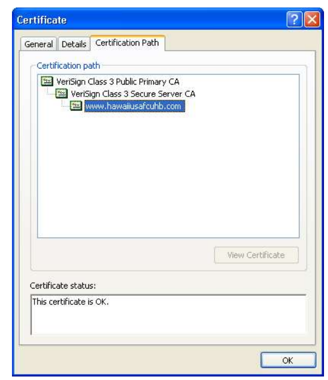


Fake certificate

Certificate comparison 3

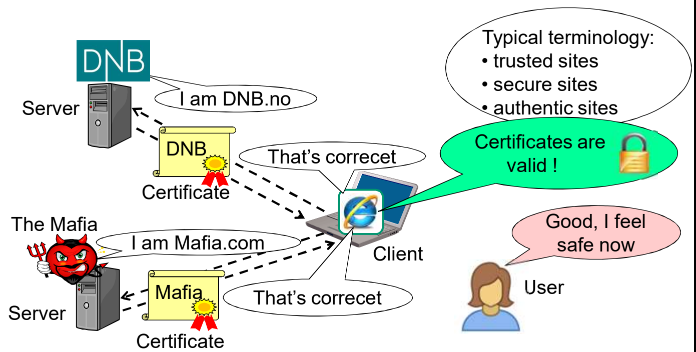


Genuine certificate



Fake certificate

Meaningless Server Authentication

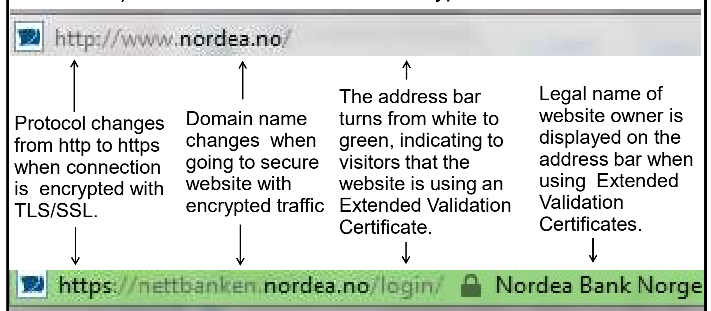


Extended validation certificates

- 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.

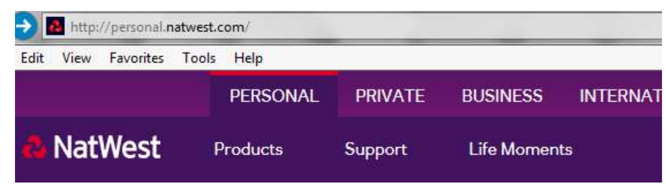
Extended validation certificates

a) Normal website without encryption

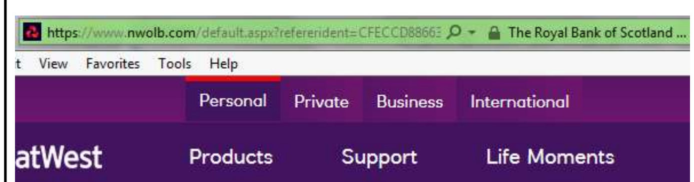


b) Secure website with EV certifiate and encryption

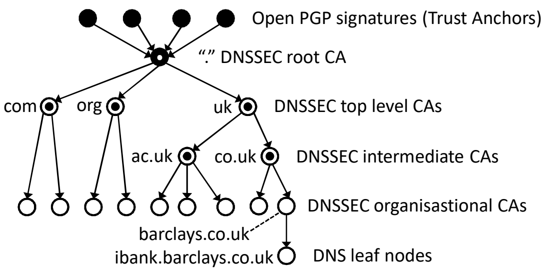
Problem of interpreting EV Certificates



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

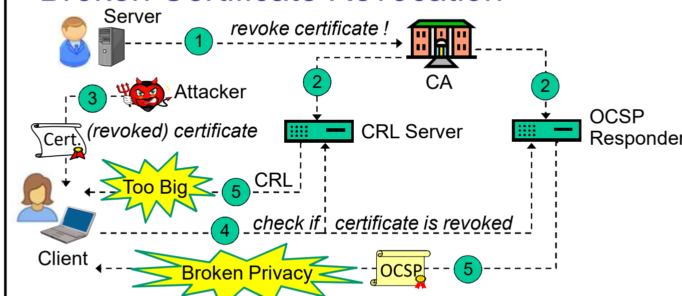


DNSSEC PKI



- 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.

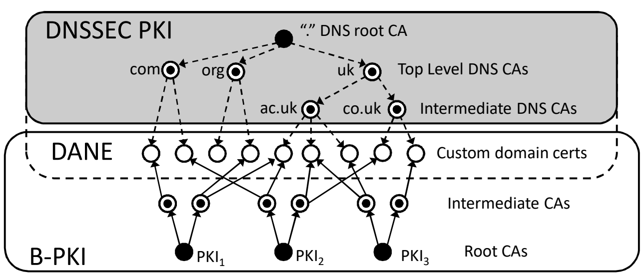
Broken Certificate Revocation



Traditional certificate revocation is broken, which is very serious

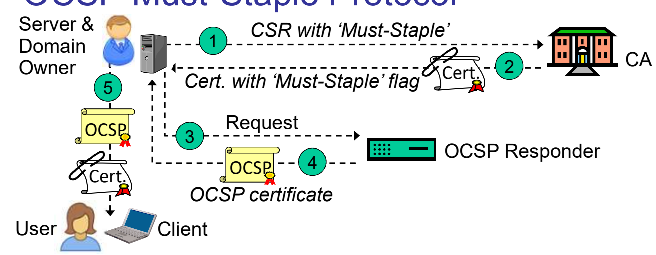
- CRL (Certificate Revocation List) Server
  - Does not scale, CRL size can be 100MByte
- OCSP (Online Certificate Status Protocol) Responder
  - Privacy issues: OCSP Responder knows user's browser habits

DNSSEC PKI vs. Browser PKI



- CAs in the DNSSEC PKI can only issue certificates under own domain
  - But normally not to custom domains
- CAs in the Browser PKI can issue certificates for arbitrary domains
- DANE: DNSSEC-based Authentication of Named Entities
  - Certificates for custom domains issued under DNSSEC PKI
  - Alternative to B-PKI, standards exist, but not widely deployed

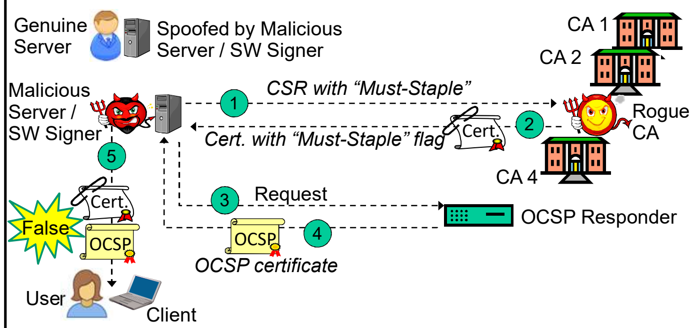
OCSP Must-Staple Protocol



The OCSP-Must-Staple protocol solves the revocation problem

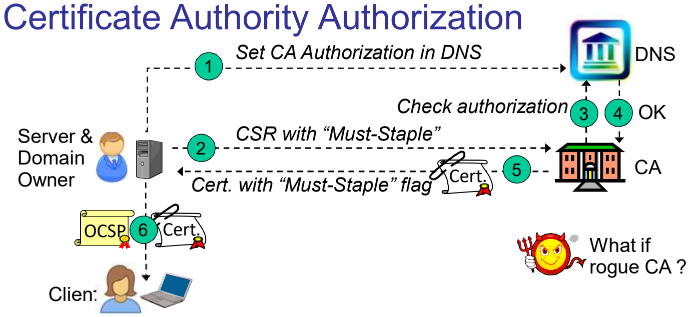
- CSR (Certificate Signature Request) with 'Must-Staple' flag
  - The 'Must-Staple' flag means that the server *must always* provide an OCSP certificate together with the server certificate
  - Client receives OCSP cert. from server, not from OCSP Responder
  - OCSP Responder does not know the user's browser habits
  - The server can request and cache a new OCSP certificate regularly

The Problem of Rogue/Compromised CAs



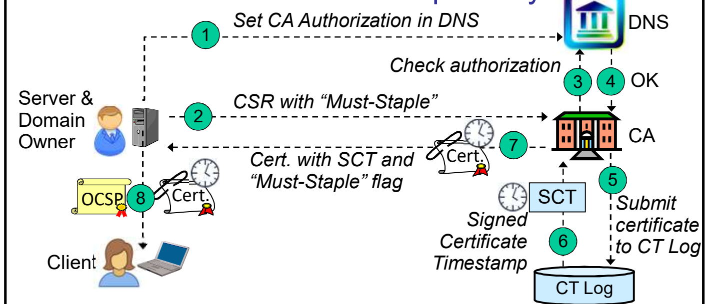
- Traditionally, any CA can issue certs for any domain (risky)
- The security of the whole Browser PKI is in principle broken if only one single CA is compromised or becomes rogue

Certificate Authority Authorization



- CA Authorization reduces problem of rogue/compromised CAs
- CA Authorization lets domain owner specify which CAs are allowed to issue certificates for that domain
- CAs must not issue certificates if they are not authorized
  - But a rogue/compromised CA could issue (false) certificates anyway

CA Auth. with Certificate Transparency



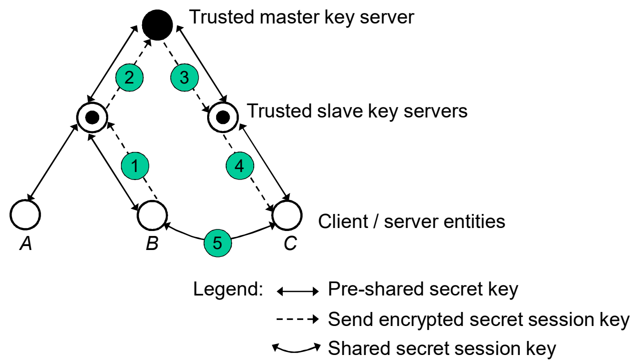
- Enforcement of CA Authorization is by logging every certificate
  - CT (Certificate Transparency) Logs are public block chains
- Certificates that have not been logged will be rejected by client
  - Domain owner must check CT Log for every certificate issued to its domain
  - Any illegal certificate found on the CT Log must be revoked !

Alternatives to PKI based on RSA and ECC

- Quantum computing (QC) breaks PKI based on RSA and ECC
  - It is predicted that practical QC will become available by 2030
- PKI supports confidentiality (encryption) and digital signature
- Questions:
  - For how long must data be kept confidential ?
  - For how long must a digital signature be valid ?
- Don't use PKI (RSA/ECC) if the answer is "Until after 2030" !!!
- Alternatives to PKI based on RSA/ECC:
  - PKI based on Post Quantum Crypto (not standardized/implemented yet)
  - Symmetric Key Infrastructure (SKI)
- SKI supports confidentiality ✓
- SKI does not support digital signatures ✗
  - Does not support non-repudiation (which requires digital signature)
  - Can only support mutual authentication between two parties

SKI: Symmetric Key Infrastructure

- Scenario for how entity *B* establishes session key with entity *C*



PKI Summary

- Public key cryptography needs a PKI in order to be practical
- It is complex and expensive to operate a robust PKI
- PKI services are called 'Trust Services' in EU's Digital Agenda
  - Intended as a security foundation for e-Id and e-Services in the EU
- Establishing initial trust in PKIs has a cost, because it is expensive to use secure out-of-band channels needed for distributing root certificates
- The Browser PKI is the most widely deployed PKI thanks to the distribution of root certificates with web browsers
- Traditional PKIs are insecure if long-term protection is required

End of lecture