

INF3510 Information Security
University of Oslo
Spring 2014

Lecture 6

Key Management and PKI



Audun Jøsang

Key Management

- The security of cryptographically protected information depends on:
 - The size of the keys
 - The robustness of cryptographic algorithms/protocols
 - The protection and management afforded to the keys
- Key management provides the foundation for the secure generation, storage, distribution, and destruction of keys.
- Proper key management is essential to the robust use of cryptography for security.
- Poor key management may easily lead to compromise systems protected with strong algorithms.

Key Usage

- A single key should be used for **only** one purpose
 - e.g., encryption, authentication, key wrapping, random number generation, or digital signatures
- 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

- How many types of keys are there?
- 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)
- NIST Special Publication 800-57, Recommendation for Key Management – Part 1: General, August 2005, defines **19** types of cryptographic keys.

Crypto Period

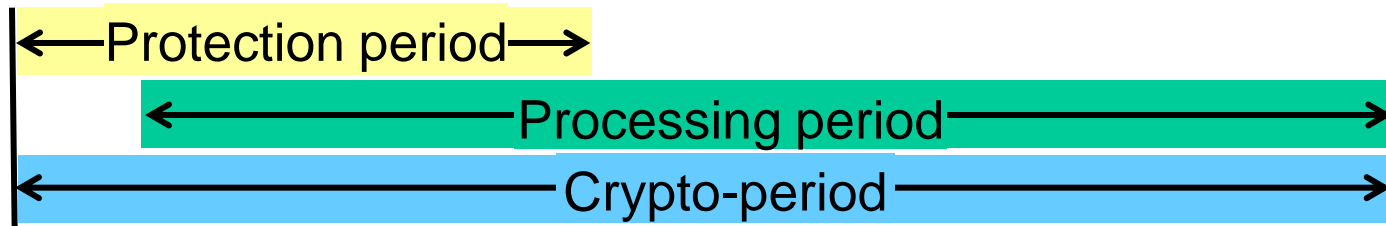
- The crypto period is the time span during which a specific key is authorized for use
- The crypto period is important because it:
 - Limits the amount of information, protected by a given key, that is available for cryptanalysis.
 - Limits the amount of exposure and damage, should a single key be compromised.
 - 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 is used for both protecting and processing.
 - Protection: Key is used to encrypt or to generate MAC and DigSig
 - Processing: Key is used to decrypt or to validate MAC and DigSig
- A cryptographic key **shall not** be used to provide protection **after** the end of the protection period.
- The processing period normally extends beyond the protection period.
- The crypto-period lasts from the beginning of the protection period to the end of the processing period.



Recommended Crypto Periods

Ref: NIST SP 800-57

Key Type	Cryptoperiod	
	Protection Period	Usage Period
1. Private Signature Key	1-3 years	
2. Public Signature Key	Several years (depends on key size)	
3. Symmetric Authentication Key	≤ 2 years	$\leq \text{OUP} + 3$ years
4. Private Authentication Key	1-2 years	
5. Public Authentication Key	1-2 years	
6. Symmetric Data Encryption Keys	≤ 2 years	$\leq \text{OUP} + 3$ years
7. Symmetric Key Wrapping Key	≤ 2 years	$\leq \text{OUP} + 3$ years

Recommended Crypto Periods (cont.)

Ref: NIST SP 800-57

Key Type	Cryptoperiod	
	Protection Period	Usage Period
8. Symmetric and asymmetric RNG Keys	Upon reseeding	
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	

Recommended Crypto Periods (cont.)

Ref: NIST SP 800-57

Key Type	Cryptoperiod	
	Protection Period	Usage Period
14. Public Static Key Agreement Key	1-2 years	
15. Private Ephemeral Key Agreement Key	One key agreement transaction	
16. Public Ephemeral Key Agreement Key	One key agreement transaction	
17. Symmetric Authorization (Access Approval) Key	<= 2 years	
18. Private Authorization (Access Approval) Key	<= 2 years	
19. Public Authorization (Access Approval) Key	<= 2 years	

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 in their BSAFE security products a flawed method for generating random numbers.
- 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 be 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 the protective mechanisms for the key fail, and the key can no longer be trusted
- When a key is compromised, all use of the key to **protect** information shall cease, and the compromised key shall be revoked.
 - However, the continued use of the key for **processing** under controlled circumstances may be warranted, depending on the risks, and on the organization's Key Management Policy.
- The continued use of a compromised key must be limited to processing protected information.
 - In this case, the entity that uses the information must be made fully aware of the risks involved.

Key Compromise Recovery Plan

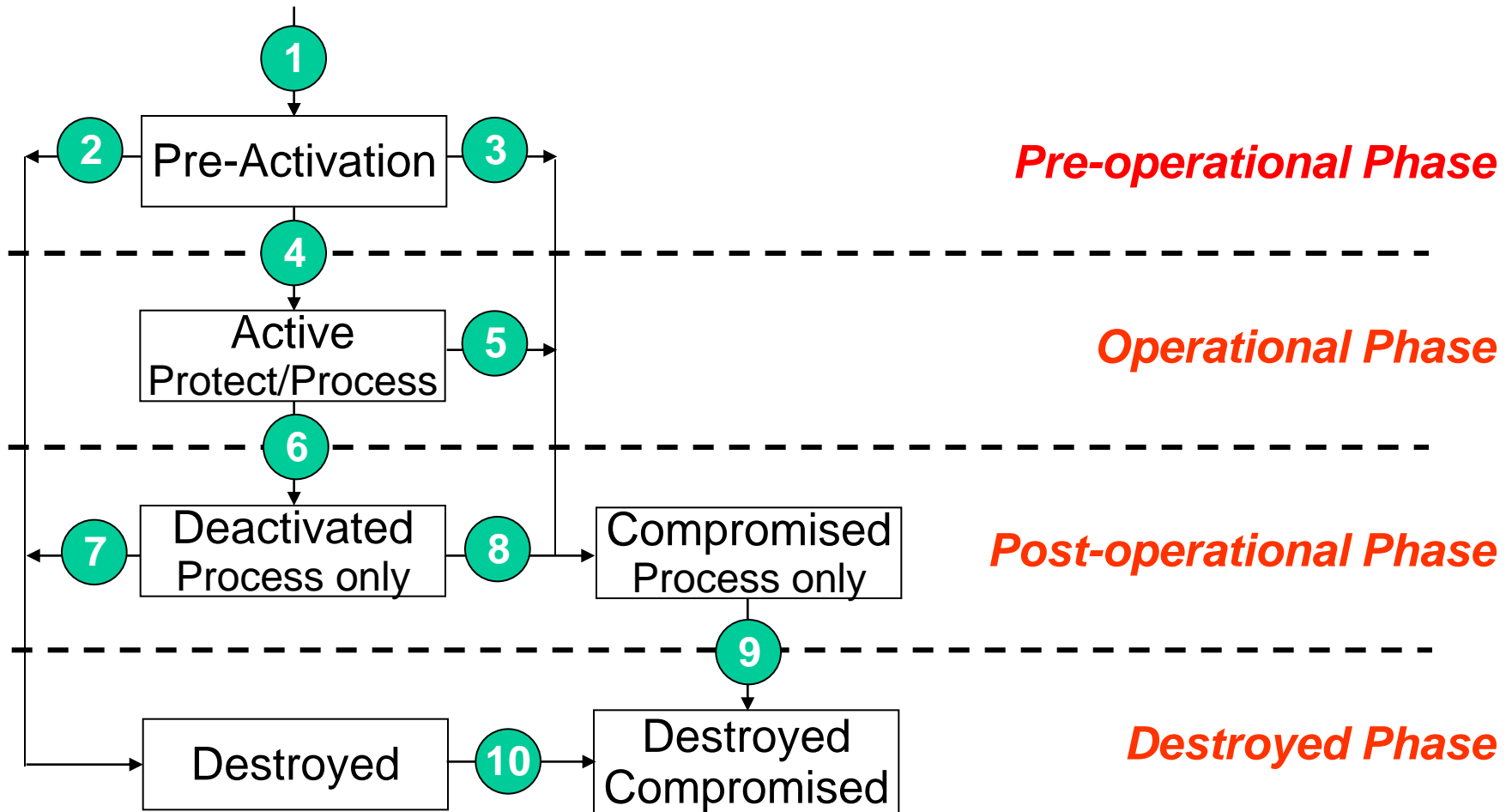
- A compromise recovery plan **should** contain:
 - The identification of the parties to notify.
 - The identification of the personnel to perform the recovery actions.
 - The re-key method.
 - Any other recovery procedures, such as:
 - Physical inspection of equipment.
 - Identification of all information 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 a 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.

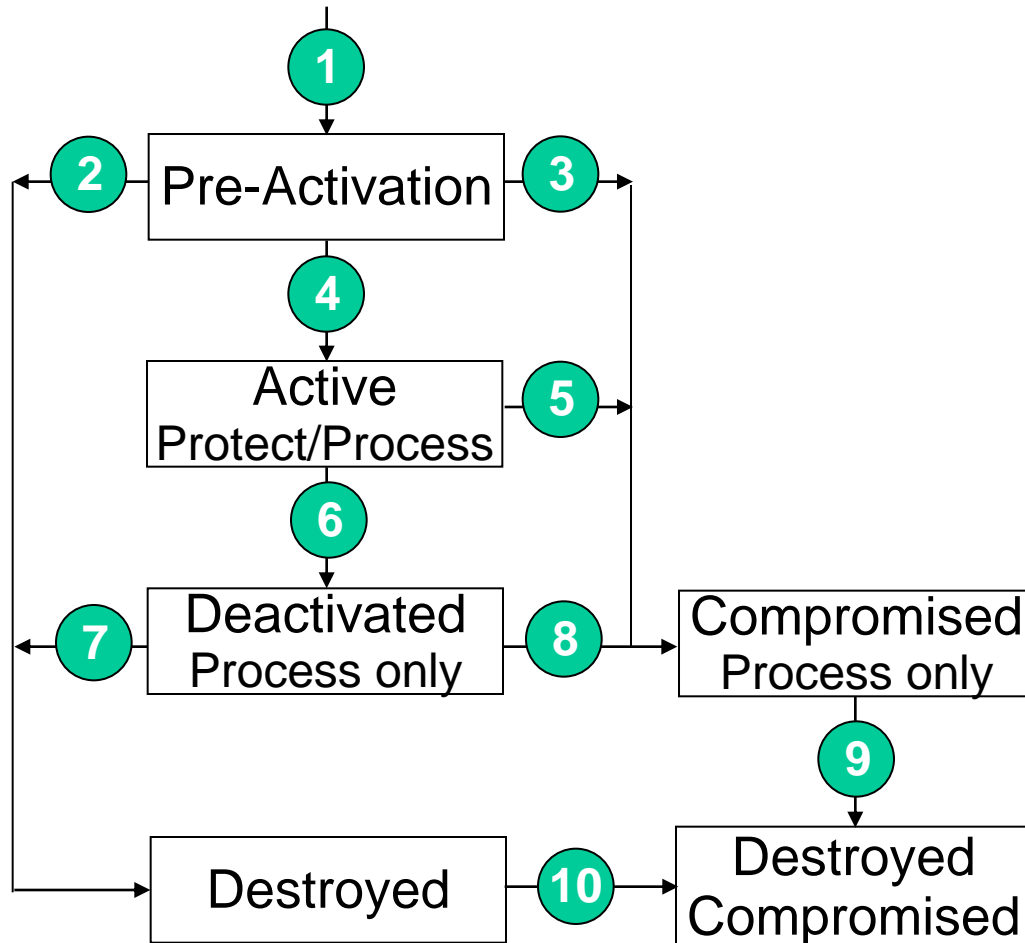
Key States, Transitions and Phases

Ref: NIST SP 800-57



Key States and Transitions

Ref: NIST SP 800-57



1) Pre-Activation

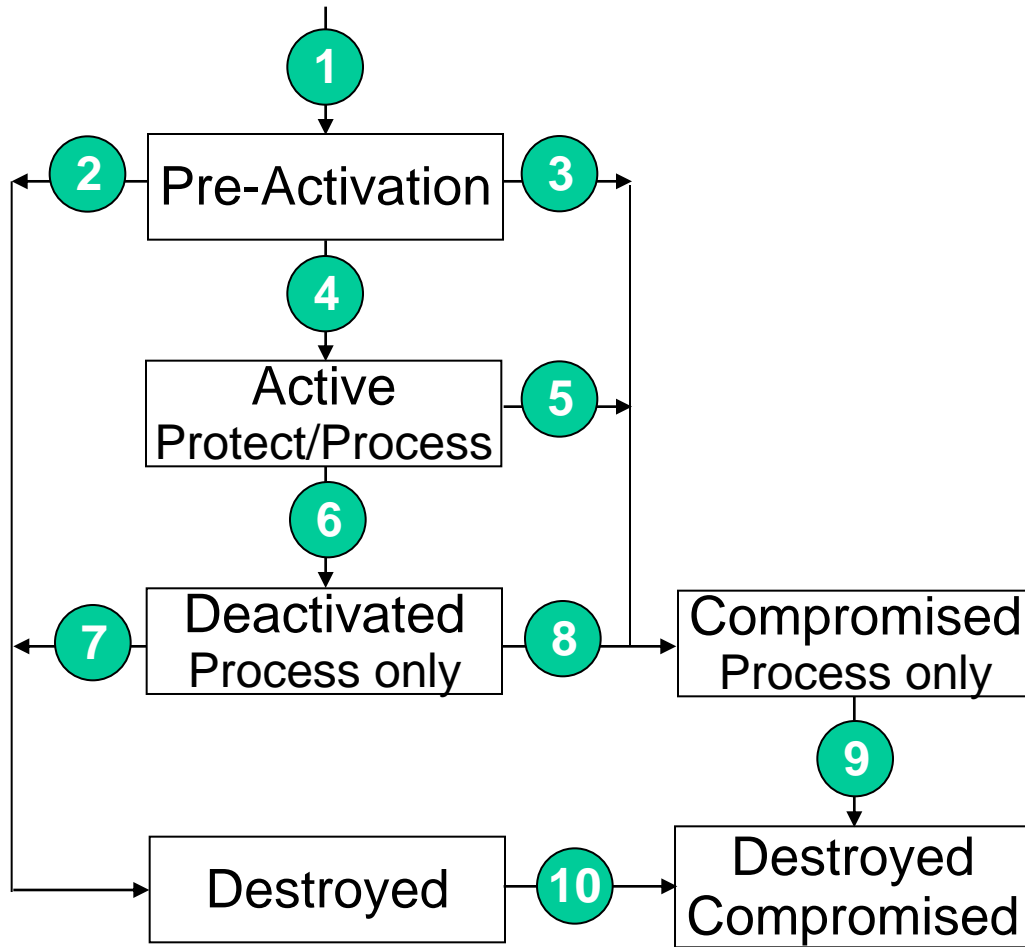
- The key material has been generated but is not yet authorized for use

4) Active

- The key may be used to cryptographically protect information or cryptographically process previously protected information, or both. When a key is active, it may be designated to protect only, process only, or both.

Key States and Transitions (cont.)

Ref: NIST SP 800-57



6) Deactivated

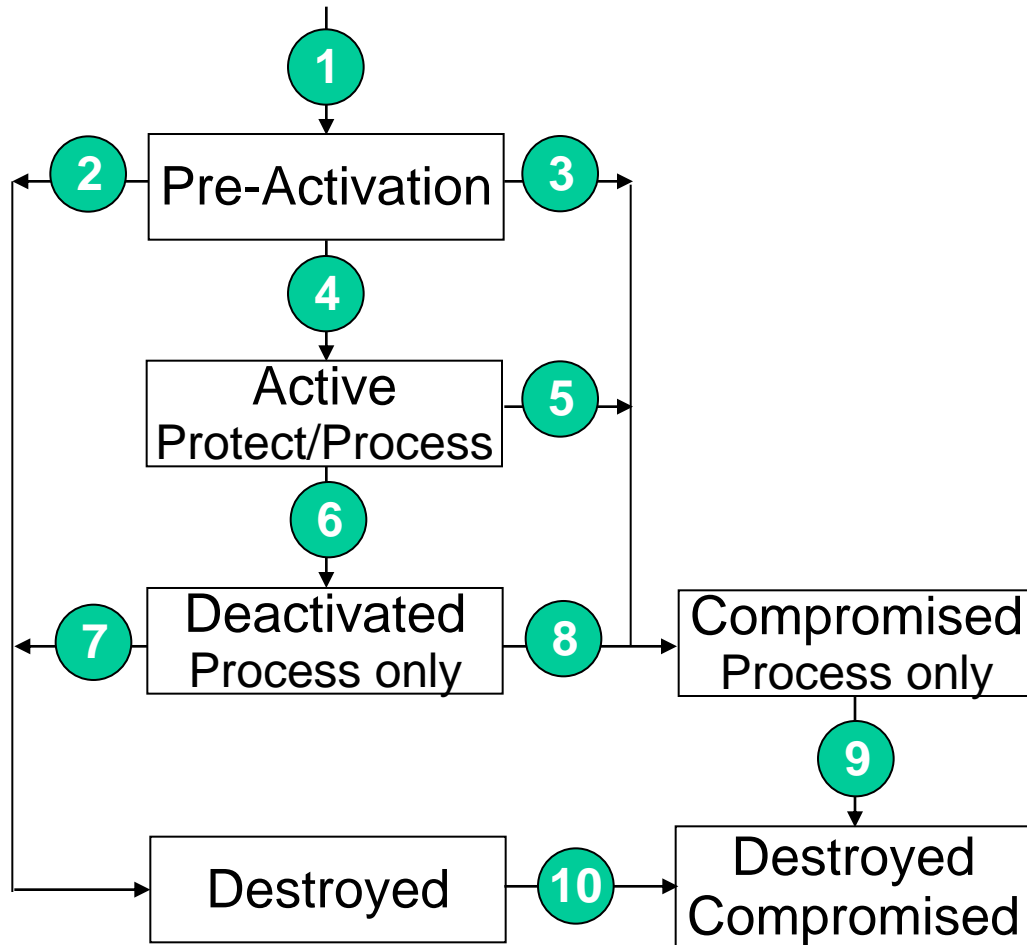
- A key whose cryptoperiod has expired but is still needed to perform cryptographic processing, gets deactivated until it is destroyed.

2), 7) Destroyed

- The key has been destroyed. Even though the key no longer exists in this state, certain key attributes (e.g. key name, type and cryptoperiod) may be retained.

Key States and Transitions (cont.)

Ref: NIST SP 800-57

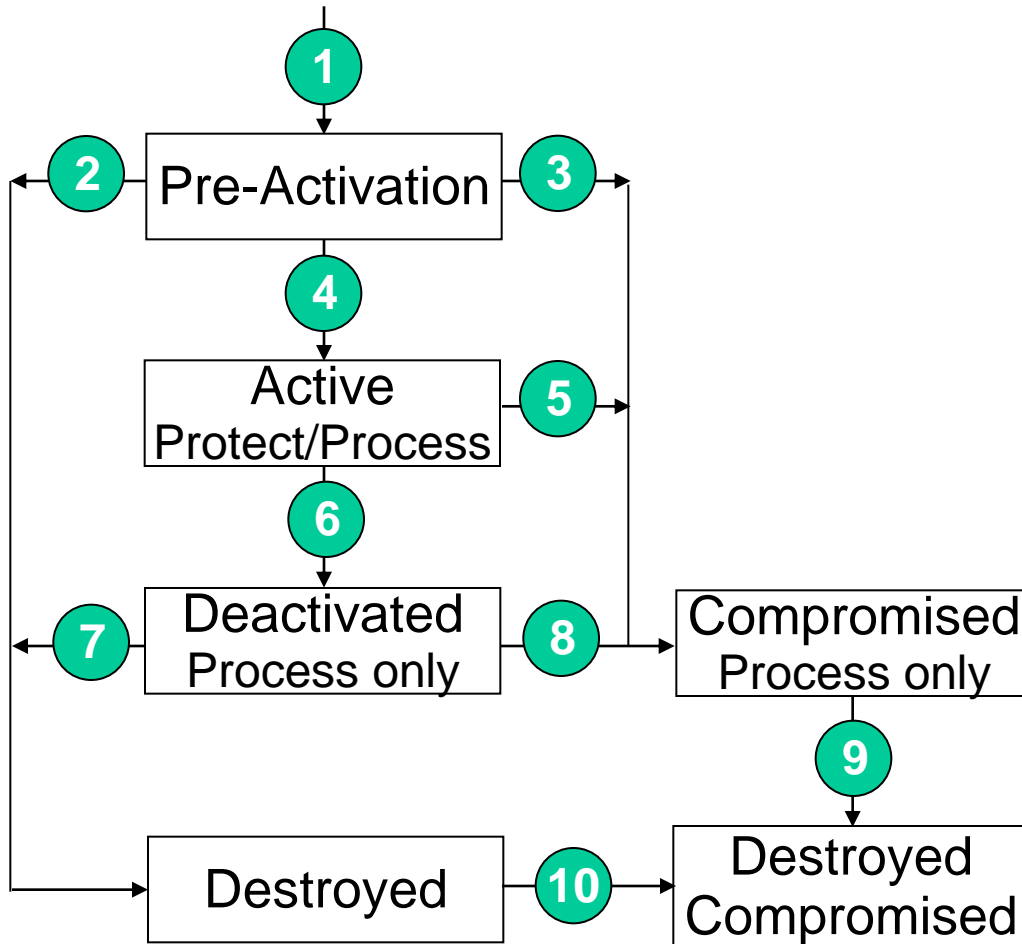


3), 5), 8) Compromised

- Generally, keys are compromised when they are released to or determined by an unauthorized entity. If the integrity or secrecy of the key is suspect, it is revoked. The key is not used to apply protection to information. In some cases, the key may be used for processing.

Key States and Transitions (cont.)

Ref: NIST SP 800-57



9), 10) Destroyed Compromised

- The key is destroyed after a compromise, or the key is destroyed and a compromise is later discovered. Key attributes may be retained.

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
 - Where will they be kept?
 - How will they be kept securely?
 - Who will know how to access them 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 ?

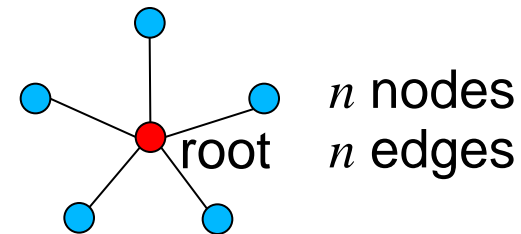
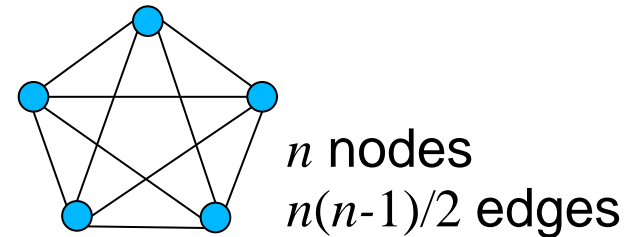
Cryptography solves security problems in open networks,
... but creates key management complexity.



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

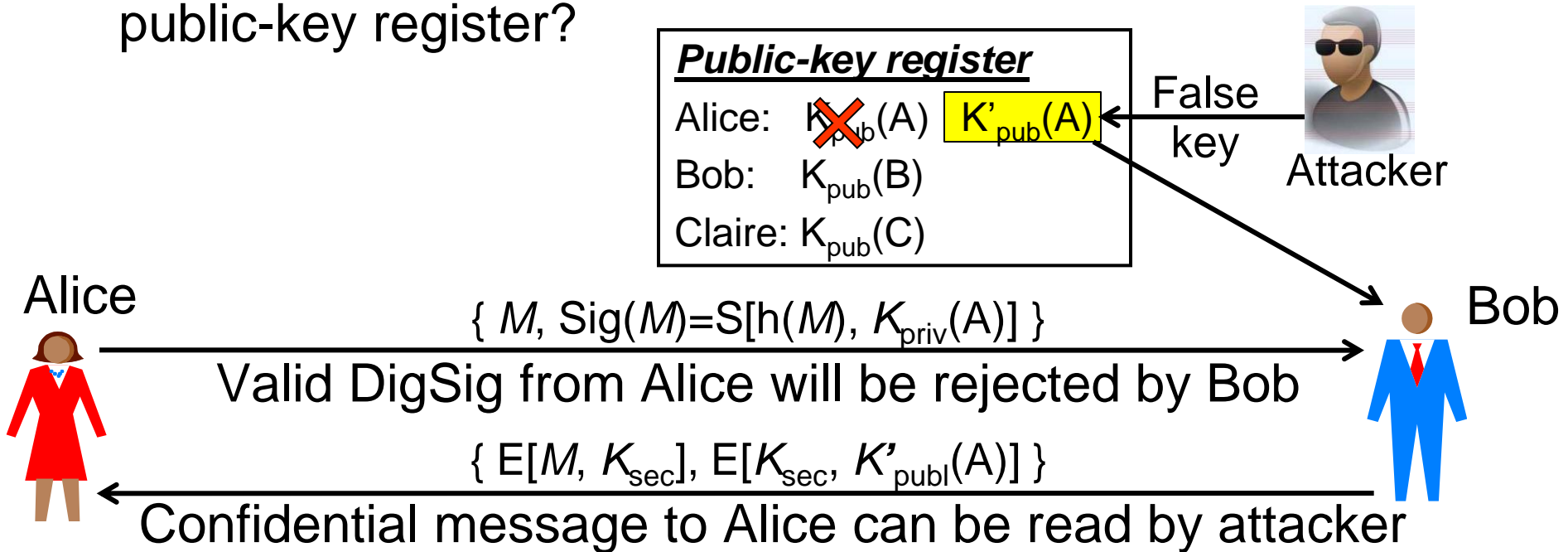
Key distribution: The problem

- Network with n nodes
- Every pair of nodes need to communicate securely under cryptographic protection
- How many secure key **distributions** needed ?
 - Symmetric secret keys: $n(n-1)/2$ distributions
 - Secrecy needed, quadratic growth
 - Impractical in open networks
 - Asymmetric public keys: $n(n-1)/2$ distributions
 - authenticity needed, quadratic growth
 - Impractical in open networks
 - Asymmetric public keys with PKI: 1 root public key distributed to n parties
 - authenticity needed, linear growth
 - ... more difficult than you might think



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)

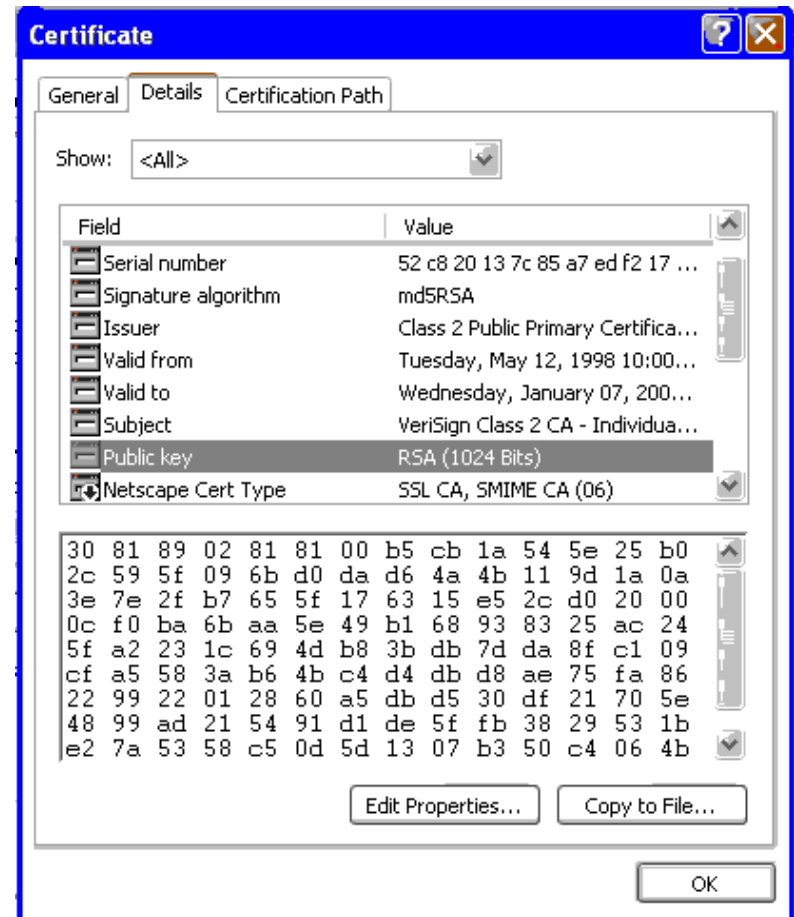
X.509 Digital Certificate

- Version
- Serial Number
- Algorithm Identifier
- CA Name
- CA Unique Identifier
- User Name
- **User Unique Name**
- **User Public Key**
- Validity Period
- Extensions

CA Digital
Signature

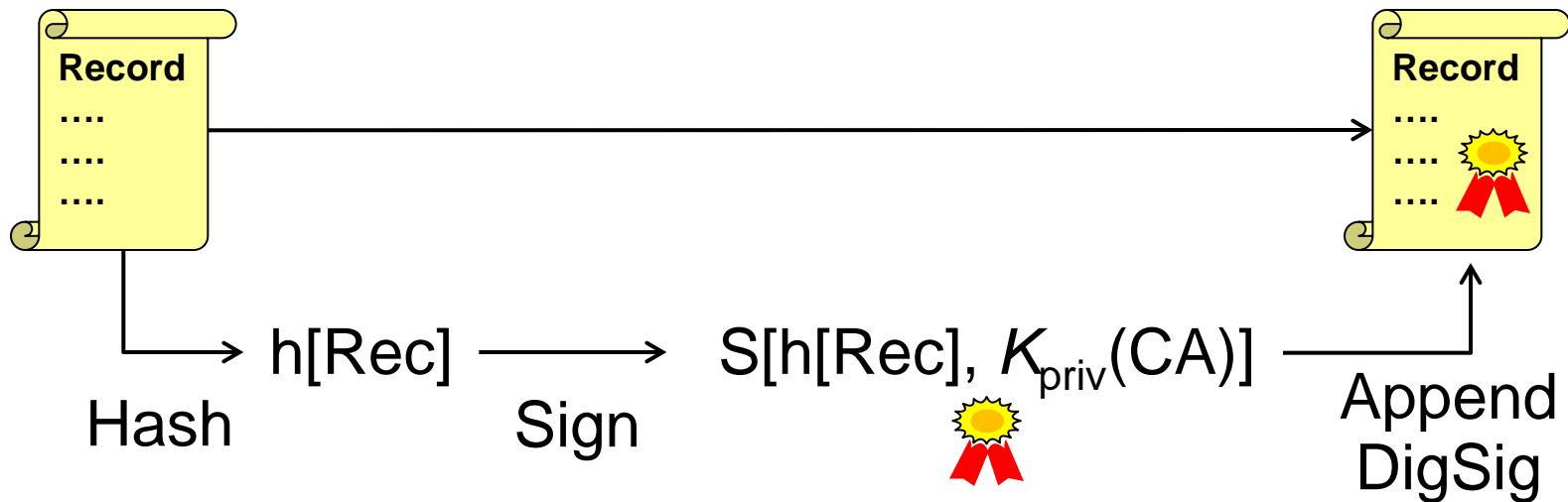


Example of X.509 certificate

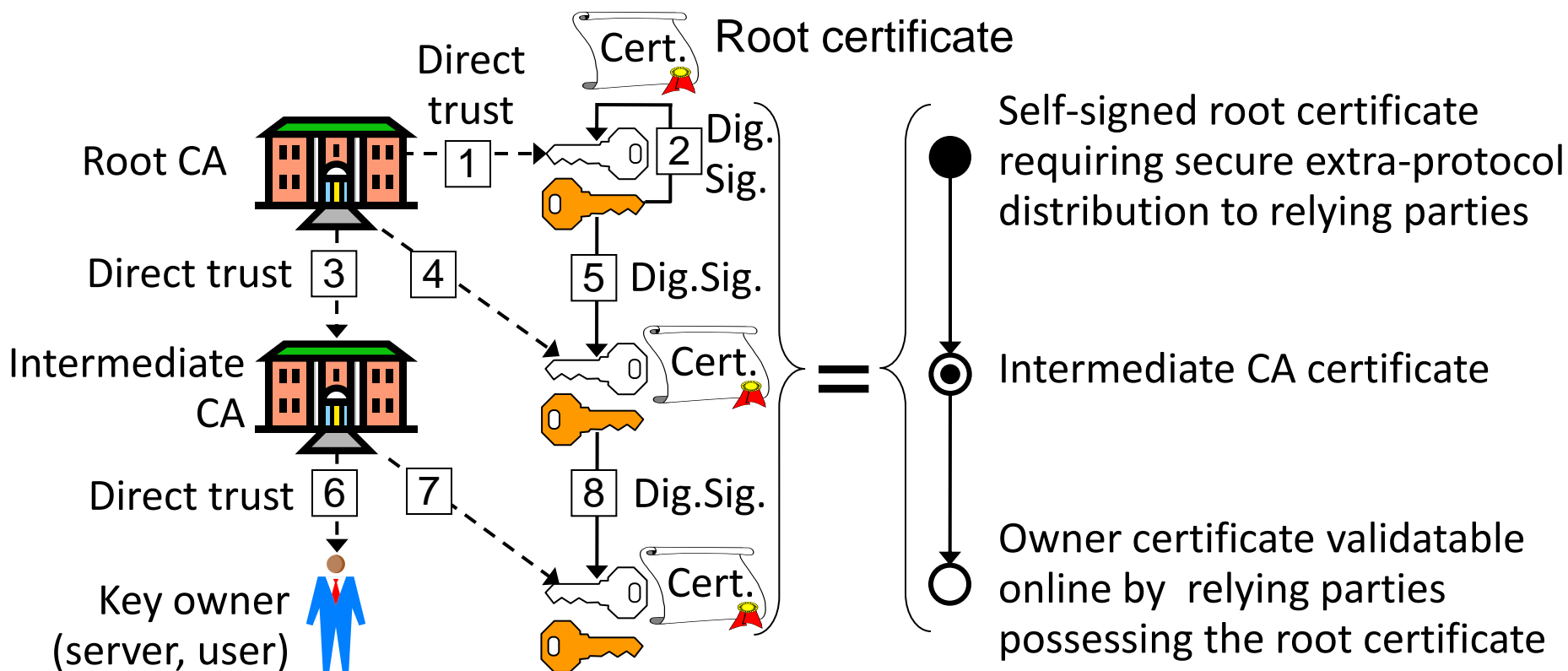


How to generate a digital certificate?

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



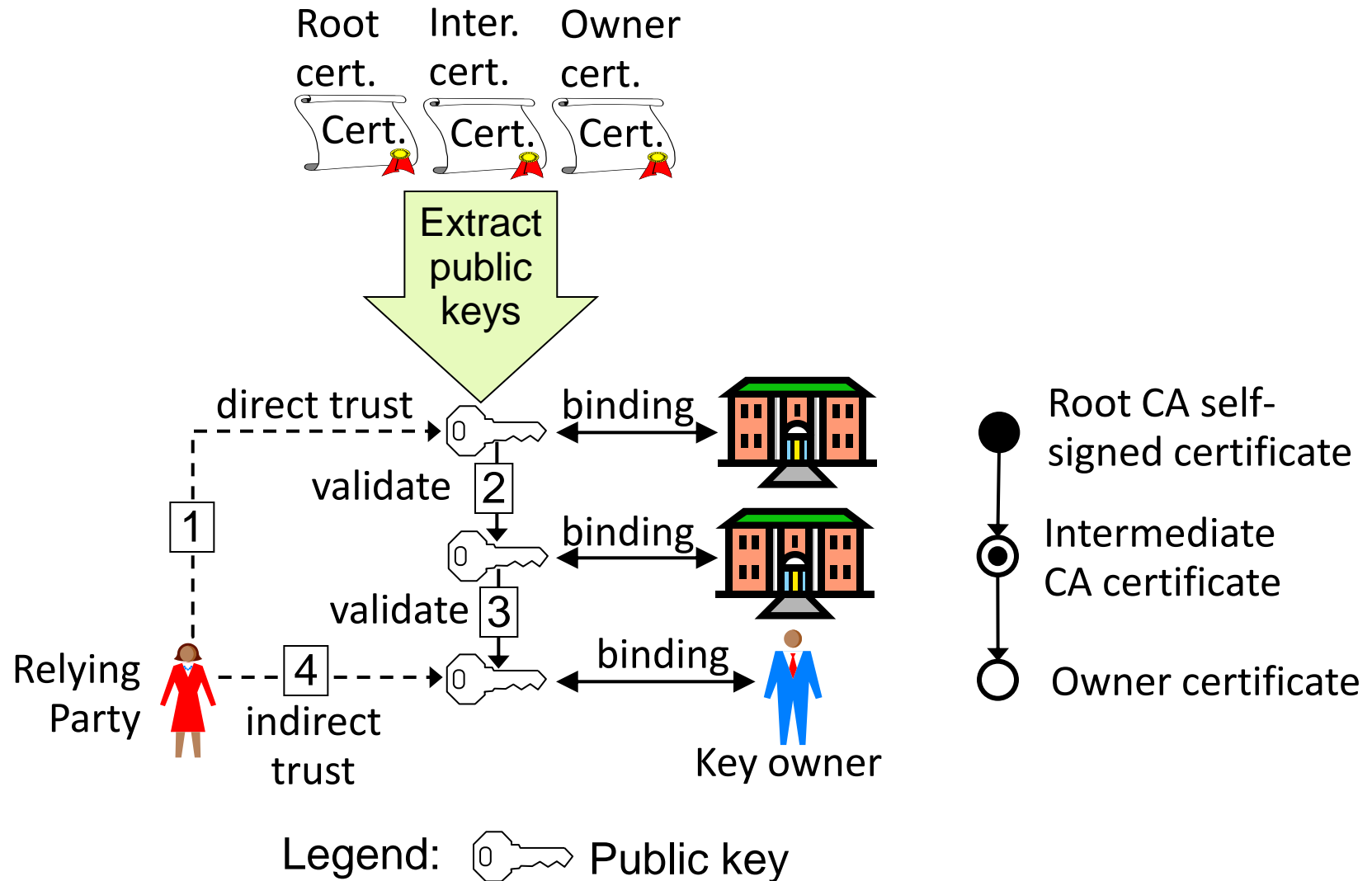
Legend:  Public key  Private key

Self-signed root keys: Why?

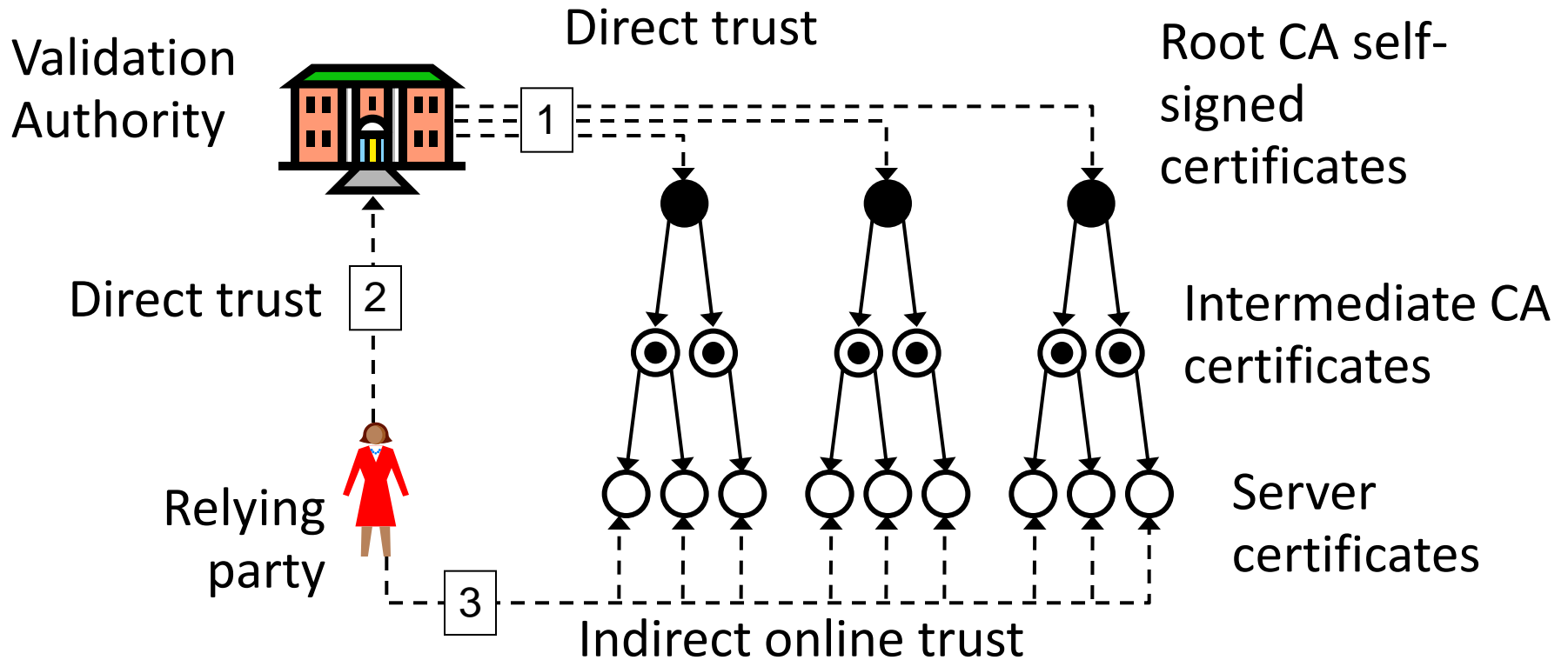
- Many people think a root public key is authentic just because it is self-signed
- Can be deceptive
 - Gives impression of assurance
 - Disguises insecure practice
 - Gives false trust
- Self-signing provides absolutely no security
- Useful purpose of self-signing
 - X.509 certificates have a field 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

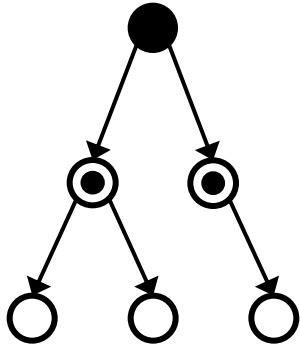


Validation Authorities

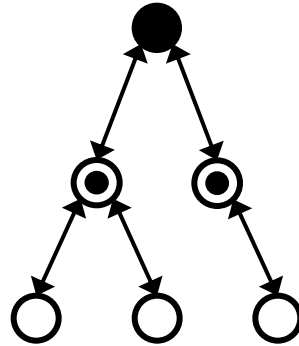


- A validation authority can assist relying parties to validate certificates

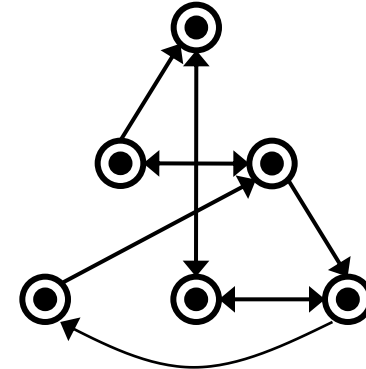
PKI Trust Models



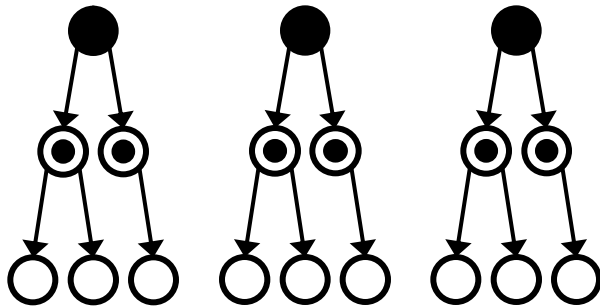
Strict hierarchy
e.g. `DNSSEC PKI`



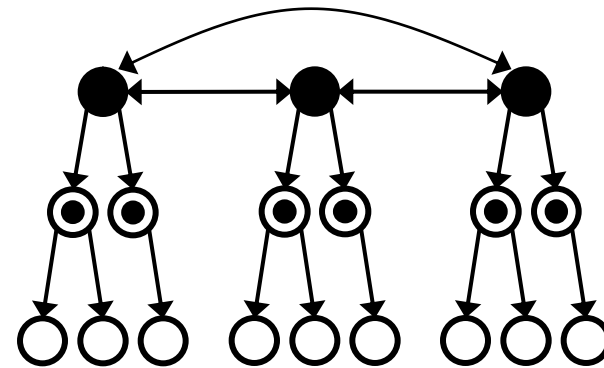
Bi-directional
hierarchy



Ad-hoc anarchic PKI



Isolated strict hierarchies
e.g. `Browser PKIX`



Cross-certified strict hierarchies

PKI trust models

Strict hierarchical model

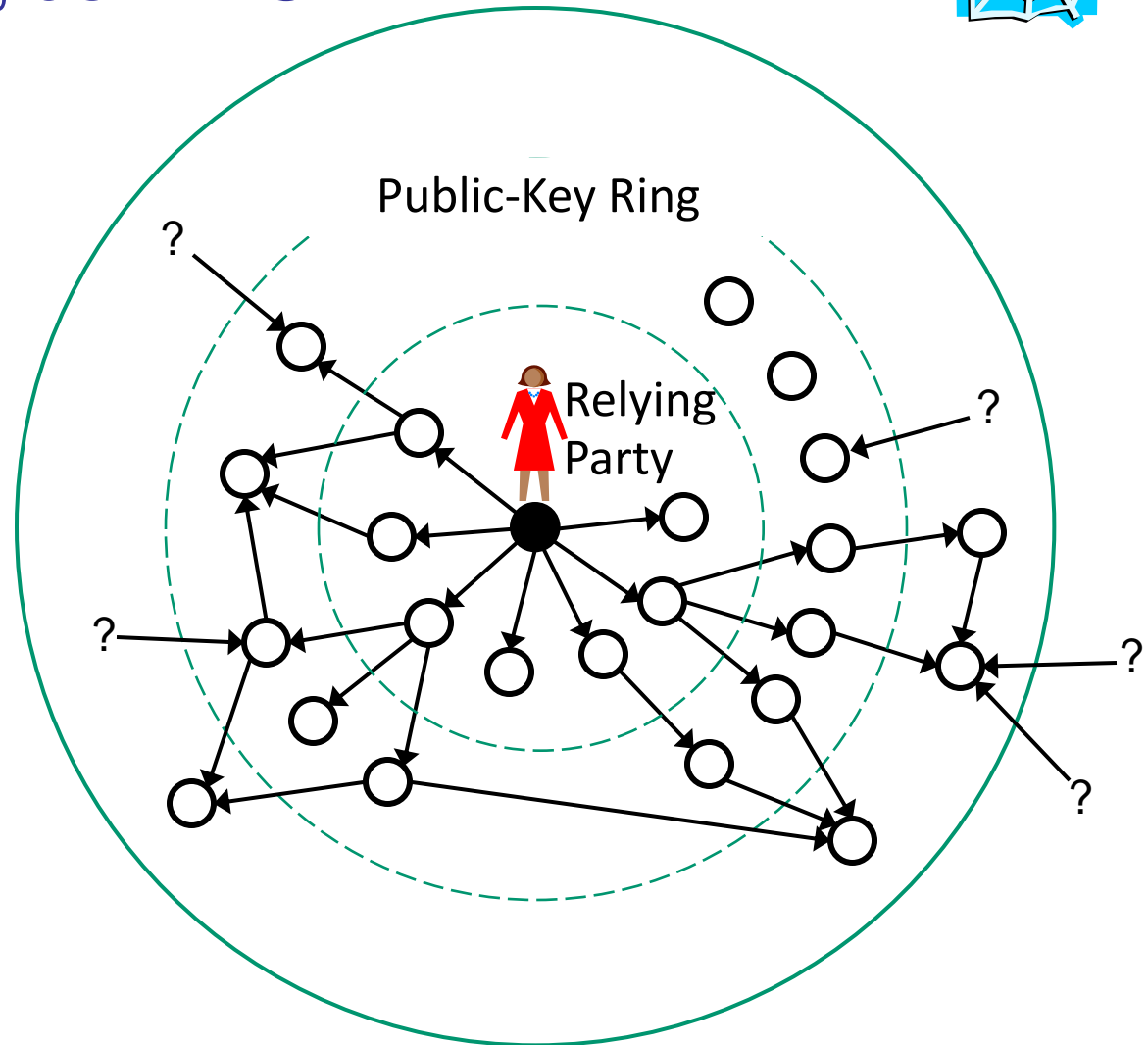
- Advantages:
 - works well in highly-structured setting such as military and government
 - unique certification path between two entities (so finding certification paths is trivial)
 - scales well to larger systems
- Disadvantages:
 - need a trusted third party (root CA)
 - ‘single point-of-failure’ 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/>
- PGP or GPG – What is the difference?

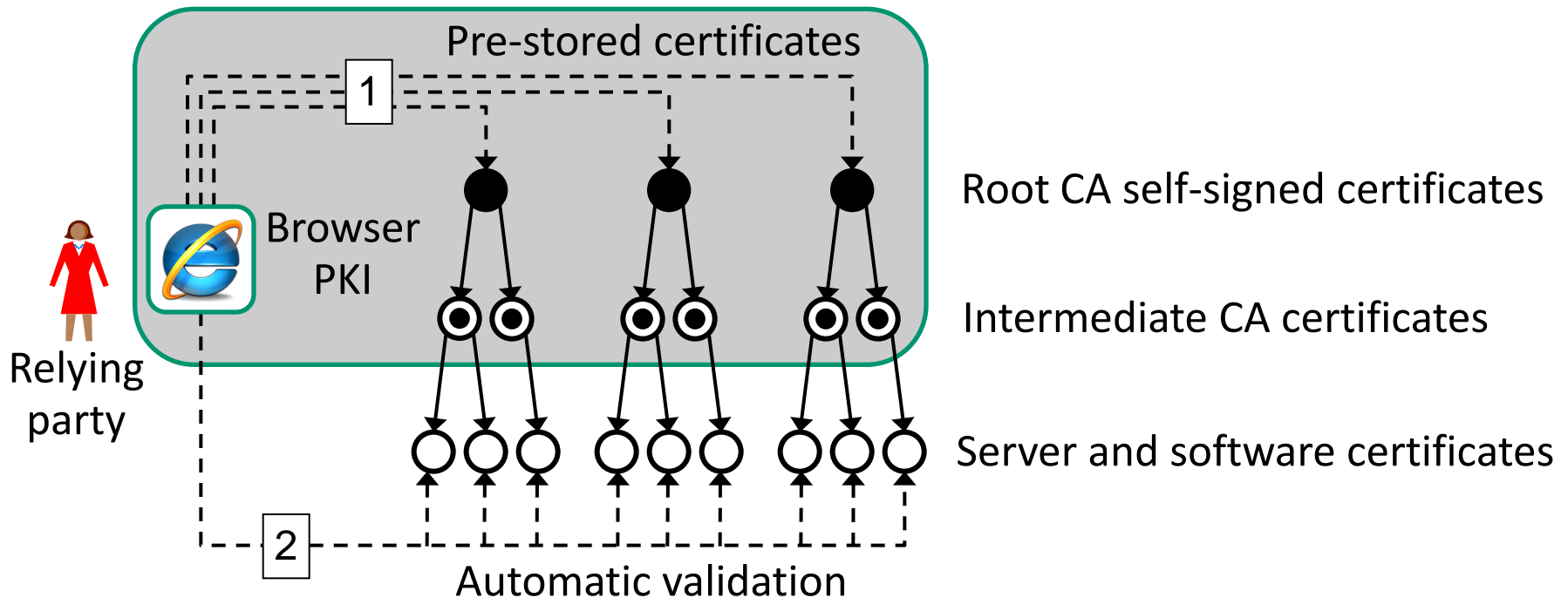
PKI trust models

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 roots operation
- Disadvantages:
 - More effort, and relies on human judgment
 - Works well with technical users who are aware of the issues, but not the general public
 - Not appropriate for more trust-sensitive areas such as finance and government

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



The browser PKIX 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

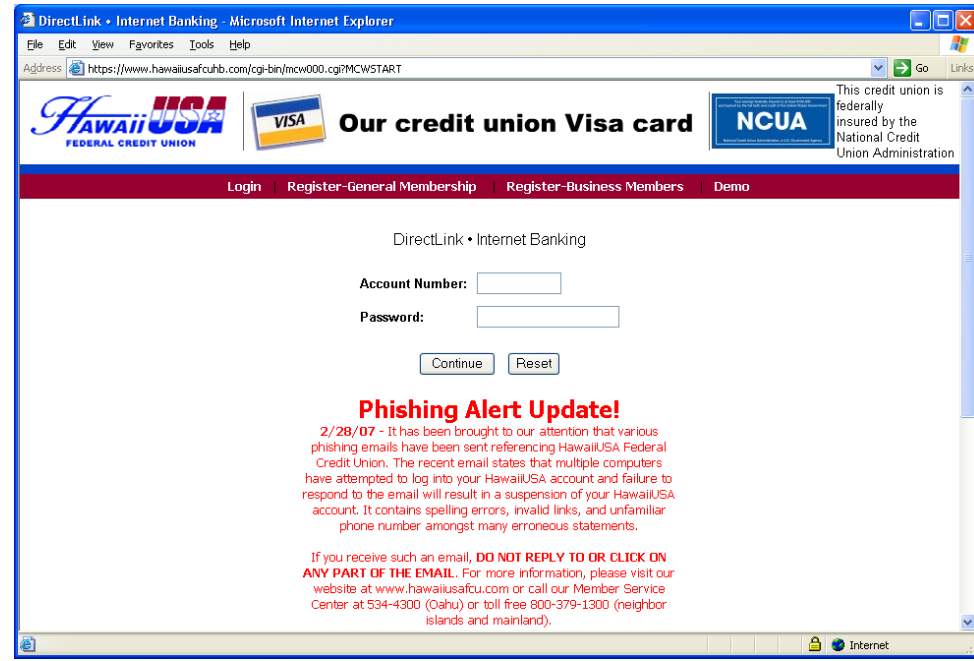
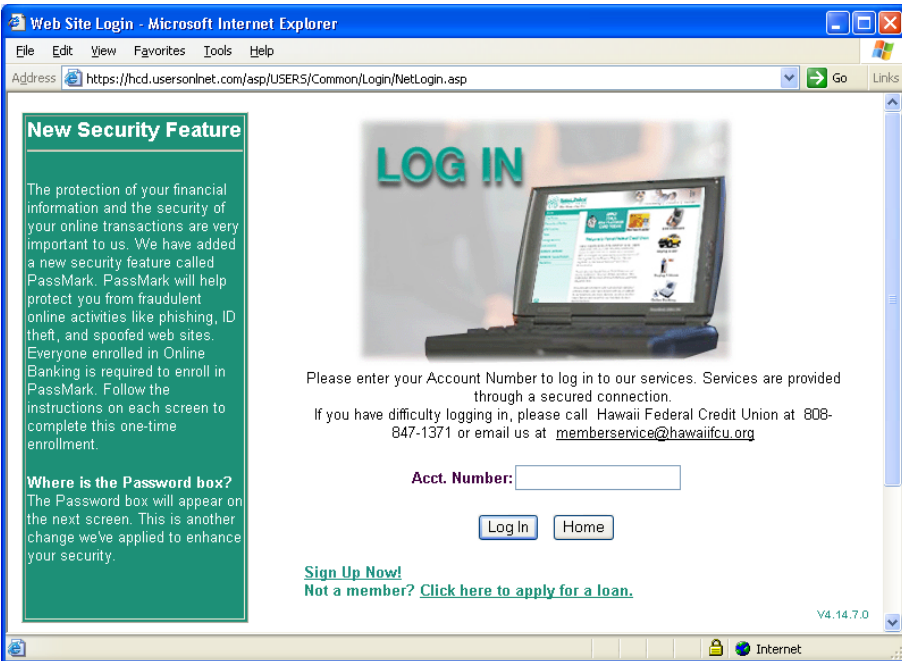
Browser PKIX and malicious certificates

- 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



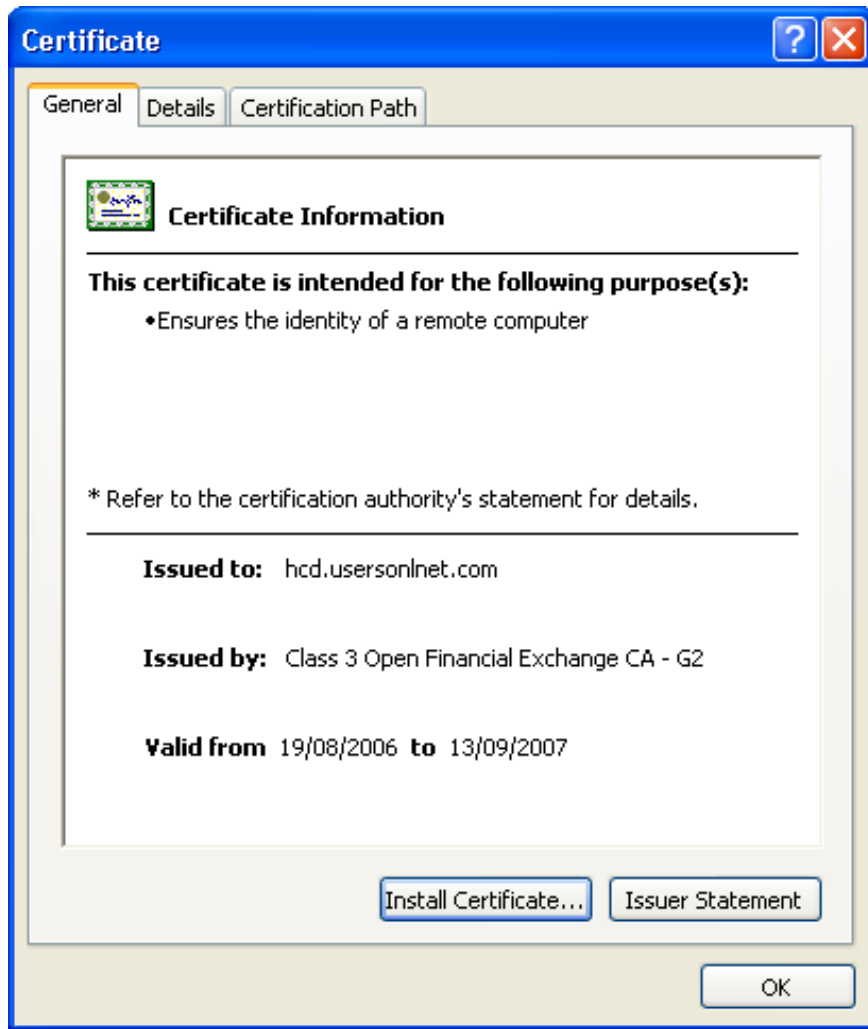
Authentic bank login

<https://hcd.usersonline.com/asp/USERS/Common/Login/NetLogin.asp>

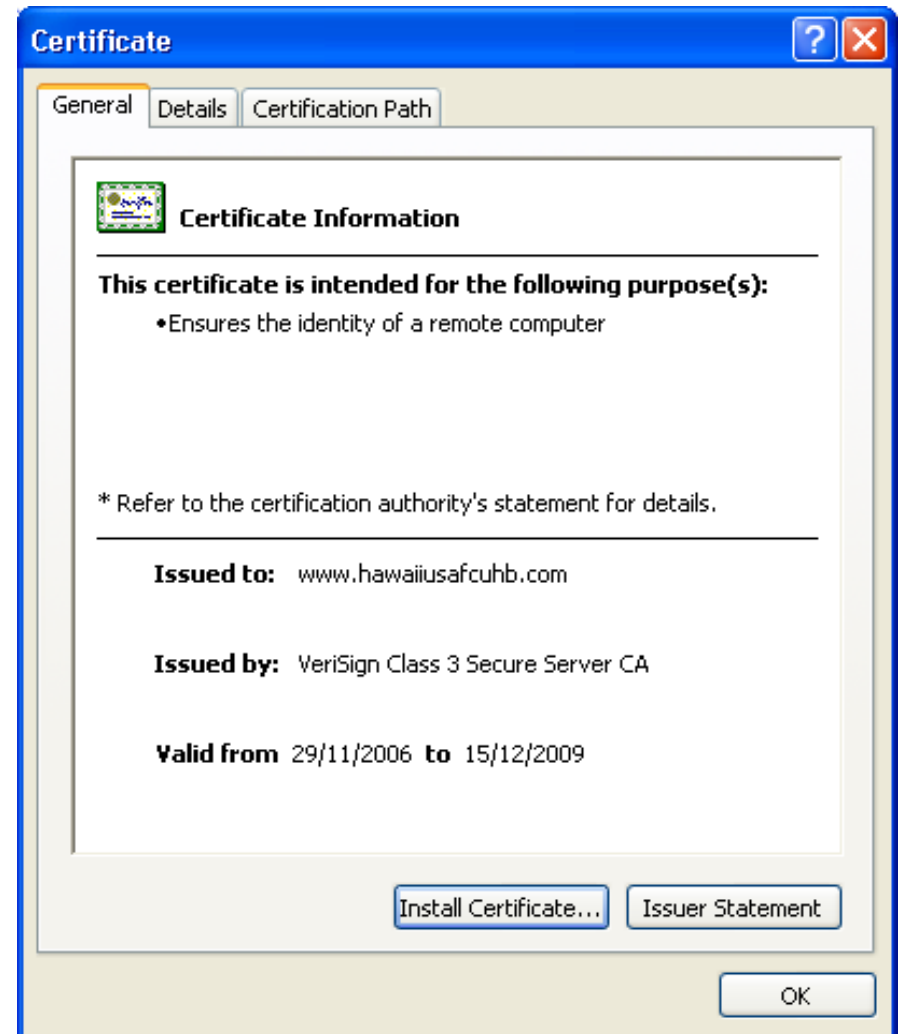
Fake bank login

<https://hawaiiusafcuhb.com/cgi-bin/mcw000.cgi?MCWSTART>

Authentic and Fake Certificates

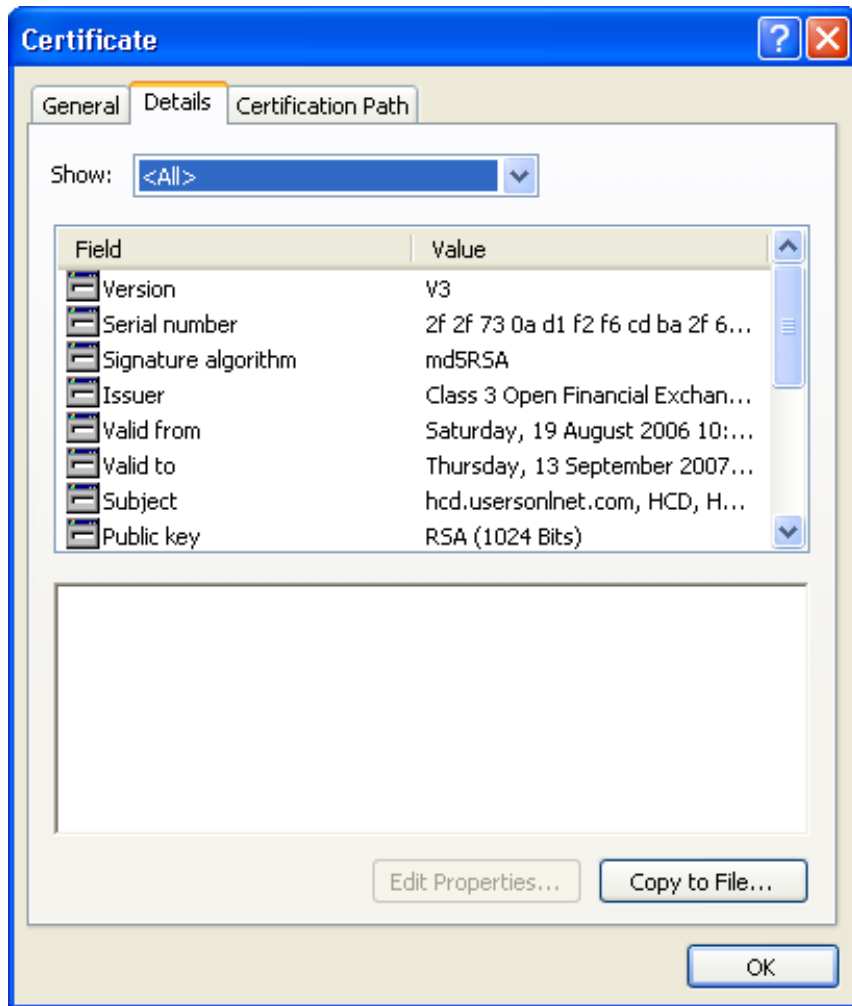


Authentic certificate

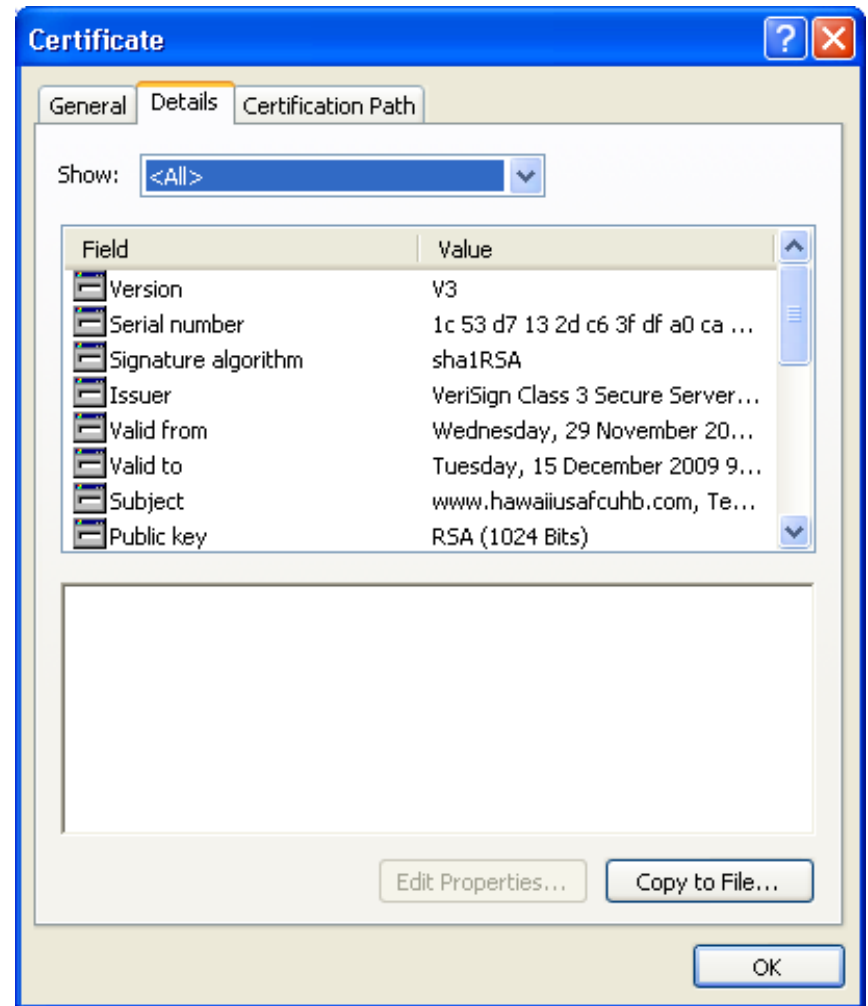


Fake certificate

Certificate comparison 2

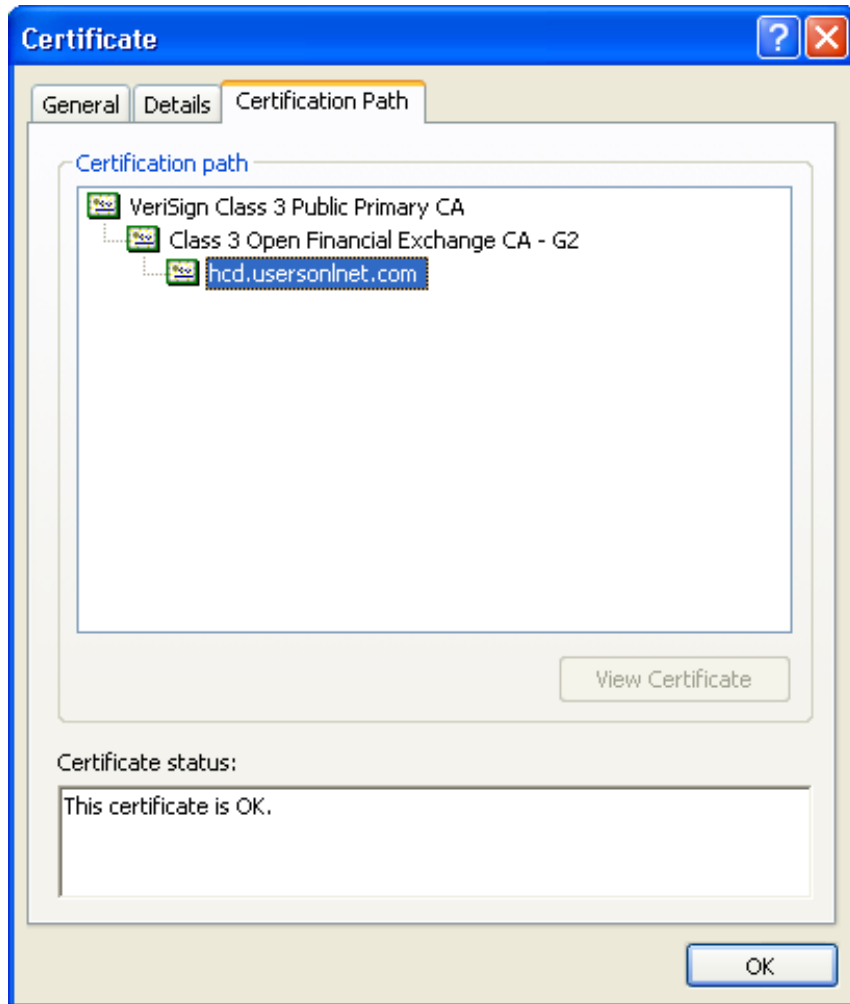


Genuine certificate

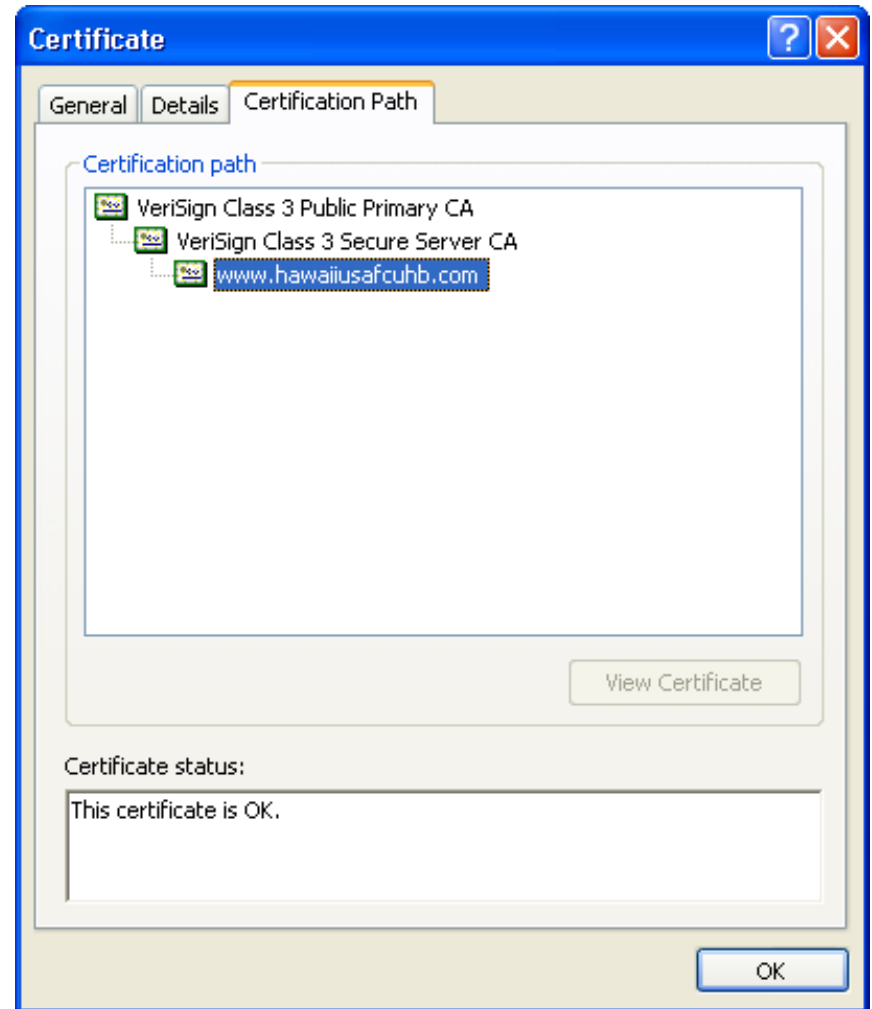


Fake certificate

Certificate comparison 3



Genuine certificate



Fake certificate

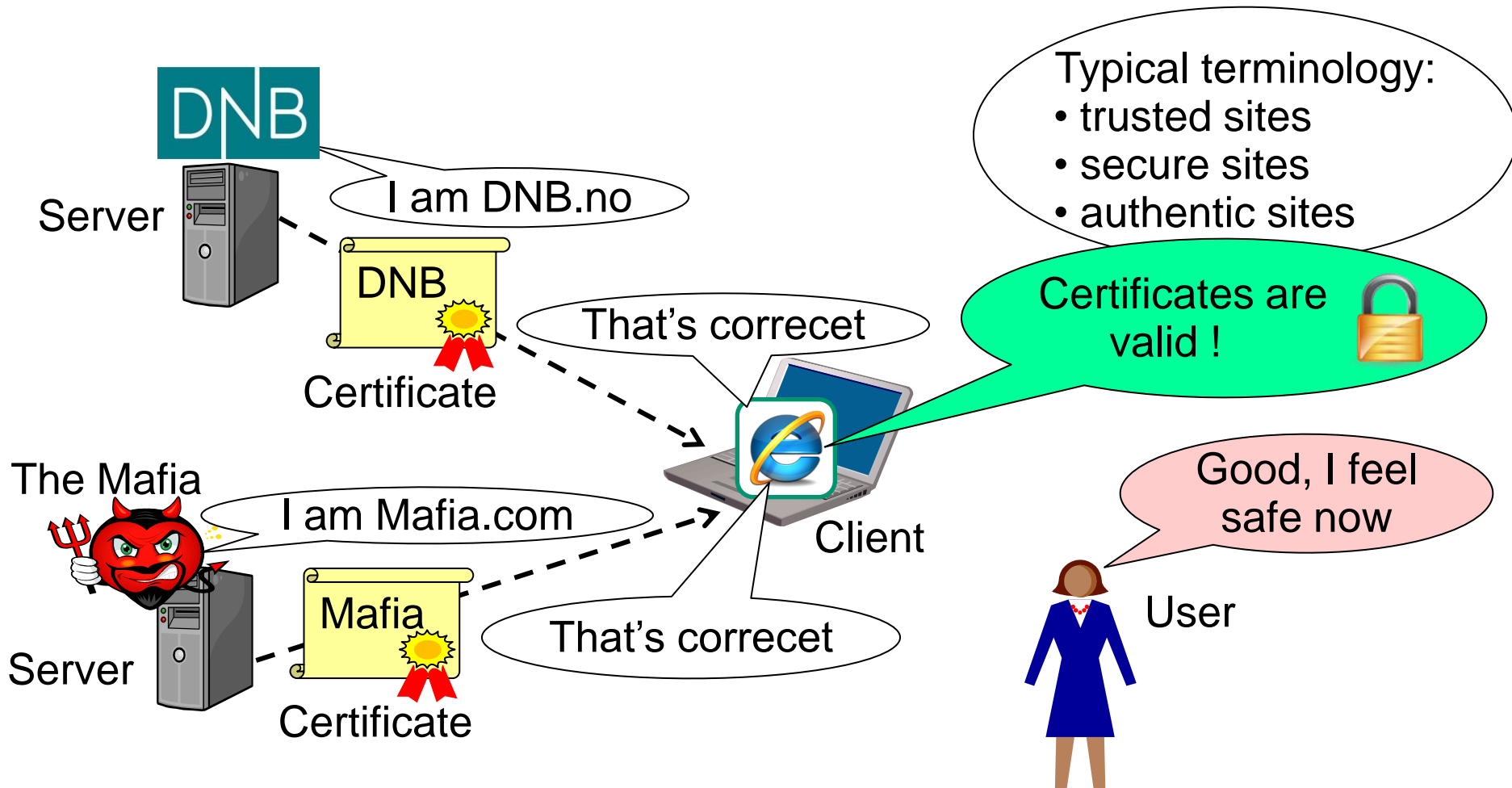
Public-key certificate meaning

- Public-key certificates are only about identity, not about honesty, reliability or anything you normally associate with trust
- Public-key certificates are not even suitable to verify identity.

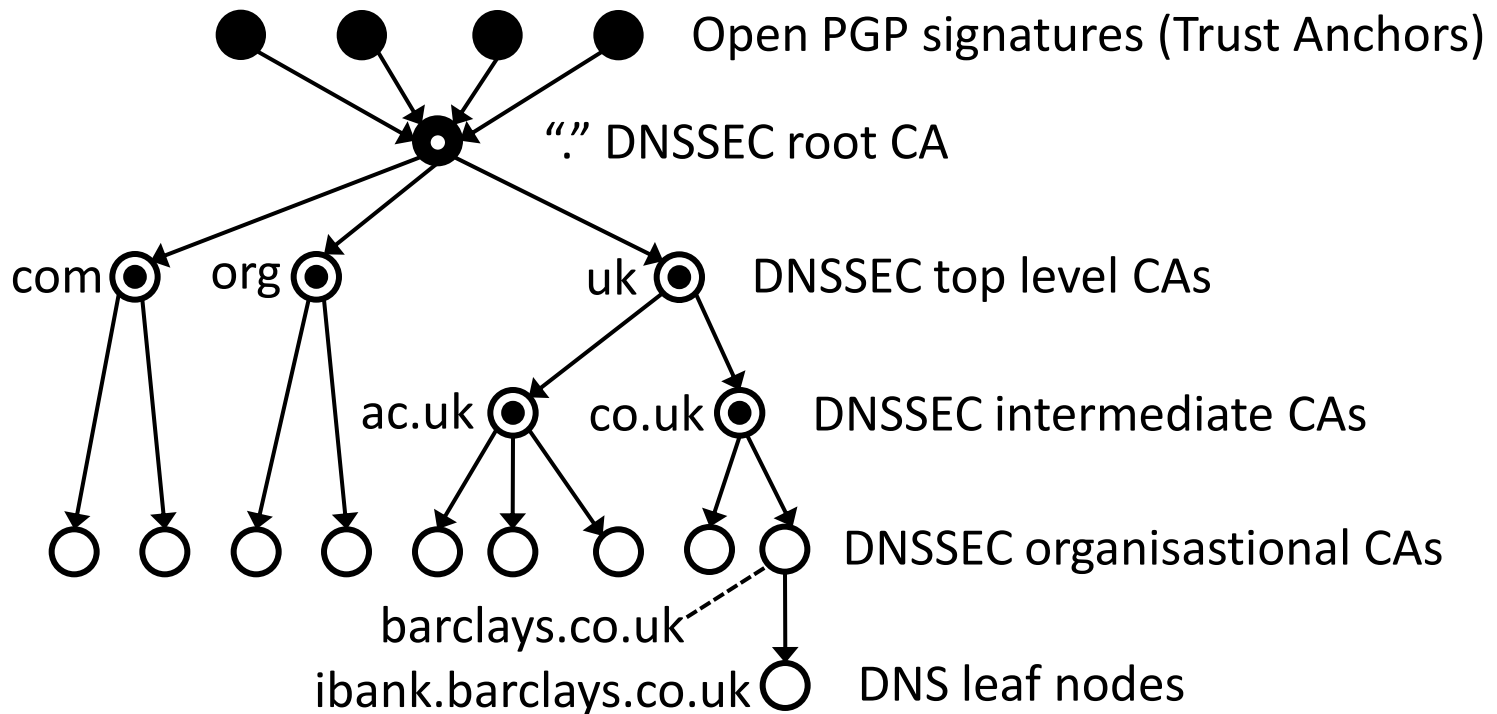
Stuxnet with valid signature

- 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 certificate
- Stuxnet worm would be automatically validated by every browser in the whole world
- Point about SW certificate is meaningless
- Anybody can buy software certificates and sign whatever they want, even the Mafia !!!

Meaningless PKIX Server Authentication

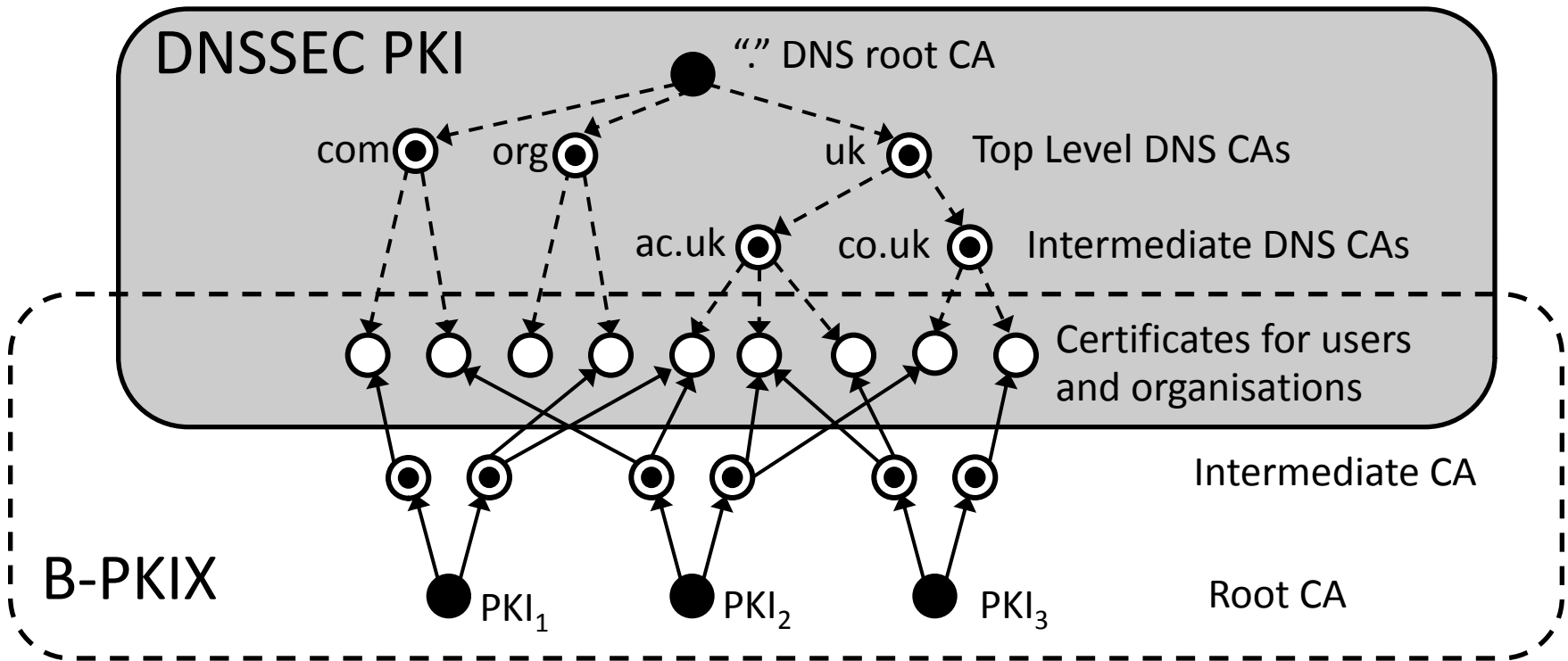


DNSSEC PKI



- 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 PKIX



- In B-PKIX, 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-PKIX, 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 disclosed
 - 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 PKIX is the most widely deployed PKI thanks to piggy-backing on browsers and the lax security requirements
- DNSSEC PKI might replace the browser PKIX

PKI Summary

- Public key cryptography needs a PKI to work
 - 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