

# **Digital Signatures and Certificate Authorities**

*Piotr Kucharski*

## Overview

- Digital Signatures
- Certificate Authorities

## Overview: true one

- Introduction to Cryptography
- Digital Signatures Basics
- Public Key Infrastructure, which includes **Certificate Authorities**

## Overview: Introduction to Cryptography

- Basic Terminology
- Basic Cryptographic Algorithms
- Cryptographic Hash Functions
- Other important stuff
- Cryptanalysis and Attacks on Cryptosystems

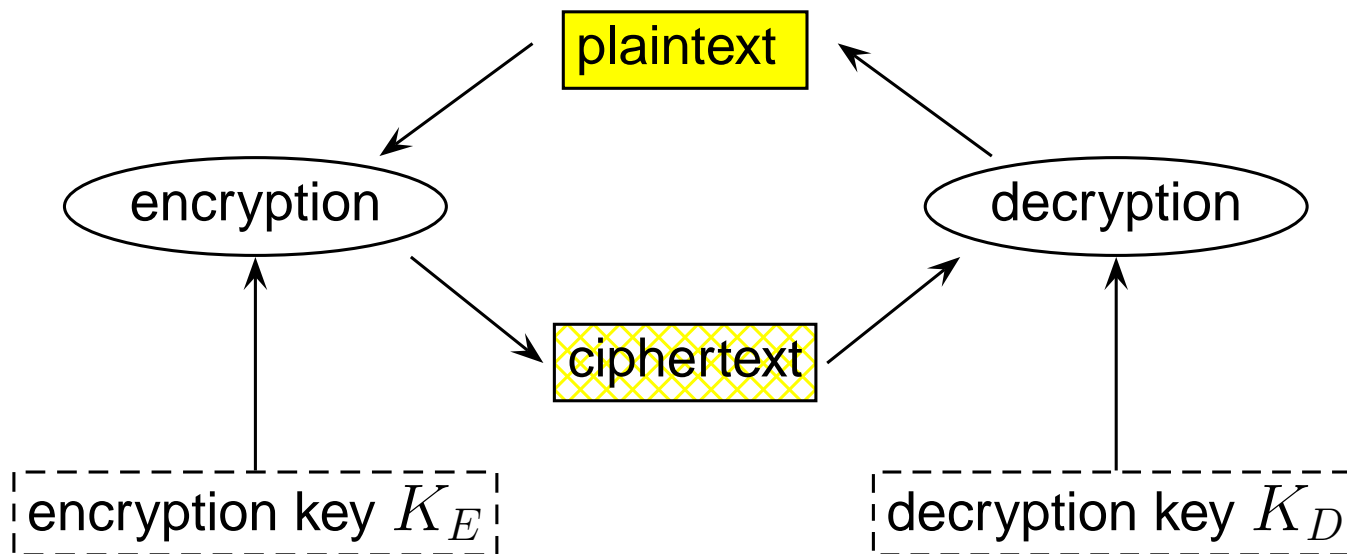
## Cryptography: Basic Terminology

**cryptography** art or science of mathematical techniques related to such aspects of data security as confidentiality, data integrity, authentication, non-repudiation

**cryptanalysis** study of mathematical methods which are used in attempting to defeat cryptographic techniques

**cryptology** study of cryptography and cryptanalysis

# Cryptography: Basic Terminology



## Cryptography: Ciphers

Method of encryption and decryption is called **cipher**. Some cryptographic methods rely on the secrecy of the encryption algorithms. Naive.

The Dutch cryptographer Auguste Kerckhoff von Nieuwenhof formulated this in 1883:

**Kerckhoff's Principle** *The security of a cryptosystem shall not be based on keeping the algorithm secret but solely on keeping the key secret.*

## Cryptography: Algorithms

Two classes of ciphers:

**symmetric** secret-key, the same key is used for both encrypting and decrypting, thus the key **must** be kept secret; those algorithms can be divided in two groups: stream ciphers (bit by bit) and block (block of 64 bits or more by block) ciphers.

**asymmetric** public-key, one key is used for encrypting, another for decrypting and *vice-versa*; this allows for one key to be widely published and for keeping security by keeping the other key (private key) secrecy

## Cryptography: Symmetric Algorithms

**The One-Time Pad** proven unbreakable; Vernam invented in 1917 a cipher: equal bits of plaintext and random “key”. Not reusable. Shannon’s proof in 1949.

**DES** developed in 70s, NIST made it standard, 64-bit blocks, 56-bit keys (secure, but susceptible to full key search), slow; its 3DES variant fixes short key problem, but is three times slower...

**AES** National Institute of Standards and Technology launched the competition for new **A**dvanced **E**ncryption **S**tandard. All finalist use 128-bit blocks and accept 128, 192 and 256 bit keys.

**Rijndael** Winner by Joan Daemen and Vincent Rijmen; performs well in hardware, short setup key time, low memory (best attack 7 rounds of 10... not that it matters much, it still requires  $2^{120}$  steps and  $2^{100}$  bytes)

**Serpent** by Anderson, Biham, and Knudsen; innovative design, may be implemented by vector gate logic, most secure, but slow (best attack 10 of 32 rounds)

**Twofish** by Schneier et al., Counterpane Security; somewhat balanced between Rijndael and Serpent, seems quite good (best attack 8 of 16 rounds)

**RC6** by Rivest, Robshaw, and Yin, RSA Laboratories; improvements over RC5 (best attack 17 of 20 rounds)

**MARS** by Zunic et al., IBM; depends on 32-bit processor instruction sets, problems with other architectures, many operations, thus costly

**other** Blowfish, CAST-128 (RFC2144), IDEA (considered very secure, known, but patented for commercial use and in US), Rabbit (very new), RC4 (source allegedly leaked, unknown security, fast, variant of OTP)

**before 1970** Fish (German high-command army WWII, lead to Colossus first computer), Enigma (rotors, Polish cryptanalysts, Bletchley Park (not secure today)), Vernam cipher, substitution ciphers (Caesar), ancient

## Cryptography: Asymmetric Algorithms

In late 1970s it was observed that based on a very, very difficult problem, that would take thousands of years to solve, a cryptosystem with two keys, a public (for encryption) and a private one (for decryption), could be developed.

Key exchange. It was noted that public key cryptosystem could be used to generate secret key used to do symmetric bulk encryption.

Whitfield Diffie and Martin Hellman constructed key exchange protocol that changed the cryptographic world.

Soon the Ron Rivest, Adi Shamir, and Leonard Adleman developed first real public-key cryptosystem.

Public-key cryptosystem in general is built from difficult problem:

1. take difficult problem (NP-hard), for which you can find specific instance solvable in short time (trapdoor function with high computational complexity)
2. convert the cleartext into such easy instance of difficult problem
3. use public-key to convert easy instance back to difficult problem
4. send it over insecure channel
5. use the private key to convert difficult problem to easy instance
6. solve easy instance to get the cleartext

**RSA** created by **R**ivest, **S**hamir and **A**dleman; the most commonly used public-key algorithm; factoring product of two large primes;

Interactive RSA tutorial at <http://www.youdzone.com/rsa.html>

**ElGamal** similar to DH, slower, but patent free

**Diffie-Hellman** key exchange protocol from 1976(!); allows two parties without any initial shared secret to create one; then they can use that shared secret as a key for a symmetric cryptography (block cipher, fast) or as the basis for key exchange; discrete logarithm problem.

## Diffie-Hellman scheme

Given known prime  $p = 23$  and generator  $g = 3$  (mathematically carefully chosen)

**Alice** generates random number  $a = 6$

calculates  $A = g^a \bmod p = 3^6 \bmod 23 = 16$

sends  $A = 16$  to Bob

**Bob** generates random number  $b = 15$

calculates  $B = g^b \bmod p = 3^{15} \bmod 23 = 12$

sends  $B = 12$  to Alice

**shared secret**  $s = g^{a \cdot b} \bmod p$  can now be calculated:

- Alice knows  $a$  and  $B$ , so she computes  $s = B^a \bmod p = 12^6 \bmod 23 = 9$
- Bob knows  $b$  and  $A$ , so  $s = A^b \bmod p = 16^{15} \bmod 23 = 9$

## Cryptographic Hash Functions

“Compress” the message to a fixed-size **hash value**. Designed such that different messages produce different hash values. Not entirely possible. At least small changes produce completely different hash values and it is hard to find two messages to produce the same hash value (*collision*)

**MD5** developed by RSA Labs, published RFC1321, hashes to 128-bit value, example:

MD5 (irc2.11.1p1.tgz) = c5a2b3097a5fbeb91b39412730b02ab5

**RIPEND-160** meant to replace MD5, produces 160-bit values

**SHA-1** published by USA Gov (FIPS PUB 180-1), gives 160-bit, has longer (256, 384, 512) variants

## Other important stuff

**MAC** (message authentication codes) based on hash (HMAC-SHA-1) or symmetric block ciphers (DES-CBC-MAC), used to verify integrity of messages

**shared secret key** distribute the key among  $N$  people (or more) and guarantee that less than  $N$  of them will not be able to get the whole key

**Random Number Generators** used heavily in cryptography (like keys);

we need at least 128 bit of entropy

**true** physical, hardware (semiconductor leak) noise, least significant

bit of audio input, intervals between interrupts or user keystrokes

**pseudo** best to get some physical (even small sample), stir it later;

ANSI X9.17; FIPS-186 (DSS)

***Make RNG really good or it ruins crypto system.***

# Cryptanalysis and Attacks on Cryptosystems

**brute-force** aka exhaustive search of keys

**man in the middle** get between Alice and Bob, act as a proxy

**cryptanalysis** art of deciphering encrypted

**ciphertext-only attack** nothing known

**known-plaintext attack** decrypt rest or get the key

**chosen-plaintext attack** to get the key

**quantum computing** solve *ALL* computations at once

**human factor** cheating

## Overview: Digital signatures basics

- Traditional Signatures
- Digital Signatures
  - concepts
  - creation and verification
  - problems
  - example

## Traditional Signatures

Signature is not substance of transaction, but rather representation.

Traditional signatures serve general purposes:

**evidence** hand written signature is attributable to the signer, hence it connects the writing with a signer

**ceremony** brings attention to legal significance, which helps to prevent “inconsiderate engagements”

**approval** some legal acts require written form to make it work; establishes sense of legal transaction

**efficiency and logistics** imparts sense of clarity and finality (negotiations) and is easy to do!

*Assuring the parties of transactions validity and enforceability.*

## Digital Signatures

Digital documents' advantages (easily duplicated, fast travel, computer-processable) are also their disadvantages. They need more:

**signer authentication** signature must indicate who signed some data and it should be difficult to forge

**document authentication** signature must indicate what was signed exactly, with no possibility to forge or alter document or signature

**efficiency** signature, its creation and verification should provide greatest possible assurance of signer and document authenticity with least possible expense

*This is PKI non-repudiation service job.*

Digital signatures use public key cryptography, which uses two mathematically related keys:

**private key** known only to the signer, used to create signatures

**public key** known widely, used to verify signatures

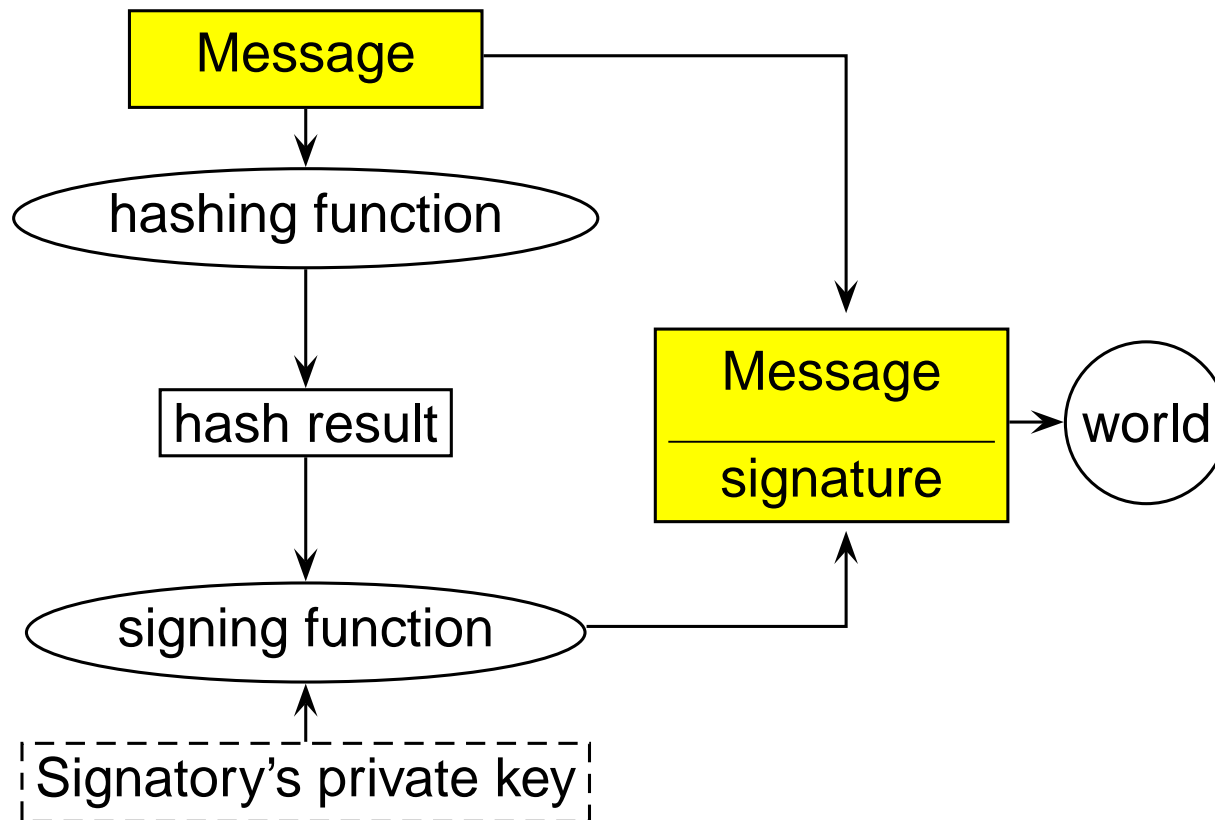
Keys are related, but it is computationally infeasible to derive private key from the public one.

For the efficiency purpose, the whole message is not signed (though short one could be), only its hash value

**hash value, hash result** result of applying hash function on the message

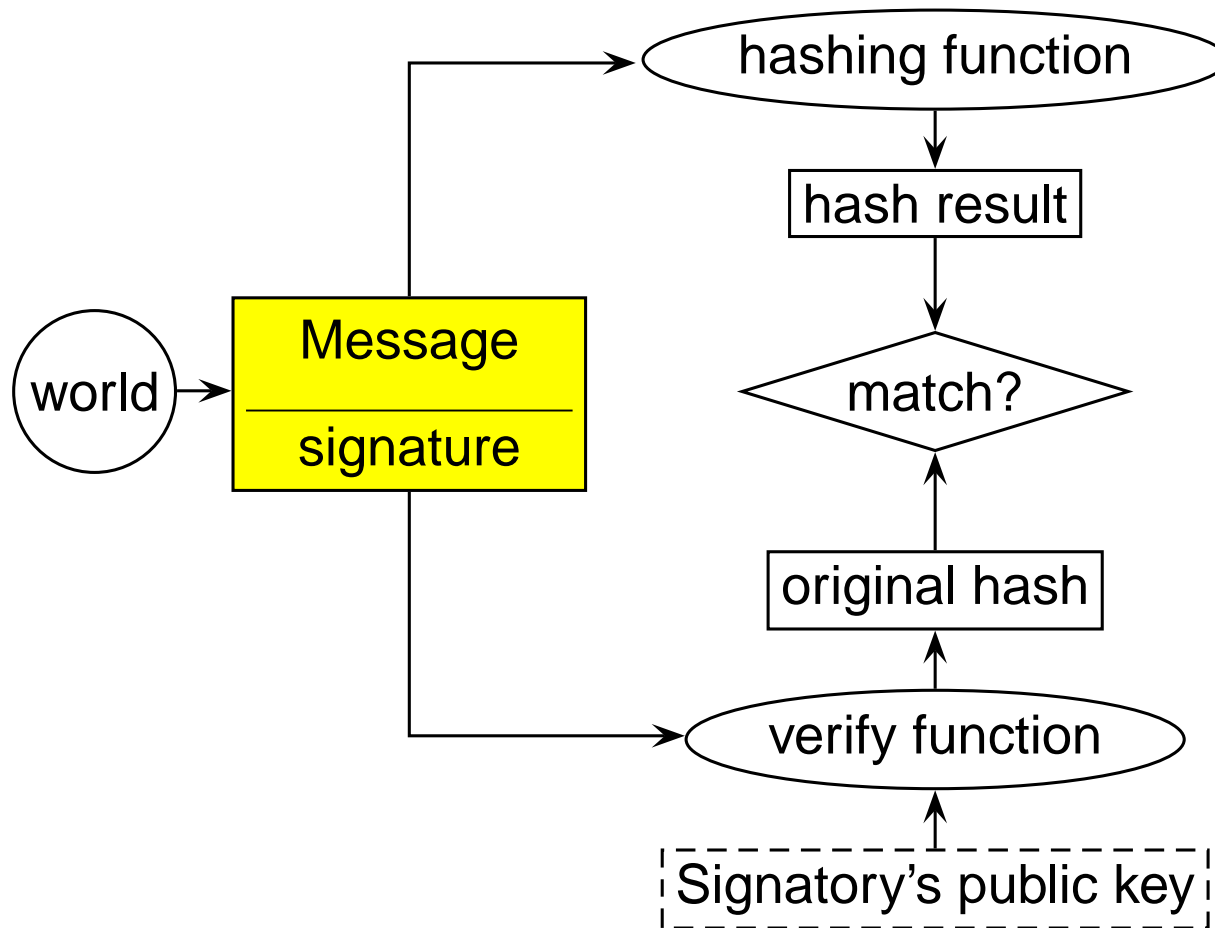
**hash function** one-way mechanism to produce a unique, fixed-size fingerprint of an input data

# Digital Signature Creation



Note: signing function is de facto encryption of hash of message

# Digital Signature Verification



Note: verify function is de facto decryption of signature

## Digital Signatures problems

**hash function** we depend on, has ex definition the possibility of collisions. . .

there may be *more than one* message producing the same hash value  $\Rightarrow$  forgery

MD5 collisions <http://www.cits.rub.de/MD5Collisions/>

**message** signer must be fully aware of what s/he signs

**keys** how to know which key belongs to whom?  $\Rightarrow$  Public Key Infrastructure

**PKI** can we trust it at all?

## Digital Signature example with PGP

```
% cat test.txt
Hello, this is a valid example file for CEENet Workshop
Secret data for later: 981cff74a11a7ffacab4ce26656b3f55

% pgp -sba -u chopin@sgh.waw.pl test.txt
Pretty Good Privacy(tm) 2.6.3ia - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-03-04
International version - not for use in the USA. Does not use RSAREF.
```

```
A secret key is required to make a signature.
You need a pass phrase to unlock your RSA secret key.
Key for user ID: Piotr Kucharski <chopin@sgh.waw.pl>
1024-bit key, key ID D818A489, created 1995/10/31
```

```
Enter pass phrase: s3kr3t... Pass phrase is good. Just a moment...
Transport armor file: test.txt.asc
```

```
% cat test.txt.asc
-----BEGIN PGP MESSAGE-----
Version: 2.6.3ia
```

```
iQCVAwUAQr899kABjqnYGKSJAQFabgP9GsOfNMs25xNTrMPcdNEodZfmtpzTSJg5
3vEYR200sVD9txVklLlqchhIRznssgYNIeTOYQwVrt1XRedgEnkAUyO8iRjPn4DQm
ETP17NReV3bcwHZI5SFdNsZiLNe7YIujycIbywuP4C/a/MrteAdp5tLgHTBX01l/
uFVYM6fnoLw=
=IRV1
-----END PGP MESSAGE-----
```

Let's verify.

```
% pgp +verbose=0 test.txt.asc -o /dev/null
Good signature from user "Piotr Kucharski <chopin@sgh.waw.pl>".
Signature made 2005/06/20 23:20 GMT using 1024-bit key, key ID D818A489
```

Now I remove the line with "secret data" and verify again.

```
% pgp +verbose=0 test.txt.asc -o /dev/null
WARNING: Bad signature, doesn't match file contents!

Bad signature from user "Piotr Kucharski <chopin@sgh.waw.pl>".
Signature made 2005/06/20 23:20 GMT using 1024-bit key, key ID D818A489
```

Now I restore the line with "secret data", modify signature and verify again.

```
% pgp +verbose=0 test.txt.asc -o /dev/null
ERROR: Bad ASCII armor checksum

Error: Transport armor stripping failed for file test.txt.asc
```

For a usage summary, type: `pgp -h`  
For more detailed help, consult the PGP User's Guide.

## **Overview: Public Key Infrastructure**

- Infrastructure Model
- Offered Services
- Certificate Authorities in detail
- Trust Model
- Problems

## PKI: Infrastructure Model

**Infrastructure** in general: like electricity or LAN

**Needs** applications related, secure (single) sign-on, transparency

**Gains** cost savings, interoperability (internal, external), uniform solution  
(easy administration), *way to possibly achieve security*

**Infrastructure Model** on the next slide

## Infrastructure

---

- Certification Authority
- Key Backup
- Key History Management
- Certification Repository
- Key Recovery
- Cross-Certification
- Certificate Revocation
- Automatic Key Update
- Client Software

## Services

---

- Authentication
- Secure Time Stamping
- Secure Data Archive
- Integrity
- Notarization
- Privilege/Policy Creation
- Confidentiality
- Non-Repudiation
- Privilege/Policy Verification

## PKI Infrastructure overview

**Certificate Authority** trusted third party, binds given public key to some entity by issuing CA-signed certificate

**Certification Repository** mechanism of actually reaching certificates; efficiency, scalability, on-line accessibility

**Certificate Revocation** change of name or private key stolen – there must be a way to tell that certificate is no longer valid; published Certificate Revocation Lists (CRLs), using Online Certificate Status Protocol (OCSP)

**Key Backup & Recovery** keys (for signing and encrypting) will be getting lost (fragile human memory or hardware malfunction) – for organizations it may be unacceptable to *not* be able to decrypt encrypted data

**Automatic Key Update** certificates do not live forever — if update is left up to user, it will surely be forgotten

**Key History Management** new keys will not decrypt old data encrypted with old keys (and reencrypting is practically infeasible)

**Cross-Certification** universal (world-wide) PKI is a myth; different PKI systems cooperating need some way of trusting each other users

**Client Software** all infrastructure (servers) is nothing without clients; it's the client that requests certificates, parses CRLs, verifies procedures...

## PKI Services overview

**Authentication** assurance to one entity that another entity is who it claims to be

**Integrity** assurance to an entity that data has not been altered

**Confidentiality** assurance to an entity that no one can read a particular piece of data except the receiver(s) explicitly intended.

**Secure Time Stamping** sequence of events must be out of question

**Notarization** certification of data validity (being "correct" is totally subjective, though)

**Non-repudiation** once you sign, you cannot claim you did not; cannot be 100% certain, but helps

**Secure Data Archive** all old certs, CRLs, timestamped documents etc.  
must be stored for later use in arguments

**Privilege/Policy Creation** mechanisms to create fine-grained access control  
for individuals or groups (actually, their certificates)

**Privilege/Policy Verification** mechanisms to verify above

## PKI core services: Authentication

**Authentication** is assurance to one entity that another entity is who it claims to be. Kind of.

**Entity authentication** with no regards to what it does, first step before anything else is done

- local auth: entry level, usually requires manual intervention and manifests evidently and directly (pressing the thumb, typing the password etc.)
- remote auth: wherever user accesses remote services, which need to know who s/he is; it may involve the user directly but because of possible problems with securing auth data over transport and (which seems more important) users' discomfort (SSO may be not supported), only the *result of local auth* is carried on

**Entity authentication** can be achieved in many ways

**something you have** like a smart card or token

**something you know** like a password or a PIN code

**something you are** like your unique thumb print, retina shape

**something you do** like handwriting pattern

Single-factor is less secure than multi-factor. Commercial world uses two-factor, eg. smart card (pin+chip), securID (tokencode + pin).

**Data origin identification** identifies the entity being source or origin of some data – statically and irrevocably; (provides support for non-repudiation)

## PKI core services: Integrity

- assurance of non-alteration (intentional or not), both during transport (between *there* and *here*) and storage (between *then* and *now*)
- parity bits or CRC codes can detect accidental bit errors, but are no match for evil intruders
- signing with own private key (one to many (basically it's derivative of authentication)) or with recipient public key (one to one) is the answer — noone but the key owners will be able to manipulate data *and* keep integrity of the signature

Message Authenticating Code (MAC) are built from symmetric block cipher like DES-CBC-MAC (FIPS113) or cryptographic hashes like HMAC-SHA-1 (RFC2104) – they use symmetric cryptography, but the keys are usually distributed using PKI.

What Alice has to do to MAC the code for Bob?

if Bob has PKI public key	if Bob doesn't have PKI public key
	make Bob generate DH public key and use it with her private DH key to
generate new symmetric key	
MAC the data with this symmetric key	
encrypt symmetric key with Bob's public key	
attach encrypted symmetric key to data	attach her public DH key to data
send the data to Bob	

## PKI core services: Confidentiality

Confidentiality is the assurance to an entity that no one can read a particular piece of data except the receiver(s) explicitly intended. First and obvious thought: encrypt all data with recipient public key.

This. Is. S. L. O. W... Instead use fast symmetric cryptography (regardless of data size, to make it simpler):

- create symmetric key
- use symmetric key (using some block cipher) to encrypt data
- send encrypted data to Bob with
  - either own DH public key
  - or copy of symmetric key encrypted with Bob's public key

## **PKI enabled services: secure communication**

Data transmission secured by core PKI service(s) (authentication, integration, confidentiality) and applied onto the existing network framework creates *secure communication*:

**e-mail** S/MIMEv2 (RFC2311, RFC2312) or PGP

**web** SSL/TLS (RFC2246)

**VPN** IPsec/IKE (RFC2401, RFC2411)

## **PKI enabled services: secure time stamping**

- sequence of events must be out of question
- especially when dealing with certificates lifetime and revocations
- especially when digital signatures are used

## PKI enabled services: non-repudiation support

Product of other PKI services: authentication and secure time stamping.

- once you securely sign a document (either as an author, or as a receiver), you **cannot** deny this signing
- of course when you can prove it was compromised before... the judge may prevail

## **PKI enabled services: notarization**

- notary (third party) certifies that some data is "correct"
- what is "correct" is very subjective

## **PKI enabled services: privilege/policy management**

- mechanisms to create fine-grained access control for individuals or groups (actually, their certificates)
- who (client certificate) can do what and when
- sometimes included directly in the certificate, but it's difficult to manage

## Certificates

**X.509v3** ISO standard, public-key, global trust to single authority, tries to bind persons (their names) to single certificates globally, profiled by IETF to PKIX as RFC 3280

**SET** Secure Electronic Transaction: credit card payments, "private" extension of X.509v3

**Attributes** role based, RFC 3281, based on X.509, without public key, foundation for Privilege Management Infrastructures

**SPKI/SDSI** (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure), detaches persons' names from certificates, meant to be simple, but noone really uses it

**PGP Certificates** (Pretty Good Privacy), shifts authority from global central to user central (more about it in Trust Models slide later)

## X.509 (RFC 3280) structure

- Certificate
  - Version** identifies the version of certificate
  - Serial Number** CA-unique integer identifier for the certificate
  - Algorithm ID** algorithm used to sign the certificate
  - Issuer** unique name of certificate issuer
  - Validity** not before and not after
  - Subject** unique name of the owner
  - Subject Public Key Info** public key (and its algorithm) of the owner
  - Issuer Unique ID** optional unique ID of issuing CA
  - Subject Unique ID** optional unique ID of the subject
  - Extensions** optional extensions
- Certificate Signature Algorithm
- Certificate Signature

## X.509 Extensions

**Authority Key Identifier** unique id of key to be used to verify digital signature of certificate

**Subject Key Identifier** unique id of public key in this certificate (to distinguish between multiply keys of the same certificate owner)

**Key Usage** bit string to restrict usage of the key for one or more of: digital signature, non-repudiation, key encipherment, data encipherment, key agreement, certificate signature, CRL signature, encipher only, decipher only

**Extended Key Usage** sequence of OIDs identifying usage of public key: TLS server authentication, TLS client authentication, code signing, e-mail protection, time stamping, OCSP signing

**CRL Distribution Point** obviously, where (URI mostly) you can get newest CRLs (either full or delta)

**Private Key Usage Period** when private key may be used (as opposed to public key lifetime); helps managing keys

**Certificate Policies** policy OIDs and optional qualifiers: Certification Practice Statement (CPS) and User Notice

**Policy Mappings** policy OIDs equivalences between two CA

**Subject Alternative Name** alternative names for certificate owner, like e-mail, IP, URL etc. (must be used if **Subject** is empty)

**Issuer Alternative Name** alternative names for certificate issuer

**Basic Constraints** end-entity (false) or CA certificate (true)

**Path Length Constraint** max number of CAs that can be in certificate chain after this one

**Name Constraints** Permitted/Excluded Subtrees attributes in form of DN, URI, e-mail, anything hierarchical, allows to manage trees more flexibly (delegating authority)

**other** Subject Directory Attributes, Policy Constraints, Inhibit Any Policy, Freshest CRL Pointer, private extensions like Authority Information Access or Subject Information Access

```
% openssl s_client -connect www.sgh.waw.pl:443 -showcerts | \
openssl x509 -noout -text
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 191757 (0x2ed0d)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=PL, O=Unizeto Sp. z o.o., CN=Certum Level III
  Validity
    Not Before: Apr 18 15:29:40 2005 GMT
    Not After : Apr 18 15:29:40 2006 GMT
  Subject: C=PL, O=Szkola Glowna Handlowa, OU=Centrum Informatyczne,
    CN=*.sgh.waw.pl/emailAddress=unix@sgh.waw.pl
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
      Modulus (1024 bit):
        00:b2:95:e7:08:25:72:13:89:be:7f:2e:78:9a:25:
        2f:87:aa:f6:37:43:f6:a7:3c:6a:41:33:70:cb:cd:
        a0:1c:62:19:9f:3c:e5:9e:e9:53:bc:15:96:04:57:
        ff:d6:29:52:cf:50:ab:7f:6d:b6:0a:a0:1e:11:64:
        79:53:d4:5b:1d:31:d3:b5:21:6f:7b:03:06:51:b9:
        b4:ea:83:a9:ce:7b:e6:3c:ef:9d:f9:ec:08:ce:03:
        3b:64:52:b8:ad:c1:be:36:bf:2b:1a:91:6f:9c:5e:
        7c:9b:20:15:c0:47:b7:35:ac:79:92:d1:01:10:e8:
        bf:e4:07:4f:2b:e7:0f:21:75
      Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Extended Key Usage:
      TLS Web Server Authentication
```

X509v3 Basic Constraints:

CA:FALSE

X509v3 CRL Distribution Points:

URI:http://crl.certum.pl/class3.crl

Authority Information Access:

OCSP - URI:http://ocsp.certum.pl

X509v3 Certificate Policies:

Policy: 1.2.616.1.113527.2.2.3

CPS: http://www.certum.pl/CPS

User Notice:

Organization: Unizeto Sp. z o.o.

Number: 1

Explicit Text: Usage of this certificate is strictly subjected to the Certum Certification Practice Statement (CPS) incorporated by reference herein and in the Certum Repository at <https://www.certum.pl/repository>. This CPS is also available by mail at Unizeto Sp. z o.o. 70-486 Szczecin, Krolowej Korony Polskiej 21, Poland. Copyright (c) 1998-2004 Unizeto Sp. z o.o. All Rights Reserved.

Signature Algorithm: sha1WithRSAEncryption

b5:8e:03:9a:b6:e5:93:ed:a6:11:53:8b:e0:7e:9f:e1:87:63:  
27:32:27:d1:6f:76:45:4d:bb:fc:4f:4a:60:0b:1b:fa:84:15:  
49:f7:16:6f:48:ba:0e:d6:90:19:fe:01:47:91:3e:a7:4e:6a:  
28:9c:51:b4:35:0e:26:b1:79:49:c4:94:49:40:f2:0e:25:aa:  
24:5e:e5:c9:54:ae:5b:a2:e3:82:56:2c:b5:6f:b9:65:00:7a:  
fc:92:63:d6:72:d5:7a:ec:f7:86:ac:9f:58:d9:32:75:f8:5a:  
2d:31:c1:ff:76:1a:1e:07:2d:88:f0:e3:9e:6c:7e:8f:5f:5f:  
7a:a7

## Certificate Authorities

CA provides a level of assurance that the public key contained in the certificate does indeed belong to the entity named in the certificate.

Digital signature placed on the public key certificate by the CA provides *cryptographic binding* between the entity's public key, the entity's name, and other information in the certificate, such as a validity period.

## PKI parties

**certificate authorities** issue certificates

**subscribers** entities (individuals or organizations) operating the private key corresponding to the public key within the certificate

**relying parties** entities relying on the public key within the certificate

**registration authorities** entities assisting CA to authenticate the identity or other attributes of certificate applicant, initiating revocation of certificate upon subscribers request, approving or rejecting requests to rekey the certificate

**repositories** entities providing publication, storage and access to certificates, CRLs and other PKI documents

## CA basic duties

**register subscribers** identification and authentication (often delegated to Registration Authorities (RAs))

**securely manage the certificate** generation and issuance, distribution, renewal and rekey, revocation, suspension

**provide certificate status information** issuance of certificate revocation lists (CRLs), maintenance of an online status-checking mechanism (OCSP)

**enable access to certificates and CRLs** through Certificate Repository (online directory, may be operated by outside party) accessible to relying parties

## Subscriber registration

- subscriber generates own private/public key pair
- subscriber produces proof of identity – depends on certificate policies and assurance level required
- subscriber demonstrates s/he holds private key (by signing some data, so CA can verify it with public key)
- CA issues certificate
- CA signs the certificate with its private key
- subscriber publishes certificate (so other users may use it)
- CA publishes the certificate in repository

## Certificate Authorities: basic paper work

**Certificate Policy** a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements

**Certification Practice Statement** statement of practices and procedures, more detailed than CP, where CP establishes requirements, CPS describe how CA meets them

**Subscriber Agreement** between CA or RA; focuses on subscriber's responsibilities and terms and conditions of use of the certificate

**Relying Parties Agreements** governs terms and conditions under which the relying party is permitted to rely upon the certificate (like: requires to check the status of certificates in the chain)

**PKI Disclosure Statement** clear and concise (as opposed to CP and CPS)

**other agreements** agreements to facilitate interoperability, vendor agreements, agreements aiding certificate distribution, internal agreements related to trustworthiness of a PKI

**other CA Business Practices**

## **Certificate Authorities: more paper work**

CA Business Practices Disclosure include Service Integrity:

**CA Environmental Controls** documenting the work environment

**Key Management** documenting CA key procedures

**Certificate Life Cycle Management** documents subscribers certificates  
procedures

## CA Environmental Controls

- CPS and Certificate Policy Management
- Security Management
- Asset Classification and Management
- Personal Security
- Physical and Environmental Security
- Operations Management
- System Access Management
- Systems Development and Maintenance
- Business Continuity Management
- Monitoring and Compliance
- Event Journaling

## Key Management

- CA Key Generation
- CA Key Storage, Backup and Recovery
- CA Public Key Distribution
- CA Key Escrow (if applicable)
- CA Key Usage
- CA Key Destruction
- CA Archival
- CA Cryptographic Hardware Life Cycle Management
- CA-provided Subscriber Key Management Services (if applicable)

## Certificate Life Cycle Management

- Subscriber Registration
- Certificate Renewal (if applicable)
- Certificate Rekey
- Certificate Issuance
- Certificate Distribution
- Certificate Revocation
- Certificate Suspension (if applicable)
- Certificate Status Information Processing
- Integrated Circuit Life Cycle Management

## Certificate Revocation

- certificate binds entity to public key
- if entity changes (name, job) or private key is compromised... revoke certificate!
- **revocation delay** is time between knowledge that certificate should be revoked and actual posting of revocation information – very important factor
- methods

**periodic publications** like Certificate Revocation Lists

**online systems** like Online Certificate Status Protocol

## Periodic publications (of Certificate Revocation Lists)

**Complete CRL** scalability and validity problems (huge!)

**Authority Revocation Lists** exclusively for CA revocation, using "issuing distribution point" extension

**CRL Distribution Points** (Partitioned CRLs) multiple CRLs from one CA, easier management, can point to the location; however must be static

**Enhanced CRL Distribution Points and Redirect CRLs** pointers to CRL partitions, restricted with some scope statement

**Delta CRLs** deltas since last publication of complete list

**Indirect CRLs** sum of CRLs from different CAs (easier verification)

**Certificate Revocation Trees** invented by Valicert, based on Merkle hash trees, represent large revocation lists in efficient manner

## Online systems

Online certificate status protocol (OCSP) RFC2560

**request** protocol version, service request, certificate(s) identifier(s)

**response message** version, name of responder, responses for certificates in request, optional extensions, signature algorithm, signature

**response for certificates** target certificate identifier, status value (good, revoked, unknown), response validity interval (thisUpdate, nextUpdate (empty=always), producedAt), optional extensions

**CRL**

**Certificate Revocation Lists** digitally signed data structures

- contain list of revoked certificates
- integrity and authenticity come from digital signature
- signer usually is the same as issuer of revoked certificate
- can be cached

**CRL v1** X.509 spec, 1998; internal problems:

- scalability problem (could grow immensely large)
- problems with extending
- CRL substitution attack

**CRL v2** adds significant enhancement: *extensions*

## CRL structure

**version** either 2 or none (then 1 assumed)

**signature** algorithm used (like md5WithRSAEncryption)

**issuer** distinguished name of CRL issuer

**thisUpdate** issue time

**nextUpdate** next issue time (equal to when this one expires)

**revokedCertificates** list of certificate serial number, revocation time (and extensions for v2)

**extensions** v2 only

- per entry extensions

**reason code** unspecified, key compromise, CA compromise, affiliation changed, superseded, cessation of operation, certificate hold, remove from CRL, privilege withdrawn...

**invalidity date** time on which the key pair was invalid

**certificate issuer** for redirect CRLs

**hold instruction code** for certificate suspension (oid actions: none, call issuer, reject)

- per CRL extensions

**authority key identifier**

**issuer alternative name**

**CRL number**

**issuing distribution point**

**Delta CRL indicator**

```
% wget http://crl.thawte.com/ThawteServerCA.crl
% openssl crl -inform der -text -noout -in ThawteServerCA.crl
Certificate Revocation List (CRL):
  Version 1 (0x0)
  Signature Algorithm: md5WithRSAEncryption
  Issuer: /C=ZA/ST=Western Cape/L=Cape Town/O=Thawte Consulting
         cc/OU=Certification Services Division/CN=Thawte Server
         CA/emailAddress=server-certs@thawte.com
  Last Update: Jun 28 10:00:01 2005 GMT
  Next Update: Jul 26 10:00:01 2005 GMT
Revoked Certificates:
  Serial Number: 0A42FE
    Revocation Date: May 30 17:38:19 2003 GMT
  Serial Number: 0A486A
    Revocation Date: Jul 15 14:14:40 2004 GMT
  Serial Number: 0A4AFE
    Revocation Date: Jul 29 17:17:57 2003 GMT
[... ]
  Serial Number: 3F83F4
    Revocation Date: Jun 23 10:36:25 2005 GMT
  Signature Algorithm: md5WithRSAEncryption
  48:d2:f3:69:07:c3:94:fe:5f:78:ac:52:05:df:9a:16:27:74:
  80:49:88:64:6f:12:ee:a5:39:9b:7d:20:63:87:d6:4b:b3:6b:
  1d:69:3a:db:ea:3e:00:0f:bb:35:59:ab:4a:68:8c:6a:ff:d8:
  db:f4:79:cb:a6:2e:a8:ef:bc:38:29:cd:29:80:95:76:64:2c:
  6b:0c:74:6f:33:b1:73:5a:c0:81:9e:e8:cf:94:45:ec:0a:63:
  5a:43:6e:e3:fb:38:e4:e0:ae:a3:e8:cf:60:69:a8:54:c6:3d:
  9d:d3:c0:6f:38:a4:88:3b:67:92:e9:2c:b0:63:ec:32:91:0e:
  68:a0
```

## Gargantuan PKI

**infrastructure** certification authority, certification repository, certificate revocation, key backup and recovery, automatic key update, key history management, cross certification, client software

**services** authentication, integrity, confidentiality, secure time stamping, notarization, non-repudiation support, secure data archive, privilege/policy creation, privilege/policy verification

Is it *all* really necessary?

### ssl web and emails

<b>Certification Authority</b>		
<b>Authentication</b>	<b>Integrity</b>	<b>Confidentiality</b>

ssl web and emails **that matter**

Certification Authority		<b>Certificate Revocation</b>
		<b>Client Software</b>
Authentication	Integrity	Confidentiality
	<b>Privilege/Policy Creation</b>	<b>Privilege/Policy Verification</b>

ssl web and emails **that matter** for more institutions

Certification Authority	Certification Repository	Certificate Revocation
Key Backup		
Key History Management	Cross-Certification	Client Software
Authentication	Integrity	Confidentiality
Secure Time Stamping		Non-Repudiation
	Privilege/Policy Creation	Privilege/Policy Verification

ssl web and emails **that matter** for more institutions **governmental**

Certification Authority	Certification Repository	Certificate Revocation
Key Backup	Key Recovery	Automatic Key Update
Key History Management	Cross-Certification	Client Software
Authentication	Integrity	Confidentiality
Secure Time Stamping	Notarization	Non-Repudiation
Secure Data Archive	Privilege/Policy Creation	Privilege/Policy Verification

## PKI Trust Model

Who can you trust?

**trust** defined by ITU-T:

*Entity "A" trusts entity "B" when "A" assumes that "B" will behave exactly as "A" expects*

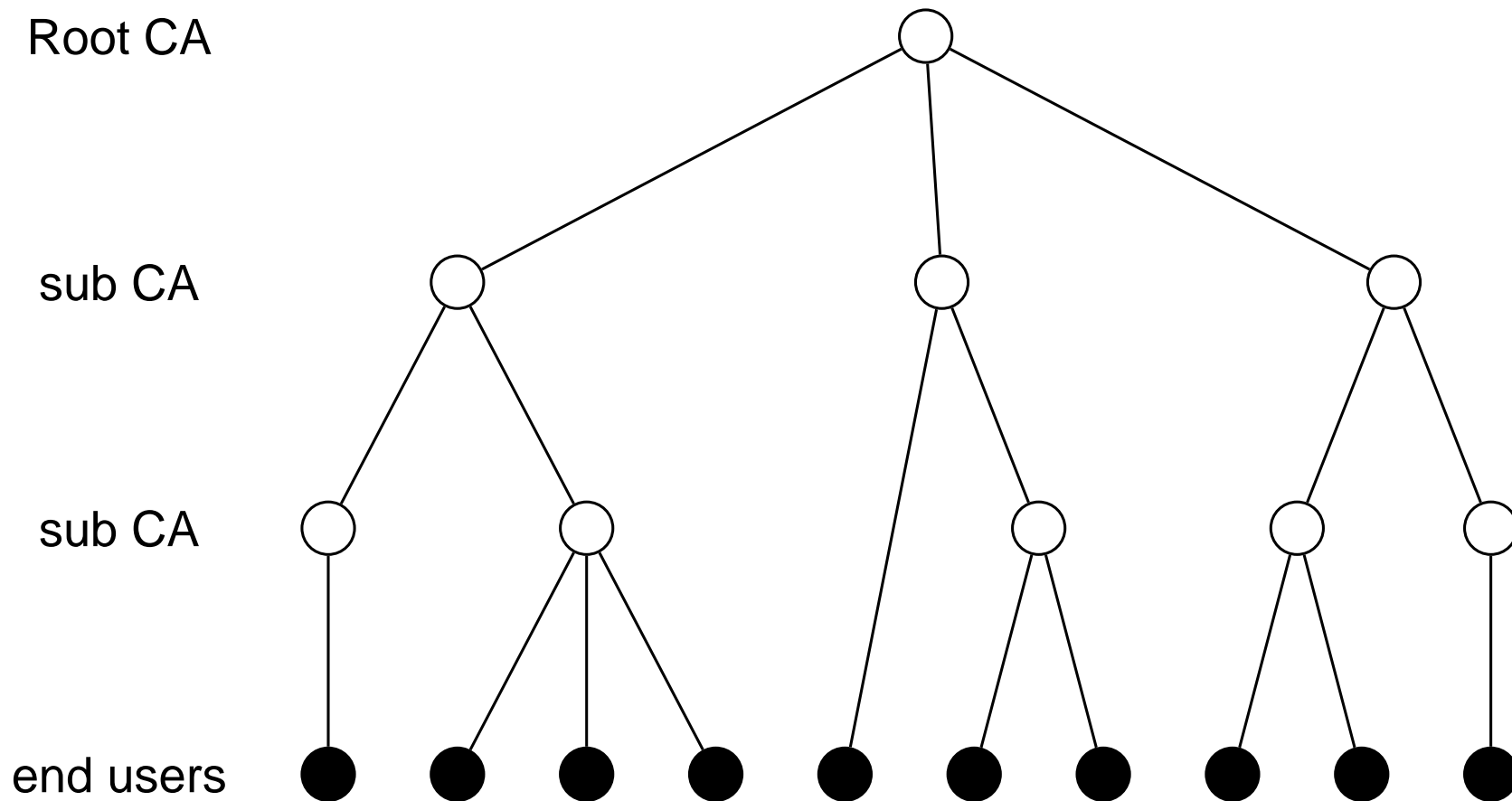
**PKI trust** derives from above:

*End-entity trusts a CA when the end-entity assumes that the CA will establish and maintain an accurate binding of attributes to a public key*

**public-key trust** if you are convinced that public and private keys belong only to a given entity

Assumptions, expectations, human behaviour. . .

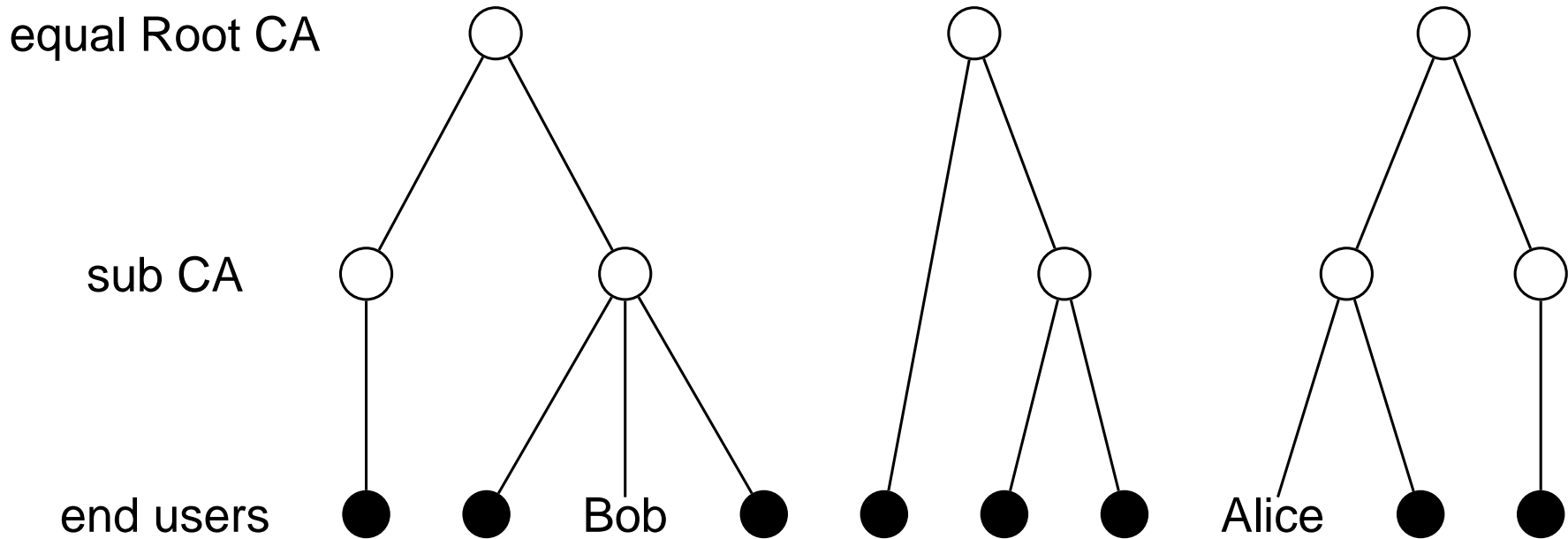
# Strict Hierarchy of Certificate Authorities

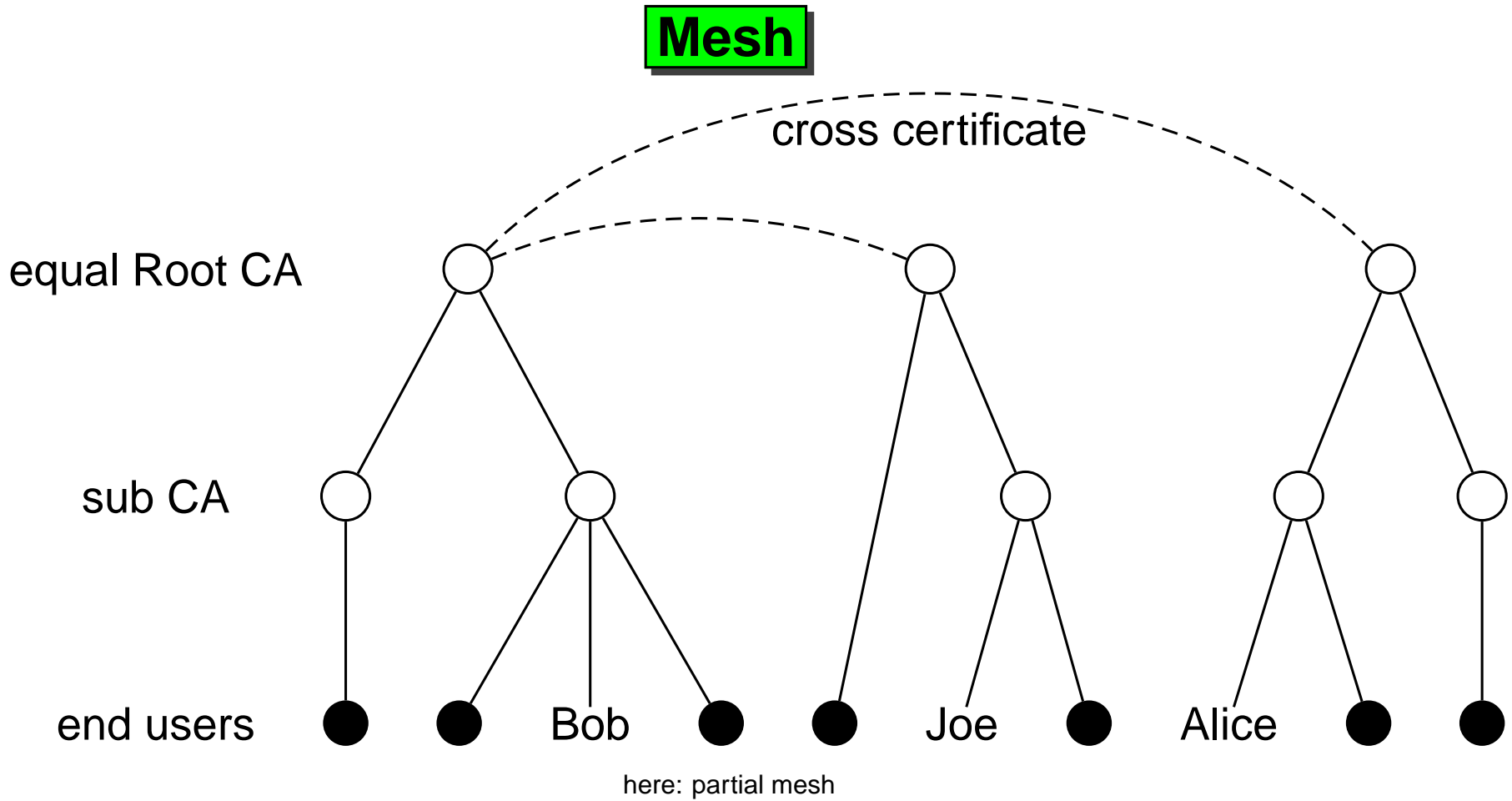


Root CA is *trust anchor*

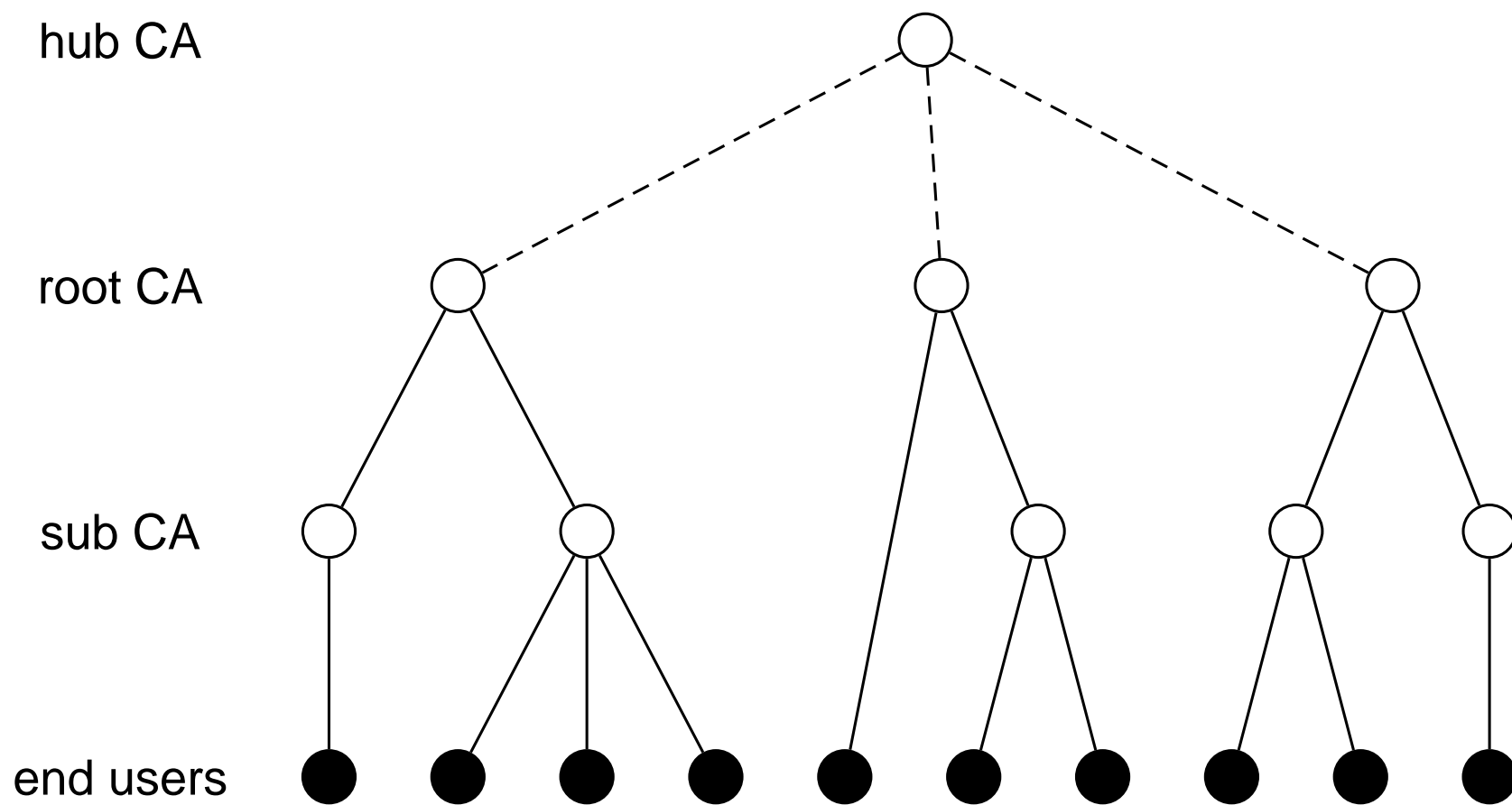
You can create your own self-signed Root CA to play with!

# Distributed Trust Architecture

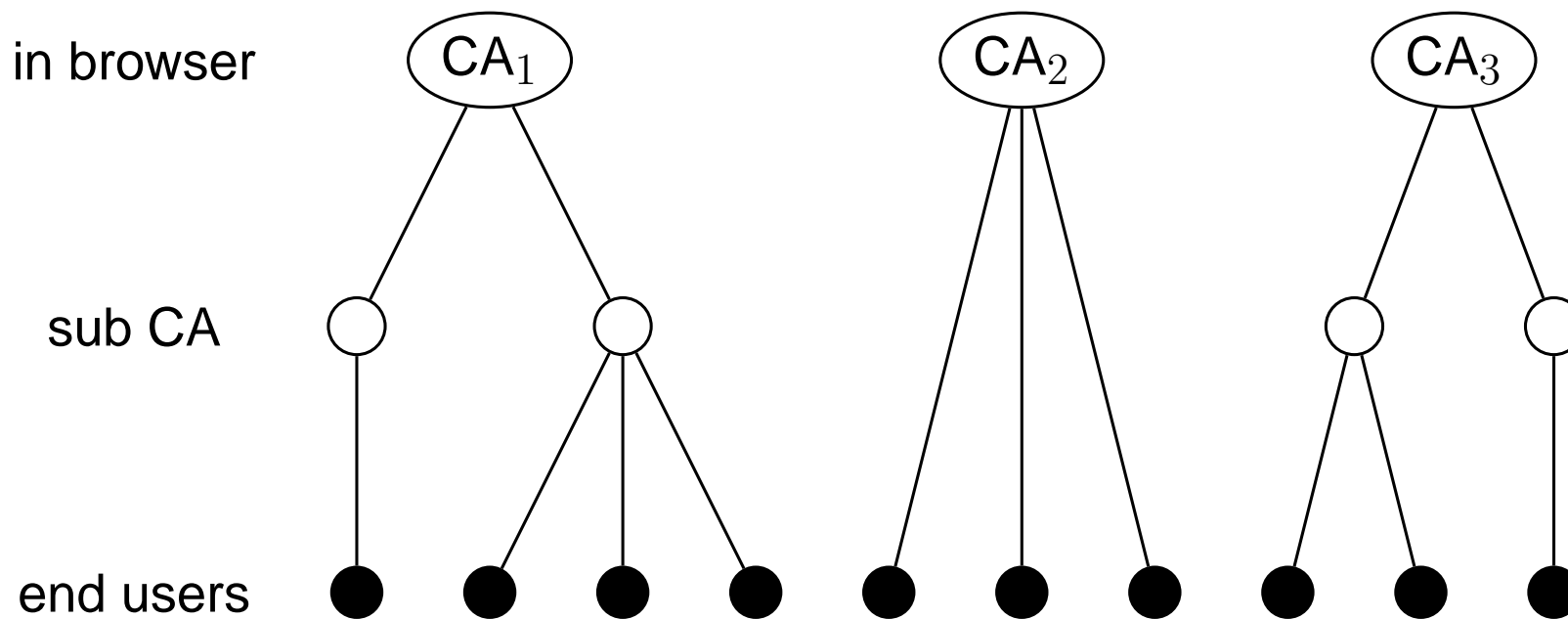




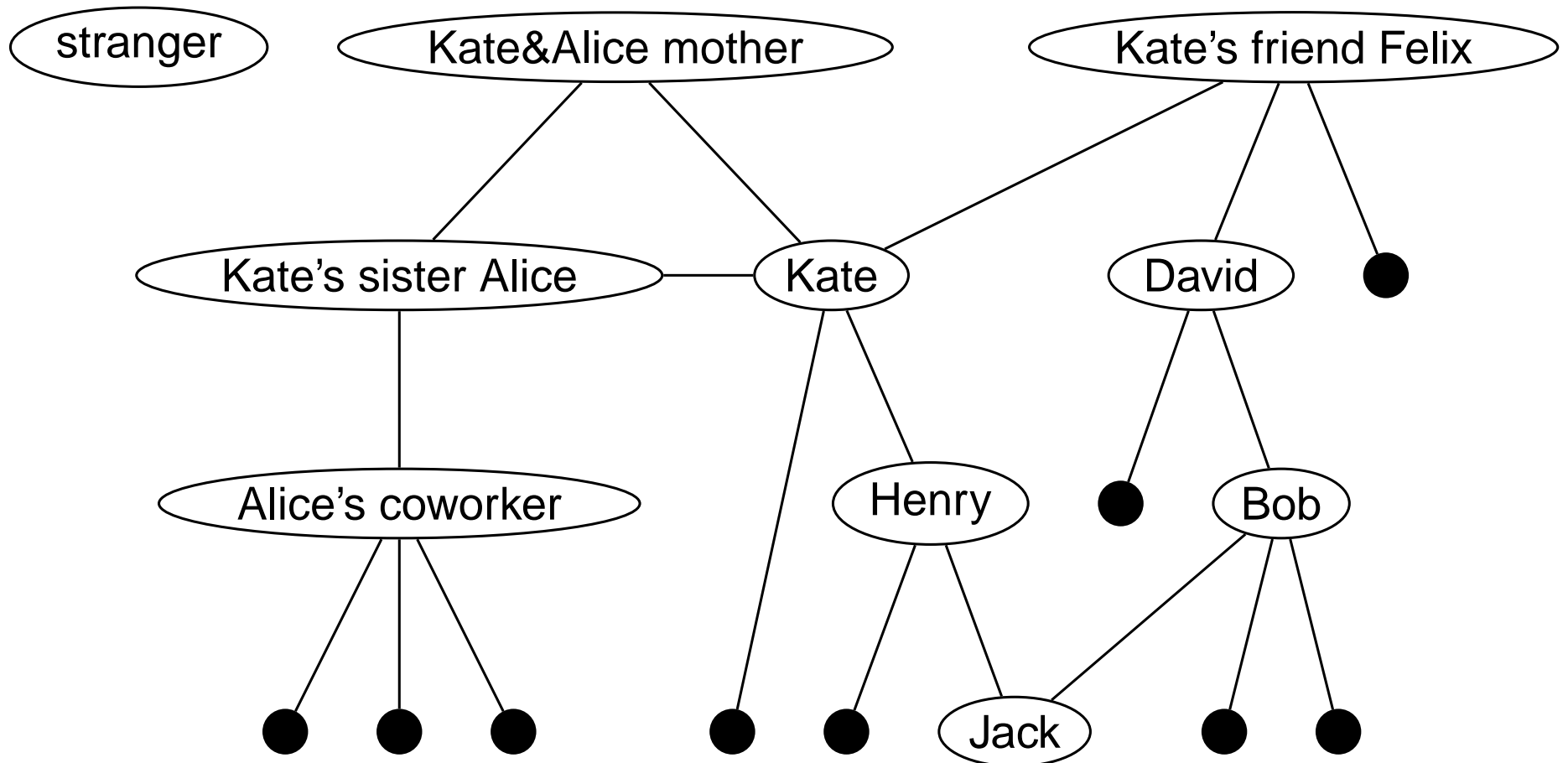
# Hub and Spoke



**Web Model**



# User-centric Trust



PGP model

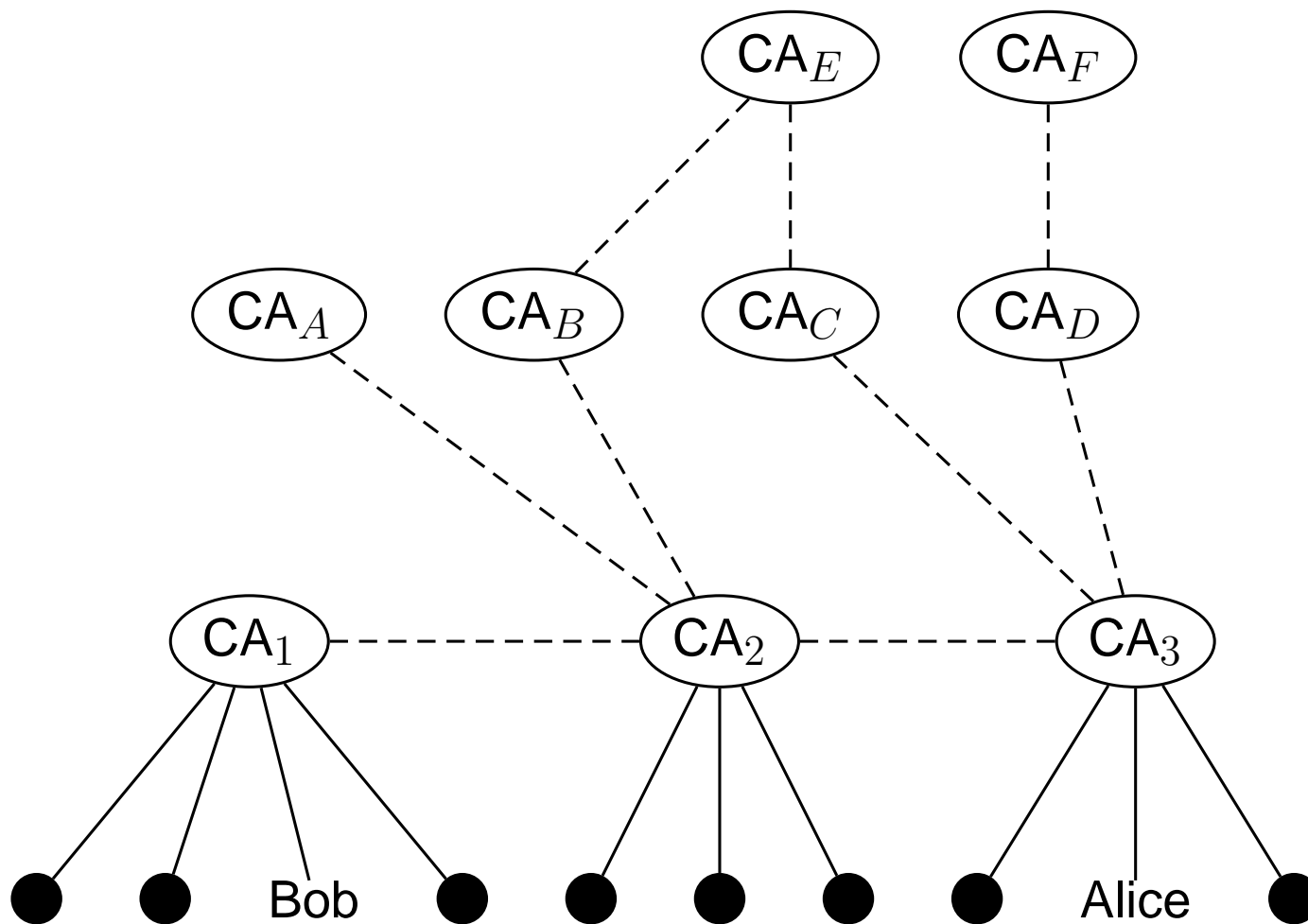
"Six Degrees of Separation"

```
% pgp +verbose=0 -kvv chopin@sgh
Type Bits/KeyID      Date           User ID
pub   1024/D818A489  1995/10/31    Piotr Kucharski <chopin@sgh>
sig           D818A489                Piotr Kucharski <chopin@sgh>
sig           61094A7D             Szymon Sokol <szymon@uci.agh>
sig           56A1520D            Tomasz R. Surmacz <ts@wroc.apk>
% pgp +verbose=0 -kvv ts@wroc
Type Bits/KeyID      Date           User ID
pub   1024/56A1520D  1995/04/23    Tomasz R. Surmacz <ts@wroc.apk>
sig           D818A489                Piotr Kucharski <chopin@sgh>
sig           C38B2AAD             Jan Rychter <jwr@itc.pw>
sig           61094A7D             Szymon Sokol <szymon@uci.agh>
sig           56A1520D            Tomasz R. Surmacz <ts@wroc.apk>
% pgp +verbose=0 -kvv szymon@uci
Type Bits/KeyID      Date           User ID
pub   512/61094A7D  1993/08/26    Szymon Sokol <szymon@uci.agh>
sig           8DF27F3D                <Wojtek.Sylwestrzak@icm>
sig           DA9F6825             Zbigniew Zych <zych@onet>
sig           56A1520D            Tomasz R. Surmacz <ts@wroc.apk>
```

MIT PGP Public Key Server <http://pgp.mit.edu/>

```
% pgp +verbose=0 -kvc chopin@sgh
Type Bits/KeyID      Date           User ID
pub   1024/D818A489  1995/10/31    Piotr Kucharski <chopin@sgh>
Key fingerprint =
      CE B8 CA 5D AC 3B 97 85   94 D5 8A 25 F3 4E 55 C6
```

# Certificate Path building and verification



## Problems with conventional PKI: technology

<http://www.anu.edu.au/people/Roger.Clarke/II/PKIMisFit.html>

**digital signatures problems** third party must check if private key is in good hands, third party must be trustworthy, private key must be subject to strong security (so no one can gain access to it), public key must be appropriate, infrastructure cooperation must be secured, information about private key compromised and/or certificates revoked must be transmitted fast to all (and without network abuse), d.s. rely on hash function, which produce collisions

**public keys** are subject to spoofing (either sent over with the message, stored on ftp/http or in central repository), hence certificates

## Problems with conventional PKI: technology

**authentication process of issuing certificate** (binding public key with some entity) must be secure in all its steps (showing at RA, providing physical id (with photo), providing public key and evidence of being holder of private key (signing in front of RA), *provide evidence that private key is secure*, giving contact point, nominating delivery point for certificate

**CA** must be trusted and there is no easy way to discover if CA was compromised and the certificate issued valid

## Problems with conventional PKI: technology

**X.509v3 certificate** single key-pair per person, 'distinguished name' unique across vast name-space, denies pseudonyms, little choice in how key-pair is generated and in many cases is outside control of person (meaning private key may be not private from beginning), little choice on how private key and certificate is stored (token, diskette), issuer-ownership of certificates and tokens (user is merely licensed to use it); revocation problems;

## Problems with conventional PKI: model of trust

**hierarchical model of trust** trust in CA is not automatic, each CA must be signed by some higher level until some mythical authority in which everyone is assumed to have ultimate trust

**authoritarian** all parties are required to disclose their identity even if it is needed only to communicate eligibility (age, qualifications etc.)

**unique DN** there can be no two "Joe Average", they must be further attributed; it also denies to have more certificates for individuals

## Problems with conventional PKI: private key [in]security

**corporate servers** are broken into, users' private keys can be stolen

**storage** users usually store their private keys on their workstations, which are directly connected to Internet! many ways to discover, copy or use private keys by malware

## Problems with conventional PKI: limited assurance

**assurance** is basically non-existent; CAs try to minimise financial liabilities in "Certificate Policy Statements", even worse, they issue certificates for different prices and different degree of scrutiny applied when reviewing certification requests

**NO assurance** about whether

- the private key was (during certificate issuing) available only to "owner",
- the private key *now* is available only to "owner",
- the private key invocation was done by "owner",
- the private key invocation was done by "owner" freely and consently

## Problems with conventional PKI: privacy issues

**identification** registration requires intrusive demands for documents and even biometrics

**registers** of certificates are full of personal data

**CRLs** should be checked for every transaction – poses a risk of trailing person's e-activity and represents opportunity for an authority to cancel person's identity

**anonymity** increased usage of named certificates may lead to breaking down traditions of anonymity and pseudonymity

## **Problems with conventional PKI: alternatives**

Models different from naive military concept of absolute trust:

**PGP web of trust** no CAs, anyone can issue certificates, fault-tolerance by multiple (paths to) certificates, assurance is relative not absolute, reflects real world

**SPKI/SDSI** global name-space abandoned, removed presumption that certificate is reliably bound to entity, up to relying party to decide; attributes associated with public keys, not identities of real world; supports nyms;

**other** Stefan Brand's Alternative Certificates, Reputation and Brand, Trust-Management systems