IN2120 Information Security
University of Oslo
Autumn 2018

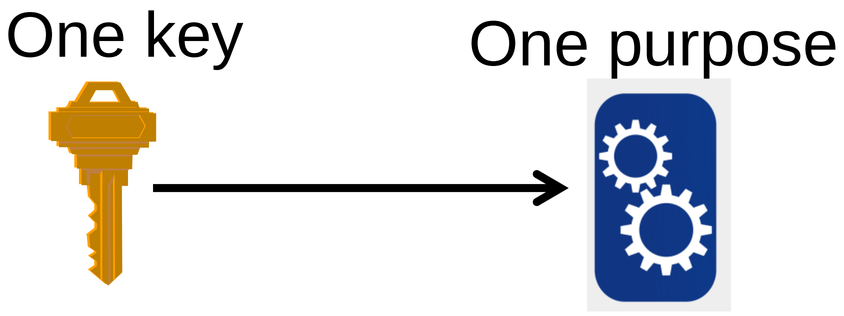
Lecture 3

Key Management and PKI

Nils Gruschka

Key Management

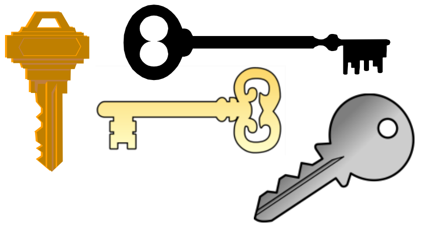
- The strength of cryptographic security depends on:
  1. The size of the keys
  2. The robustness of cryptographic algorithms/protocols
  3. The protection and management afforded to the keys
- Key management provides the foundation for the secure generation, storage, distribution, and destruction of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

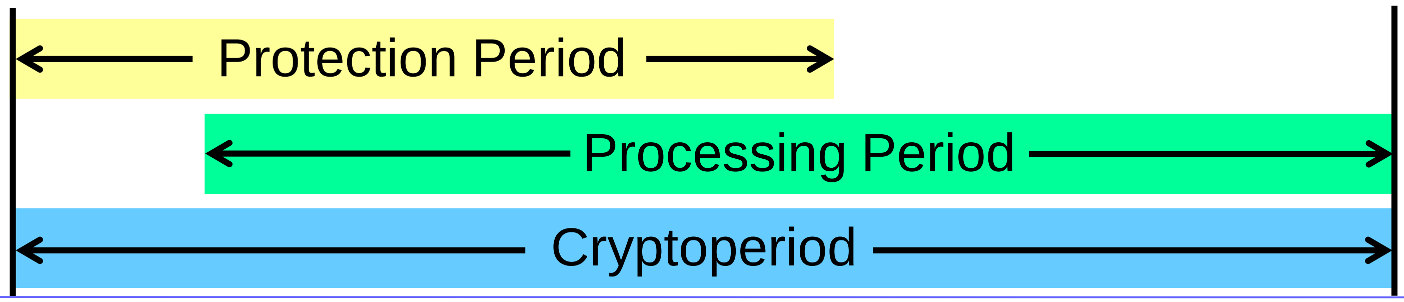
  http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

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.

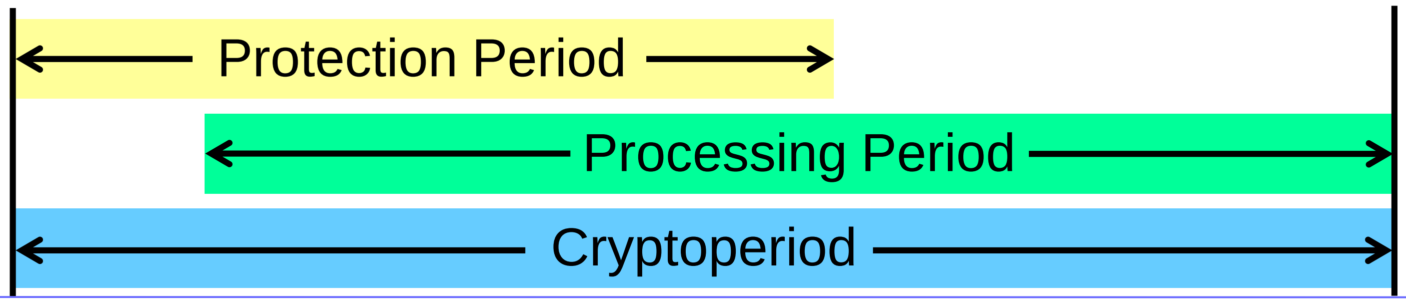
Crypto Periods

- A key can be used for <u>protection </u>and/or <u>processing</u>.
- Protection:
  - Key is used to encrypt (asymmetric or symmetric)
  - Key is used to generate a digital signature (asymmetric)
- Processing:
  - Key is used to decrypt (asymmetric or symmetric)
  - Key is used to verify a digital signature (asymmetric)
- Time frame in which key usage is allowed/recommended:
  - Protection Period (Originator-Usage Period)
  - Processing Period (Recipient-Usage Period)

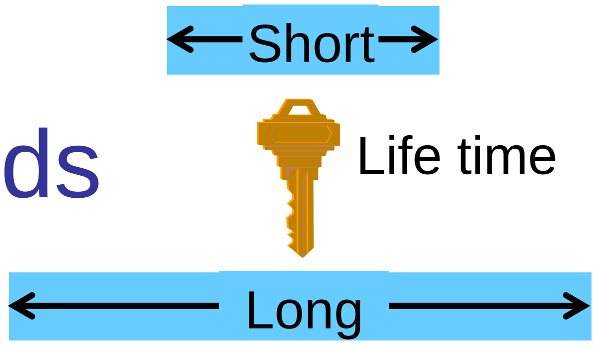


Crypto Periods

- The processing period can continue after the protection period.
- The **crypto-period** lasts from the beginning of the protection period to the end of the processing period.



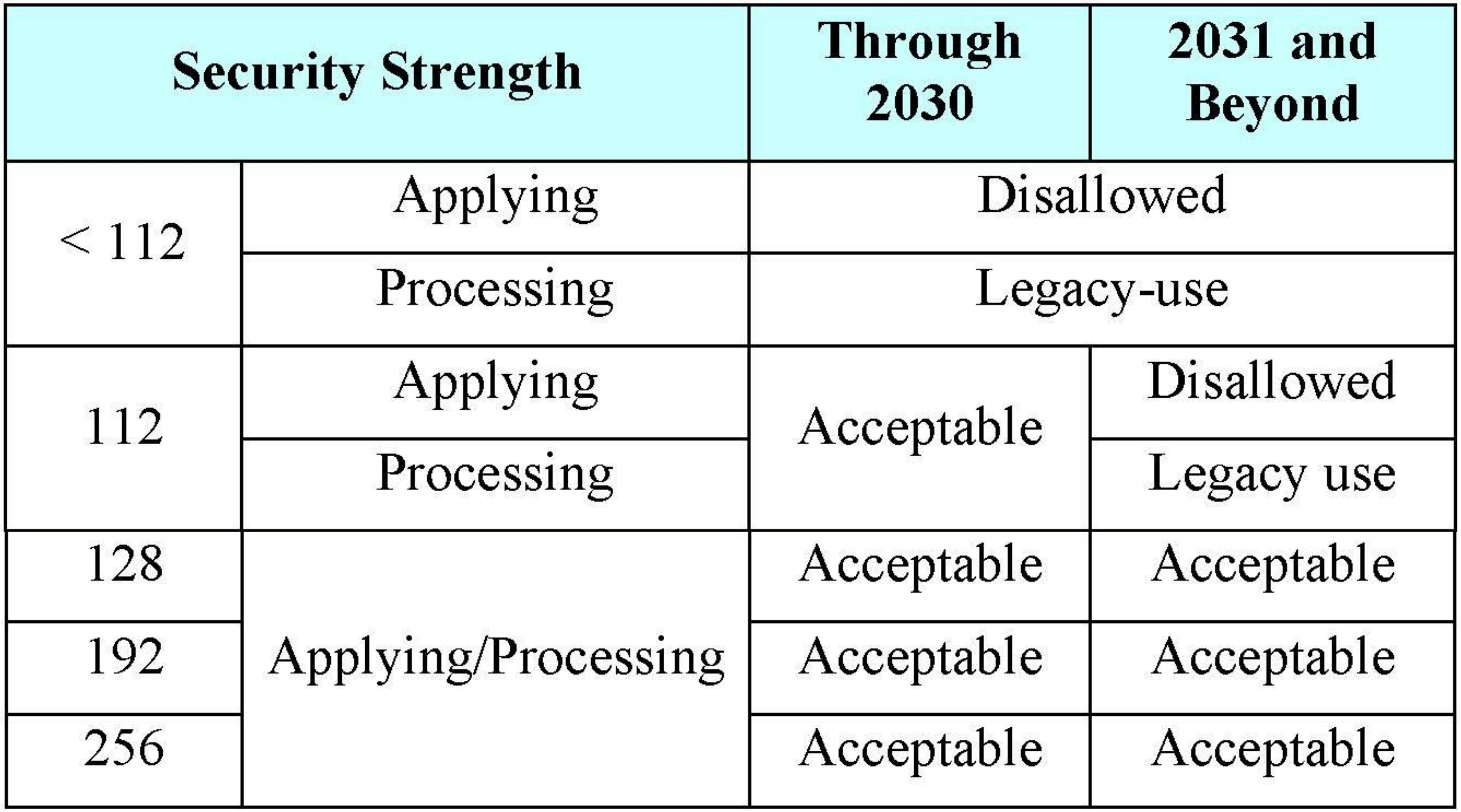
Factors Affecting Crypto-Periods



- 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

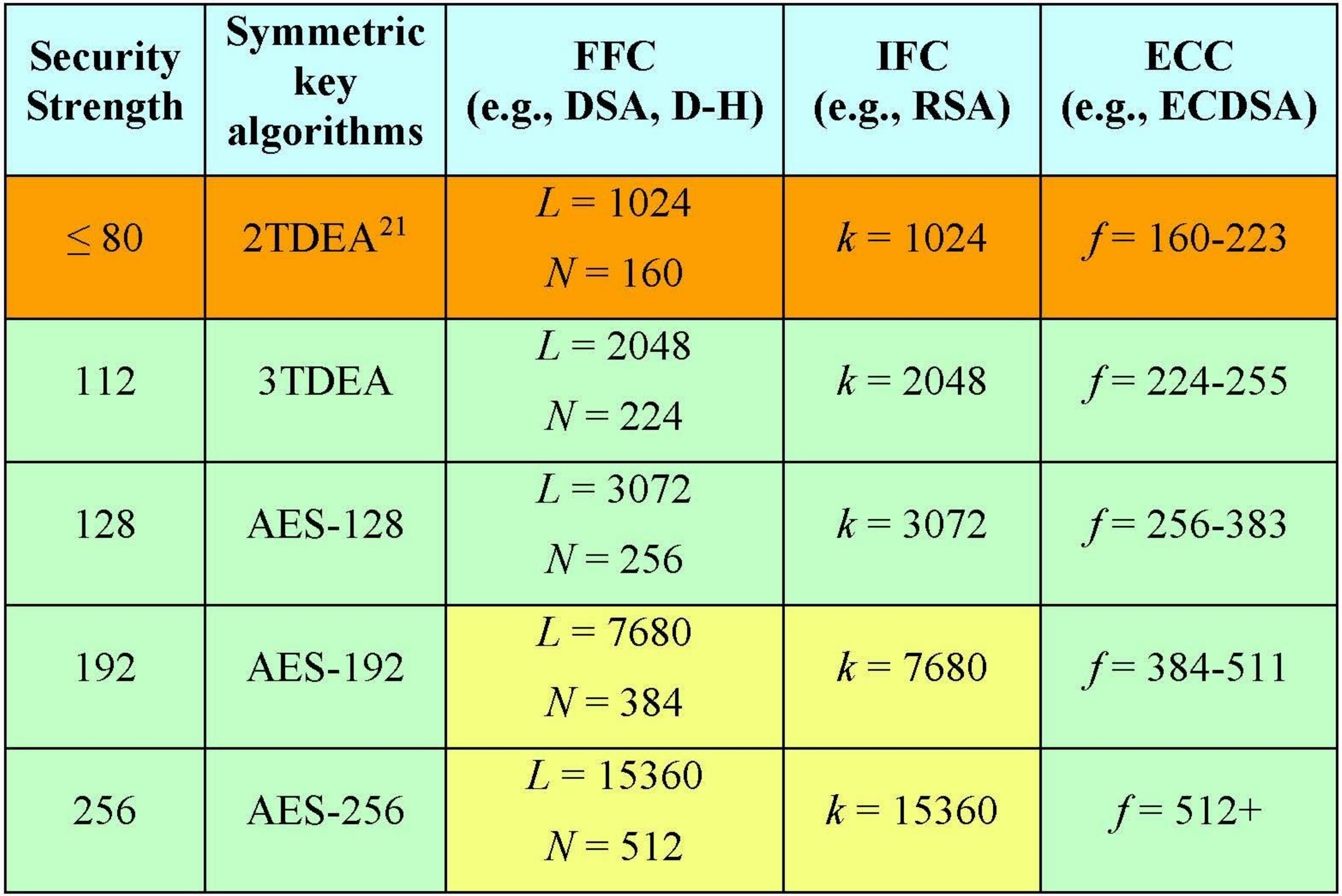


Security Strength		Through 2030	2031 and Beyond
< 112	Applying	Disallowed	
	Processing	Legacy-use	
112	Applying	Acceptable	Disallowed
	Processing		Legacy use
128	Applying/Processing	Acceptable	Acceptable
192		Acceptable	Acceptable
256		Acceptable	Acceptable

Key strength comparison (ignoring QC)

Ref: NIST SP 800-57

Finite Field Cryptography | Integer Factorization Cryptography | Elliptic Curve Cryptography



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

Towards a Catastrophic Crypto Collapse

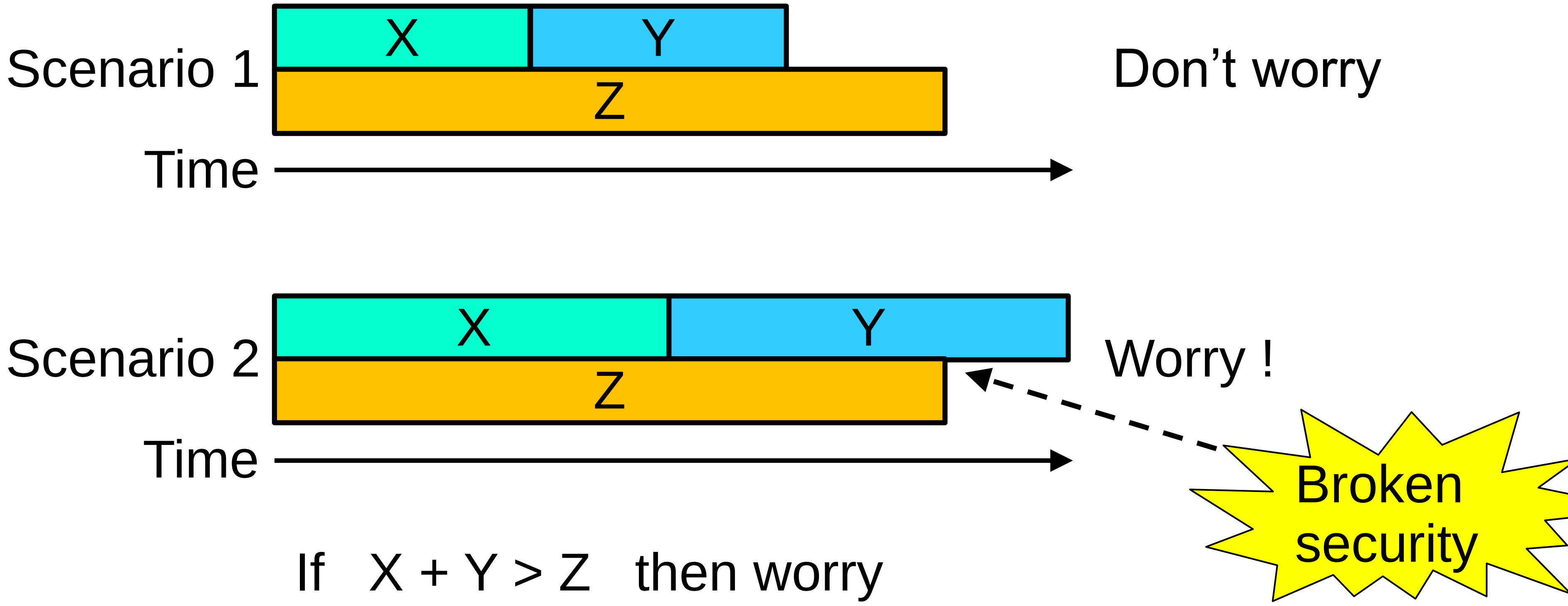
- NIST (US National Institute of Standards and Technology) expects practical quantum computers to be built in the 2020s
- Impact on public-key crypto:
  - ~~RSA~~
  - ~~Elliptic Curve Cryptography (ECDSA)~~
  - ~~Finite Field Cryptography (DSA)~~
  - ~~Diffie-Hellman key exchange~~
- Impact on symmetric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



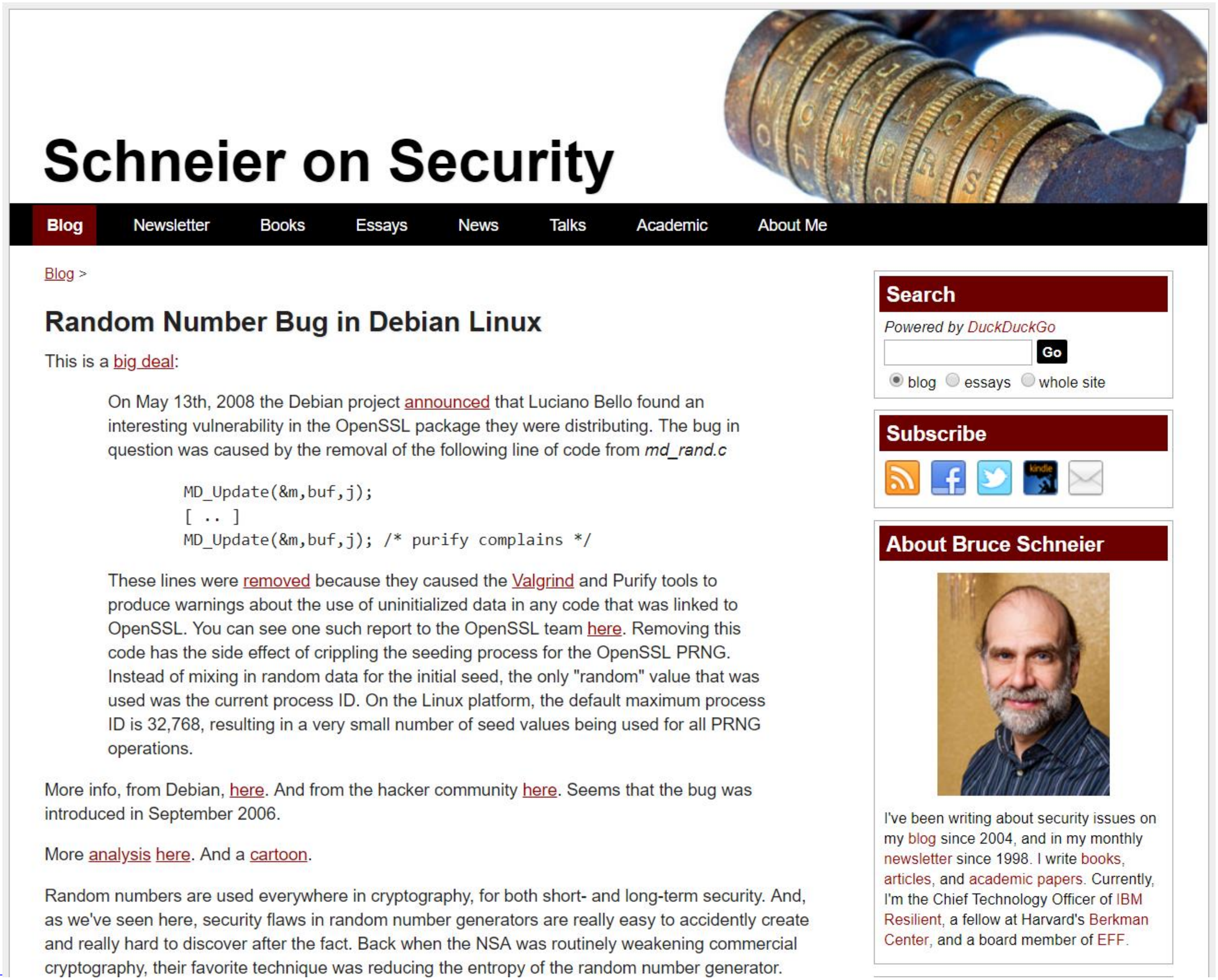
If X + Y > Z then worry

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.



Blog >

Random Number Bug in Debian Linux

This is a big deal:

> On May 13th, 2008 the Debian project announced that Luciano Bello found an interesting vulnerability in the OpenSSL package they were distributing. The bug in question was caused by the removal of the following line of code from *md_rand.c*
>
> ```
> MD_Update(&m,buf,j);
> [ .. ]
> MD_Update(&m,buf,j); /* purify complains */
> ```
>
> These lines were removed because they caused the Valgrind and Purify tools to produce warnings about the use of uninitialized data in any code that was linked to OpenSSL. You can see one such report to the OpenSSL team here. Removing this code has the side effect of crippling the seeding process for the OpenSSL PRNG. Instead of mixing in random data for the initial seed, the only "random" value that was used was the current process ID. On the Linux platform, the default maximum process ID is 32,768, resulting in a very small number of seed values being used for all PRNG operations.

More info, from Debian, here. And from the hacker community here. Seems that the bug was introduced in September 2006.

More analysis here. And a cartoon.

Random numbers are used everywhere in cryptography, for both short- and long-term security. And, as we've seen here, security flaws in random number generators are really easy to accidently create and really hard to discover after the fact. Back when the NSA was routinely weakening commercial cryptography, their favorite technique was reducing the entropy of the random number generator.

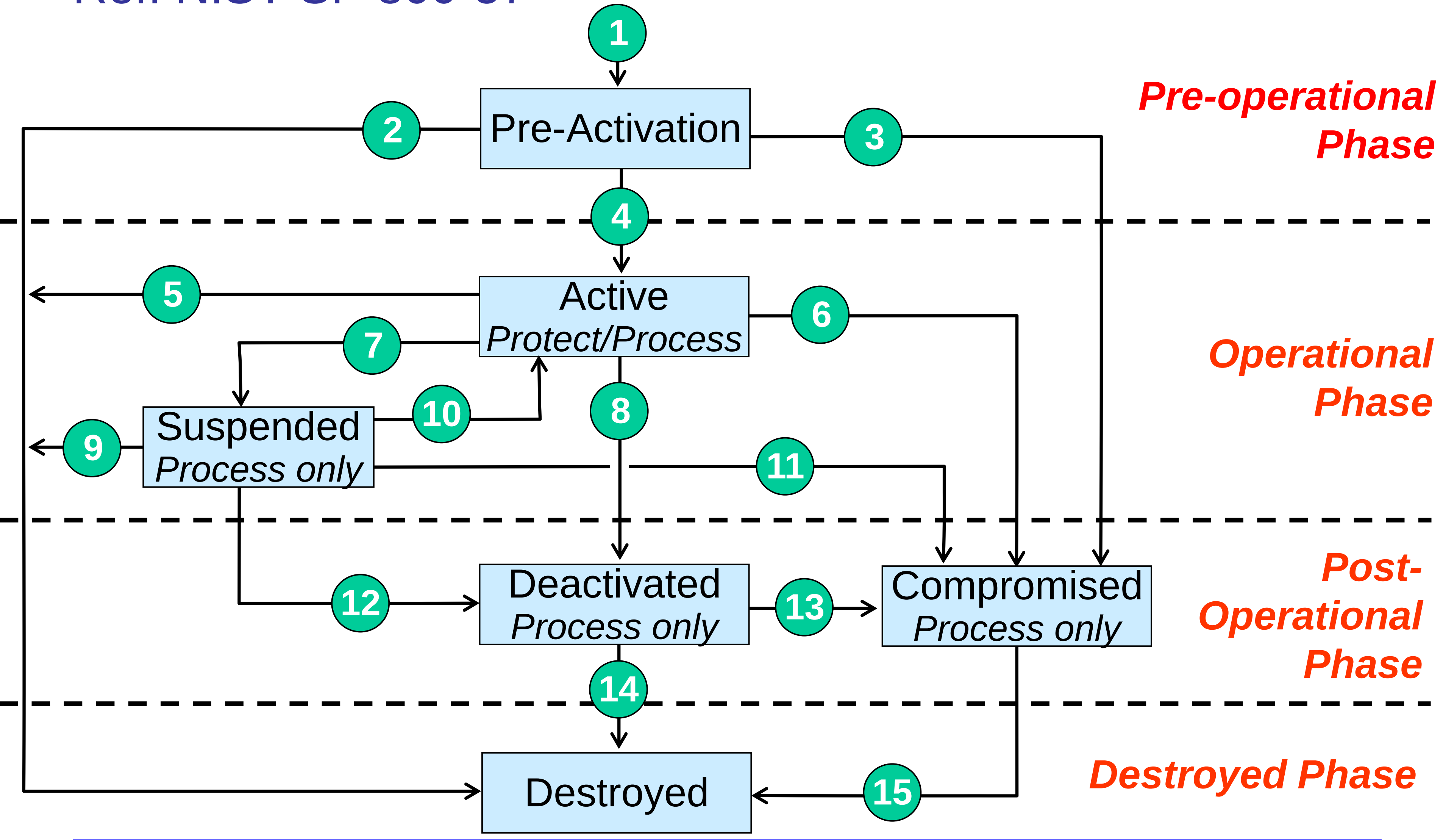
I've been writing about security issues on my blog since 2004, and in my monthly newsletter since 1998. I write books, articles, and academic papers. Currently, I'm the Chief Technology Officer of IBM Resilient, a fellow at Harvard's Berkman Center, and a board member of EFF.

Compromise of keys and keying material

- Key compromise occurs when it is known or suspected that an unauthorized entity has obtained a secret/private key.
- When a key is compromised, immediately stop using the secret/public key for <u>protection</u>, and revoke the compromised key (pair).
- A compromised key **might** be used for continued <u>processing</u> 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.
- The worst form of key compromise is when a key is compromised without detection.

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 (see above “Master Key”)
  - Public keys need integrity/authenticity protection (see next section “PKI”)

Key destruction

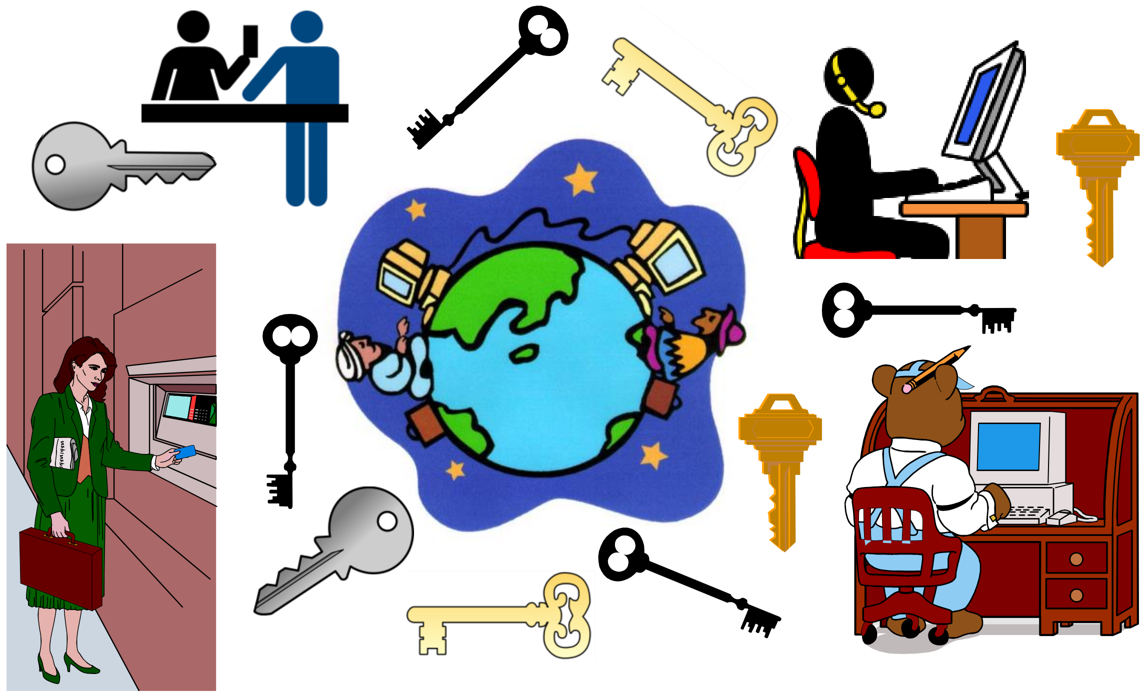
- No key material should reside in volatile memory or on permanent storage media after destruction
- Key destruction methods, e.g.
  - Simple delete operation on computer
    - may leave undeleted key e.g. in recycle bin or on disk sectors
  - Special delete operation on computer
    - that leaves no 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

Public Key Infrastructure

PKI

Why the interest in PKI ?

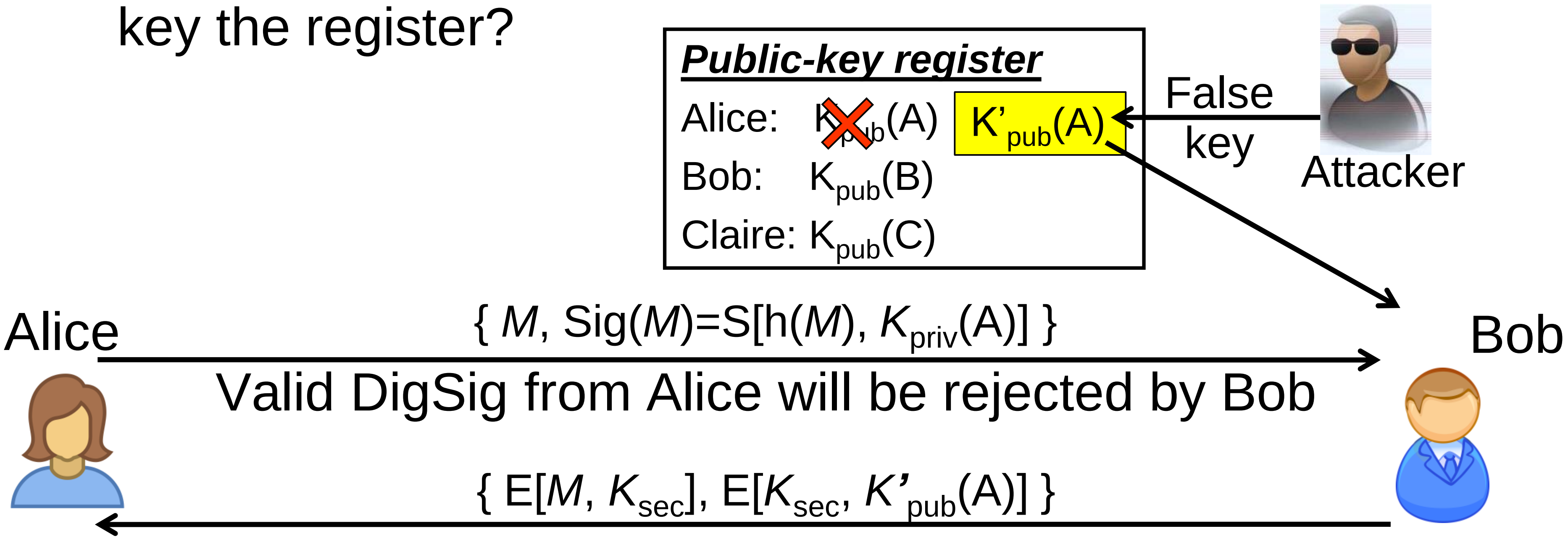
Cryptography solves security problems in open networks,
… but creates key distribution challenges.



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

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: $\cancel{K_{pub}}(A)$ $K'_{pub}(A)$
Bob: $K_{pub}(B)$
Claire: $K_{pub}(C)$
False key
Attacker
Alice
Bob
$\{ M, \text{Sig}(M)=\text{S}[\text{h}(M), K_{priv}(A)] \}$
$\{ \text{E}[M, K_{sec}], \text{E}[K_{sec}, K'_{pub}(A)] \}$

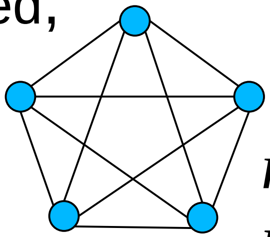
Valid DigSig from Alice will be rejected by Bob

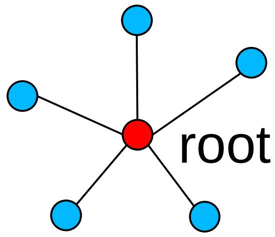
Confidential message to Alice can not be read by Alice, but can be read by the attacker

- Broken public-key authenticity breaks security assumptions

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**,**
    - $n(n\text{-}1)/2$ distributions, quadratic growth
    - Impractical in open networks
  - Asymmetric public keys: **Authenticity** required**,**
    - $n(n\text{-}1)/2$ 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



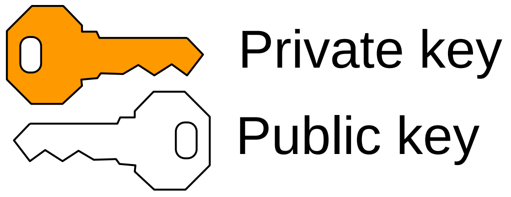


Public-key infrastructure

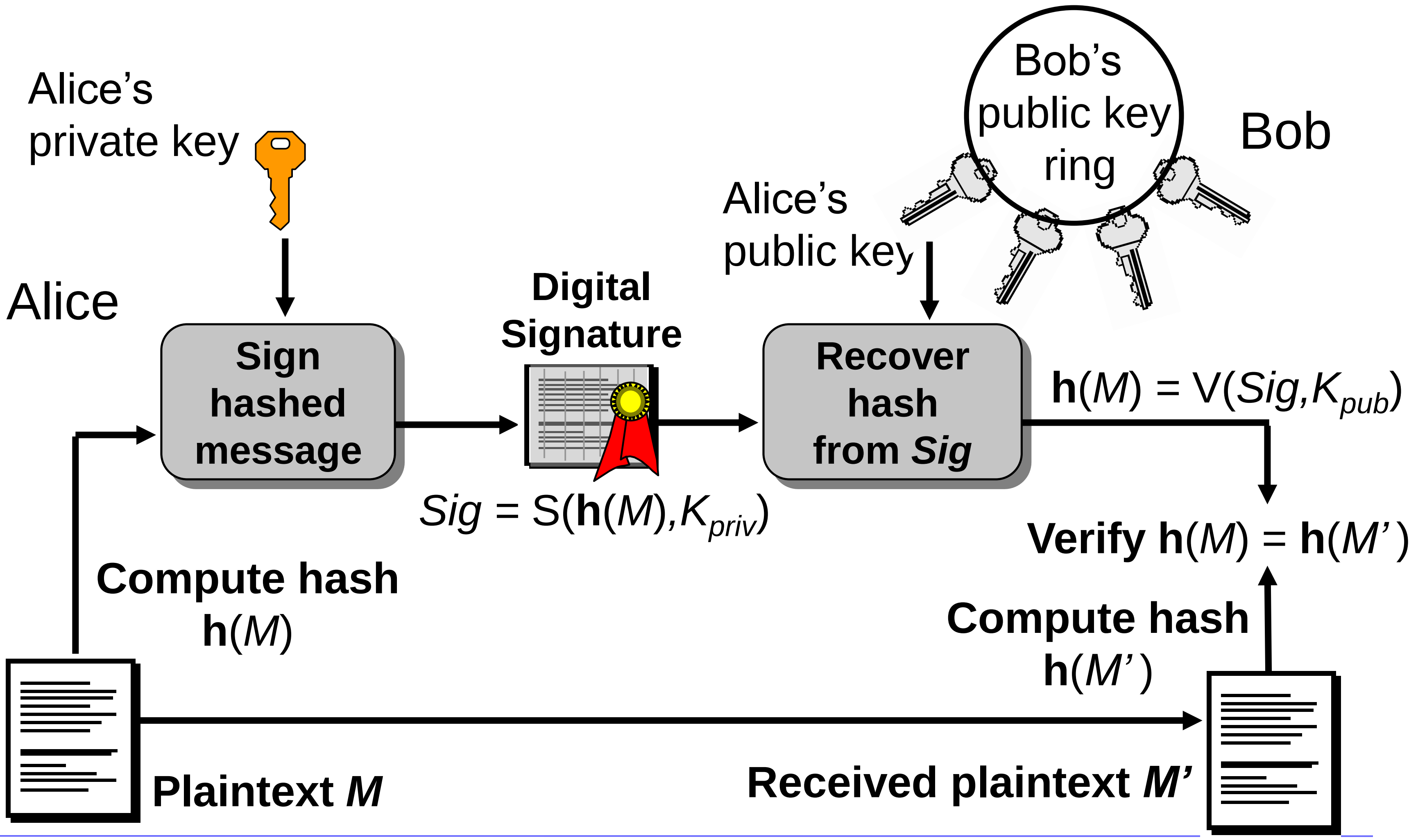
- Due to spoofing problem, public keys must be digitally signed before distribution.
- The main purpose of a PKI is to ensure authenticity of public keys.
- PKI consists of:
  - **Policies** (to define the rules for managing certificates)
  - **Technologies** (to implement the policies and generate,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

Recapitulation: Digital Signature

- 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)

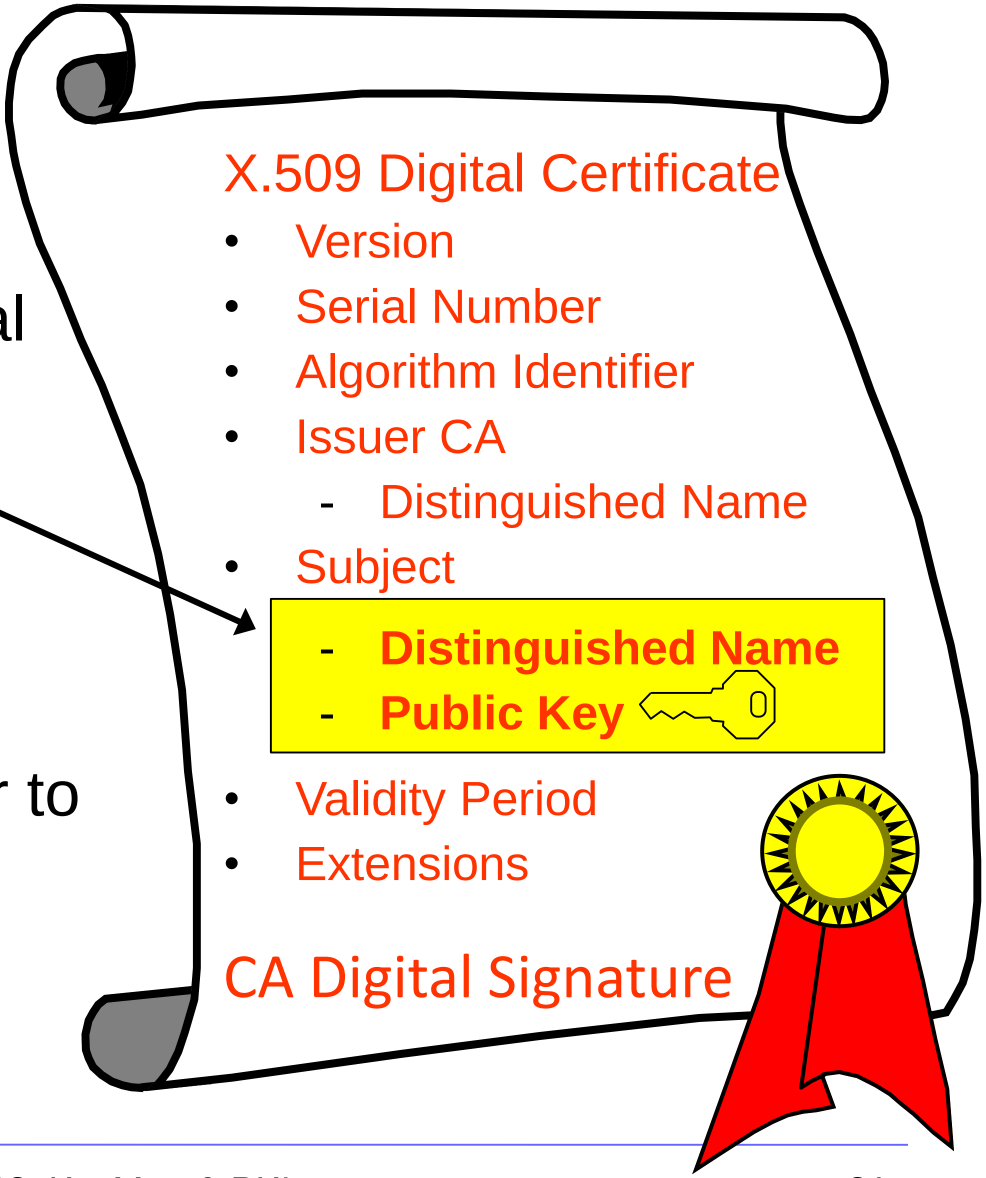


Recapitulation: Digital signature

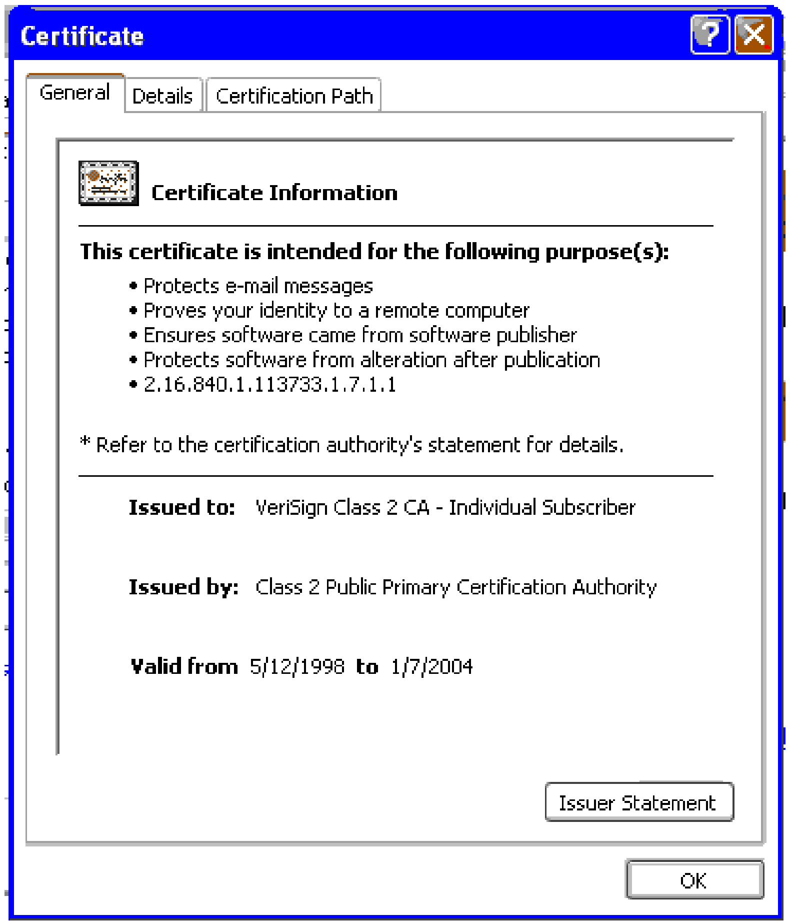

Alice's private key
Alice
Sign hashed message
Digital Signature
$Sig = S(\mathbf{h}(M), K_{priv})$
Alice's public key
Bob's public key ring
Bob
Recover hash from *Sig*
$\mathbf{h}(M) = V(Sig, K_{pub})$
Verify $\mathbf{h}(M) = \mathbf{h}(M')$
Compute hash $\mathbf{h}(M)$
Compute hash $\mathbf{h}(M')$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

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



Certificate

General | Details | Certification Path

Certificate Information

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

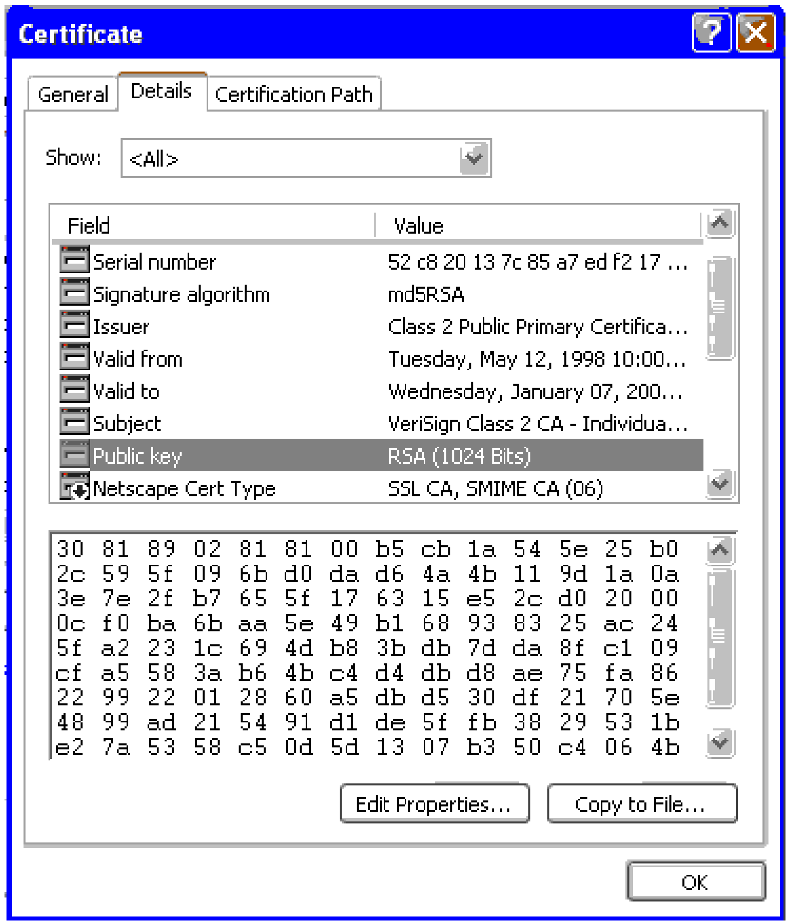
- Protects e-mail messages
- Proves your identity to a remote computer
- Ensures software came from software publisher
- Protects software from alteration after publication
- 2.16.840.1.113733.1.7.1.1

* 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



Certificate

General | Details | Certification Path

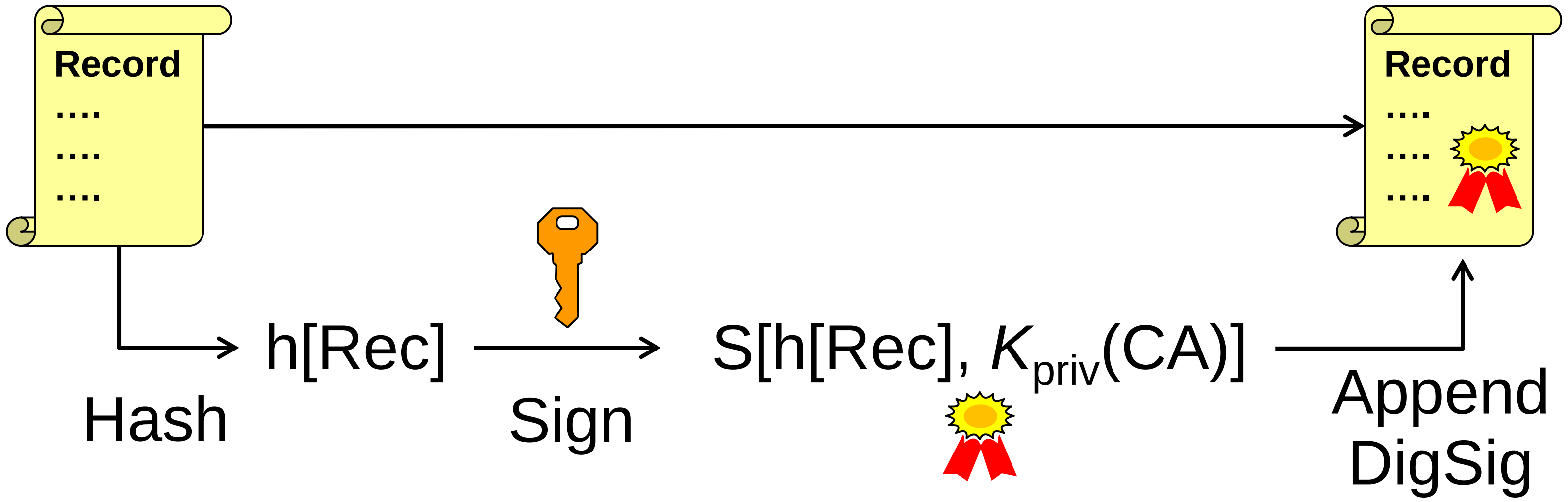
Show: <All>

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)

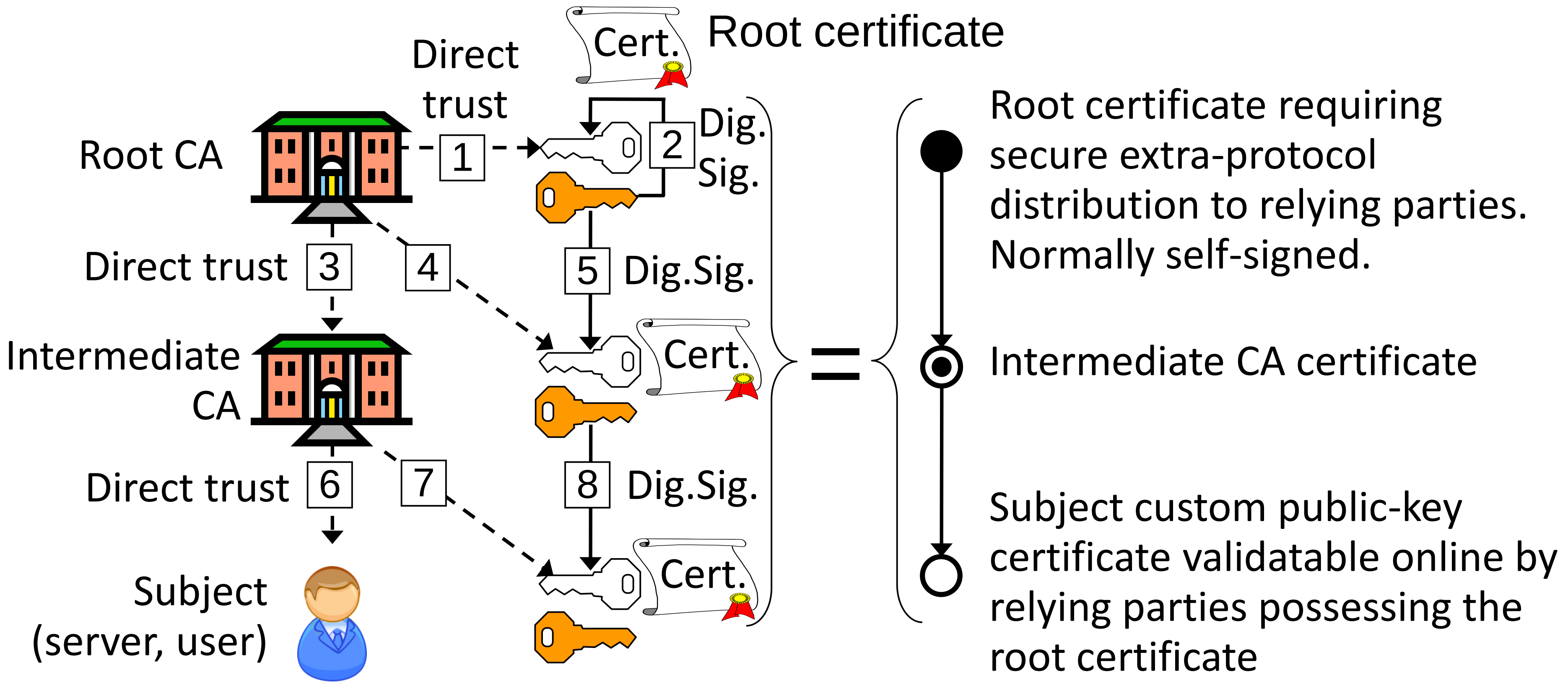
```
30 81 89 02 81 81 00 b5 cb 1a 54 5e 25 b0
2c 59 5f 09 6b d0 da d6 4a 4b 11 9d 1a 0a
3e 7e 2f b7 65 5f 17 63 15 e5 2c d0 20 00
0c f0 ba 6b aa 5e 49 b1 68 93 83 25 ac 24
5f a2 23 1c 69 4d b8 3b db 7d da 8f c1 09
cf a5 58 3a b6 4b c4 d4 db d8 ae 75 fa 86
22 99 22 01 28 60 a5 db d5 30 df 21 70 5e
48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

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

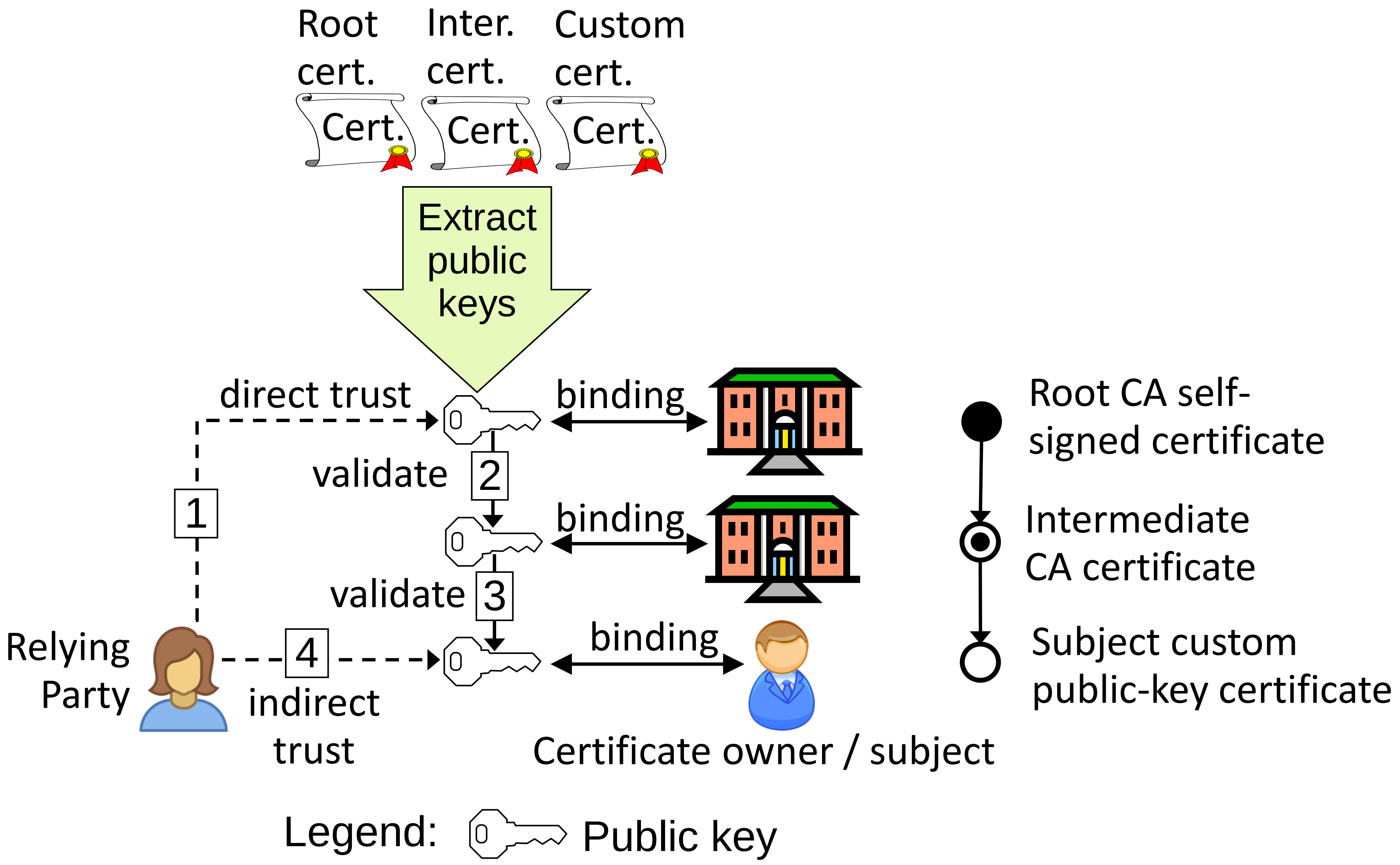


PKI certificate generation


Root certificate
Cert.
Direct trust
Root CA
1
2
Dig. Sig.
Direct trust
3
4
5
Dig.Sig.
Intermediate CA
Cert.
Direct trust
6
7
8
Dig.Sig.
Cert.
Subject (server, user)
=
Root certificate requiring secure extra-protocol distribution to relying parties. Normally self-signed.
Intermediate CA certificate
Subject custom public-key certificate validatable online by relying parties possessing the root certificate

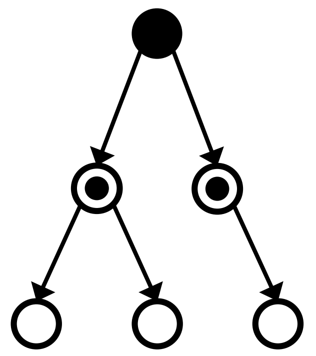
Legend: Public key Private key

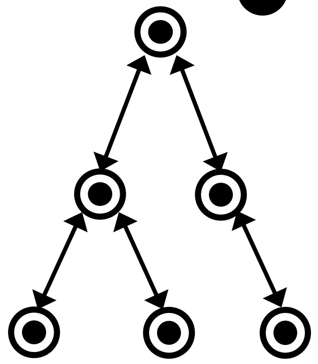
Certificate and public key validation

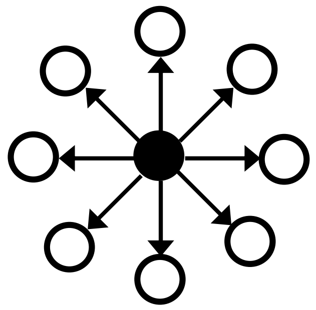


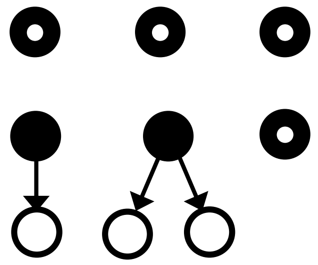
PKI Trust Models

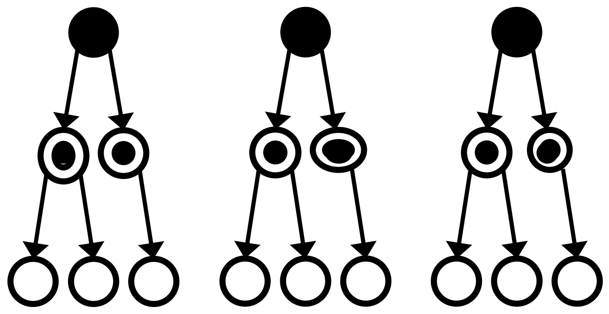
Legend:
- ● Self-signed root CA certificate
- ◉ CA-signed intermediate CA certificate
- ○ CA-signed custom (leaf) certificate (cannot sign)
- ⦿ Self-signed custom certificate

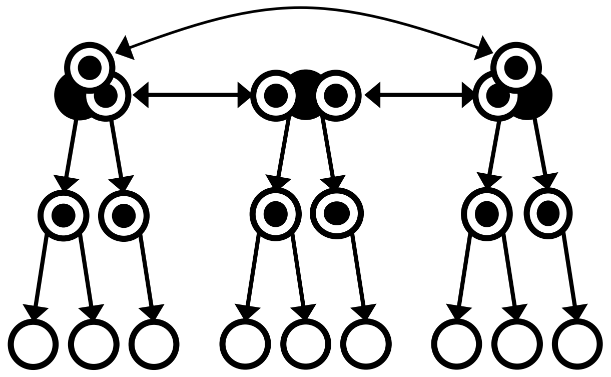

Strict hierarchy
e.g. ` DNSSEC PKI ’

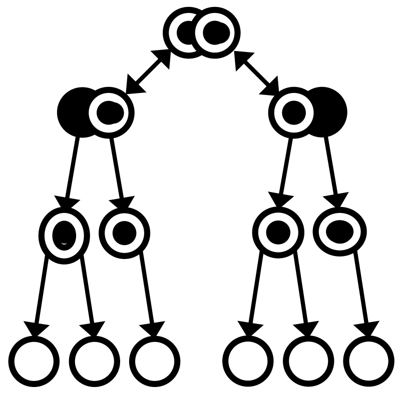

Bi-directional
hierarchy


User-centric PKI
(local view)


Unstructured PKI


Isolated strict hierarchies
e.g. ` Browser PKI ’


Mesh PKI
Cross-certified strict hierarchies


PKIs with Bridge CA

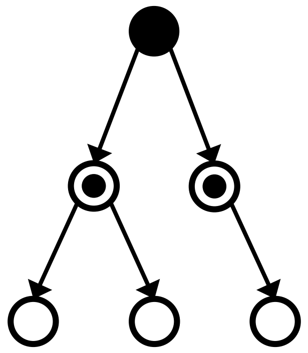
Meaning of Trust for PKI

- **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 from a honest + 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.

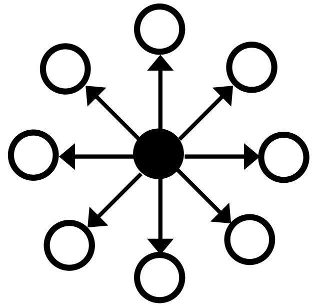
PKI trust models
Strict hierarchical model



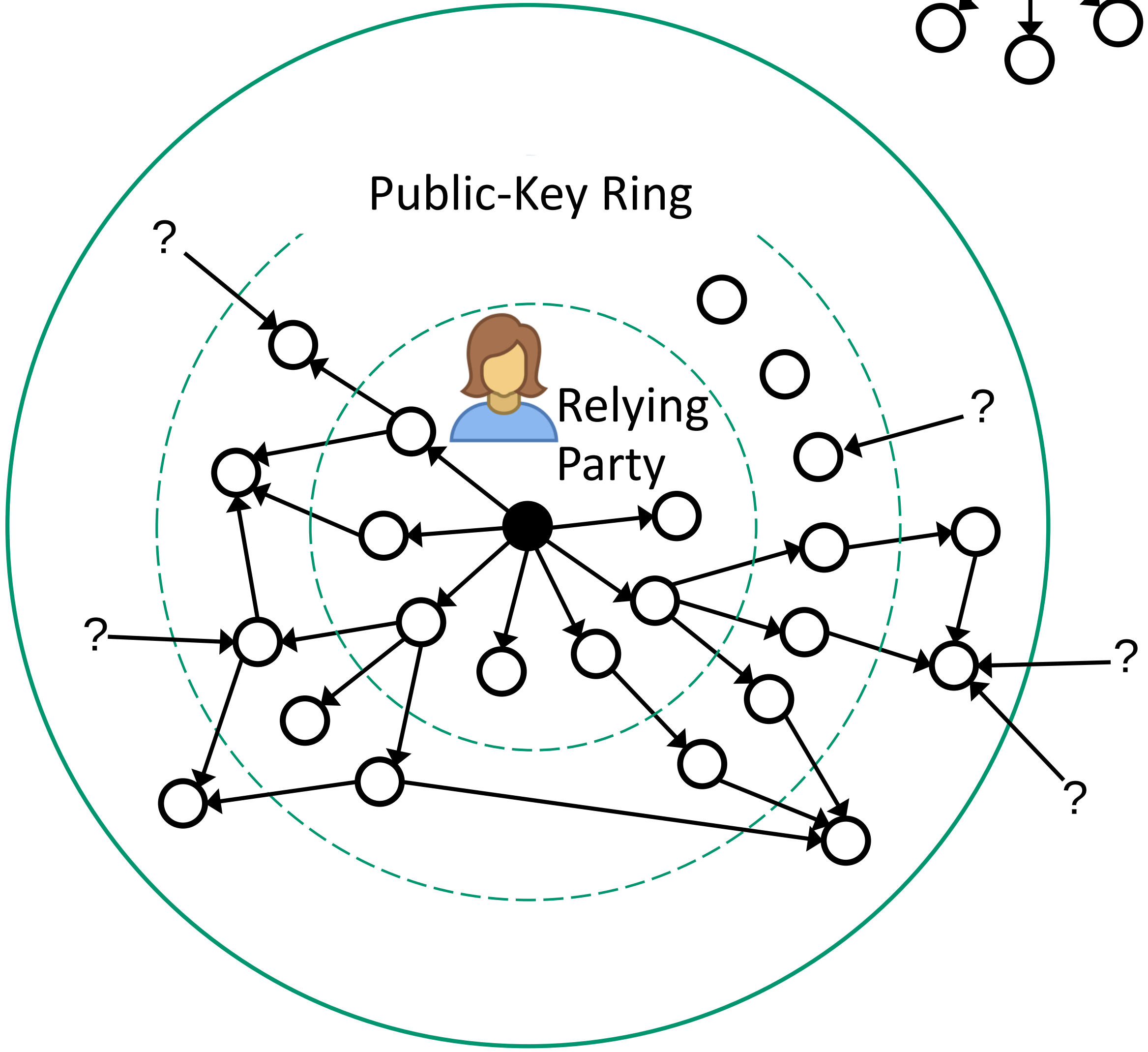
- Advantages:
  - works well in highly-structured setting such as military and government
  - unique certification path between two entities (so finding certification paths is trivial)
  - scales well to larger systems
- Disadvantages:
  - need a trusted third party (root CA)
  - ‘single point-of-failure’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

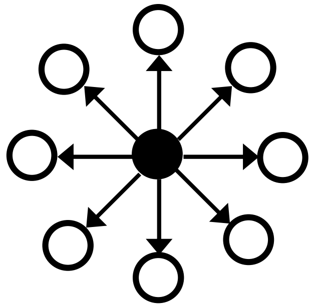
User-centric model, as in PGP



- 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.

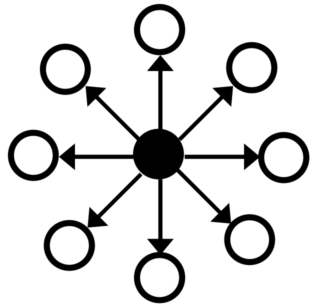


PKI trust models
User-centric model



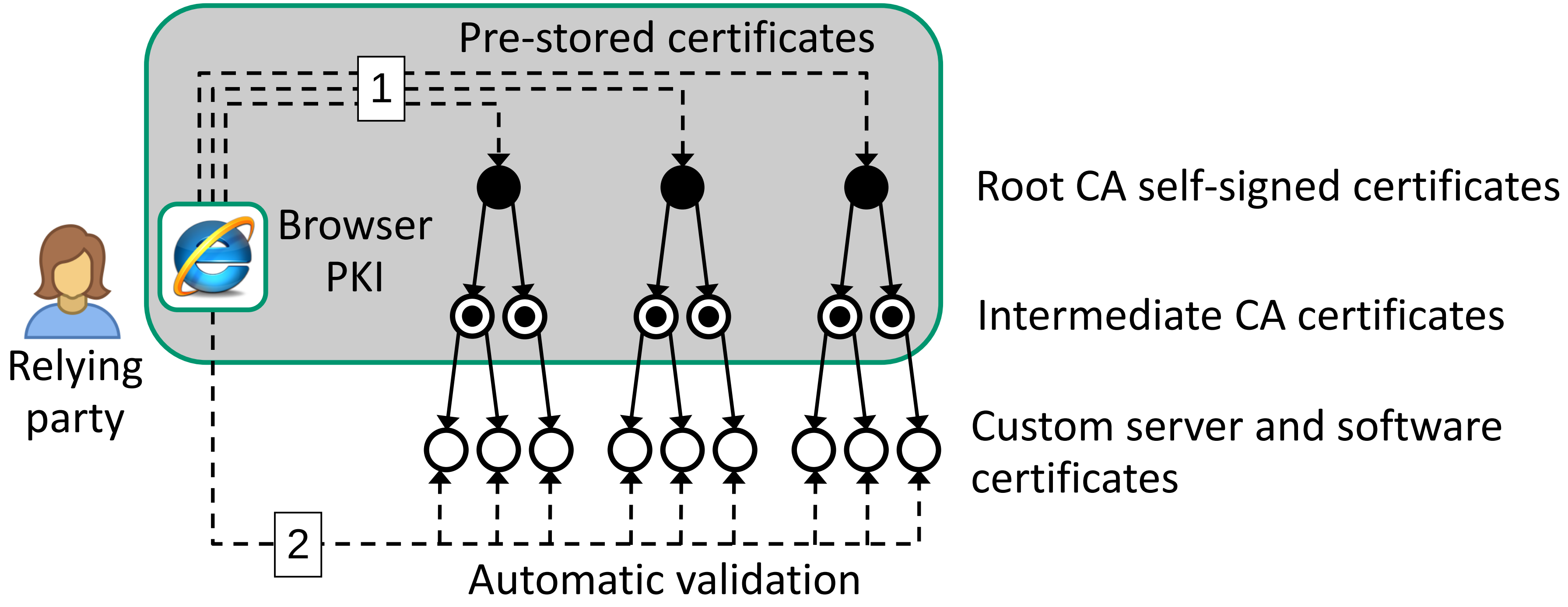
- Each user is **completely responsible** for deciding which public keys to trust
- Example: *Pretty Good Privacy (PGP)*
  - ‘Web of Trust’
  - Each user may act as a CA, signing public keys that they will trust
  - Public keys can be distributed by key servers and verified by fingerprints
  - OpenPGP 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)


Pre-stored certificates
1
Browser PKI
Root CA self-signed certificates
Intermediate CA certificates
Custom server and software certificates
Relying party
2
Automatic validation

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

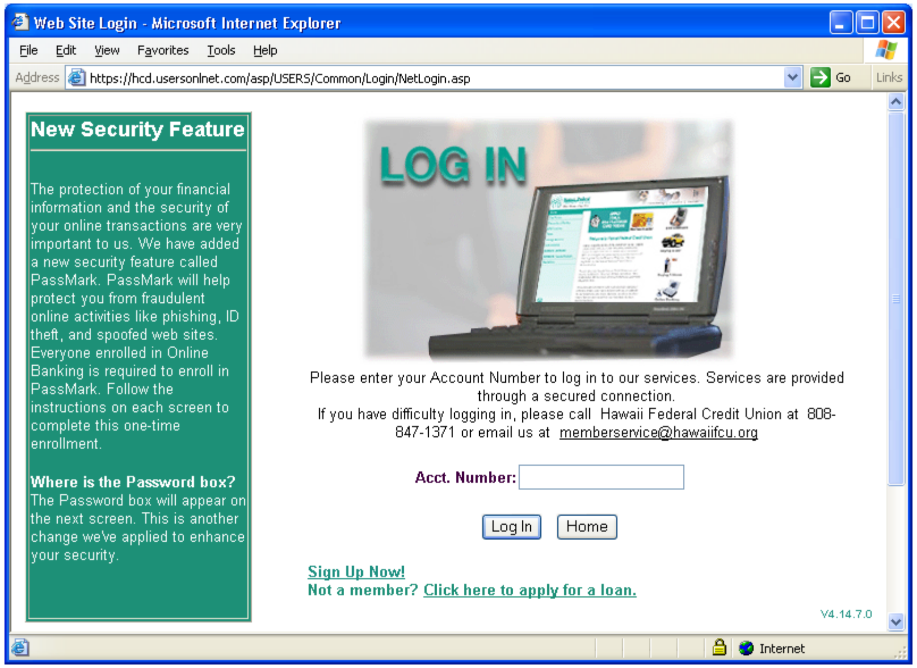
Browser PKI and malicious certificates

- 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
  - <u>Malicious sites can have legitimate certificates !!!</u>
- 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 and OS vendors 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


Web Site Login - Microsoft Internet Explorer

Address: https://hcd.usersonlnet.com/asp/USERS/Common/Login/NetLogin.asp

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.

LOG IN

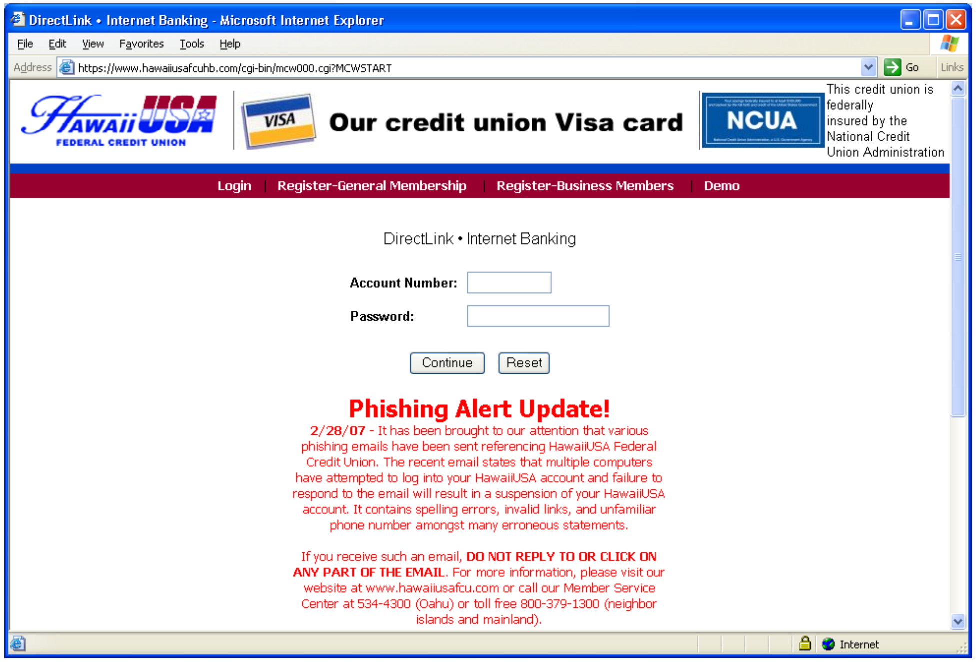
Please enter your Account Number to log in to our services. Services are provided through a secured connection.
If you have difficulty logging in, please call Hawaii Federal Credit Union at 808-847-1371 or email us at memberservice@hawaiifcu.org

Acct. Number:

Log In | Home

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

V4.14.7.0


DirectLink • Internet Banking - Microsoft Internet Explorer

Address: https://www.hawaiiusafcuhb.com/cgi-bin/mcw000.cgi?MCWSTART

HawaiiUSA FEDERAL CREDIT UNION | Our credit union Visa card | NCUA | This credit union is federally insured by the National Credit Union Administration

Login | Register-General Membership | Register-Business Members | Demo

DirectLink • Internet Banking

Account Number:

Password:

Continue | Reset

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).

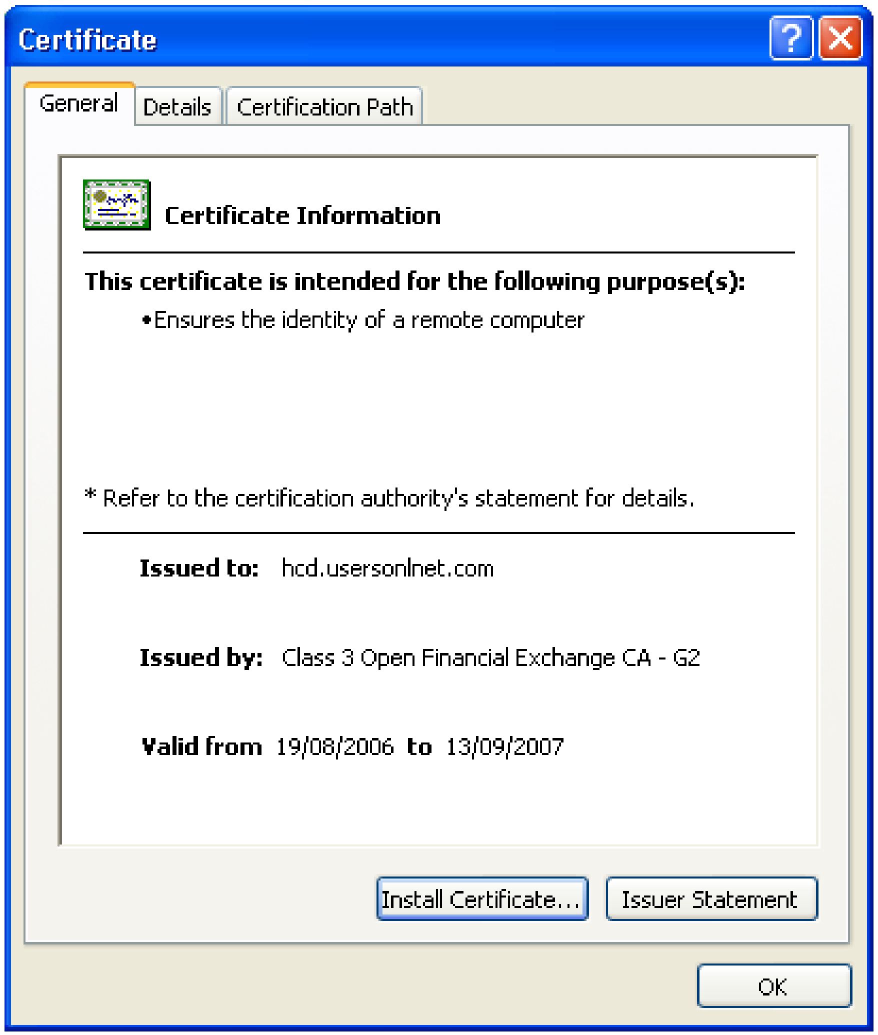
Genuine bank login

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



Certificate Information

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

- Ensures the identity of a remote computer

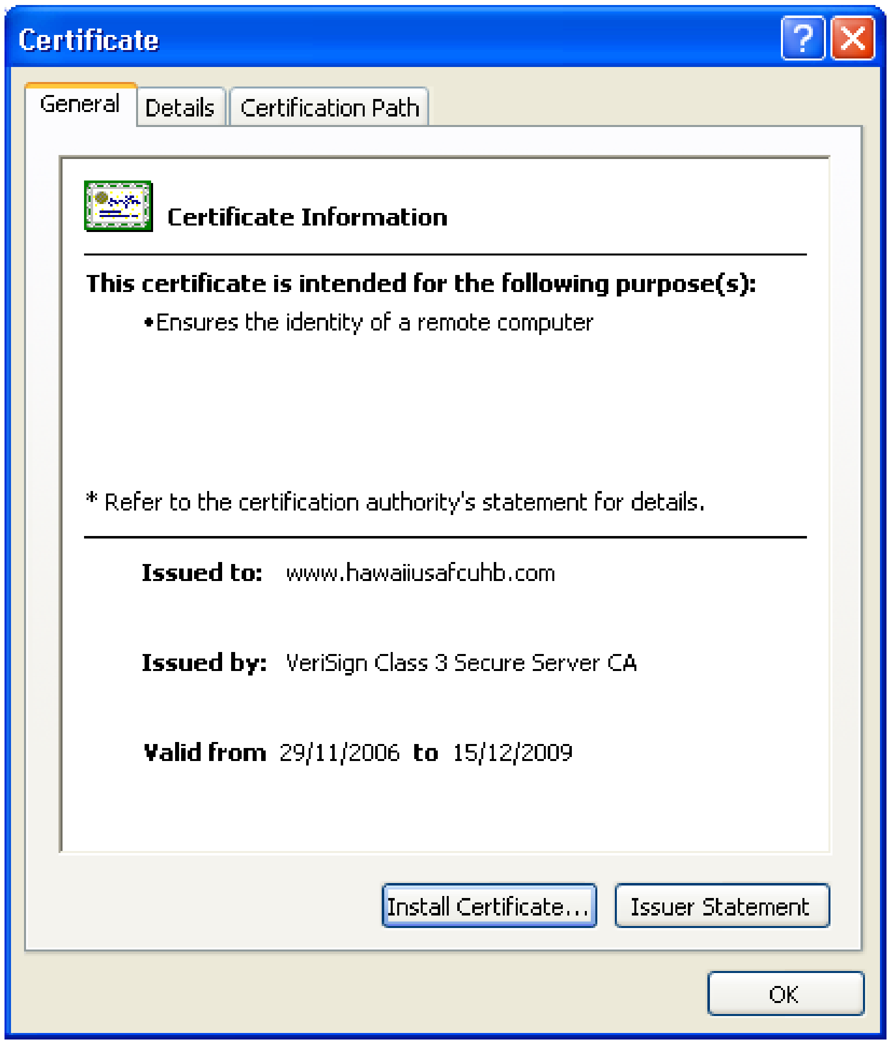
* Refer to the certification authority's statement for details.

Issued to: hcd.usersonlnet.com

Issued by: Class 3 Open Financial Exchange CA - G2

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

Genuine 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.

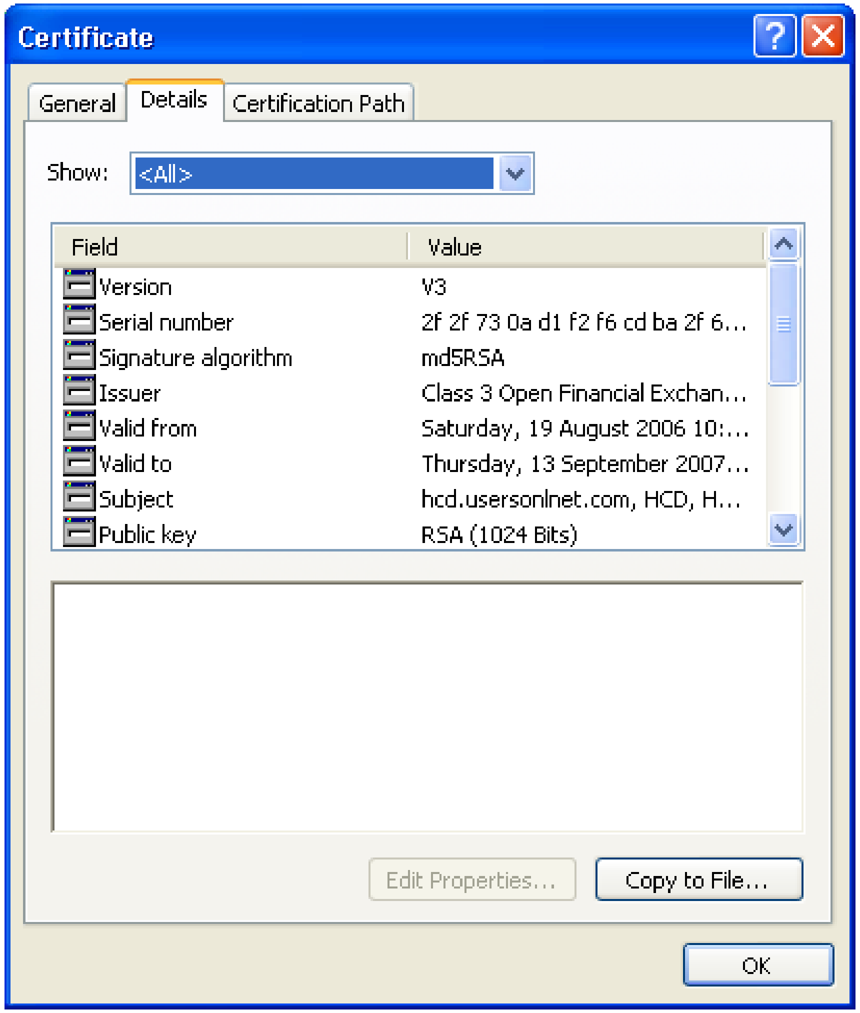
Issued to: www.hawaiiusafcuhb.com

Issued by: VeriSign Class 3 Secure Server CA

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

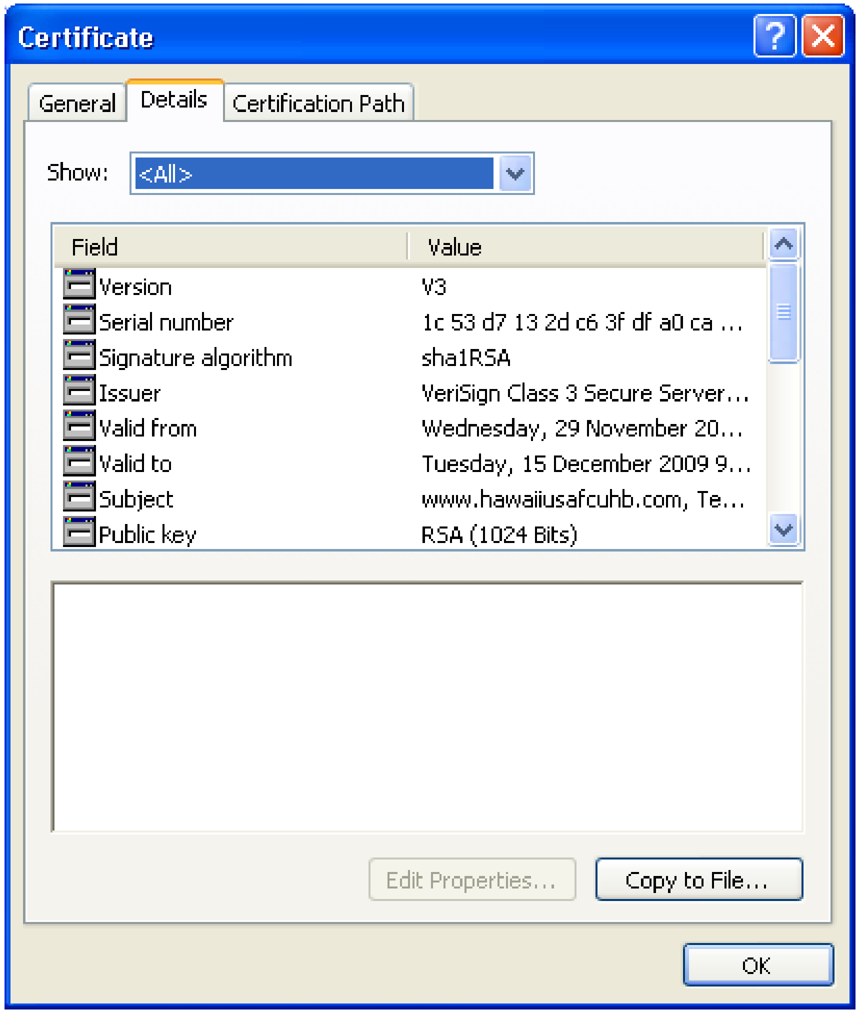
“Fake” certificate

Certificate comparison 2



Field	Value
Version	V3
Serial number	2f 2f 73 0a d1 f2 f6 cd ba 2f 6...
Signature algorithm	md5RSA
Issuer	Class 3 Open Financial Exchan...
Valid from	Saturday, 19 August 2006 10:...
Valid to	Thursday, 13 September 2007...
Subject	hcd.usersonlnet.com, HCD, H...
Public key	RSA (1024 Bits)

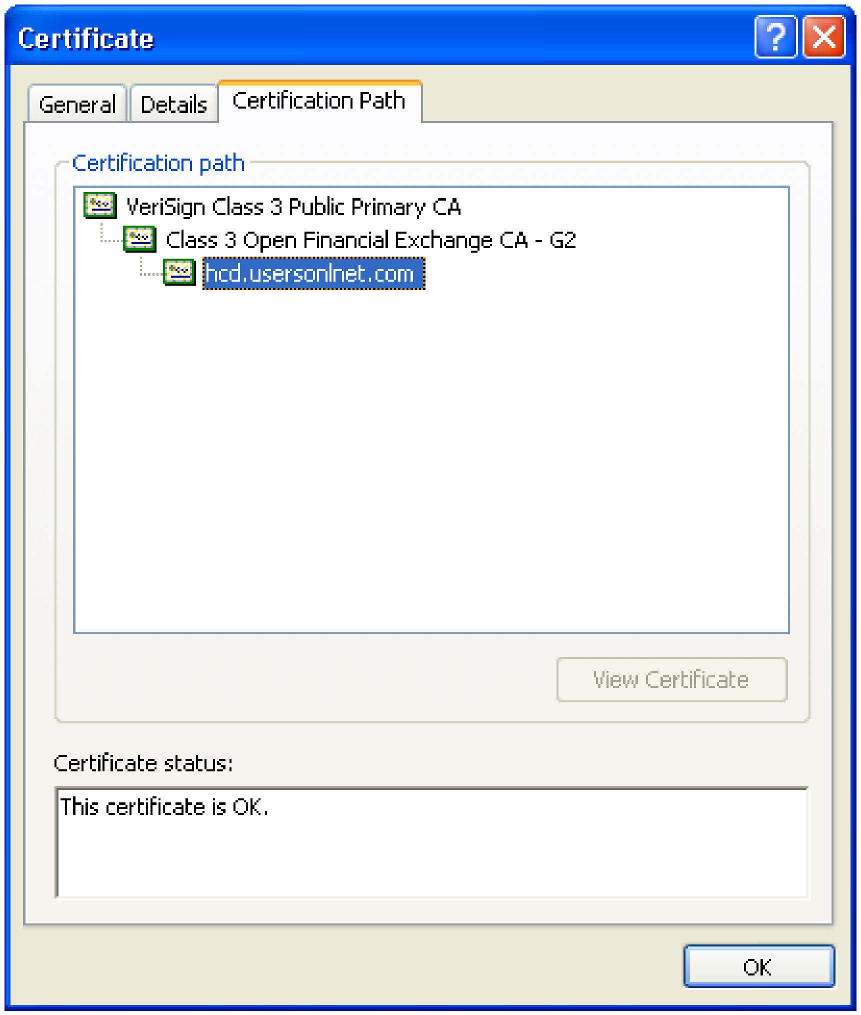
Genuine certificate



Field	Value
Version	V3
Serial number	1c 53 d7 13 2d c6 3f df a0 ca ...
Signature algorithm	sha1RSA
Issuer	VeriSign Class 3 Secure Server...
Valid from	Wednesday, 29 November 20...
Valid to	Tuesday, 15 December 2009 9...
Subject	www.hawaiiusafcuhb.com, Te...
Public key	RSA (1024 Bits)

“Fake” certificate

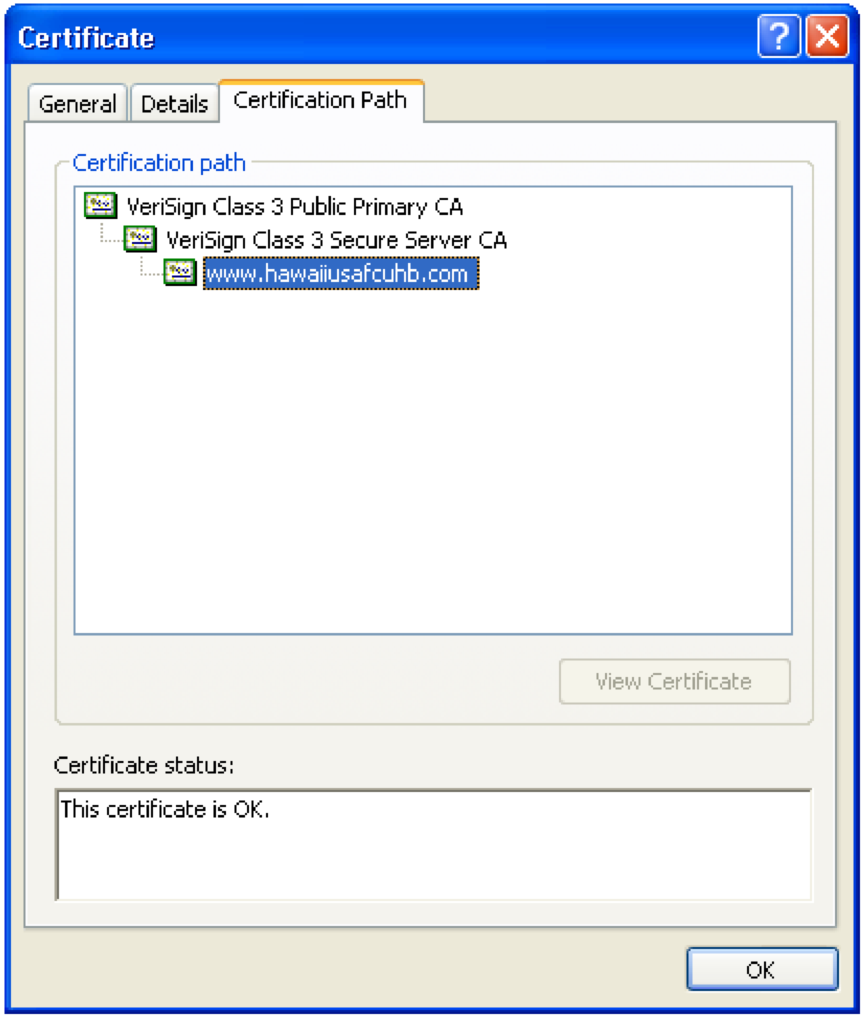
Certificate comparison 3



VeriSign Class 3 Public Primary CA
Class 3 Open Financial Exchange CA - G2
hcd.usersonlnet.com

Certificate status:

This certificate is OK.



VeriSign Class 3 Public Primary CA
VeriSign Class 3 Secure Server CA
www.hawaiiusafcuhb.com

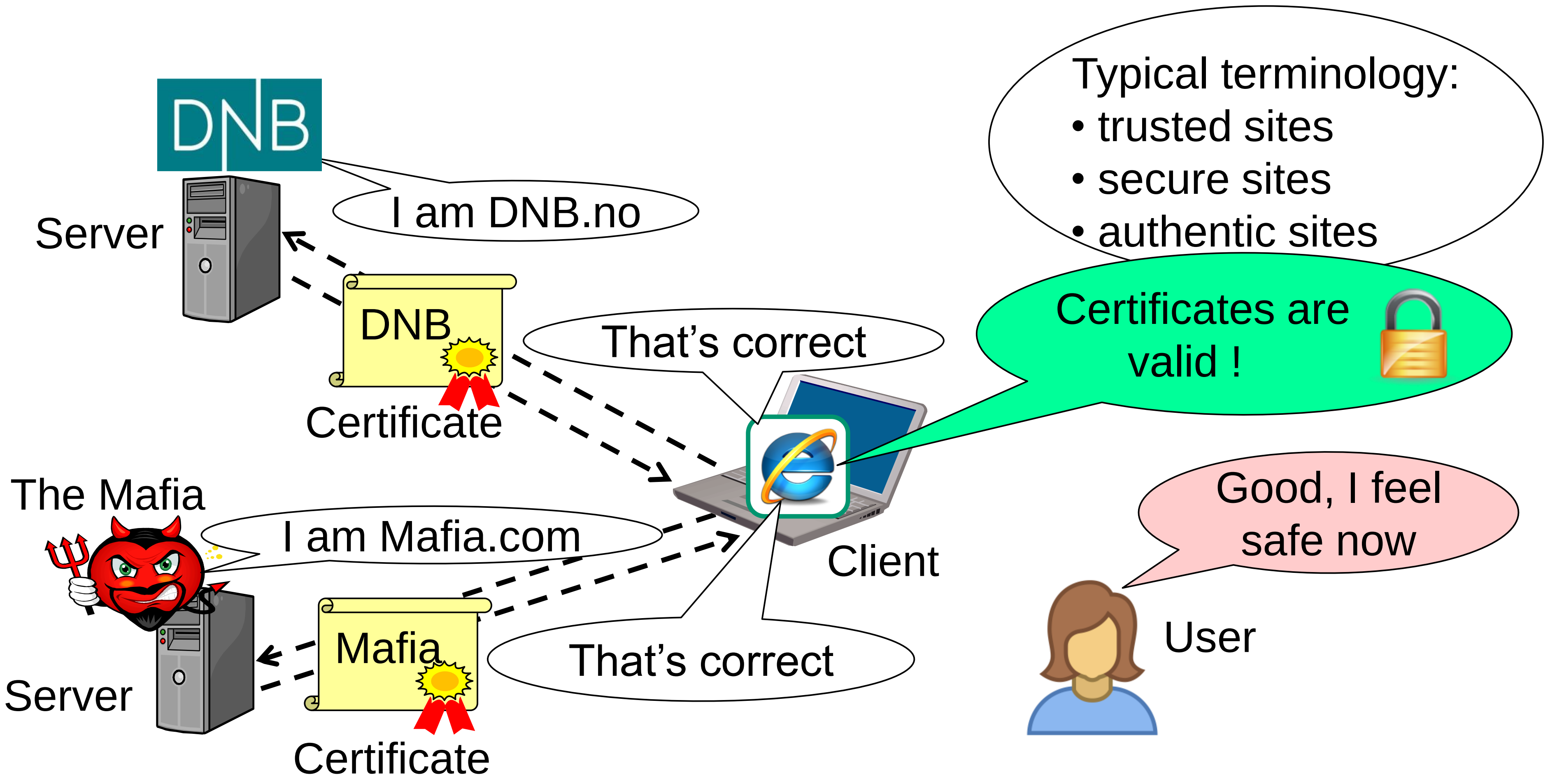
Certificate status:

This certificate is OK.

Genuine certificate

“Fake” certificate

Meaningless Server Authentication

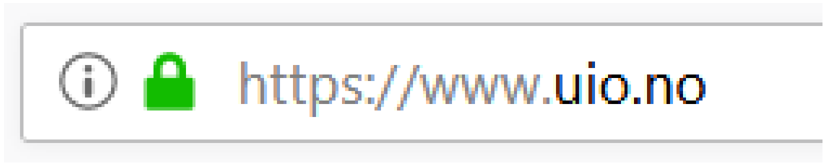


Extended validation certificates

- Problem with simple certificates:
  - Can be requested by anonymous entities (nowadays even free of charge)
- 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.

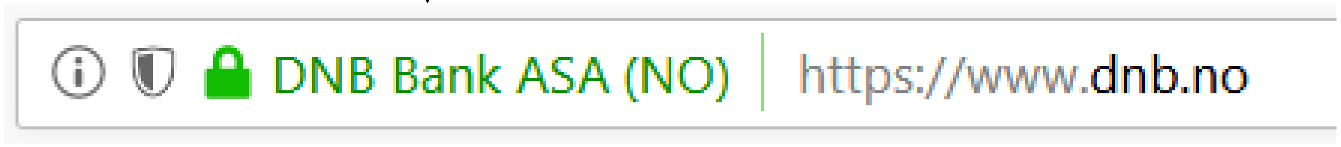
Extended validation certificates

a) "Standard" certificate


https://www.uio.no

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

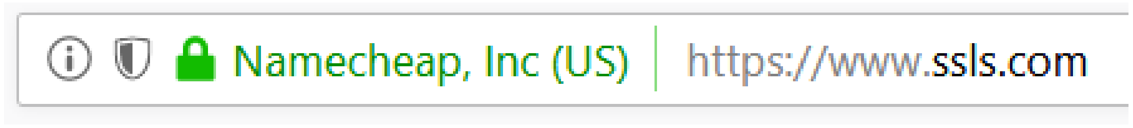
$\downarrow$


DNB Bank ASA (NO) | https://www.dnb.no

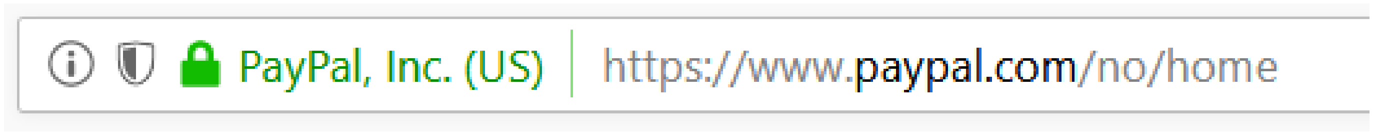
b) Extended validation certifiate

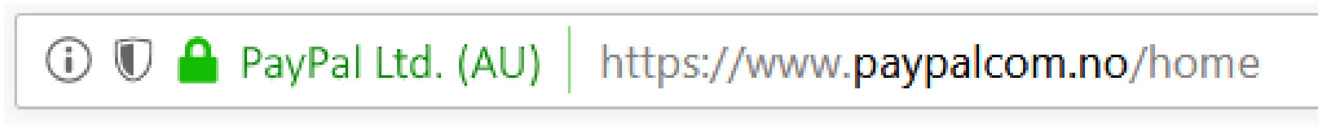
Problems of EV Certificates

- Domain name and owner name not always equal

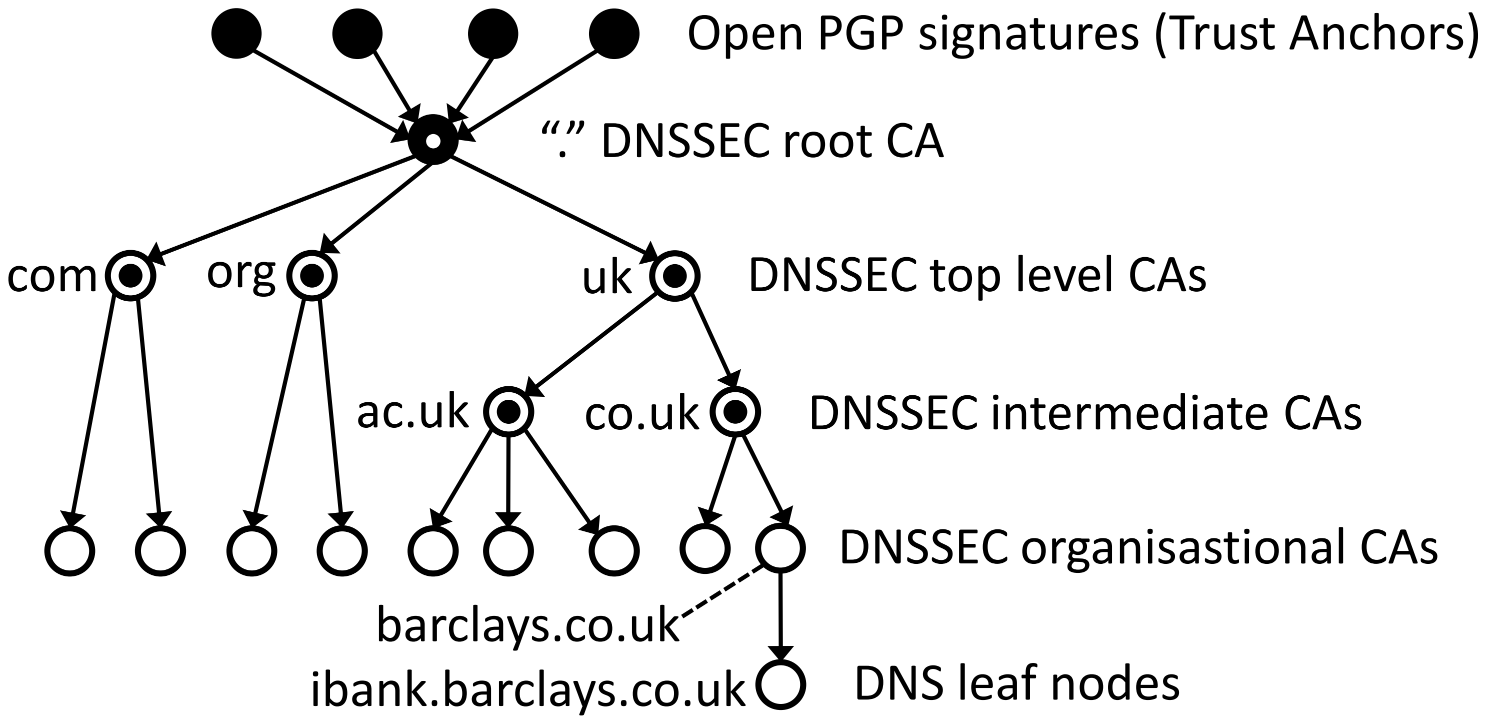

Namecheap, Inc (US) https://www.ssls.com

- If an attacker is able to register a similar business (even in a different country), most users will not recognize the difference


PayPal, Inc. (US) https://www.paypal.com/no/home

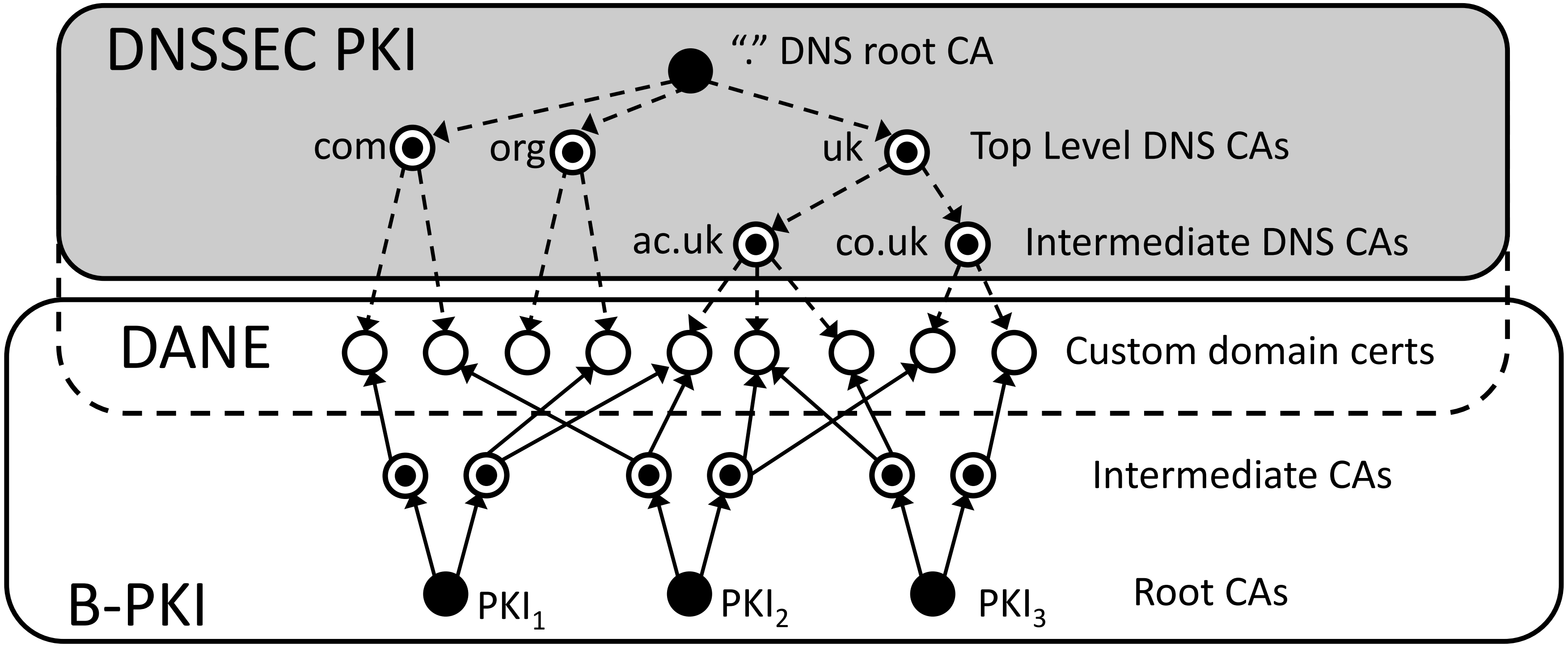

PayPal Ltd. (AU) https://www.paypalcom.no/home

DNSSEC PKI



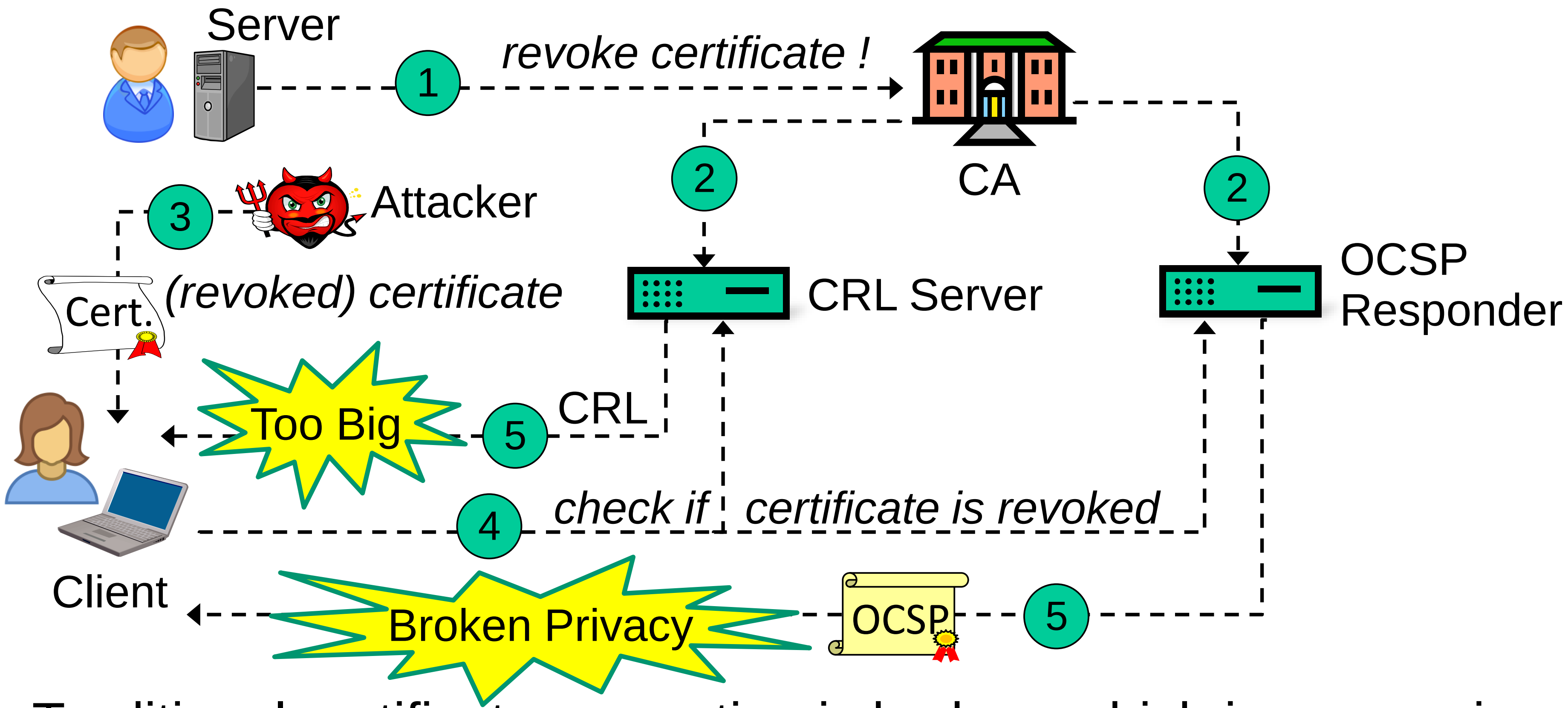
- The DNS (Domain Name System) is vulnerable to e.g. cache poisoning attacks resulting in wrong IP addresses being returned.
- DNSSEC designed to provide digital signature on every DNS reply
- Based on PKI with a single root.

DNSSEC PKI vs. Browser PKI



- 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

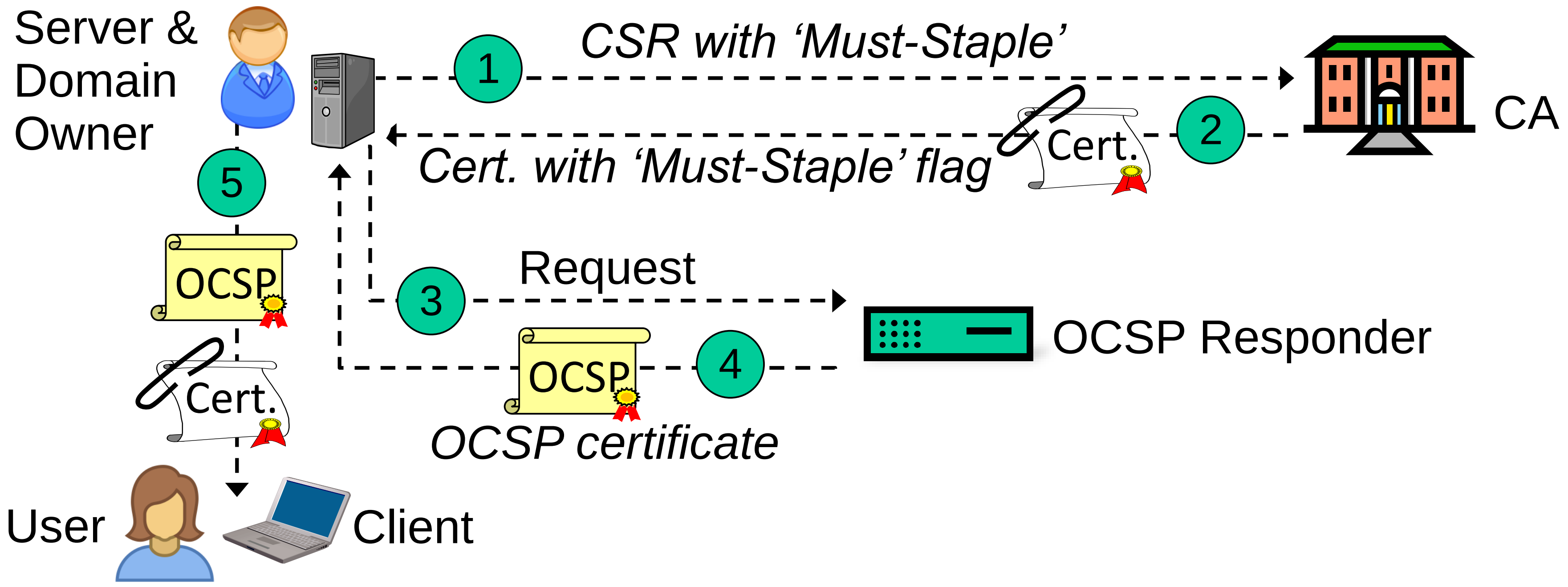
Broken Certificate Revocation



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

OCSP Must-Staple Protocol

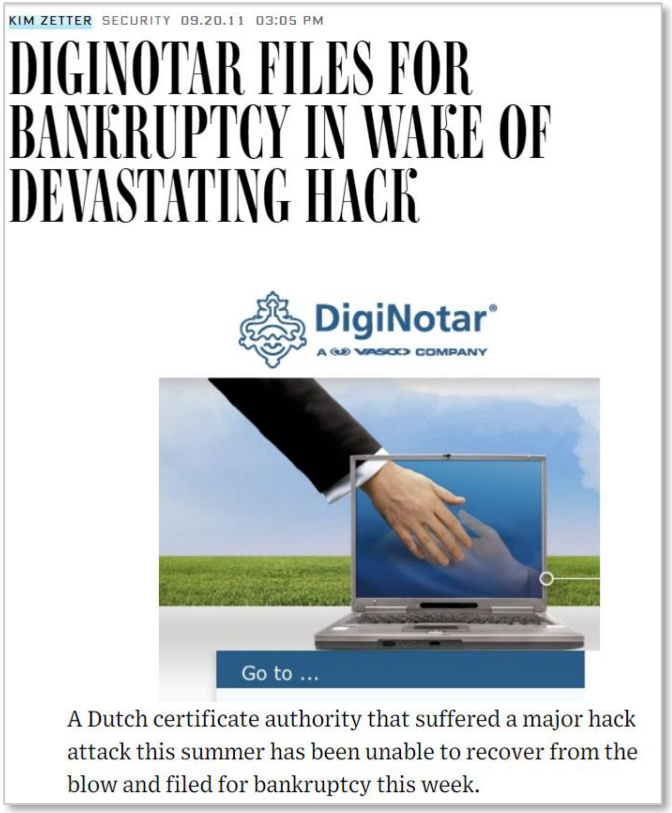


The OCSP-Must-Staple protocol solves the revocation problem

- 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

Compromised Certificate Authority

- CA DigiNotar was hacked in 2011
- A number of illegitimate certificates (e.g. *.google.com) were created by the intruders

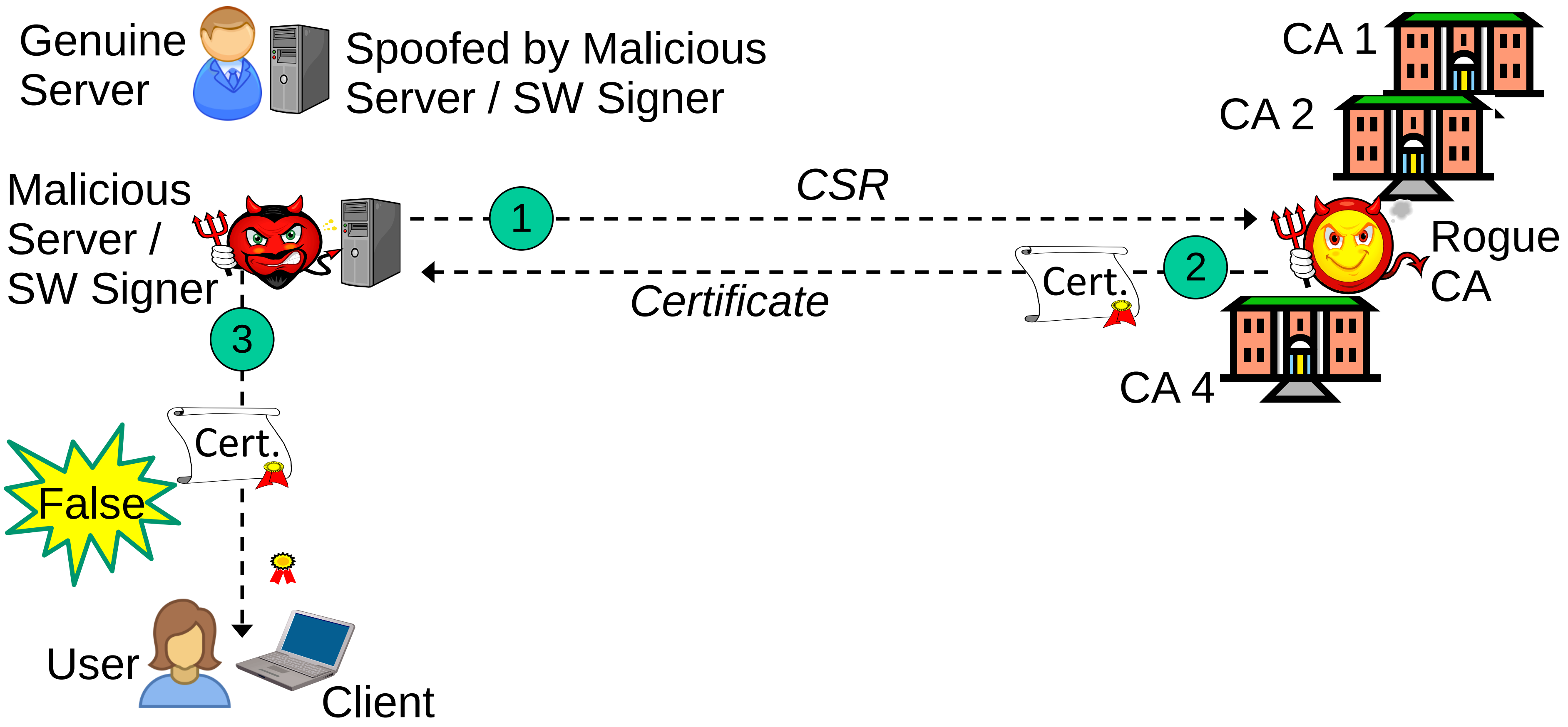


KIM ZETTER SECURITY 09.20.11 03:05 PM

DIGINOTAR FILES FOR BANKRUPTCY IN WAKE OF DEVASTATING HACK

A Dutch certificate authority that suffered a major hack attack this summer has been unable to recover from the blow and filed for bankruptcy this week.

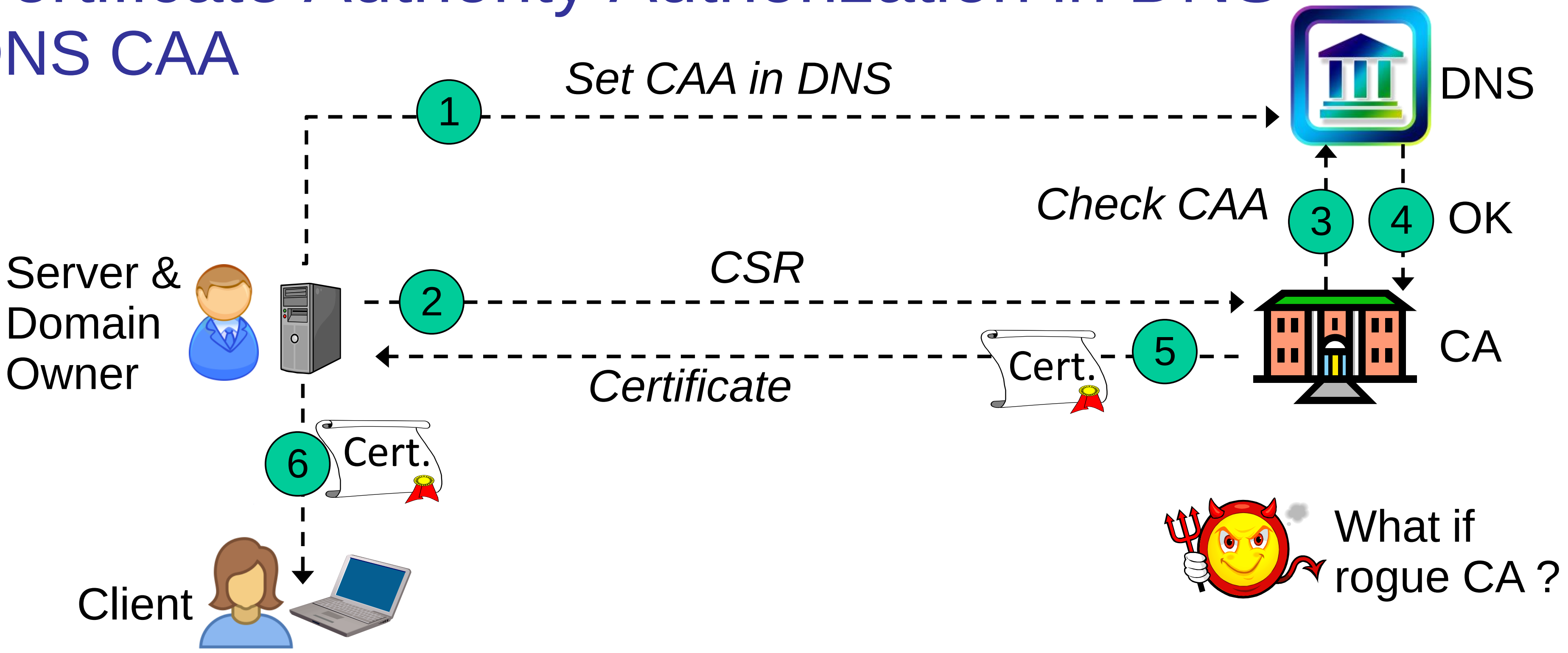
The Problem of Rogue/Compromised CAs



- 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

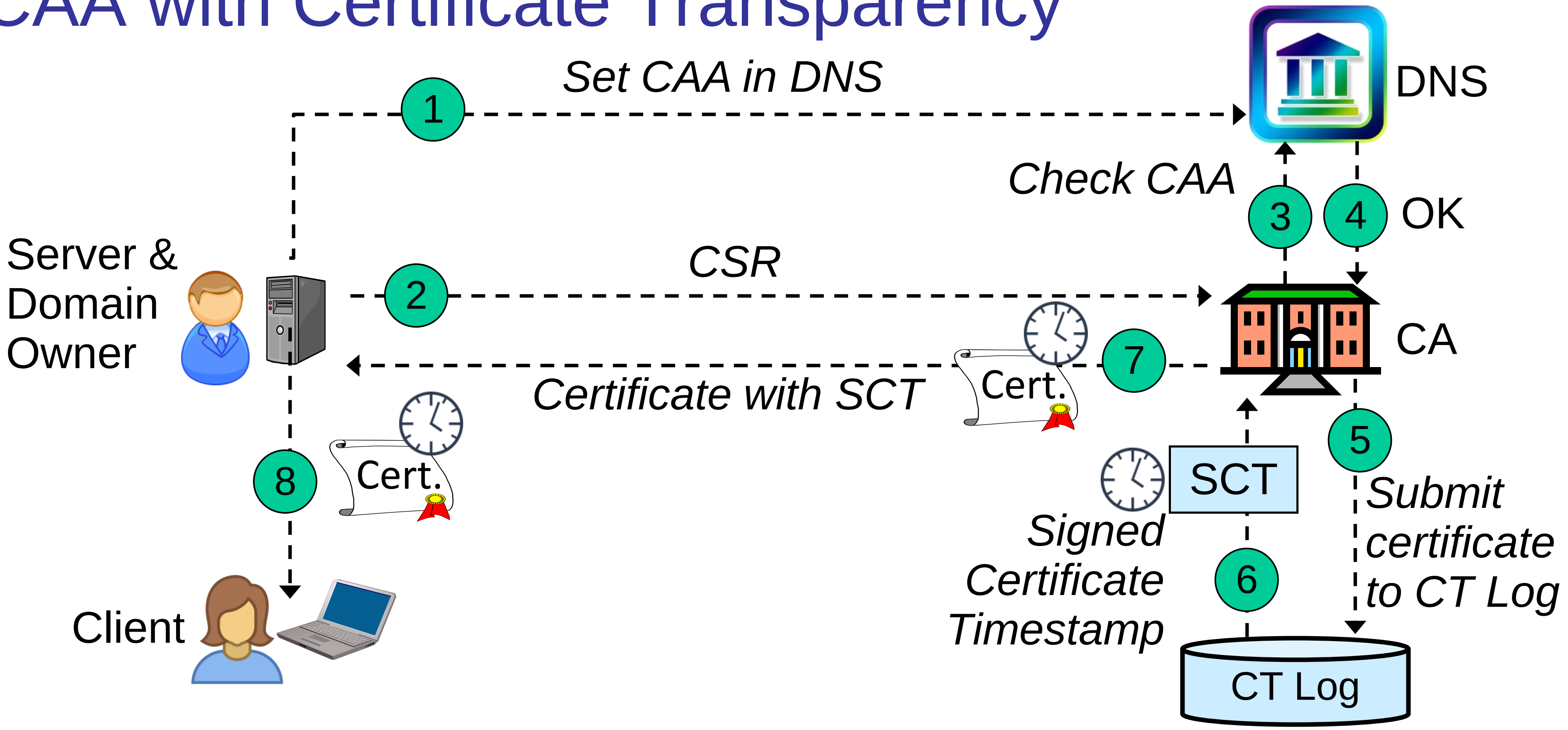
Certificate Authority Authorization in DNS

DNS CAA



- DNS CAA enables a domain owner to specify which CA(s) is/are authorized to issue certificates for the domain
- DNS CAA reduces problem of rogue/compromised CAs
- CAs must not issue certificates if they are not authorized
  - But a rogue/compromised CA could issue (false) certificates anyway

CAA with Certificate Transparency



- Enforcement of DNS CAA is by logging every certificate issued
  - 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 certificates issued to its domain
  - Any illegal certificate found on the CT Log must be revoked !

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