INF3510 Information Security
University of Oslo
Spring 2011

Lecture 6

Key Management and PKI

Audun Jøsang

Outline

- Key management
- Key establishment
- Public key infrastructure
- Digital certificates
- PKI trust models
- PKI components

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 compromise 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 processes may weaken the security of one or both of the processes.
- 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?
- They’re classified according to:
  - Whether they’re public, private or symmetric
  - Their use
  - For asymmetric keys, also whether they’re static or ephemeral
- NIST Special Publication 800-57, Recommendation for Key Management – Part 1: General, August 2005, defines **19** types of cryptographic keys.

Key types

1. Private signature key
2. Public signature verification key
3. Symmetric authentication key
4. Private authentication key
5. Public authentication key
6. Symmetric data encryption key
7. Symmetric key wrapping key
8. Symmetric and asymmetric RNG seeds/keys

   (RNG: Random Number Generation)
9. Symmetric master key

Key types (continued)

10. Private key transport key
11. Public key transport key
12. Symmetric key agreement key
13. Private static key agreement key
14. Public static key agreement key
15. Private ephemeral key agreement key
16. Public ephemeral key agreement key
17. Symmetric authorization key
18. Private authorization key
19. Public authorization key

Random Number Generator Seeds

- RNG keys are used to initialise the generation of random symmetric/asymmetric keys
- Knowing the seed may determine the key uniquely
- Requires confidentiality and integrity protection
  - Period of protection for seeds, e.g.:
    a. Used once and destroyed
    b. From generation until no longer needed for subsequent reseeding
    c. Shall be destroyed at the end of the protection period

Relevant Key Parameters

- Domain parameters
- Initialization vectors
- Shared secrets
- RNG seeds
- Other public information; e.g. nonce
- Intermediate results
- Key control information; e.g. identifier, purpose, counter
- Random numbers
- Passwords
- Audit information

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 if a single key is compromised.
  - Limits the use of a particular algorithm to its estimated effective lifetime.

Crypto Period (continued)

- The crypto period is also important because it:
  - Limits the time available for attempts to penetrate physical, procedural, and logical access mechanisms that protect a key from unauthorized disclosure.
  - Limits the period within which information may be compromised by inadvertent disclosure of keying material to unauthorized entities.
  - Limits the time available for computationally intensive cryptanalytic attacks. (in applications where long-term key protection is not required).

Factors Affecting Crypto Periods

- In general, as the sensitivity of the information or the criticality of the processes increase, the length of the cryptoperiods should **decrease** in order to limit the damage that might result from each compromise.
- Short cryptoperiods 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 cryptoperiod is therefore a trade-off

Factors Affecting Crypto Periods (cont.)

- **Communications versus Storage**
  - Cryptoperiods are generally made shorter for keys used for protection of communication, e.g. limited to a single session
  - Cryptoperiods are generally made longer for stored data because the overhead of re-encryption associated with changing keys may be burdensome.
- **Cost of Key Revocation and Replacement**
  - In some cases, the costs associated with changing keys are painfully high. (e.g. decryption and subsequent re-encryption of very large databases, and revocation and replacement of a very large number of keys)
  - In such cases, the expense of the security measures necessary to support longer cryptoperiods may be justified.

Key Usage Periods

- A key is used for both protecting and processing.
- The protection period is aka. the *originator usage period.*
- The processing period is aka. the *recipient usage period*.
- A symmetric 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 cryptoperiod of a symmetric key is the period 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	<= 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

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	

Keys and Keying Material Compromise

- 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 under controlled circumstances to remove or verify the protections may be warranted, depending on the risks of continued use and an organization's Key Management Policy.
- The continued use of a compromised key **shall** be limited to <u>processing</u> protected information.
  - In this case, the entity that uses the information **shall** be made fully aware of the dangers involved.

Key Compromise Recovery Plan

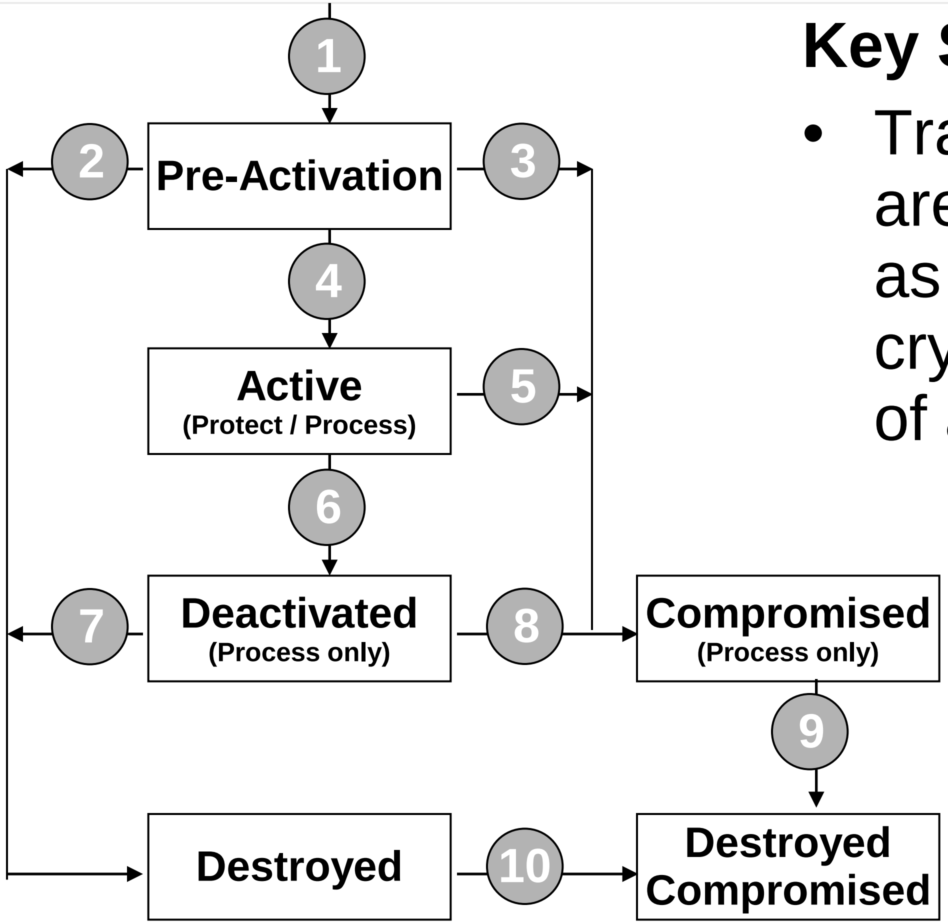
- A compromise recovery plan **should** contain:
  - The identification of the personnel to notify.
  - The identification of the personnel to perform the recovery actions.
  - The re-key method.
  - Any other recovery procedures, such as:
    - Physical inspection of equipment.
    - Identification 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 it is not detected.
- 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 could 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 to authenticate communicating entities that do not rely solely on the possession of keys.
  - Avoid building a system with catastrophic weaknesses.

Key States and Transitions

Ref: NIST SP 800-57

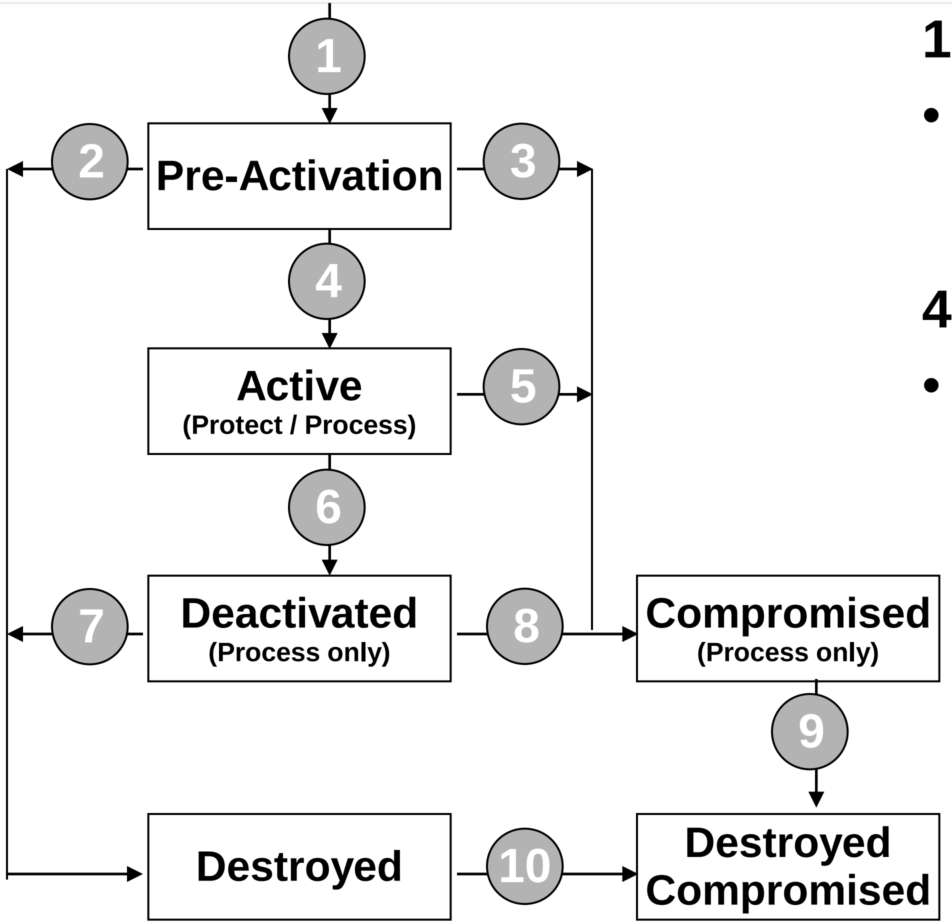


Key State Transitions

- Transitions between states are triggered by events, such as the expiration of a cryptoperiod or the detection of a compromise of a key.

Key States and Transitions (cont.)

Ref: NIST SP 800-57



1) Pre-Activation

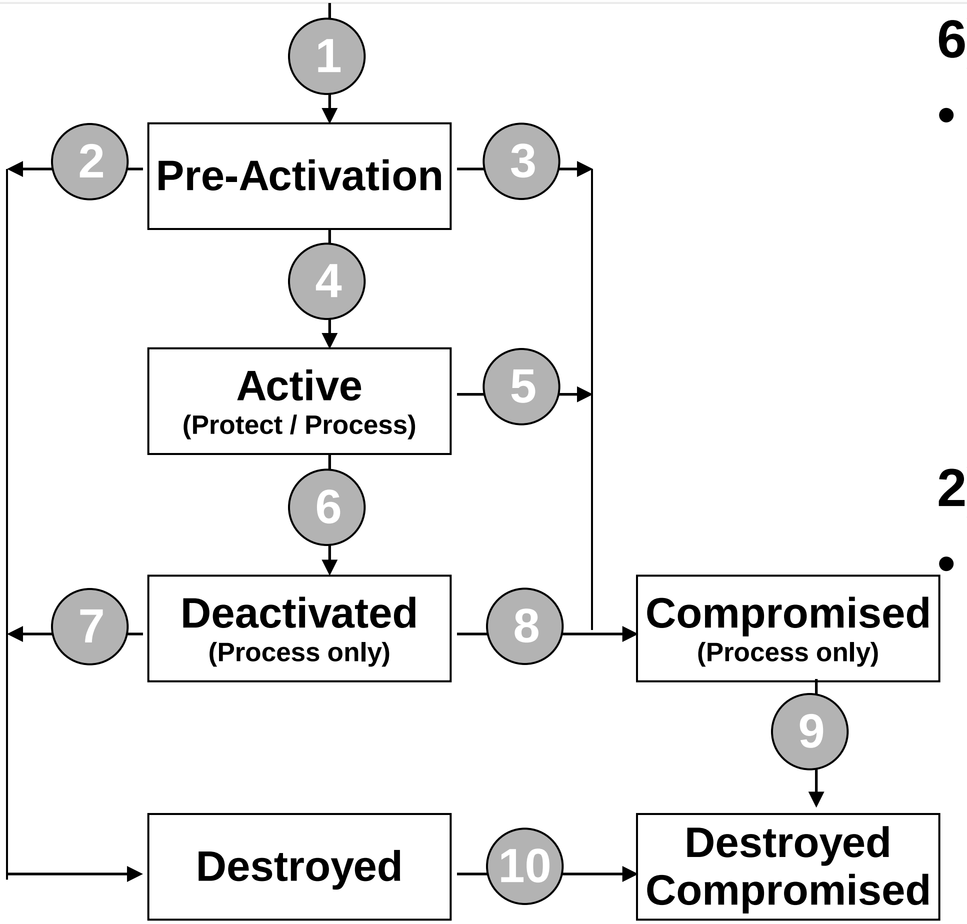
- The key material has been generated but is not yet authorized for use

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



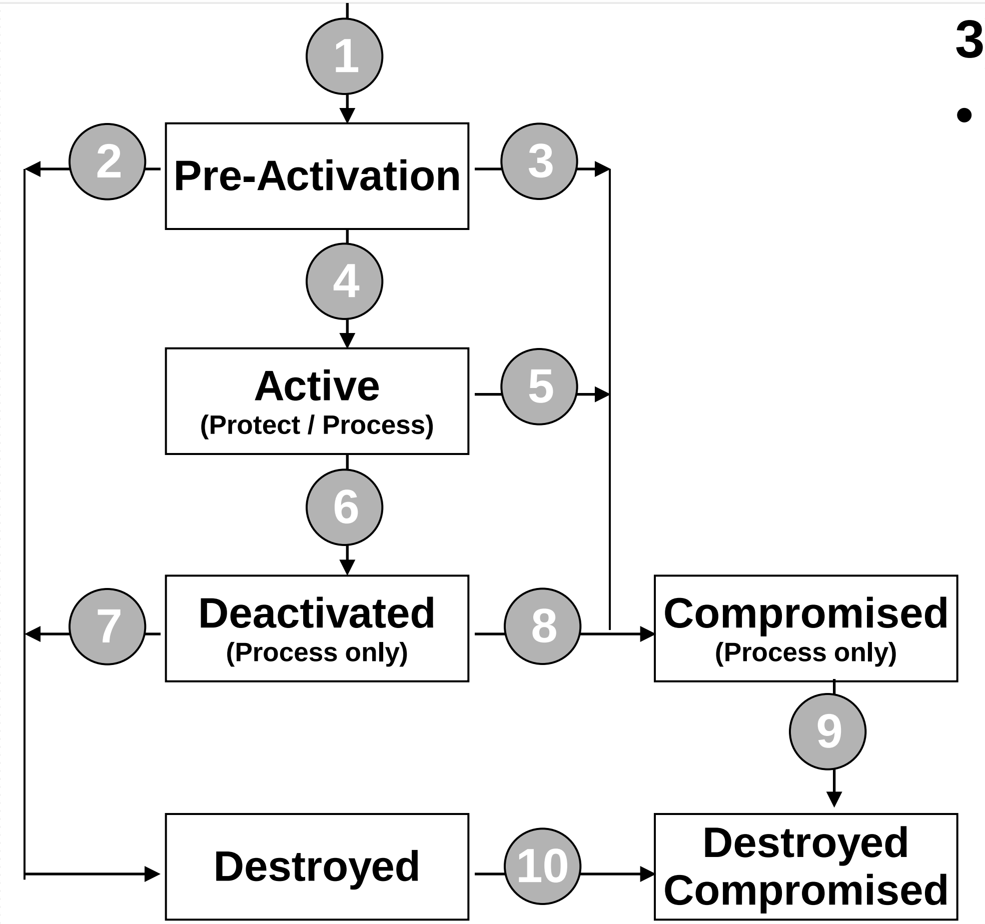
6) Deactivated

- 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

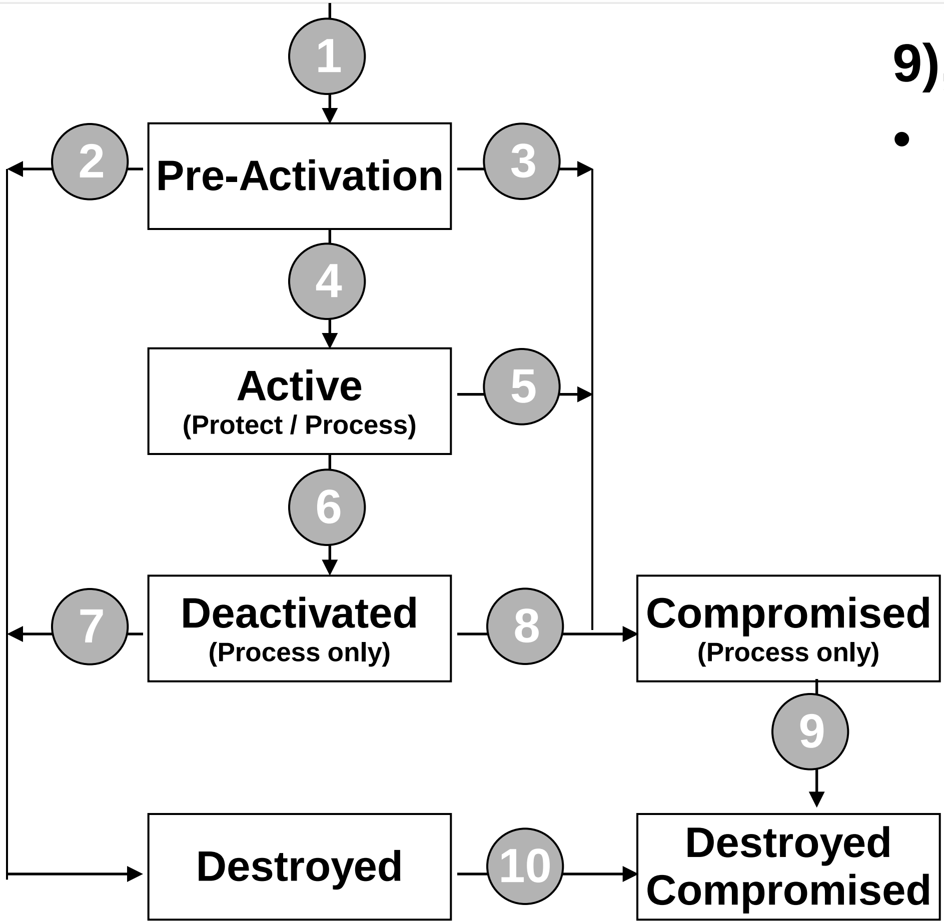


3), 5), 8) Compromised

- 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

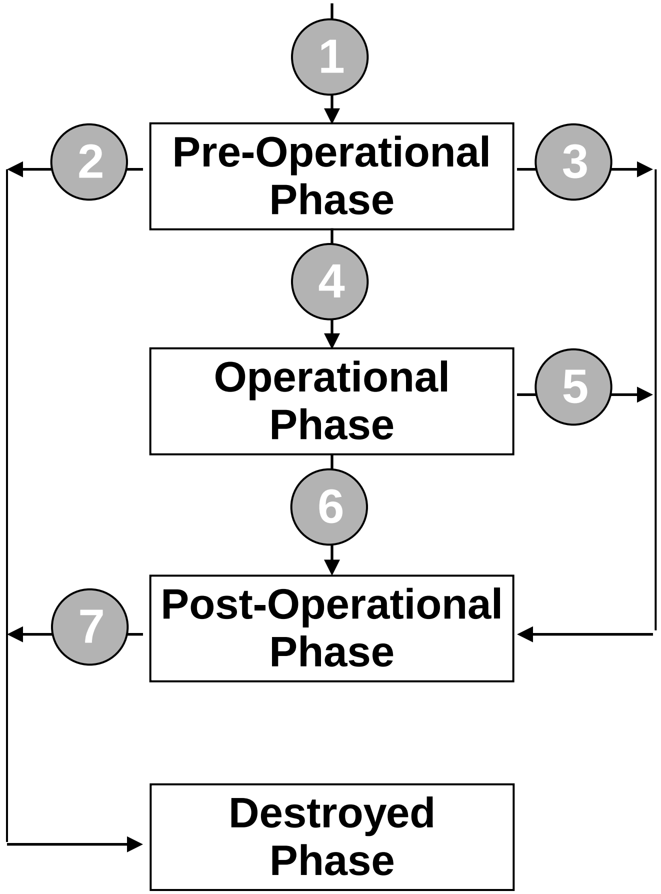


9), 10) Destroyed Compromised

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

Key Management Phases and Transitions

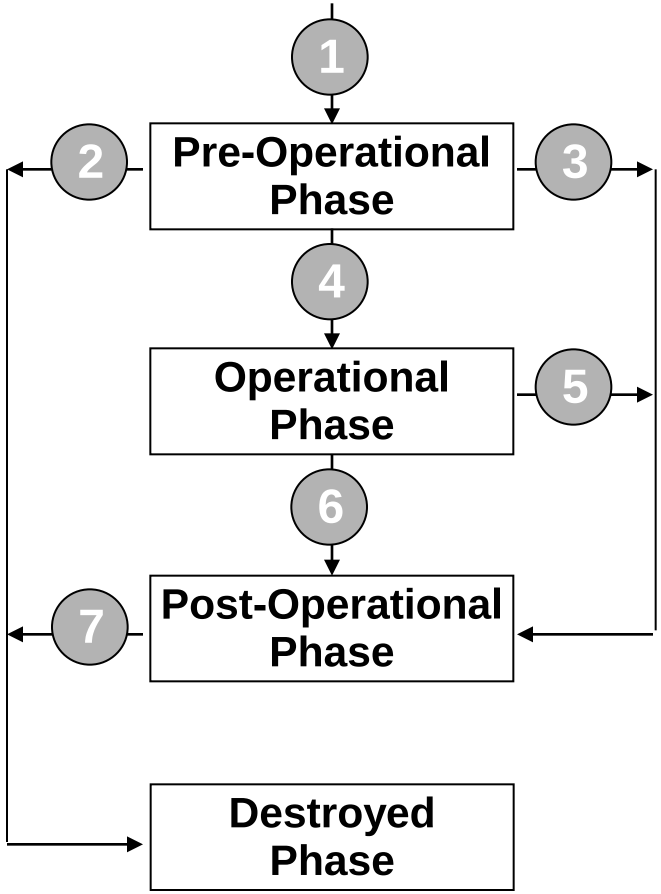
Ref: NIST SP 800-57



Management Phase Transitions

- The key management phase is a function of the state and usage of keys. Each key state requires a different management procedure.

Key Management Phases (cont.)
Ref: NIST SP 800-57



1) Pre-operational Phase

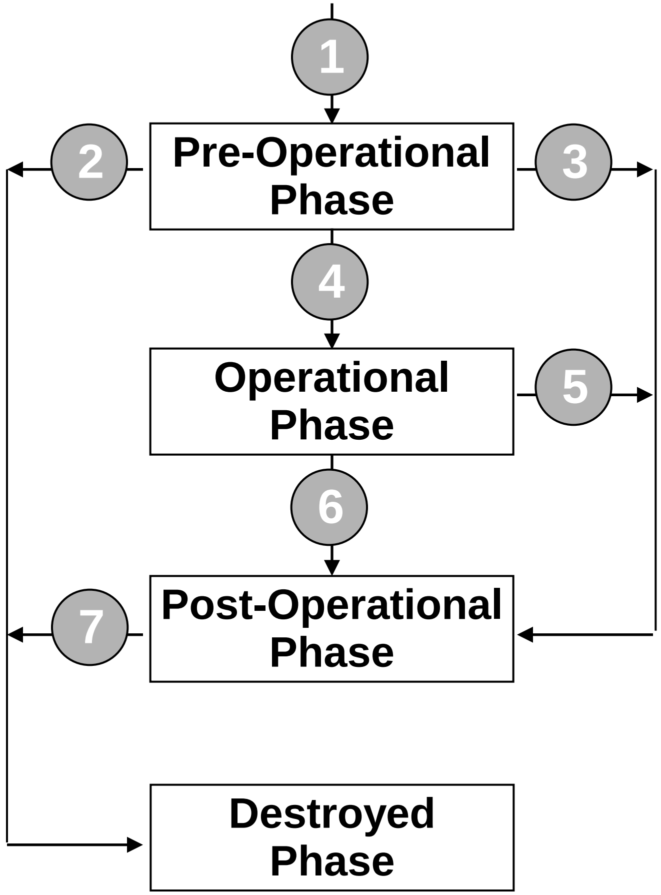
- The keying material is not yet available for normal cryptographic operations. Keys are generated, and may be in the pre-activation state. System or enterprise attributes are established during this phase as well.

4) Operational Phase

- The keying material is available and in normal use. Keys are in the active state. Keys may be designated as protect only, process only, or protect and process.

Key Management Phases (cont.)

Ref: NIST SP 800-57



3), 5), 6) Post-operational Phase

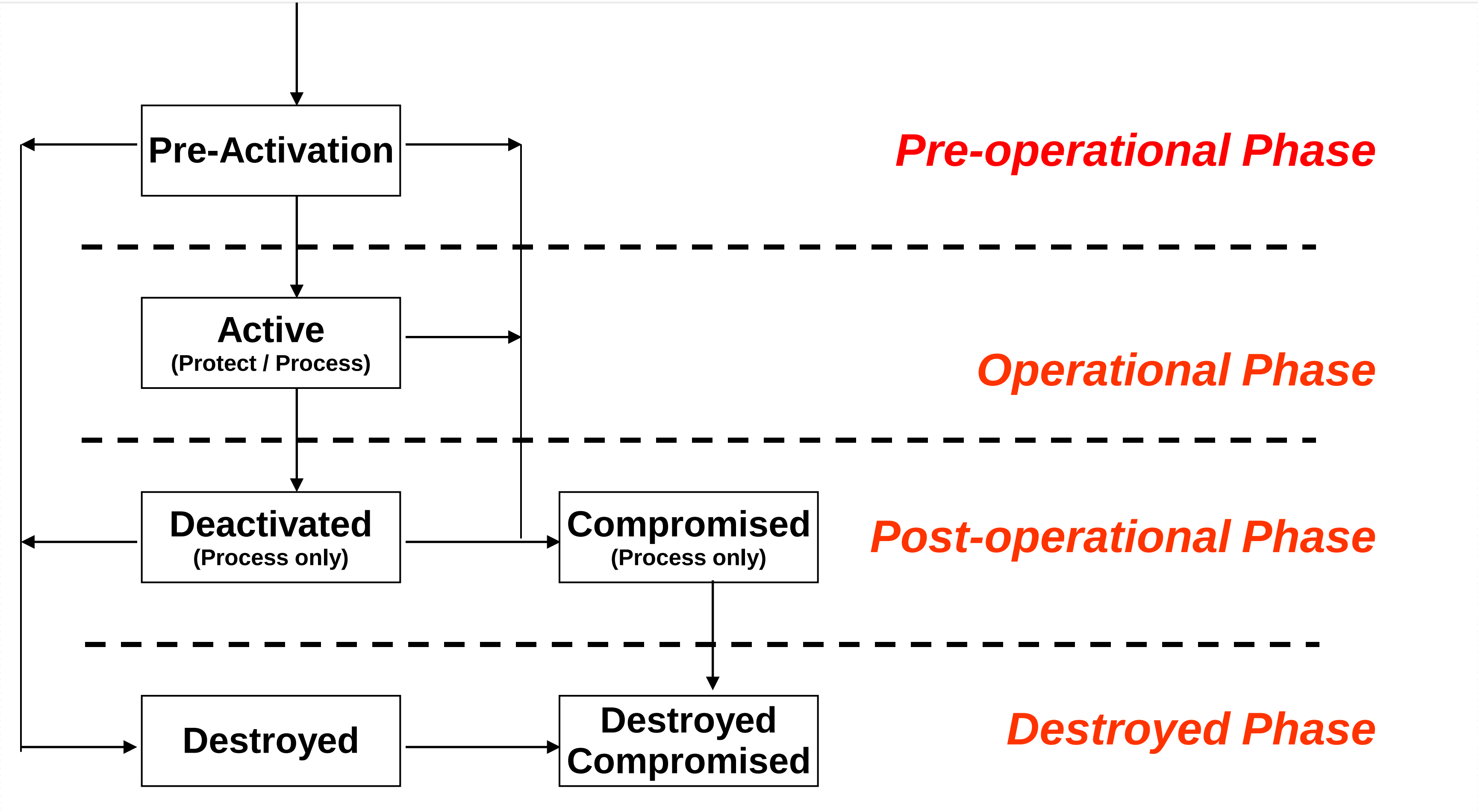
- The keying material is no longer in normal use, but access to the keying material is possible and the keying material may be used for process only in certain circumstances. Keys are in the deactivated or compromised states. Keys are archived when not processing data.

2), 7) Destroyed Phase

- Keys are no longer available. All records of their existence may have been deleted. Although the keys themselves are destroyed, the key attributes may be retained.

Key Management States and Phases

Ref: NIST SP 800-57



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 full range of key space.

Key Generation Examples

- Stream cipher
  - capture true random stream (OTP), or
  - generate pseudorandom stream with keystream generator using initial short random key (initial state or seed).
- AES symmetric block cipher
  - Ensure that any key is as probable as any other
- RSA asymmetric cipher
  - make sure *n* (modulus) is sufficiently large to prevent factoring

Key Protection

- Keys should be
  - accessible for authorised users,
  - protected from unauthorised users
- Old keys must be kept, if messages encrypted under these keys are stored
  - 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
  - Need to protect private keys only
  - Public keys made public but must be authentic

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

Outline

- Key management
- Key Establishment
- Public key infrastructure
- Digital certificates
- PKI trust models
- PKI components

Key Establishment: The Problem

- Two parties wish to communicate securely using cryptography.
- Symmetric ciphers are much more efficient than asymmetric ciphers.
- In practice, for large amounts of data, symmetric ciphers are preferred over asymmetric ciphers.
- Therefore need to establish a new shared key: a *session key*.

Key Establishment: Options

Three options to distribute session key

1. Use existing shared keys
   - Does not scale to large groups or open networks
2. Use a trusted third party (server) who shares a symmetric (long-term) key with each user
   - Need protocol such as Kerberos (lecture on authent.)
3. Use asymmetric cipher to protect session key
   - Requires PKI (see next)

Outline

- Key management
- Key Establishment
- Public-key infrastructure
- Digital certificates
- PKI trust models
- PKI components

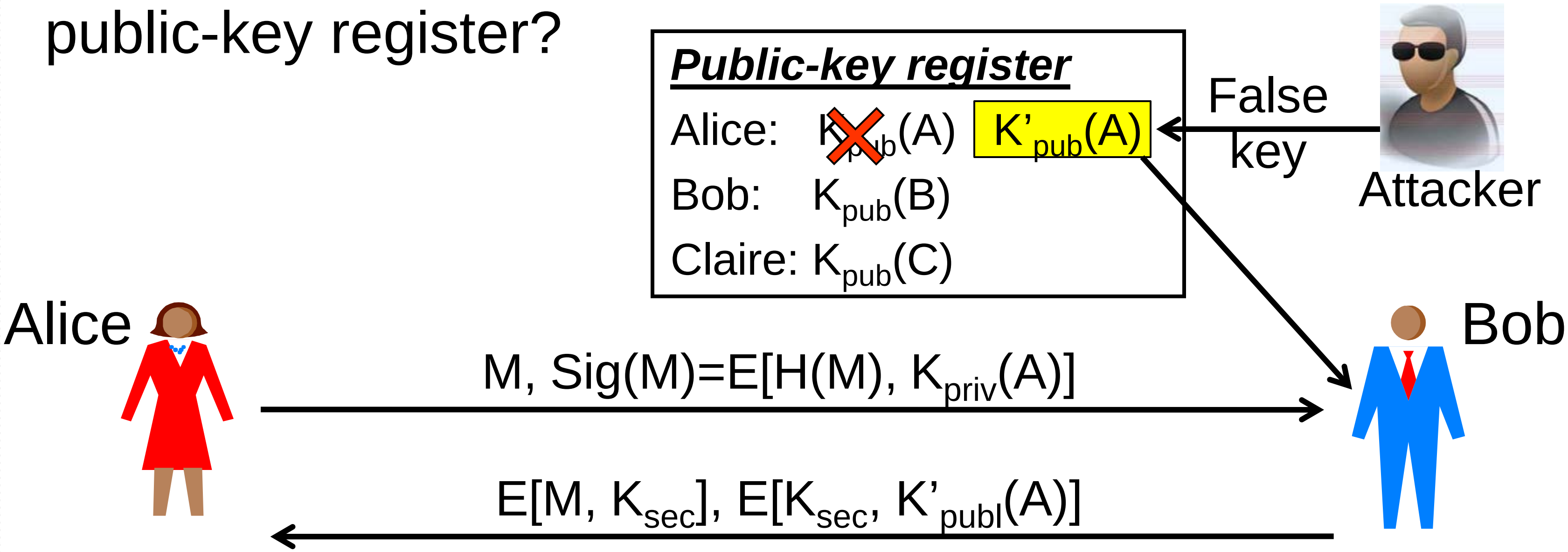
Why the interest in PKI ?

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

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

Public Keys and the Spoofing Problem

- Consequence of attacker inserting false key for Alice in the public-key register?



Public-key register

Alice: $\cancel{K_{pub}}(A)$ $K'_{pub}(A)$

Bob: $K_{pub}(B)$

Claire: $K_{pub}(C)$

$M, Sig(M)=E[H(M), K_{priv}(A)]$

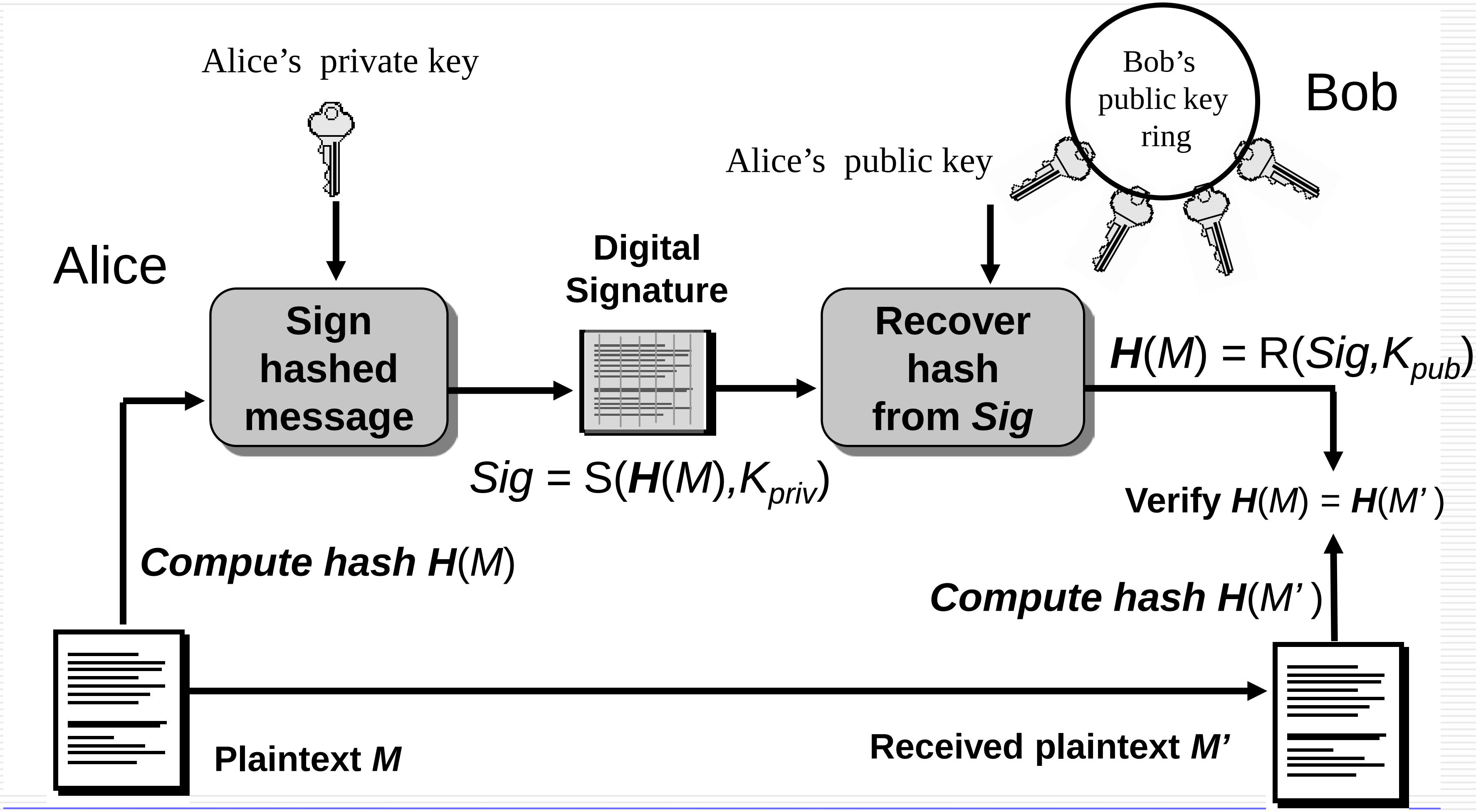
$E[M, K_{sec}], E[K_{sec}, K'_{publ}(A)]$

- This breaks the security because:
  - a valid digital signature from Alice will be rejected
  - a confidential message to Alice can be read by attacker

Digital signature notations

- Cryptographic terminology and notation:
  - Private Key **K_{priv}**: confidential key only known by owner
  - Public Key **K_{pub}**: publicly known key
  - Plaintext message **M**: the original message or data
  - Hash function **H**: used to create hash block
  - Digital signature Sig: cryptographic authentication code
  - Signature generation **S**: Function for creating the digital signature Sig on message M or on hash H(M)
  - Recovery function R: Function for recovering the message M or hash H(M) from the digital signature Sig

Digital signature


Alice's private key
Bob's public key ring
Bob
Alice's public key
Alice
Digital Signature
Sign hashed message
Recover hash from *Sig*
$\boldsymbol{H}(M) = \mathrm{R}(Sig, K_{pub})$
$Sig = \mathrm{S}(\boldsymbol{H}(M), K_{priv})$
Verify $\boldsymbol{H}(M) = \boldsymbol{H}(M')$
Compute hash $\boldsymbol{H}(M)$
Compute hash $\boldsymbol{H}(M')$
Plaintext *M*
Received plaintext *M'*

Public-key infrastructure

- **Integrity and trustworthiness of public keys:**
  - How can a user be sure who a public key belongs to?
  - How can a user be sure a public key has not been altered - intentionally or unintentionally?
- **How can public keys be made available in a trusted way?**
  - Use public-key certificates (aka. digital certificates) issued by a trusted third party
    - a Certification Authority (CA).
  - A public-key certificate is a public key digitally signed by a CA.
  - A hierarchy of public-key certificates becomes a PKI.

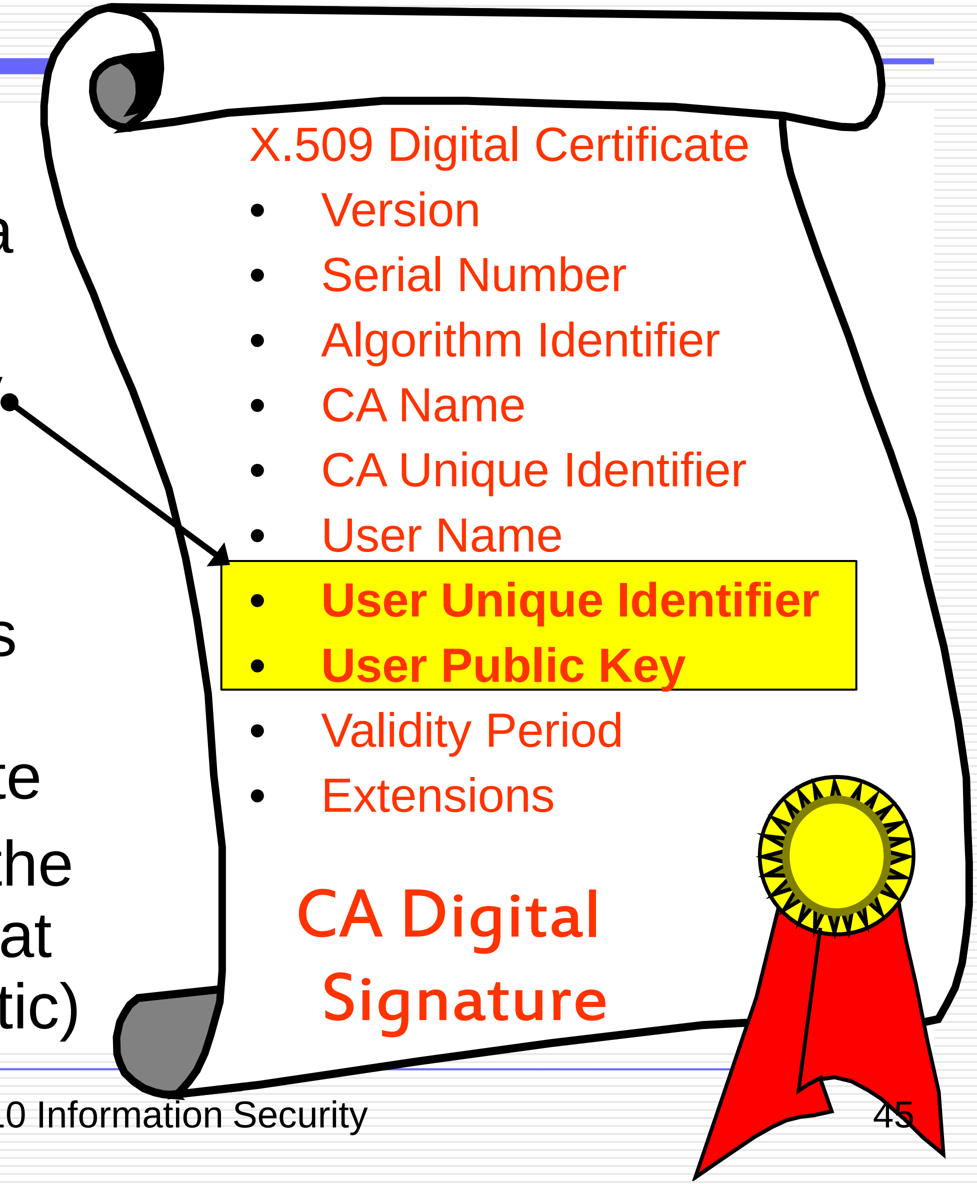
Public-key infrastructure

- Public-key cryptography needs a **PKI** to work
- **A PKI is a set 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)

that enables practical application of public key cryptography often in large, distributed settings

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)



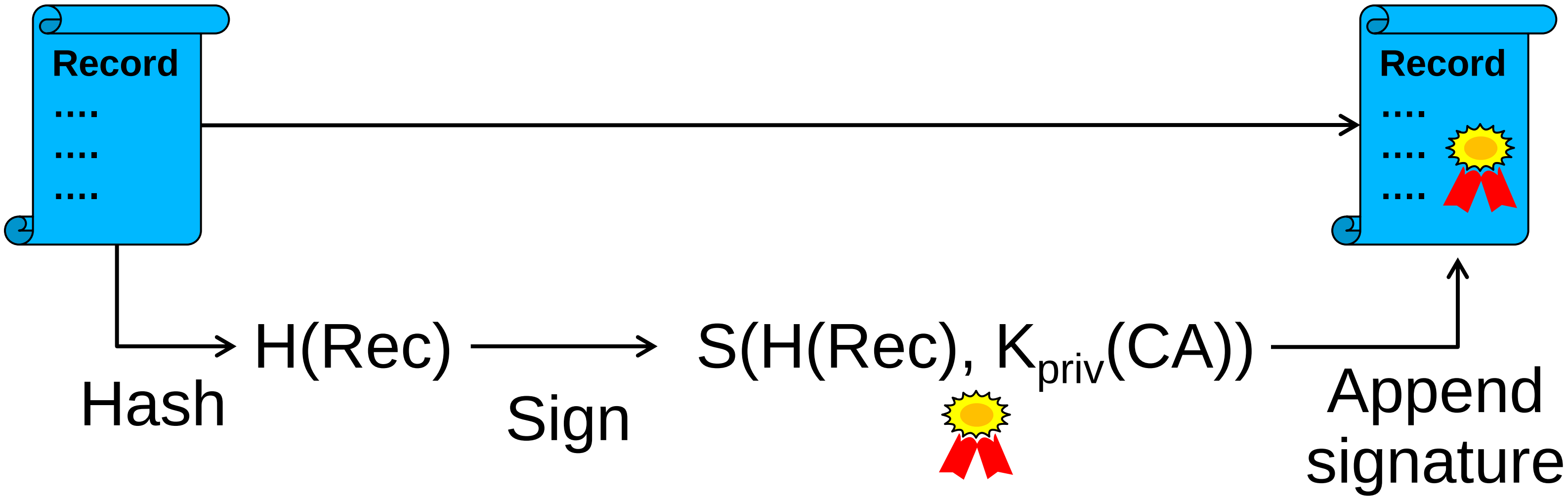
X.509 Digital Certificate

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

CA Digital Signature

How to generate a digital certificate?

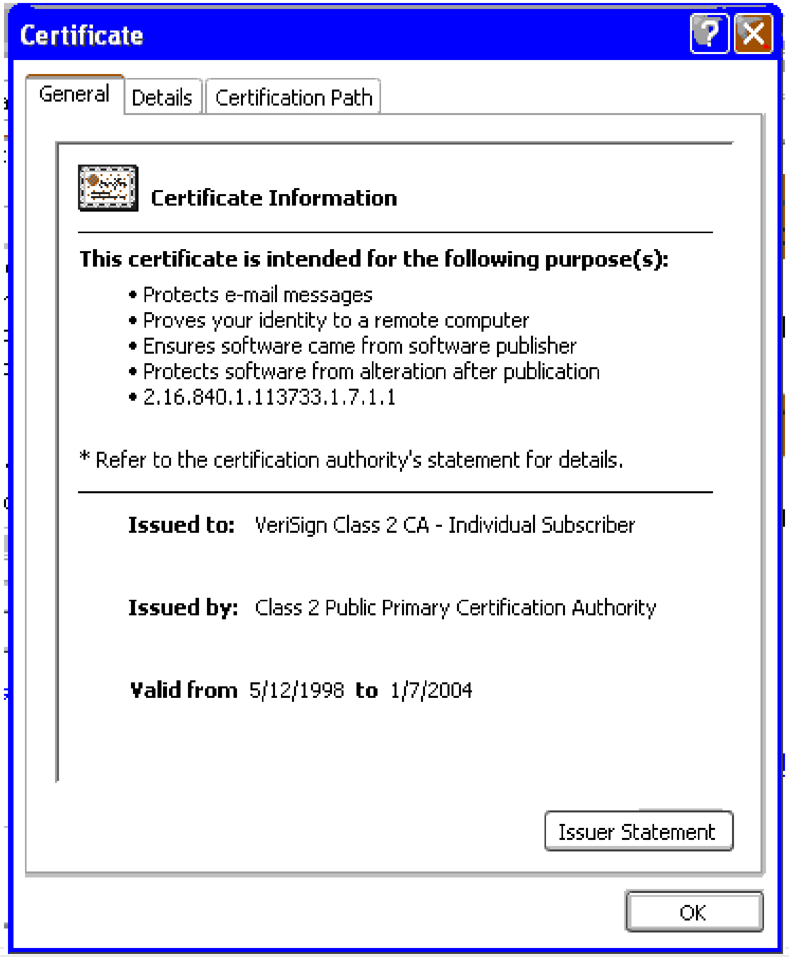
1. Assemble the information in single record Rec
2. Hash the record
3. Sign the hashed record
4. Append the digital signature to the record



Public-key certificates

- Public-key certificates (aka. digital certificates) logically bind a public key to a unique user identifier (“distinguished name”)
- A digital certificate contains:
  - the user’s public key
  - the user’s unique identifier
  - + some other information e.g. validity period
- A Certificate Authority (CA) creates and digitally signs the certificate

Example of X.509 Certificate



Certificate Information

This certificate is intended for the following purpose(s):

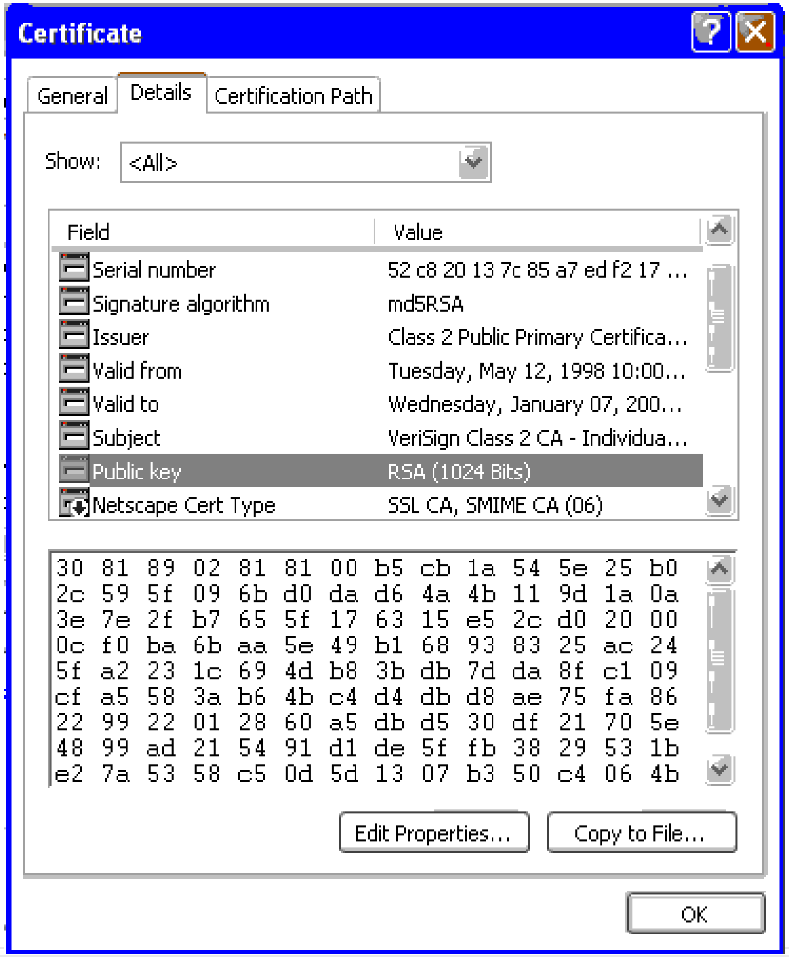
- Protects e-mail messages
- Proves your identity to a remote computer
- Ensures software came from software publisher
- Protects software from alteration after publication
- 2.16.840.1.113733.1.7.1.1

* Refer to the certification authority's statement for details.

Issued to: VeriSign Class 2 CA - Individual Subscriber

Issued by: Class 2 Public Primary Certification Authority

Valid from 5/12/1998 **to** 1/7/2004



Show: <All>

Field	Value
Serial number	52 c8 20 13 7c 85 a7 ed f2 17 ...
Signature algorithm	md5RSA
Issuer	Class 2 Public Primary Certifica...
Valid from	Tuesday, May 12, 1998 10:00...
Valid to	Wednesday, January 07, 200...
Subject	VeriSign Class 2 CA - Individua...
Public key	RSA (1024 Bits)
Netscape Cert Type	SSL CA, SMIME CA (06)

```
30 81 89 02 81 81 00 b5 cb 1a 54 5e 25 b0
2c 59 5f 09 6b d0 da d6 4a 4b 11 9d 1a 0a
3e 7e 2f b7 65 5f 17 63 15 e5 2c d0 20 00
0c f0 ba 6b aa 5e 49 b1 68 93 83 25 ac 24
5f a2 23 1c 69 4d b8 3b db 7d da 8f c1 09
cf a5 58 3a b6 4b c4 d4 db d8 ae 75 fa 86
22 99 22 01 28 60 a5 db d5 30 df 21 70 5e
48 99 ad 21 54 91 d1 de 5f fb 38 29 53 1b
e2 7a 53 58 c5 0d 5d 13 07 b3 50 c4 06 4b
```

Using certificates to send confidential messages

Alice wants to send confidential $C = E(M, K_{pub}(B))$ to Bob

1. Alice is the relying party and must get Bob's public key
   - Alice can obtain $Cert_B$
   - Alice validates $Cert_B$
   - Alice obtains $K_{pub}(B)$ from $Cert_B$
2. Alice uses $K_{pub}(B)$ to encrypt message M

- If Alice
  - trusts the CA that issued $Cert_B$ (i.e. to be competent and honest)
  - and is certain of CA's public key and unique identifier
  - and is certain of Bob's unique identifier
- then Alice can be sure that **only** Bob can read message
- Message M can be a secret session key.

Using certificates to verify signatures

Bob sends signed message {M, Sig$_B$(h(M)), Cert$_B$} to Alice

h(M) is the hash value of message M.

1. Alice is the relying party and must first validate Cert$_B$
   - Alice uses CA's public key K$_{pub}$(CA) to verify CA's signature on the binding between the public key and Bob's unique identifier.
2. Alice obtains K$_{pub}$(B) from the certificate Cert$_B$
3. Alice uses K$_{pub}$(B) to verify signature Sig$_B$(h(M)) on M

- If Alice
  - trusts the CA that issued Cert$_B$ (i.e. to be competent and honest)
  - and is certain of CA's public key and unique identifier
  - and is certain of Bob's unique identifier
- then Alice is certain that message M came from Bob

Digital certificates
Some questions

1. *What advantage is there for Bob in having digital certificate Cert$_B$?*
2. *Who can have access to Cert$_B$?*
3. *Why would someone want to verify the signature in Cert$_B$?*
4. *What does Alice need for verifying Cert$_B$?*
5. *After someone has verified Cert$_B$, of what can they be assured?*

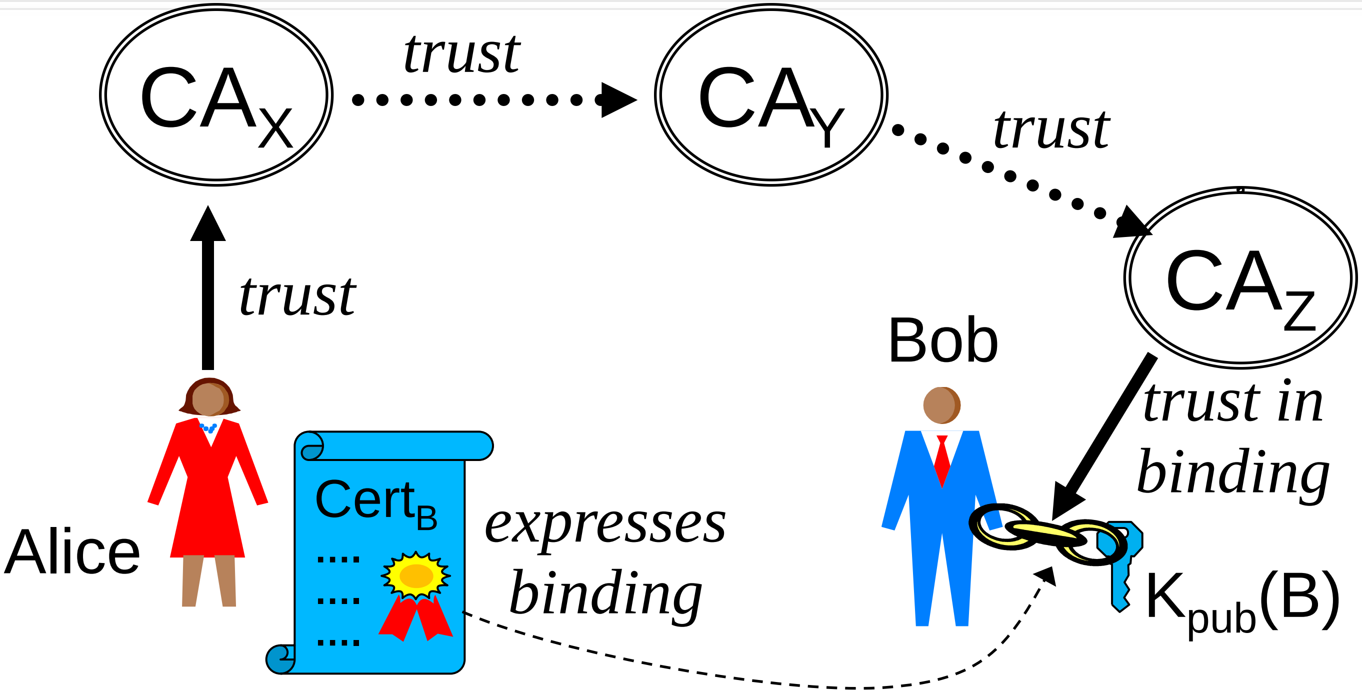
PKI trust models

- **Q: Can Alice trust Bob’s digital certificate?**
- **Q: Can Alice trust the CA that issued the cert.** and how does she get the CA’s public key?
  - To answer these questions we need to
- **Establish trust relationships**
  - between different Certificate Authorities, and
  - between Certificate Authorities and end users,

Define PKI trust models

Diagram

PKI Certification Path



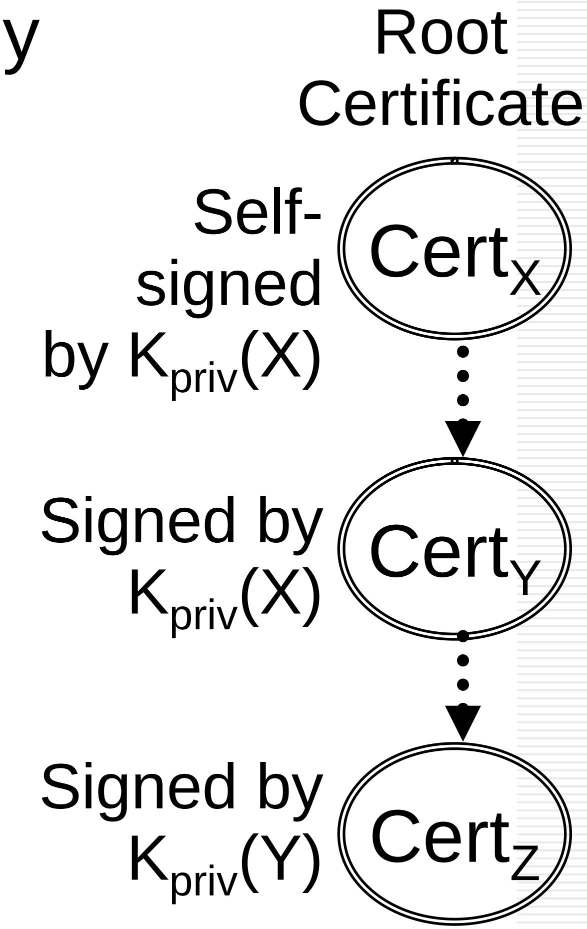
- For Alice to validate CertB and be assured of authenticity of Bob’s public key, Alice must create path of trust to Bob’s certificate which binds public key to Bob’s name.

PKI Certification Path Explanation

- Alice receives Bob's digital certificate Cert_B with Bob's public key issued by CA_Z
- Alice needs to validate Bob's certificate
  - Cert_B can be verified with Cert_Z
  - Cert_Z can be verified with Cert_Y
  - Cert_Y can be verified with Cert_X
- Alice trusts CA_X and has obtained authentic copy of CA_X's public key $K_{pub}(\text{CA}_X)$ via secure channel
- A *certification path* is established !
- The relying party Alice can validate Cert_B

Root CA’s and Self Signed Certificates

- A root CA is at the top of a hiearchy
- No other CA certifies the root CA’s public key
- **Self-signed certificate** is created by signing the root CA public key with the corresponding private key
  - Semantically not a digital signature and not a certificate
  - Syntactically looks like a certificate
  - Enables processing in PKI software
  - **Must be sent securely out-of-band**



Root Certificate

Self-signed by $K_{priv}(X)$

$Cert_X$

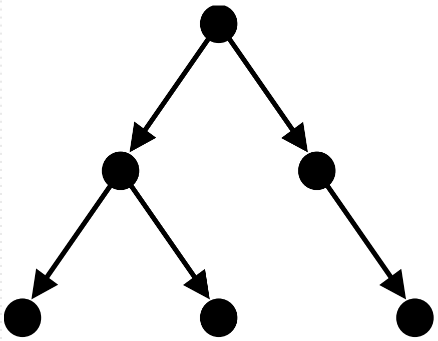
Signed by $K_{priv}(X)$

$Cert_Y$

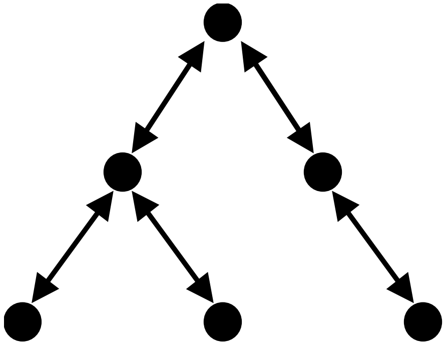
Signed by $K_{priv}(Y)$

$Cert_Z$

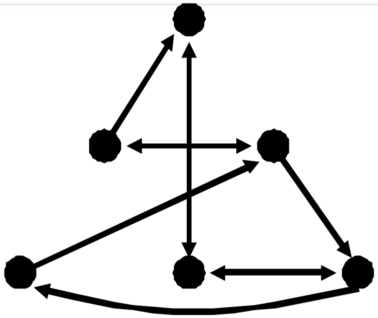
PKI Trust Models



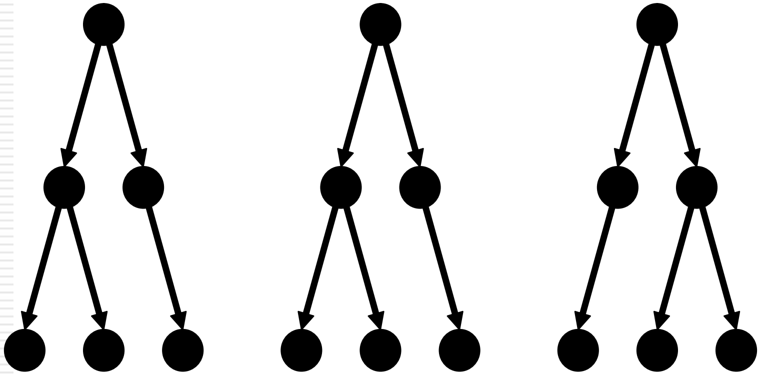
Strict hierarchy



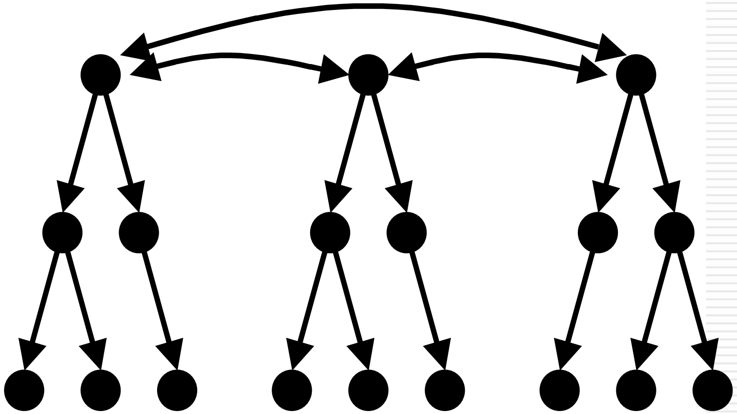
General hierarchy



Mesh model /
User-centric model



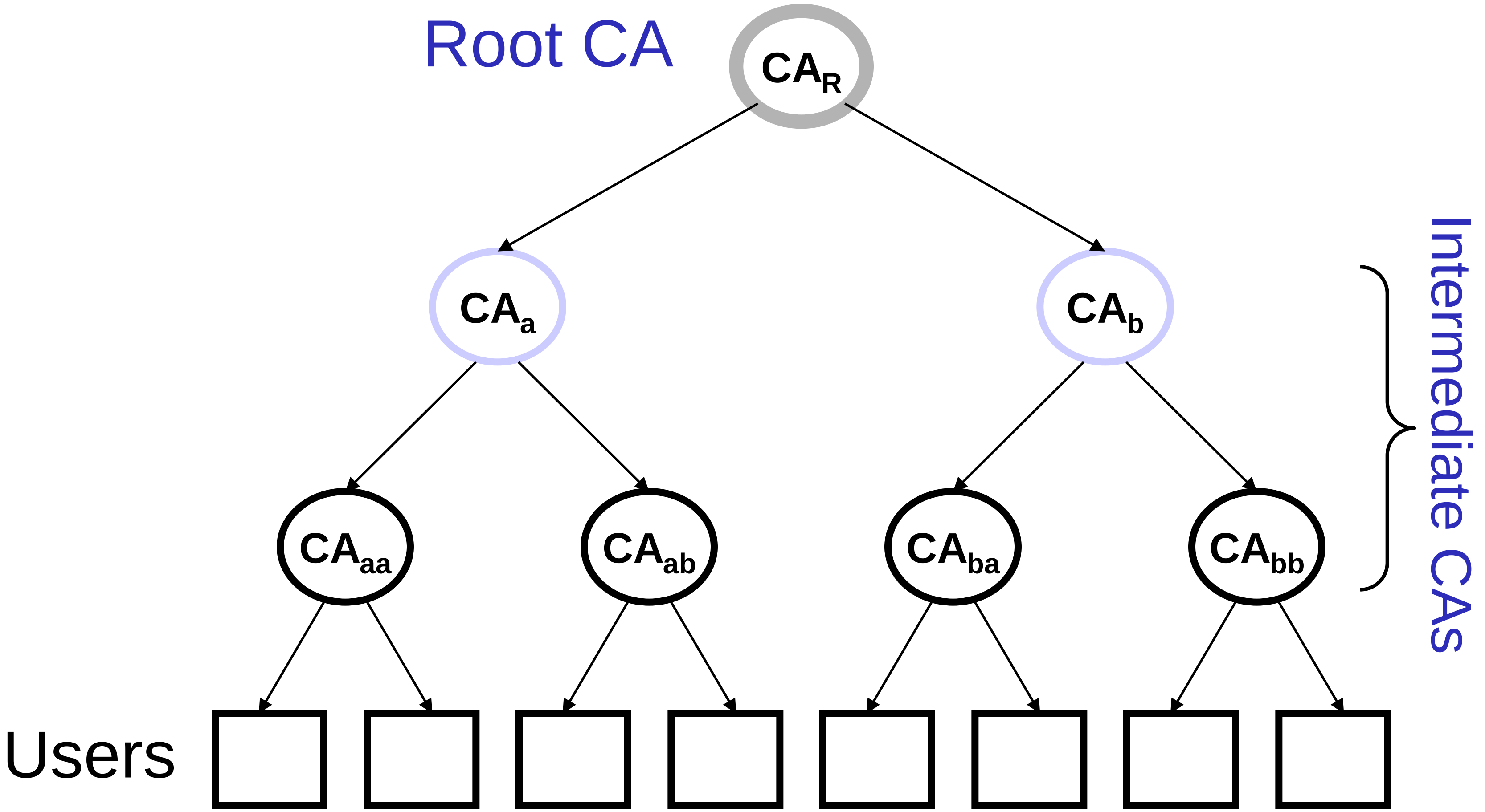
Isolated strict hierarchies: "Browser PKI"



Cross certified strict hierarchies

PKI trust models

Strict hierarchical model



PKI trust models

Strict hierarchical model

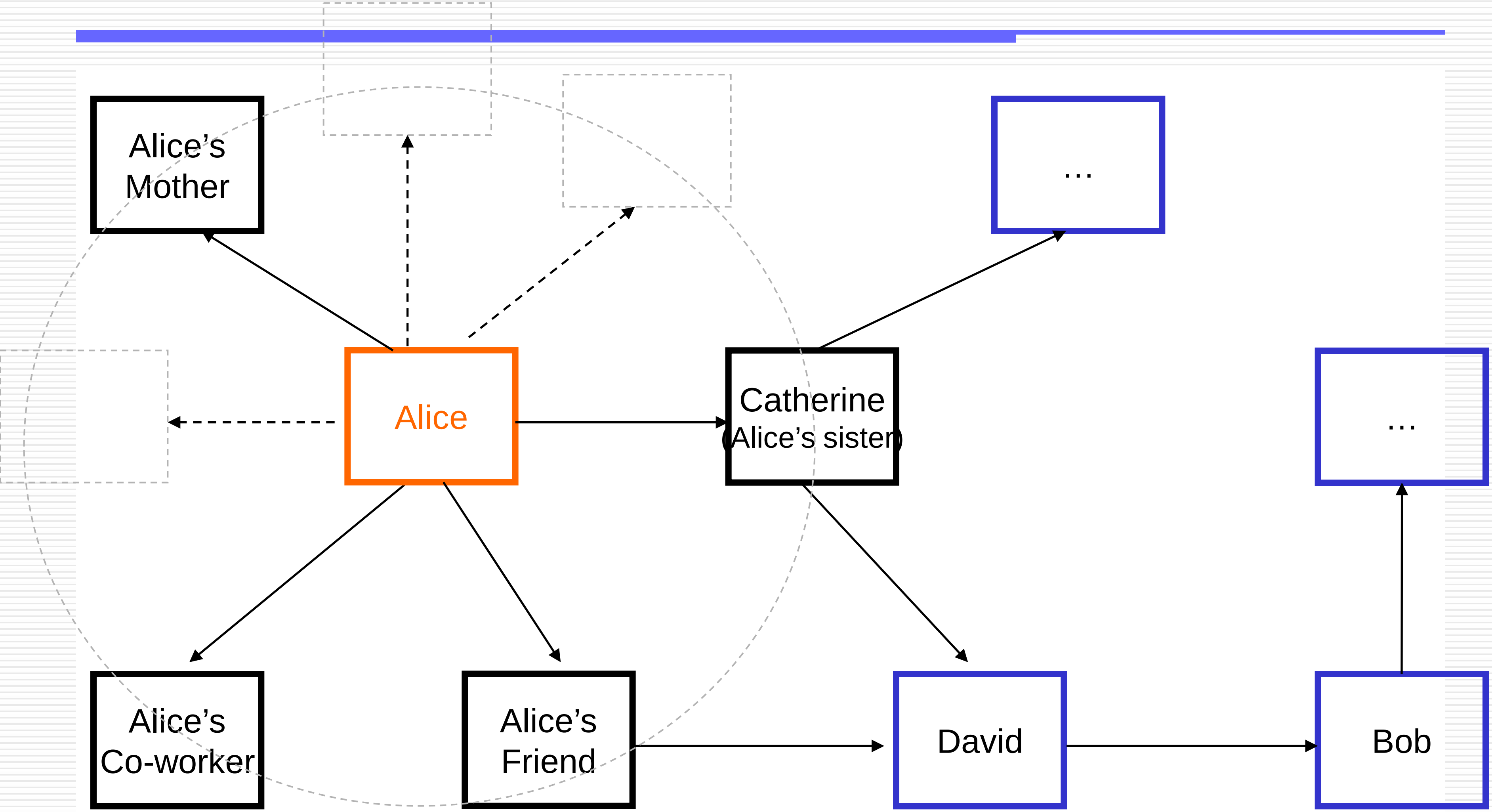
- Highly regulated
  - each CA must follow rules regarding to whom they may issue certificates
- Tree structure
  - Single root CA
  - Users are leaves of the tree
  - Each node is certified by its immediate parent CA
- Root CA
  - Starting point for trust
  - All users trust the root CA, and must receive its public key through a secure out-of-band channel

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?)

PKI trust models

User-centric model, as in PGP



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?

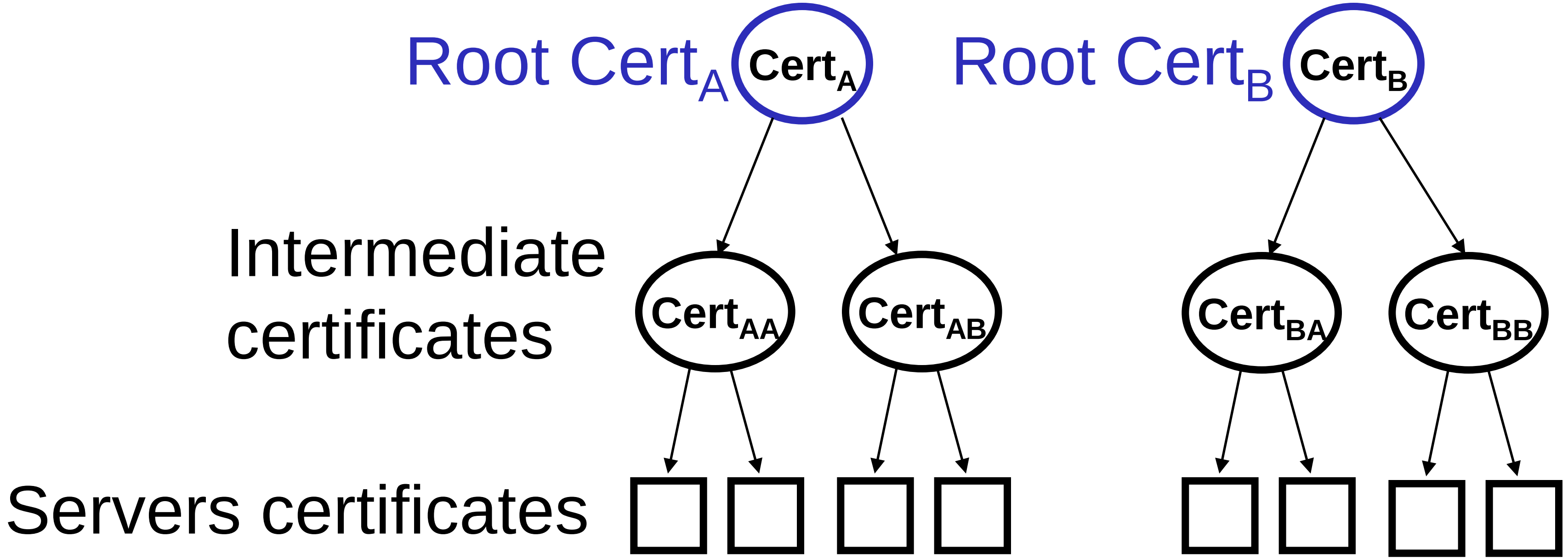
PKI trust models
User-centric model

- 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:
  - It 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

PKI Trust Models

Browser PKI

- The browser PKI model consists of isolated strict hierarchies where the (root) CA certificates are installed together with the browser



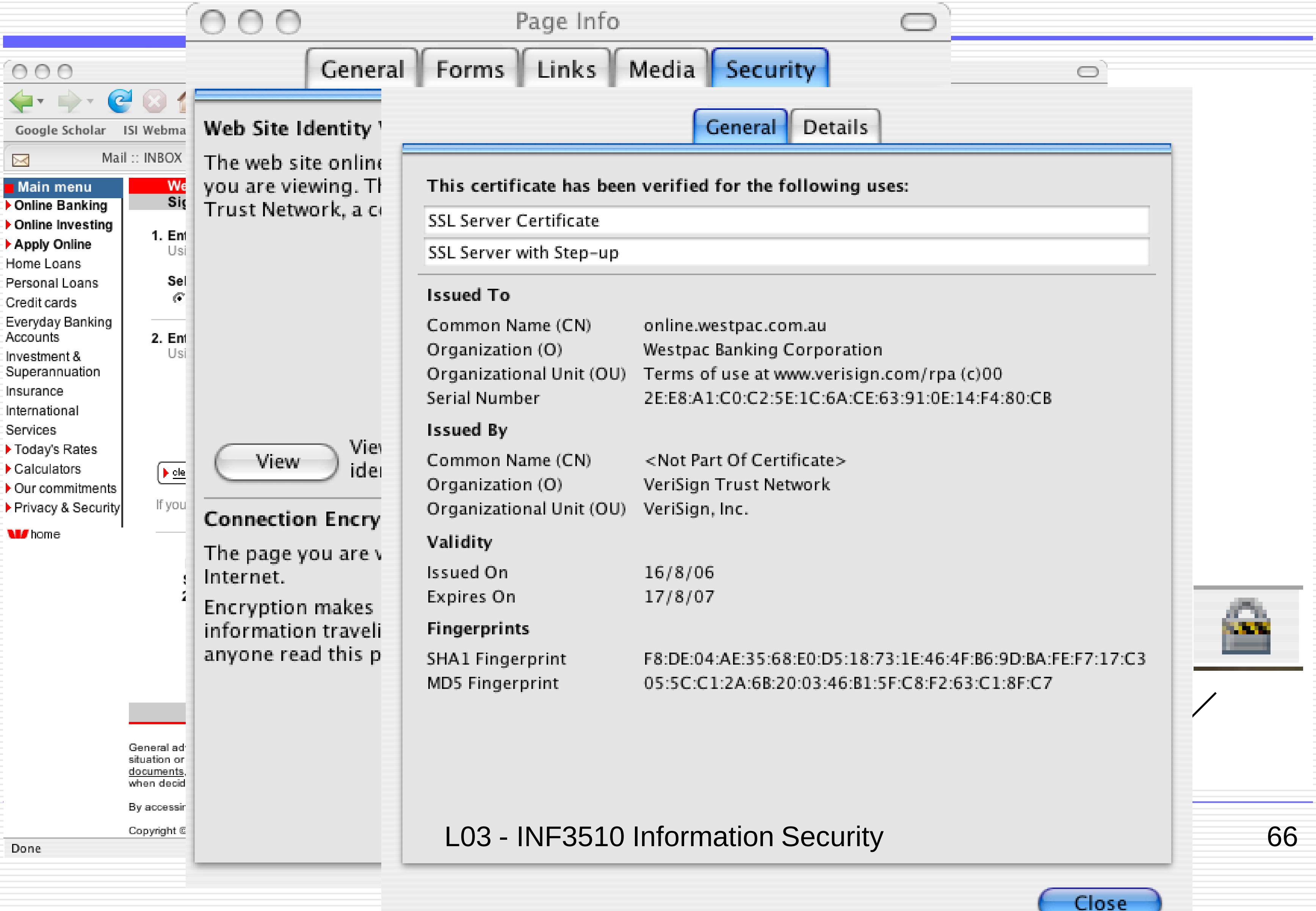
Browser PKI root certificate installation

- Distribution of root certificates which should happen securely out-of-band, is now 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*
- CAs pay browser vendors to carry CA certs

Browser PKI limitations and weaknesses

- Certification path processing is limited
  - Certificate validation only by stored certificates
- List of trusted certificates controlled by user - not well protected from modification attacks
  - Users habitually accept incoming certificates that the browser cannot automatically verify
  - Malware can install fake server and root certificates
- Cross certification not supported
- Certificate Revocation not supported
- No legal agreement between users and CAs
  - Liability rests with the users and not with the CAs

Authenticating a Web Site



Web Site Identity V

The web site onlin
you are viewing. Th
Trust Network, a c

Connection Encry

The page you are v
Internet.

Encryption makes
information traveli
anyone read this p

This certificate has been verified for the following uses:

SSL Server Certificate

SSL Server with Step-up

Issued To

Common Name (CN)	online.westpac.com.au
Organization (O)	Westpac Banking Corporation
Organizational Unit (OU)	Terms of use at www.verisign.com/rpa (c)00
Serial Number	2E:E8:A1:C0:C2:5E:1C:6A:CE:63:91:0E:14:F4:80:CB

Issued By

Common Name (CN)	<Not Part Of Certificate>
Organization (O)	VeriSign Trust Network
Organizational Unit (OU)	VeriSign, Inc.

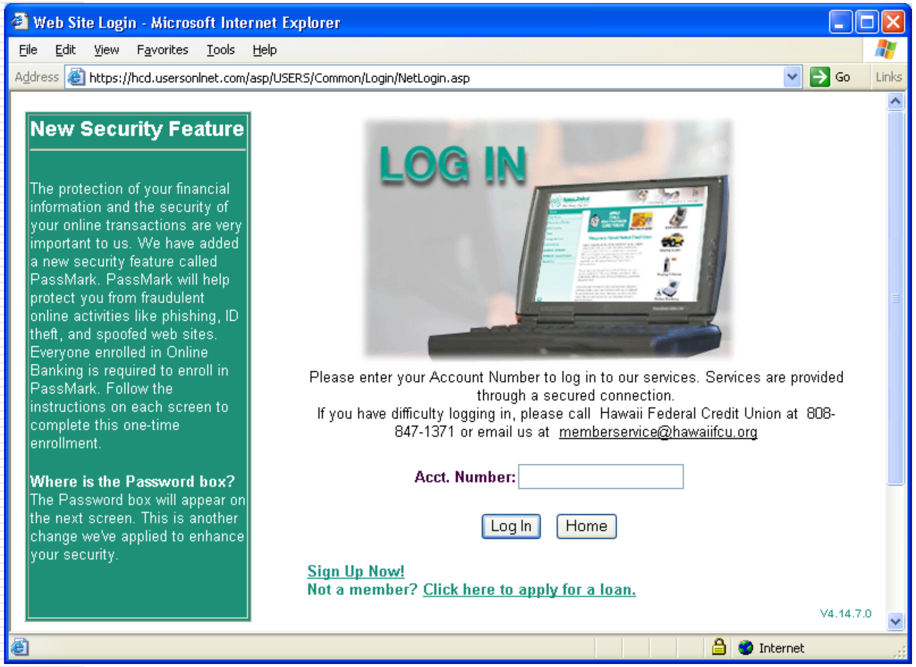
Validity

Issued On	16/8/06
Expires On	17/8/07

Fingerprints

SHA1 Fingerprint	F8:DE:04:AE:35:68:E0:D5:18:73:1E:46:4F:B6:9D:BA:FE:F7:17:C3
MD5 Fingerprint	05:5C:C1:2A:6B:20:03:46:B1:5F:C8:F2:63:C1:8F:C7

Phishing and fake certificates Hawaii Federal Credit Union


Web Site Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://hcd.usersonlnet.com/asp/USERS/Common/Login/NetLogin.asp
Go
Links
LOG IN
Log In
Home
V4.14.7.0
Internet

New Security Feature

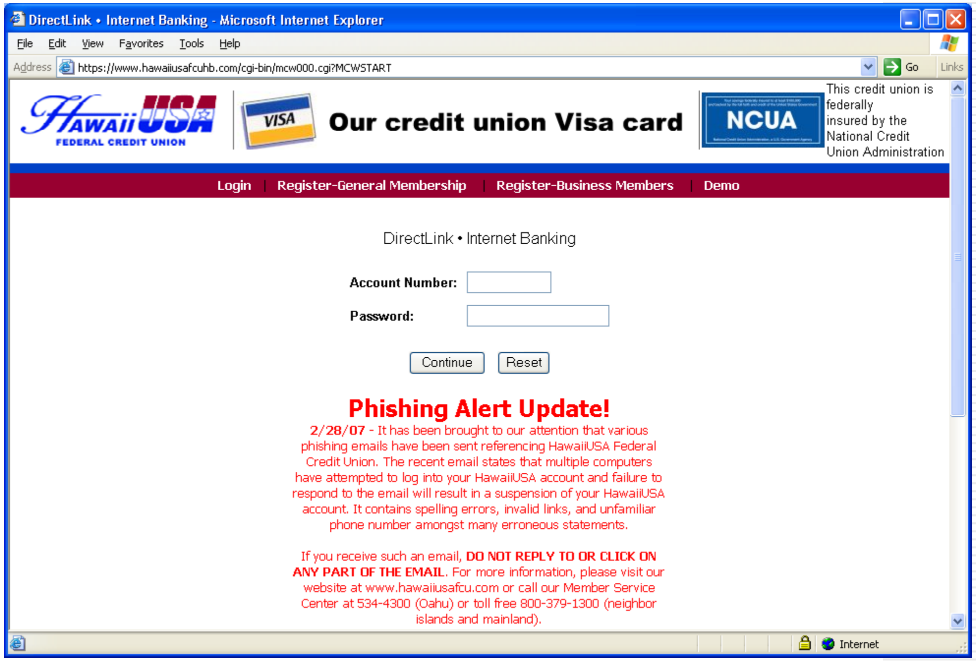
The protection of your financial information and the security of your online transactions are very important to us. We have added a new security feature called PassMark. PassMark will help protect you from fraudulent online activities like phishing, ID theft, and spoofed web sites. Everyone enrolled in Online Banking is required to enroll in PassMark. Follow the instructions on each screen to complete this one-time enrollment.

Where is the Password box? The Password box will appear on the next screen. This is another change we've applied to enhance your security.

Please enter your Account Number to log in to our services. Services are provided through a secured connection.
If you have difficulty logging in, please call Hawaii Federal Credit Union at 808-847-1371 or email us at memberservice@hawaiifcu.org

Acct. Number:

Sign Up Now!
Not a member? Click here to apply for a loan.


DirectLink • Internet Banking - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://www.hawaiiusafcuhb.com/cgi-bin/mcw000.cgi?MCWSTART
Go
Links
HawaiiUSA
FEDERAL CREDIT UNION
VISA
Our credit union Visa card
NCUA
Login
Register-General Membership
Register-Business Members
Demo
Continue
Reset
Internet

This credit union is federally insured by the National Credit Union Administration

DirectLink • Internet Banking

Account Number:

Password:

Phishing Alert Update!

2/28/07 - It has been brought to our attention that various phishing emails have been sent referencing HawaiiUSA Federal Credit Union. The recent email states that multiple computers have attempted to log into your HawaiiUSA account and failure to respond to the email will result in a suspension of your HawaiiUSA account. It contains spelling errors, invalid links, and unfamiliar phone number amongst many erroneous statements.

If you receive such an email, **DO NOT REPLY TO OR CLICK ON ANY PART OF THE EMAIL**. For more information, please visit our website at www.hawaiiusafcu.com or call our Member Service Center at 534-4300 (Oahu) or toll free 800-379-1300 (neighbor islands and mainland).

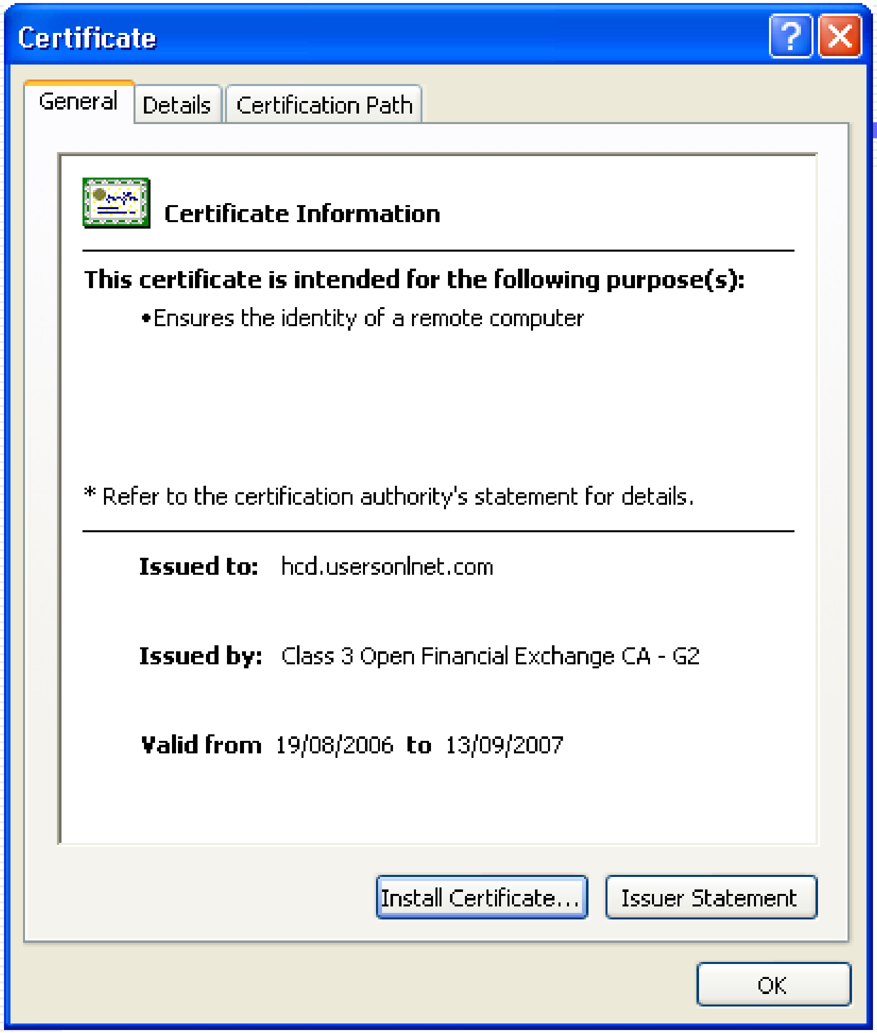
Authentic bank login

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



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer

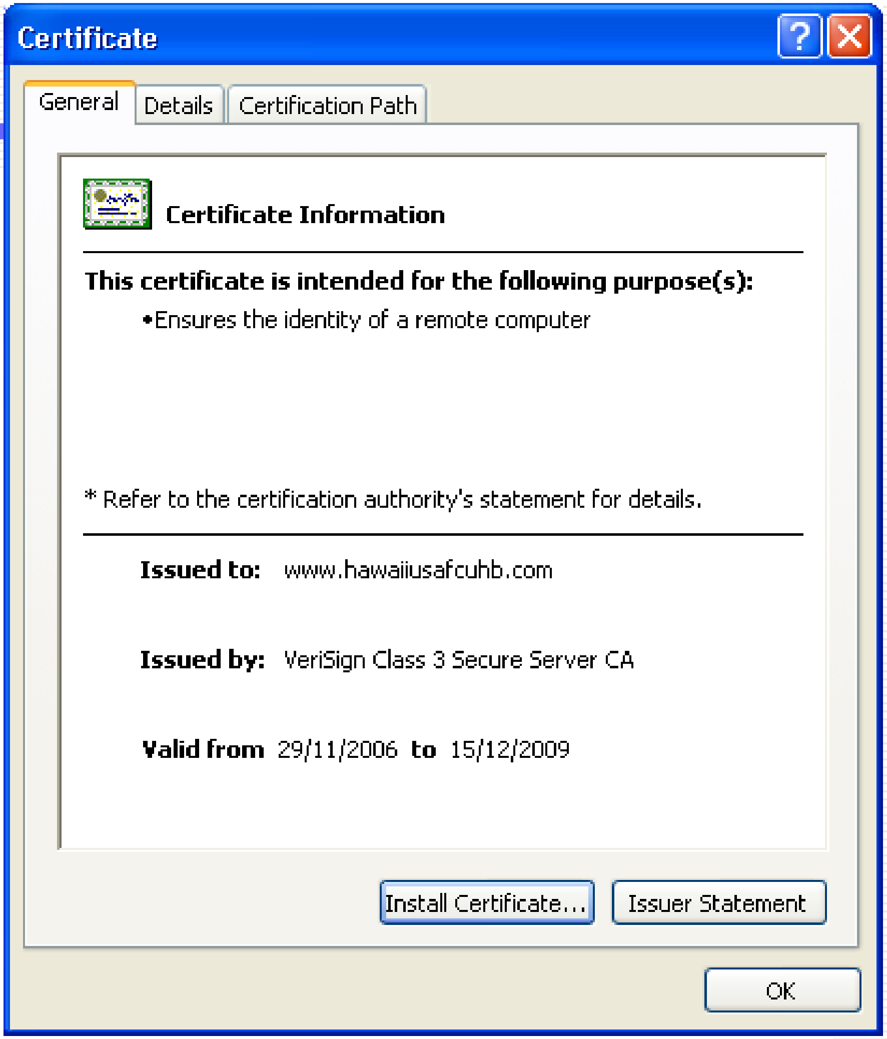
* Refer to the certification authority's statement for details.

Issued to: hcd.usersonlnet.com

Issued by: Class 3 Open Financial Exchange CA - G2

Valid from 19/08/2006 **to** 13/09/2007

Authentic certificate



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer

* Refer to the certification authority's statement for details.

Issued to: www.hawaiiusafcuhb.com

Issued by: VeriSign Class 3 Secure Server CA

Valid from 29/11/2006 **to** 15/12/2009

Fake certificate

Browser PKI and Fake Certificates

- The browser validates server certificates by
  - using public key of root certificate stored in browser
  - checking that the server certificate distinguished name and the domain name of web server are the same
- Attackers buy legitimate certificates which are automatically validated by browsers
  - **Fake certificates are legitimate certificates !!!**
- Server certificate validation is not authentication
  - Users who don’t know the server domain name cannot distinguish between authentic and fake server certificates

PKI components

- Common PKI components include:
  - Certification authorities (CA)
  - Certification Practice Statement (CPS)
    - Aka. Certification Policy (CP)
  - Registration authorities (RA)
  - Validation authorities (VA)
  - Certificate Revocation Lists (CRL)

PKI components

- Certification authorities
  - Are the primary building elements of a PKI
  - Create, issue, and revoke certificates for subscribers and other CAs
  - Have a Certification Practice Statement (CPS)
    - legally binding document
    - must be approved before CA can operate
    - outlines practices and procedures that the CA will follow in issuing certificates

PKI components CPS and CP

- Certification Practice Statement may specify:
  - ID checks performed before certificate issue
  - physical, personnel and procedural security controls for the CA
  - technical and key pair protection and management controls
  - certificate revocation management procedures
  - audit procedures for the CA
  - accreditation information
  - legal and privacy issues and liability limitations
  - profiles and descriptions covering issued certificates including naming conventions

PKI components

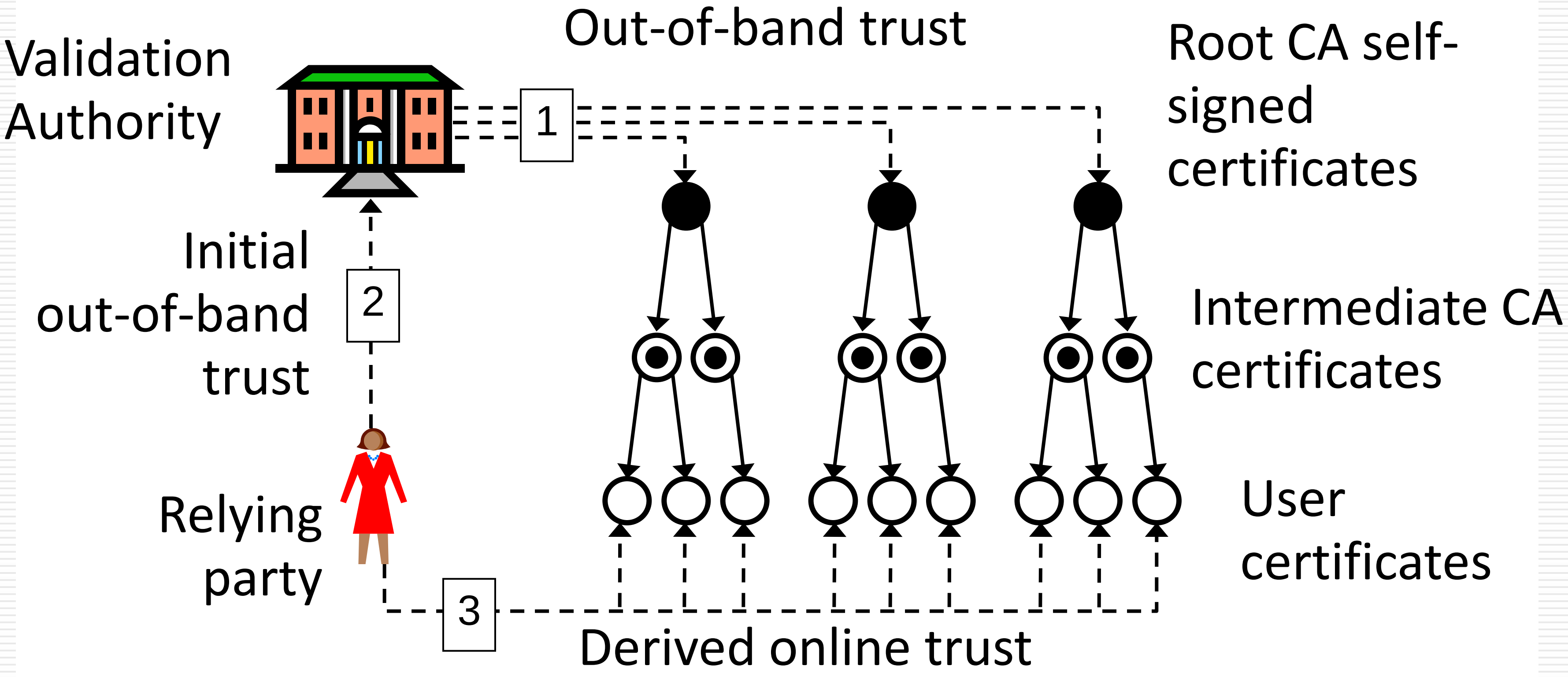
- Registration Authorities
  - – act as interface between user and CA
    - collects details from certificate requesters
    - authenticates identity, and
    - submits certificate request to CA
- Use of RA is optional, CA *may* employ an RA
    - can provide cost benefits for CA
    - quality of RA determines level of trust in issued certificates

PKI components
Validation Authority

- **Q: What if the relying party needs to validate a certificate but can't establish a trust path to it?**
- **A:** The relying party can use a validation service from a VA (Validation Authority) to validate the certificate
  - The relying party needs to trust the VA
  - The relying party needs an authentic copy of the VA's public key, e.g. received securely out-of-band
- The VAs must establish trust relationships and get authentic public keys from Root CAs of many different PKIs

PKI Components

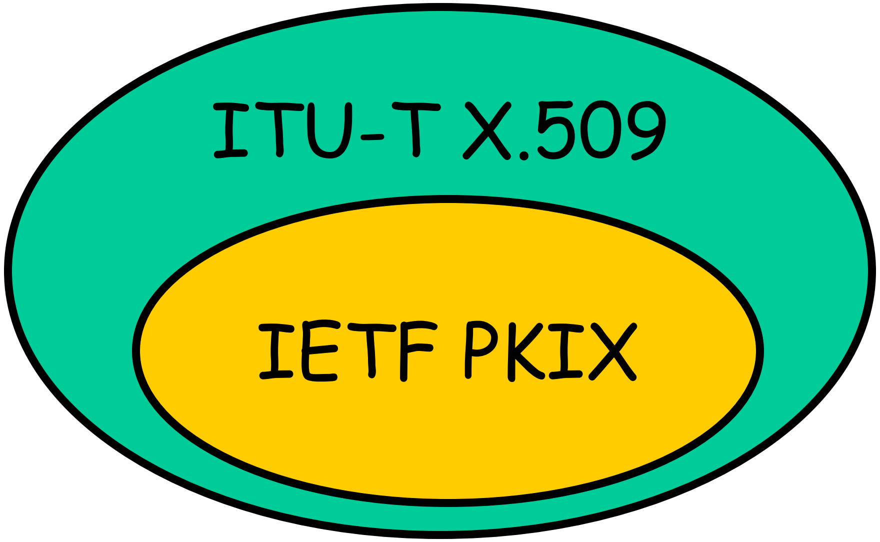
Validation Authorities



PKI components: CRL

- 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

Standardisation Activities



- X.509 specifies the the overall structure of certificates
- IETF has specified a series of standards for implementing PKIs based on X.509 certificates

PKI services

- Several organisations operate PKI services
  - Private sector
  - Public sector
  - Military sector
- Mutual recognition and cross certification between PKIs is difficult
- Can be 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

PKI Summary

- Public key cryptography needs a PKI to work
  - – Digital certificates used to provide 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