Key Management and Certificates

By the power vested in me I now declare this text and this bit string ‘name’ and ‘key’. What RSA has joined, let no man put asunder

— Bob Blakley

Key Management

Key management is the hardest part of cryptography

Two classes of keys

- Short-term session keys (sometimes called ephemeral keys)
  – Generated automatically and invisibly
  – Used for one message or session and discarded
- Long-term keys
  – Generated explicitly by the user

Long-term keys are used for two purposes

- Authentication (including access control, integrity, and non-repudiation)
- Confidentiality (encryption)
  – Establish session keys
  – Protect stored data

Key Management Problems

Key certification

Distributing keys

- Obtaining someone else's public key
- Distributing your own public key

Establishing a shared key with another party

- Confidentiality: Is it really known only to the other party?
- Authentication: Is it really shared with the intended party?

Key storage

- Secure storage of keys

Revocation

- Revoking published keys
- Determining whether a published key is still valid

Key Lifetimes and Key Compromise

Authentication keys

- Public keys may have an extremely long lifetime (decades)
- Private keys/conventional keys have shorter lifetimes (a year or two)

Confidentiality keys

- Should have as short a lifetime as possible

If the key is compromised

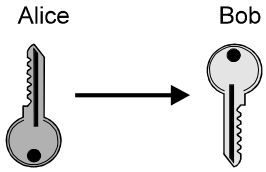
- Revoke the key

Effects of compromise

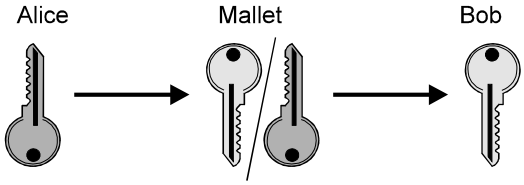
- Authentication: Signed documents are rendered invalid unless timestamped
- Confidentiality: All data encrypted with it is compromised

Key Distribution

Alice retains the private key and sends the public key to Bob



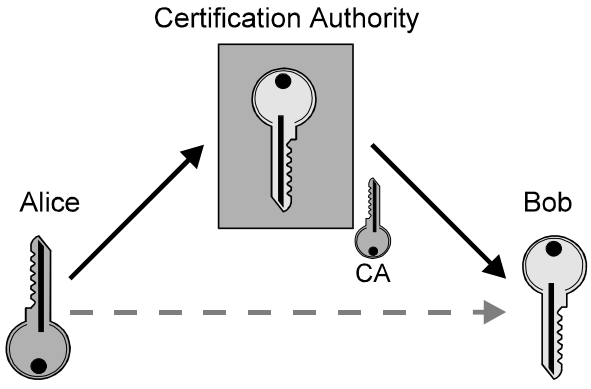
Mallet intercepts the key and substitutes his own key



Mallet can decrypt all traffic and generate fake signed message

Key Distribution (ctd)

A certification authority (CA) solves this problem



CA signs Alice's key to guarantee its authenticity to Bob

- Mallet can't substitute his key since the CA won't sign it

Certification Authorities

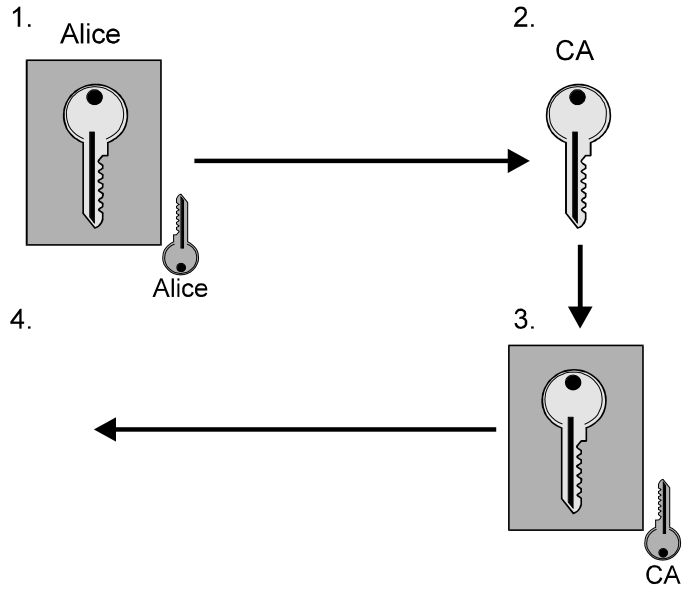
A certification authority (CA) guarantees the connection between a key and an end entity

An end entity is

- A person
- A role (“Director of marketing”)
- An organisation
- A pseudonym
- A piece of hardware or software
- An account (bank or credit card)

Some CA’s only allow a subset of these types

Obtaining a Certificate



Obtaining a Certificate (ctd)

1. Alice generates a key pair and signs the public key and identification information with the private key
   - Proves that Alice holds the private key corresponding to the public key
   - Protects the public key and ID information while in transit to the CA
2. CA verifies Alice's signature on the key and ID information

2a. Optional: CA verifies Alice's ID through out-of-band means

- email/phone callback
- Business/credit bureau records, in-house records

Obtaining a Certificate (ctd)

3. CA signs the public key and ID with the CA key, creating a certificate
   - CA has certified the binding between the key and ID
4. Alice verifies the key, ID, and CA's signature
   - Ensures the CA didn't alter the key or ID
   - Protects the certificate in transit
5. Alice and/or the CA publish the certificate

Role of a CA

Original intent was to certify that a key really did belong to a given party

Role was later expanded to certify all sorts of other things

- Are they a bona fide business?
- Can you trust their web server?
- Can you trust the code they write?
- Is their account in good standing?
- Are they over 18?

When you have a certificate-shaped hammer, everything looks like a nail

Certificate History

Original 1970s research work saw certificates as a one-time assertion about public keys

- "This key is valid at this instant for this person"
- Never put into practice

Certificates in practice were applied to protect access to the X.500 directory

- All-encompassing, global directory run by monopoly telcos

Certificate History (ctd)

Concerns about misuse of the directory

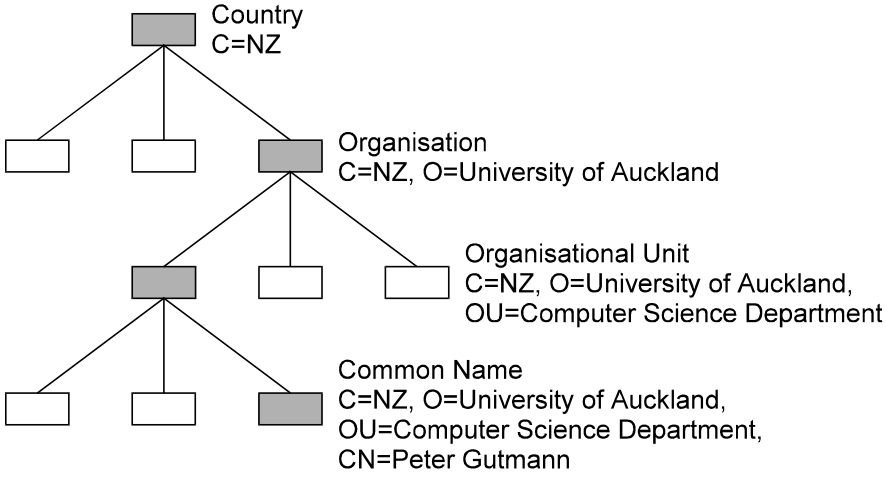
- Companies don't like making their internal structure public
  - Directory for corporate headhunters
- Privacy concerns
  - Directory of single women
  - Directory of teenage children

X.509 certificates were developed as part of the directory access control mechanisms

- Acted as an RSA analog to a password
- Strictly a password replacement, no concept of CAs, key usage, etc

X.500 Naming

X.500 introduced the Distinguished Name (DN), a guaranteed unique name for everything on earth



X.500 Naming (ctd)

Typical DN components

- Country C
- State or province SP
- Locality L
- Organisation O
- Organisational unit OU
- Common name CN

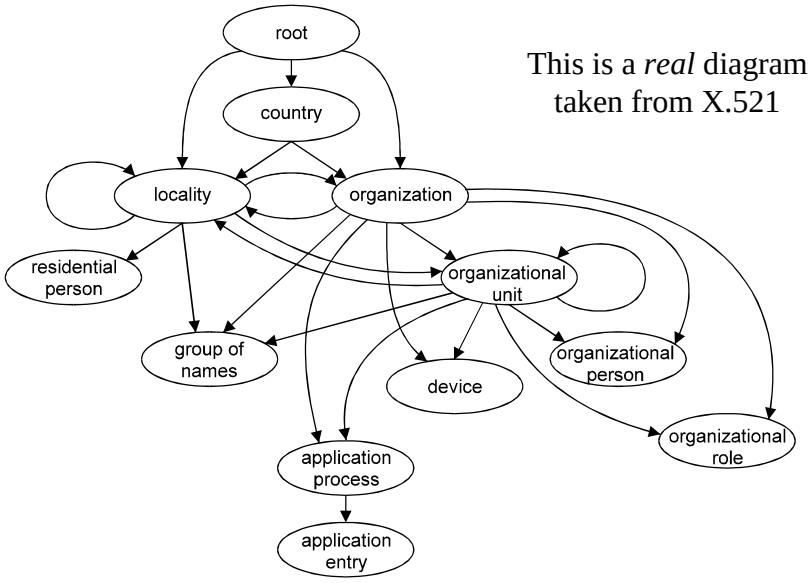
Typical X.500 DN

C=US/L=Area 51/O=Hanger 18/OU=X.500 Standards Designers/CN=John Doe

– When the X.500 revolution comes, your name will be lined up against the wall and shot

Problems with X.500 Names

No-one ever managed to figure out how to make DNs work



This is a *real* diagram taken from X.521

Problems with X.500 Names (ctd)

No clear plan on how to organise the hierarchy

- Attempts were made to define naming schemes, but nothing really worked
- People couldn't even agree on what things like 'localities' were

Hierarchical naming model fits the military and governments, but doesn't work for businesses or individuals

Problems with X.500 Names (ctd)

DNs provide the illusion of order while preserving everone's God-given Freedom to Build a Muddle

Simple problem cases

- Communal living (jails, boarding schools)
- Nomadic peoples
- Merchant ships
- Quasi-permanent non-continental structures (oil towers)
- US APO addresses
- LA phone directory contains > 1,000 people called "Smith" in a nonexistant 90000 area code
  - A bogus address is cheaper than an unlisted number
  - Same thing will happen on a much larger scale if people are forced to provide information (cf cypherpunks login)

Problems with X.500 Names (ctd)

For a corporation, is C, SP, L

- Location of company?
- Location of parent company?
- Location of field office?
- Location of incorporation?

For a person, is C, SP, L

- Place of birth?
- Place of residence/domicile?
  - Dual citizenship
  - Stateless persons
  - Nomads
- Place of work?

Solution: Specify it in the CPS, which no-one reads anyway

DNs in Practice

Public CAs typically set

C = CA country

O = CA name

OU = Certificate type/class

CN = User name

email = User email address

- Some European CAs add oddball components required by local signature laws
- Some CAs modify the DN with a nonce to try and guarantee uniqueness

DNs in Practice (ctd)

Private CAs (organisations or people signing their own certs) typically set any DN fields supported by their software to whatever makes sense for them

- Some software requires that all of { C, O, OU, SP, L, CN } be set
- Resulting certificates contain strange or meaningless entries as people try and guess values, or use dummy values
- Windows 2000 has given up on issuer → subject chaining by names entirely and instead chains by hash of the public key

Solving the DN Problem

Two solutions were informally adopted

1. Users put whatever they felt like into the DN
2. X.509v3 added support for alternative (non-DN) names
   – These are largely ignored in favour of the DN though

General layout for a business-use DN

Country + Organisation + Organisational Unit + Common Name

– C=New Zealand
  O=Dave's Wetaburgers
  OU=Procurement
  CN=Dave Taylor

Solving the DN Problem (ctd)

General layout for a personal-use DN

Country + State or Province + Locality + Common Name

– C=US
SP=California
L=San Francisco
CN=John Doe

There are dozens of other odd things which can be specified

- teletexTerminalIdentifier
- destinationIndicator
- supportedApplicationContext

Luckily these are almost never used

Non-DN Names

X.509 v3 added support for other name forms

- email addresses
- DNS names
- URL's
- IP addresses
- EDI and X.400 names
- Anything else (type+value pairs)

For historical reasons, email addresses are often stuffed into DN's rather than being specified as actual email addresses

Problems with Naming/Identity Certificates

"The user looks up John Smith's certificate in a directory"

- Which directory?
- Which John Smith?

X.509-style PKI turns a key distribution problem into a name distribution problem

- Cases where multiple people in same O, OU have same first, middle, and last name
- Solve by adding some distinguishing value to DN (eg part of SSN)
  - Creates unique DNs, but they're useless for name lookups
  - John Smith 8721 vs John Smith 1826 vs John Smith 3504

Qualified Certificates

Certificate designed to identify a person with a high level of assurance

Precisely defines identification information in order to identify the cert owner in a standardised manner

- Defines additional parameters such as key usage, jurisdiction where certificate is valid, biometric information, etc
- Qualified certificates only apply to natural persons

Some jurisdictions don't allow this type of unique personal identifier

- Any government that can issue this type of identifier can create unpersons by refusing to issue it

Qualified Certificates (ctd)

Allows use of a pseudonym

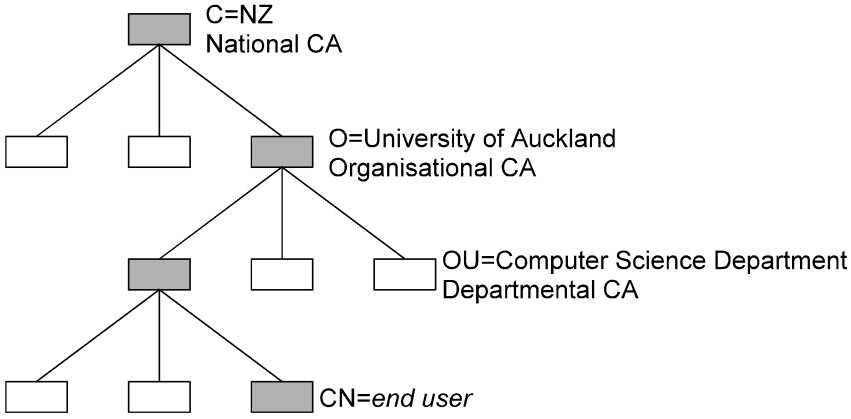
- Pseudonym must be registered, ie can be mapped to a real name via an external lookup
- Most implementations assume every DN contains a CN, so some approximation to a CN must be supplied even if a pseudonym is used

Defines personalData, a new subjectAltName subtype

- Registration authority for personal data information
- Collection of personal data
  - Full (real, not DN) name, gender
  - Date and place of birth
  - Country of residence and/or citizenship
  - Postal address

CA Hierarchy in Theory

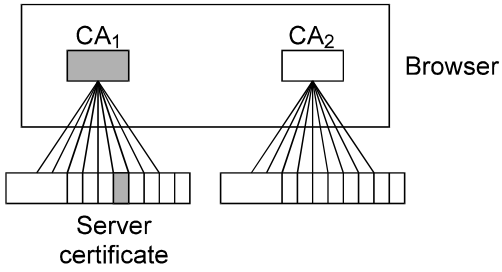
Portions of the X.500 hierarchy have CA's attached to them



Top-level CA is called the root CA, aka "the single point of failure"

CA Hierarchy in Practice

Flat or Clayton's hierarchy

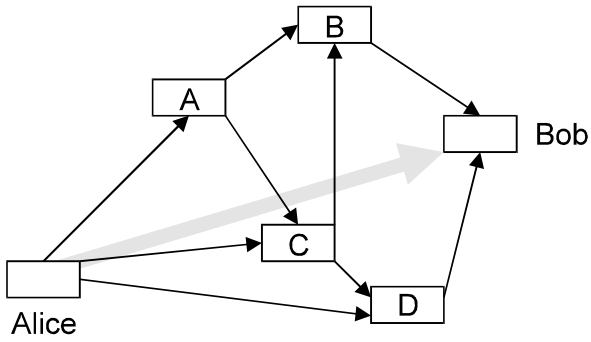


CA certificates are hard-coded into web browsers or email software

- Later software added the ability to add new CAs to the hardcoded initial set

Alternative Trust Hierarchies

PGP web of trust



Bob knows B and D who know A and C who know Alice
$\Rightarrow$ Bob knows the key came from Alice

Web of trust more closely reflects real-life trust models

Key Databases/Directories

Today, keys are stored in

- Flat files (one per key)
- Relational databases
- Proprietary databases (Netscape)
- Windows registry (MSIE)

Pragmatic solution uses a conventional RDBMS

- Already exists in virtually all corporates
- Tied into the existing corporate infrastructure
- Amenable to key storage
  - SELECT key WHERE name='John Doe'
  - SELECT key WHERE expiryDate < today + 1 week

In the future keys might be stored in X.500 directories

The X.500 Directory

The directory contains multiple objects in object classes defined by schemas

A schema defines

- Required attributes
- Optional attributes
- The parent class

Attributes are type-and-value pairs

- Type = CN, value = John Doe
- Type may have multiple values associated with it
- Collective attributes are attributes shared across multiple entries (eg a company-wide fax number)

The X.500 Directory (ctd)

Each instantiation of an object is a directory entry

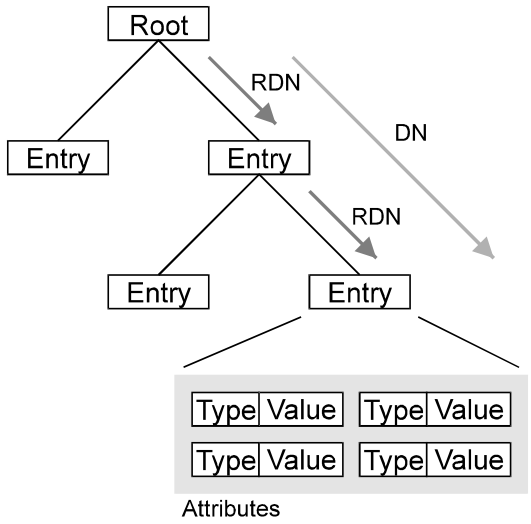
Entries are identified by DN's

- The DN is comprised of relative distinguished names (RDN's) which define the path through the directory

Directory entries may have aliases which point to the actual entry

The entry contains one or more attributes which contain the actual data

The X.500 Directory (ctd)



Data is accessed by DN and attribute type

Searching the Directory

Searching is performed by subtree refinement

- Base specifies where the start in the subtree
- Chop specifies how much of the subtree to search
- Filter specifies the object class to filter on

Example

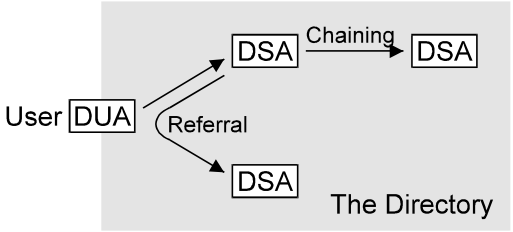
- Base = C=NZ
- Chop = 1 RDN down from the base
- Filter = organisation

Typical application is to populate a tree control for directory browsing

- SELECT name WHERE O=*

Directory Implementation

The directory is implemented using directory service agents (DSA's)



Users access the directory via a directory user agent (DUA)

- Access requests may be satisfied through referrals or chaining

One or more DSA's are incorporated into a management domain

Directory Access

Typical directory accesses:

- Read attribute or attributes from an entry
- Compare supplied value with an attribute of an entry
- List DN's of subordinate entries
- Search entries using a filter
  - Filter contains one or more matching rules to apply to attributes
  - Search returns attribute or attributes which pass the filter
- Add a new leaf entry
- Remove a leaf entry
- Modify an entry by adding or removing attributes
- Move an entry by modifying its DN

LDAP

X.500 Directory Access Protocol (DAP) adapted for Internet use

- Originally Lightweight Directory Access Protocol, now closer to HDAP

Provides access to LDAP servers (and hence DSAs) over a TCP/IP connection

- `bind` and `unbind` to connect/disconnect
- `read` to retrieve data
- `add`, `modify`, `delete` to update entries
- `search`, `compare` to locate information

LDAP (ctd)

LDAP provides a complex hierarchical directory containing information categories with sub-categories containing nested object classes containing entries with one or more (usually more) attributes containing actual values

- In one large-scale interop test the use of a directory for cert storage was found to be the single largest cause of problems

Simplicity made complex

"It will scale up into the billions. We have a pilot with 200 users running already"

LDAP (ctd)

Most practical way to use it is as a simple database

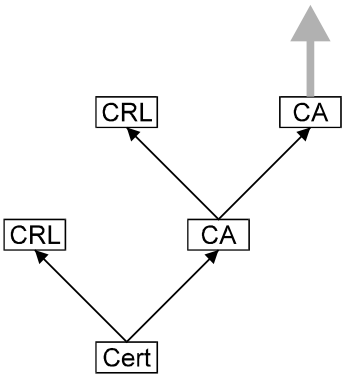
SELECT key WHERE name='John Doe'

LDAP equivalent query

S(&(|(&(objectclass=inetorgperson)(objectclass= organizationalperson))(objectClass= StrongAuthenticationUser))(usercertificate;binary=*) (|(commonname=*name*)(rfc822mailbox=*email address*)))

Certificate Verification using the Directory

Checking works in reverse order to normal lookup



Check certificate
Check certificate's CRL
repeat
    Check CA's certificate
    Check CA's CRL
until root reached

Certificate Revocation

Revocation is managed with a certificate revocation list (CRL), a form of anti-certificate which cancels a certificate

- Equivalent to 1970s-era credit card blacklist booklets
- Relying parties are expected to check CRLs before using a certificate
  - "This certificate is valid unless you hear somewhere that it isn't"

CRL Problems

CRLs don't work

- Violate the cardinal rule of data-driven programming
  "Once you have emitted a datum you can't take it back"
- In transaction processing terms, viewing a certificate as a PREPARE and a revocation as a COMMIT
  – No action can be taken between the two without destroying the ACID properties of the transaction
  – Allowing for other operations between PREPARE and COMMIT results in nondeterministic behaviour
- Blacklist approach was abandoned by credit card vendors 20 years ago because it didn't work properly

CRL Problems (ctd)

CRLs mirror credit card blacklist problems

- Not issued frequently enough to be effective against an attacker
- Expensive to distribute
- Vulnerable to simple DOS attacks
  – Attacker can prevent revocation by blocking CRL delivery

CRLs add further problems of their own

- Can contain retroactive invalidity dates
- CRL issued right now can indicate that a cert was invalid last week
  – Checking that something was valid at time *t* isn't sufficient to establish validity
  – Back-dated CRL can appear at any point in the future
- Destroys the entire concept of nonrepudiation

CRL Problems (ctd)

CA cert revocation is more difficult than end-entity revocation

- One interop test found that revoking a CA cert would require a "system rebuild"
  - Replace the current PKI software with updated software
- Testing of CA cert revocation was deferred until later

CRL Problems (ctd)

Revoking self-signed certificates is even hairier

- Cert revokes itself
- Applications may
  - Accept the CRL as valid and revoke the certificate
  - Reject the CRL as invalid since it was signed with a revoked certificate
  - Crash
- Computer version of Epimenides paradoxon "All Cretans are liars"
  - Crashing is an appropriate response

CRL Problems (ctd)

CRL Distribution Problems

- CRLs have a fixed validity period
  - Valid from *issue date* to *expiry date*
- At *expiry date*, all relying parties connect to the CA to fetch the new CRL
  - Massive peak loads when a CRL expires (DDOS attack)
- Issuing CRLs to provide timely revocation exacerbates the problem
  - 10M clients download a 1MB CRL issued once a minute = ~150GB/s traffic
  - Even per-minute CRLs aren't timely enough for high-value transactions with interest calculated by the minute

CRL Problems (ctd)

- Clients are allowed to cache CRLs for efficiency purposes
  - CA issues a CRL with a 1-hour expiry time
  - Urgent revocation arrives, CA issues an (unscheduled) forced CRL before the expiry time
  - Clients which re-fetch the CRL each time will recognise the cert as expired
  - Clients which cache CRLs won't
  - Users must choose between huge bandwidth consumption/ processing delays or missed revocations

CRL Problems (ctd)

Various ad hoc solutions proposed

- Segment CRLs based on urgency of revocation
  - "Key compromise" issued once a minute
  - "Affiliation changed" issued once a day
  - Possible attacks
    - Substitute one CRL for another
    - Attacker can place key on low-priority CRL before victim can place it on high-priority CRL
- Delta CRLs
  - Short-term CRLs which modify a main CRL
  - Discussion on PKI mailing lists indicates that use of delta CRLs will be an interesting experience

CRL Problems (ctd)

- Stagger CRLs
  - Over-issue CRLs so that multiple overlapping CRLs exist at one time
  - Timeliness guarantees vanish
  - Plays havoc with CRL semantics
    - Cert may or may not appear on any of several CRLs valid at a given time

Bypassing CRLs

SET sidesteps CRL problems entirely

- End user certificates are "revoked" by cancelling the credit card
- Merchant certificates are "revoked" by marking them as invalid at the acquiring bank
- Payment gateways have short-term certificates which are quickly replaced

Account Authority Digital Signatures (AADS/X9.59)

- Public key is tied to an existing account
- Revocation is handled by removing the key
- Matches 1970s model of certificates: "This key is valid at this instant for this account"

Certificate Revocation (ctd)

Many applications require prompt revocation

- CA's (and X.509) don't really support this
- CA's are inherently an offline operation

Requirements for online checks

- Should return a simple boolean value "Certificate is valid/not valid right now"
- Can return additional information such as "Not valid because …"
- Historical query support is also useful, "Was valid at the time the signature was generated"
- Should be lightweight (c.f. CRLs, which can require fetching and parsing a 10,000 entry CRL to check the status of a single certificate)

Online Status Checking

Online Certificate Status Protocol, OCSP

- Inquires of the issuing CA whether a given certificate is still valid
  - Acts as a simple responder for querying CRL's
  - Still requires the use of a CA to check validity
- OCSP acts as a selective CRL protocol
  - Standard CRL process: "Send me a CRL for everything you've got"
  - OCSP process: "Send me a pseudo-CRL/OCSP response for only these certs"
    - Lightweight pseudo-CRL avoids CRL size problems
  - Reply is created on the spot in response to the request
    - Ephemeral pseudo-CRL avoids CRL validity period problems

Online Status Checking (ctd)

- Returned status values are non-orthogonal
  - Status = "good", "revoked", or "unknown"
  - "Not revoked" doesn't necessarily mean "good"
  - "Unknown" could be anything from "Certificate was never issued" to "It was issued but I can't find a CRL for it"
  - Can submit a JPEG image or Excel spreadsheet and all the responder can say is "unknown"

Online Status Checking (ctd)

- Problems are due in some extent to the CRL-based origins of OCSP
  - CRL can only report a negative result
  - "Not revoked" doesn't mean a cert was ever issued
  - Some OCSP implementations will report "I can't find a CRL" as "Good"
  - Some relying party implementations will assume "revoked" $\Rightarrow$ "not good", so any other status = "good"
  - Much debate among implementors about OCSP semantics

Online Status Checking (ctd)

Other protocols

- Simple Certificate Validation Protocol (SCVP)
  - Relying party submits a full chain of certificates
  - Server indicates whether the chain can be verified
  - Aimed mostly at thin clients
- Data Validation and Certification Server Protocols (DVCS)
  - Provides facilities similar to SCVP disguised as a general third-party data validation mechanism
- Integrated CA Services Protocol (ICAP)
- Real-time Certificate Status Protocol (RCSP)
- Web-based Certificate Access Protocol (WebCAP)
- Open CRL Distribution Protocol (OpenCDP)

Online Status Checking (ctd)

- Directory Supported Certificate Status Options (DCS)
- Data Certification Server (also DCS)
- Delegated Path Validation (DPV)
  - Offshoot of the SCVP/DVCS debate and an OCSP alternative OCSP-X
- Many, many more
  - Protocol debate has been likened to religious sects arguing over differences in dogma

Online Status Checking (ctd)

Online protocols place an enormous load on the CA

- CA must carefully protect their signing keys

… but …

- CA must be able to sign *x*,000 status requests per second
- CRL is inherently a batch operation
  - Once an hour, scan a database table and sign the resulting list
- Online status protocols have a high processing overhead
  - For each query, check for a revocation and produce a signed response
  - By their very nature, it's not possible to pre-generate responses, since they must be fresh

Rev./Status Checking in the Real World

CA key compromise: Everyone finds out

- Sun handled revocation of their CA key via posts to mailing lists and newsgroups

SSL server key compromise: Noone finds out

- Stealing the keys from a typical poorly-secured server isn't hard (c.f. web page defacements)
- Revocation isn't necessary since certificates are included in the SSL handshake
  - Just install a new certificate

email key compromise: Who cares?

- If necessary, send a copy of your new certificate to everyone in your address book

Rev./Status Checking in the Real World (ctd)

In practice, revocation checking is turned off in user software

- Serves no real purpose, and slows everything down a lot

Possible alternative revocation techniques

- Self-signed revocation (suicide note)
- Certificate of health/warrant of fitness for certificates (anti-CRL)

Certificate of health provides better proof than CRLs

- CRL is a negative statement
- Anti-CRL is a positive statement
- Proving a negative is much harder than proving a positive

Rev./Status Checking in the Real World (ctd)

PKI researchers like to tinker with revocation in the same way that petrol-heads tinker with car engines

Anyone who can figure out how to make revocation work, please see me afterwards

Revocation as Distributed Trans.Processing

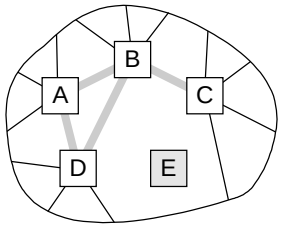
View revocation as a distributed transaction processing problem

- Allows analysis of requirements and solution using established TP mechanisms
- Goal is to distribute certificate status information in a reliable, consistent manner to all parties in the presence of hardware and software failures
- All users in a closed community are presented with a guaranteed-consistent view of certificate information
  - Meets the online status check requirements given earlier

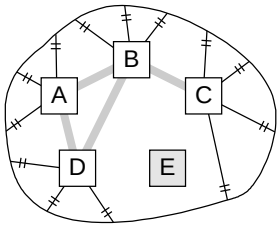
Revocation as Distributed TP (ctd)

Managing distributed status information

Initially, all hosts (except E, which is down) maintain a standard view of a valid cert



Certificate = valid

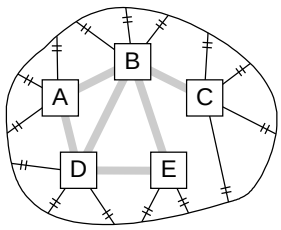


Certificate = invalid

Certificate is invalidated, atomic update propagates across all parties

Revocation as Distributed TP (ctd)

Crashed server is restarted and also updates its state



Update propagation

- In X.509 terms this is equivalent to propagating a CRL to all relying parties simultaneously using only a single transaction
- Since transaction times are recorded, this system can also resolve historical queries
  - "Was this cert valid at time *t*?"
  - "Was this cert valid at the time it signed this document?"

Key Backup/Archival

Need to very carefully balance security vs backup requirements

- Every extra copy of your key is one more failure point
- Communications and signature keys never need to be recovered — generating a new key only takes a minute or so
- Long-term data storage keys should be backed up

Never give the entire key to someone else

- By extension, never use a key given to you by someone else (eg generated for you by a third party)

Key Backup/Archival (ctd)

Use a threshold scheme to handle key backup

- Break the key into *n* shares
- Any *m* of *n* shares can recover the original
- Store each share in a safe, different location (locked in the company safe, with a solicitor, etc)
- Shares can be reconstructed under certain conditions (eg death of owner)

Defeating this setup requires subverting multiple shareholders

Never give the entire key to someone else

Never give the key shares to an outside third party

Key Destruction

Ensure all copies of a private key are destroyed

- Is *every* copy really gone?

Public keys may need to survive private keys by quite some time

- Signature on 20-year mortgage

Long-term key ownership can be a thorny issue

- CA goes bankrupt and auctions off keys
  - c.f. bankrupt dot-coms selling user lists after they promised not to
  - Only asset the CA had left
  - Bidding quickly shot up to rather high values
- Do you want a third-party CA issuing your corporate certs?

Certificate Structure

Usually either the subject name or issuer and serial number identify the certificate

Validity field indicates when certificate renewal fee is due

Certificate Structure (ctd)

Typical certificate

- Serial Number = 177545
- Issuer Name = Verisign
- ValidFrom = 12/09/98
- ValidTo = 12/09/99
- Subject Name = John Doe
- Public Key = RSA public key

Experience with PEM showed that X.509v1 didn't work properly

- X.509v3 added certificate extensions to augment X.509v1/v2 certificates

Certificate Extensions

Extensions consist of a type-and-value pair, with optional critical flag

Critical flag is used to protect CA's against assumptions made by software which doesn't implement support for a particular extension

- If flag is set, extension must be processed (if recognised) or the certificate rejected
- If flag is clear, extension may be ignored

Ideally, implementations should process and act on all components of all fields of an extension in a manner which is compliant with the semantic intent of the extension

Certificate Extensions (ctd)

Actual definitions of critical flag usage are extremely vague

- X.509: Noncritical extension "is an advisory field and does not imply that usage of the key is restricted to the purpose indicated"
- PKIX: "CA's are required to support constraint extensions", but "support" is never defined
- S/MIME: Implementations should "correctly handle" certain extensions
- MailTrusT: "non-critical extensions are informational only and may be ignored"
- Verisign: "all persons shall process the extension... or else ignore the extension"

Certificate Extensions (ctd)

Extensions come in two types

Usage/informational extensions

- Provide extra information on the certificate and its owner

Constraint extensions

- Constrain the user of the certificate
- Act as a Miranda warning ("You have the right to remain silent, you have the right to an attorney, ...") to anyone using the certificate

Certificate Usage Extensions

Key Usage

- Defines the purpose of the key in the certificate

digitalSignature

- Short-term authentication signature (performed automatically and frequently)
- "This key can sign any kind of document…
  … except one that happens to look like an X.509 certificate"

nonRepudiation

- Binding long-term signature (performed consciously)
- Another school of thought holds that nonRepudiation acts as an additional service on top of digitalSignature
- Certificate profiles are split roughly 50:50 on this

Certificate Usage Extensions (ctd)

keyEncipherment

- Exchange of encrypted session keys (RSA)

keyAgreement

- Key agreement (DH)

keyCertSign/cRLSign

- Signature bits used by CA's

No-one really knows what the nonRepudiation bit signifies

- Asking 8 different people will produce 10 different responses
- c.f. crimeFree bit
  - "This certificate will be used for transactions which are not a perpetration of fraud or other illegal activities"

Certificate Usage Extensions (ctd)

- Possible definition: "Nonrepudiation is anything which fails to go away when you stop believing in it"
  - If you can convince someone it's not worth repudiating a signature, you have nonrepudiation
  - Have them sign a legal agreement promising not to do it
  - Convince them that the smart card they used is infallible and it's not worth going to court over
  - Threaten to kill their kids
- The only definitive statement which can be made upon seeing the NR bit set is "The subscriber asked the issuing CA to set this bit"
- Suggestion that CAs set this bit at random just to prevent people from arguing that its presence has a meaning

Certificate Usage Extensions (ctd)

Extended Key Usage

Extended forms of the basic key usage fields

- serverAuthentication
- clientAuthentication
- codeSigning
- emailProtection
- timeStamping

Certificate Usage Extensions (ctd)

Two interpretations of what extended key usage values mean when set in a CA certificate

- Certificate can be used for the indicated usage
  - Interpretation used by PKIX, some vendors
- Certificate can issue certificates with the given usage
  - Interpretation used by Netscape, Microsoft, other vendors

Netscape cert-type

- An older Netscape-specific extension which performed the same role as keyUsage, extKeyUsage, and basicConstraints

Certificate Usage Extensions (ctd)

Private Key Usage Period

Defines start and end time in which the private key for a certificate is valid

- Signatures may be valid for 10-20 years, but the private key should only be used for a year or two

Alternative Names

Everything which doesn't fit in a DN

- rfc822Name
  - email address, `dave@wetaburgers.com`
- dNSName
  - DNS name for a machine, `ftp.wetaburgers.com`

Certificate Usage Extensions (ctd)

- uniformResourceIdentifier
  - URL, `http://www.wetaburgers.com`
- iPAddress
  - 202.197.22.1 (encoded as CAC51601)
- x400Address, ediPartyName
  - X.400 and EDI information
- directoryName
  - Another DN, but containing stuff you wouldn't expect to find in the main certificate DN
  - Actually the alternative name is a form called the GeneralName, of which a DN is a little-used subset
- otherName
  - Type-and-value pairs (type=MPEG, value=MPEG-of-cat)

Certificate Usage Extensions (ctd)

Certificate Policies

Originally implicit in PEM (X.509v1) certificates

- Policy was taken from the Policy Certification Authority which issued the certificate

Information on the CA policy under which the certificate is issued

- Policy identifier
- Policy qualifier(s)
- Explicit text ("This certificate isn't worth the paper it's not printed on")

Certificate Usage Extensions (ctd)

Defines/constrains what the *CA* does, not what the *user* does

- Passport issuer can't constrain how a passport is used
- Driver's licence issuer can't constrain how a driver's licence is used
- Social Security Number issuer can't even constrain how an SSN is (mis-)used

Certificate Usage Extensions (ctd)

X.509 delegates most issues of certificate semantics or trust to the CA's policy

- Many policies serve mainly to protect the CA from liability
  - "Verisign disclaims any warranties... Verisign makes no representation that any CA or user to which it has issued a digital ID is in fact the person or organisation it claims to be... Verisign makes no assurances of the accuracy, authenticity, integrity, or reliability of information"
- Effectively these certificates have null semantics
- If CAs didn't do this, their potential liability would be enormous
  - Universal ID certs → universal liability
  - Closed PKIs restrict this problem to manageable levels

Certificate Usage Extensions (ctd)

Policy Mappings

- Maps one CA's policy to another CA
- Allows verification of certificates issued under other CA policies
  - "For verification purposes we consider our CA policy to be equivalent to the policy of CA *x*"
- Mapping of constraints is left hanging

Certificate Constraint Extensions

Basic Constraints

Whether the certificate is a CA certificate or not

- Prevents users from acting as CAs and issuing their own certificates
- Redundant, since keyUsage specifies the same thing in a more precise manner
- Much confusion over its use in non-CA certificates
  - German ISIS profile mandates its use
  - Italian profile forbids its use

Certificate Constraint Extensions (ctd)

Name Constraints

Constrain the DN subtree under which a CA can issue certificates

- Constraint of C=NZ, O=University of Auckland would enable a CA to issue certificates only for the University of Auckland
- Main use is to balkanize the namespace so a CA can buy or license the right to issue certificates in a particular area
- Constraints can also be applied to email addresses, DNS names, and URLs

Certificate Constraint Extensions (ctd)

Policy Constraints

Can be used to disable certificate policy mappings

- Policy = "For verification purposes we consider our CA policy to be equivalent to the policy of CA *x*"
- Policy constraint = "No it isn't"

Certificate Profiles

X.509 is extremely vague and nonspecific in many areas

- To make it usable, standards bodies created certificate profiles which nailed down many portions of X.509

PKIX

Internet PKI profile

- Requires certain extensions (basicConstraints, keyUsage) to be critical
  - Doesn't require basicConstraints in end entity certificates, interpretation of CA status is left to chance
- Uses digitalSignature for general signing, nonRepudiation specifically for signatures with nonRepudiation
- Defines Internet-related altName forms like email address, DNS name, URL

Certificate Profiles (ctd)

FPKI

(US) Federal PKI profile

- Requires certain extensions (basicConstraints, keyUsage, certificatePolicies, nameConstraints) to be critical
- Uses digitalSignature purely for ephemeral authentication, nonRepudiation for long-term signatures
- Defines (in great detail) valid combinations of key usage bits and extensions for various certificate types

MISSI

US DoD profile

- Similar to FPKI but with some DoD-specific requirements (you'll never run into this one)

Certificate Profiles (ctd)

ISO 15782

Banking — Certificate Management Part 1: Public Key Certificates

- Uses digitalSignature for entity authentication and nonRepudiation strictly for nonrepudiation (leaving digital signatures for data authentication without nonrepudiation hanging)
- Can't have more than one flag set

Canada

- digitalSignature or nonRepudiation must be present in all signature certs
- keyEncipherment or dataEncipherment must be present in confidentiality certs

Certificate Profiles (ctd)

SEIS

Secured Electronic Information in Society

- Leaves extension criticality up to certificate policies
- Uses digitalSignature for ephemeral authentication and some other signature types, nonRepudiation specifically for signatures with nonRepudiation
  - nonRepudiation can't be combined with other flags
  - Requires three separate keys for digital signature, encryption, and nonrepudiation
- Disallows certain fields (policy and name constraints)

Certificate Profiles (ctd)

TeleTrusT/MailTrusT

German MailTrusT profile for TeleTrusT (it really is capitalised that way)

- Requires keyUsage to be critical in some circumstances
- Uses digitalSignature for general signatures, nonRepudiation specifically for signatures with nonRepudiation

ISIS

German Industrial Signature Interoperability Spec

- Only allows some combinations of key usage bits
- ISIS extensions should be marked non-critical even if their semantics would make them critical
- Requires authorityCertIssuer/SerialNumber instead of authorityKeyIdentifier

Certificate Profiles (ctd)

Australian Profile

Profile for the Australian PKAF

- Requires certain extensions (basicConstraints, keyUsage) to be critical
- Defines key usage bits (including digitalSignature and nonRepudiation) in terms of which bits may be set for each algorithm type
- Defines (in great detail) valid combinations of key usage bits and extensions for various certificate types

German Profile

Profile to implement the German digital signature law

- Requires that private key be held only by the end user

Certificate Profiles (ctd)

SIRCA Profile

(US) Securities Industry Association

- Requires all extensions to be non-critical
- Requires certificates to be issued under the SIA DN subtree

Microsoft Profile (de facto profile)

- Rejects certificates with critical extensions
- Always seems to set nonRepudiation flag when digitalSignature flag set
- Ignores keyUsage bit
- Treats all certificate policies as the hardcoded Verisign policy

Certificate Profiles (ctd)

Many, many more

> You can't be a real country unless you have a beer and an airline. It helps if you have some kind of a football team, or some nuclear weapons, but at the very least you need a beer.
>
> — Frank Zappa

> And an X.509 profile.
>
> — Peter Gutmann

Need to

- Ensure CA issues certificates conformant to the profile
- Ensure CA software conforms to the profile
- Ensure relying party software conforms to the profile
- Extensively test both to ensure they really do this (rather than just having the vendor claim they do this)

Setting up a CA

No-one makes money running a CA

- You make money by selling CA services and products

Typical cost to set up a proper CA from scratch: $1M

Writing the policy/certificate practice statement (CPS) requires significant effort

Getting the top-level certificate (root certificate) installed and trusted by users can be challenging

- Root certificate is usually self-signed

Bootstrapping a CA

Get your root certificate signed by a known CA

- Your CA's certificate is certified by the existing CA
- Generally requires becoming a licensee of the existing CA
- Your CA is automatically accepted by existing software

Get users to install your CA certificate in their applications

- Difficult for users to do
- Specific to applications and OSes
- Not transparent to users
- No trust mechanism for the new certificate

Bootstrapping a CA (ctd)

Publish your CA certificate(s) by traditional means

- Global Trust Register, `http://www.cl.cam.ac.uk/Research/Security/Trust-Register/`
- Book containing register of fingerprints of the world's most important public keys
- Implements a top-level CA using paper and ink

Install custom software containing the certificate on user PC's

- Even less transparent than manually installing CA certificates
- No trust mechanism for the new certificate

Business Expectations of a CA

Current work follows the "if you build it, they will (might) come" model

- Industry (particularly governments) make great testbeds for PKI experimentation
  - They'll even pay you for it!

Survey of US businesses revealed that they require CA's to be insurable

- Must be possible to quantify risk reliably enough to make meaningful warranties
- c.f. Verisign's null-semantics certificates

Business Expectations of a CA (ctd)

Two approaches to this problem:

1. Practical solution: CA has only two warranted responsibilities
    1. Ensure each name is unique
    2. Protect the CA's key(s)
        – Interpreting the certificate is left to the relying party
2. Legal solution: If you do *x*, the government will indemnify you
    - *x* expands to "jump through all the hoops defined in this digital signature law"
    - Type, size, and number of hoops varies from country to country

CA Business Model

Free email certs

- Noone will pay for them
- Clown suit certs

SSL certs run as a protection racket

- Buy our certs at US$200/kB/year or your customers will be scared away
- Actual CA advertising:

    If you fail to renew your Server ID prior to the expiration date, operating your Web site will become far riskier than normal […] your Web site visitors will encounter multiple, intimidating warning messages when trying to conduct secure transactions with your site. This will likely impact customer trust and could result in lost business for your site.

CA consulting services

Finding a Workable Business Model

PKI requires of the user

- Certificate management software to be installed and configured
- Payment for each certificate
- Significant overhead in managing keys and certificates

PKI provides to the user

- "…disclaims any warranties... makes no representation that any CA or user to which it has issued a digital ID is in fact the person or organisation it claims to be... makes no assurances of the accuracy, authenticity, integrity, or reliability of information"

Finding a Workable Business Model (ctd)

A PKI is not just another IT project

- Requires a combined organisational, procedural, and legal approach
- Staffing requires a skilled, multidisciplinary team
- Complexity is enormous
  - Initial PKI efforts vastly underestimated the amount of work involved
  - Current work is concentrating on small-scale pilots to avoid this issue

To be accepted, a PKI must provide perceived value

- Failure to do so is what killed SET
- Noone has really figured out a PKI business model yet

CA Policies

Serves two functions

- Provides a CA-specific mini-profile of X.509
- Defines the CA terms and conditions/indemnifies the CA

CA policy may define

- Obligations of the CA
  - Checking certificate user validity
  - Publishing certificates/revocations
- Obligations of the user
  - Provide valid, accurate information
  - Protect private key
  - Notify CA on private key compromise

CA Policies (ctd)

- List of applications for which issued certificates may be used/may not be used
- CA liability
  - Warranties and disclaimers
- Financial responsibility
  - Indemnification of the CA by certificate users
- Certificate publication details
  - Access mechanism
  - Frequency of updates
  - Archiving
- Compliance auditing
  - Frequency and type of audit
  - Scope of audit

CA Policies (ctd)

- Security auditing
  - Which events are logged
  - Period for which logs are kept
  - How logs are protected
- Confidentiality policy
  - What is/isn't considered confidential
  - Who has access
  - What will be disclosed to law enforcement/courts

CA Policies (ctd)

- Certificate issuing
  - Type of identification/authentication required for issuance
  - Type of name(s) issued
  - Resolution of name disputes
  - Handling of revocation requests
    - Circumstances under which a certificate is revoked, who can request a revocation, type of identification/authentication required for revocation, how revocation notices are distributed
- Key changeover
  - How keys are rolled over when existing ones expire
- Disaster recovery

CA Policies (ctd)

- CA security
  - Physical security
    - Site location, access control, fire/flood protection, data backup
  - Personnel security
    - Background checks, training
  - Computer security
    - OS type used, access control mechanisms, network security controls
  - CA key protection
    - Generation, key sizes, protection (hardware or software, which protection standards are employed, key backup/archival, access/control over the key handling software/hardware)
- Certificate profiles
  - Profile amendment procedures
  - Publication

CA's and Scaling

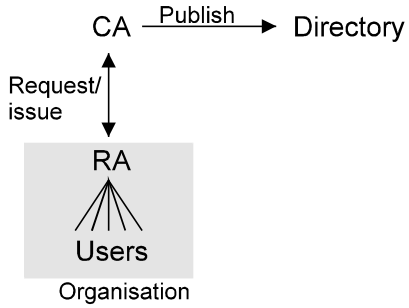
The standard certification model involves direct user interaction with a CA

This doesn't scale well

- CA has to verify details for each user
- Processing many users come from a similar background (eg a single organisation) results in unnecessary repeated work

RA's

Registration authorities offload user processing and checking from the CA



RA acts as a trusted intermediary

- RA has a trusted relationship with CA
- RA has access to user details

Timestamping

Certifies that a document existed at a certain time

Used for added security on existing signatures

- Timestamped countersignature proves that the original signature was valid at a given time
- Even if the original signatures key is later compromised, the timestamp can be used to verify that the signature was created before the compromise

Requires a data format which can handle multiple signatures

- Only PGP keys and S/MIME signed data provide this capability

Cross-Certification

Original X.500-based scheme envisaged a strict hierarchy rooted at the directory root

- PEM tried (and failed) to apply this to the Internet

Later work had large numbers of hierarchies

- Many, many flat hierarchies
- Every CA has a set of root certificates used to sign other certificates in relatively flat trees

What happens when you're in hierarchy A and your trading partner is in hierarchy B?

Solution: CAs cross-certify each other

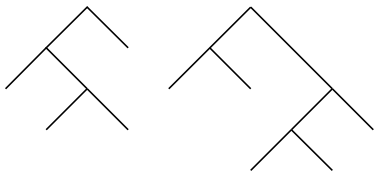
- A signs B's certificate
- B signs A's certificate

Cross-Certification (ctd)

Problem: Each certificate now has *two* issuers

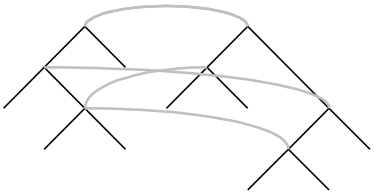
- All of X.509 is based on the fact that there's a unique issuer
- Toto, I don't think we're in X.509 any more

With further cross-certification, re-parenting, subordination of one CA to another, revocation and re-issuance/ replacement, the hierarchy of trust…

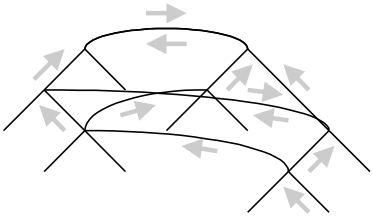


Cross-Certification (ctd)

…becomes the spaghetti of doubt…



…with multiple certificate paths possible



Cross-Certification (ctd)

Different CAs and paths have different validity periods, constraints, etc etc

- Certificate paths can contain loops
- Certificate semantics can change on different iterations through the loop
- Are certificate paths Turing-complete?
- No software in existence can handle these situations

Cross-certification is the black hole of PKI

- All existing laws break down
- Noone knows what it's like on the other side

Cross-Certification (ctd)

The theory: A well-managed PKI will never end up like this

The practice: If you give them the means, they will build it

- Allow cross-certification and it's only a matter of time before the situation will collapse into chaos
- c.f. CA vs EE certificates
  - There are at least 5 different ways to differentiate the two
  - Only one of these was ever envisaged by X.509

Problems with X.509

Most of the required infrastructure doesn't exist

- Users use an undefined certification request protocol to obtain a certificate which is published in an unclear location in a nonexistent directory with no real means to revoke it
- Various workarounds are used to hide the problems
  - Details of certificate requests are kludged together via web pages
  - Complete certificate chains are included in messages wherever they're needed
  - Revocation is either handled in an ad hoc manner or ignored entirely

Standards groups are working on protocols to fix this

- Progress is extremely slow

Problems with X.509 (ctd)

Certificates are based on owner identities, not keys

- Owner identities don't work very well as certificate ID's
  - Real people change affiliations, email addresses, even names
  - An owner will typically have multiple certificates, all with the same ID
- Owner identity is rarely of security interest
  - Authorisation/capabilities are what count
    - I am authorised to do X
    - I am the same entity you dealt with previously
  - When you check into a hotel, buy goods in a store, you're asked for a payment instrument, not a passport

Problems with X.509 (ctd)

Revocation should revoke capability, not identities

- Revoking a key requires revoking the identity of the owner
- Renewal/replacement of identity certificates is nontrivial

Authentication and confidentiality certificates are treated the same way for certification purposes

- X.509v1 and v2 couldn't even distinguish between the two

Users should have certified authentication keys and use these to certify their own confidentiality keys

- No real need to have a CA to certify confidentiality keys
- New confidentiality keys can be created at any time
- Doesn't require the cooperation of a CA to replace keys

Problems with X.509 (ctd)

Aggregation of attributes shortens the overall certificate lifetime

- Steve's Rule of Revocation: Frequency of certificate change is proportional to the square of the number of attributes
- Inflexibility of certificate conflicts with real-world IDs
  - Can get a haircut, switch to contact lenses, get a suntan, shave off a moustache, go on a diet, without invalidating your passport
  - Changing a single bit in a certificate requires getting a new one
  - Steve's certificate is for an organisation which no longer exists

Problems with X.509 (ctd)

Certificates rapidly become a dossier as more attributes are added

```
SEQUENCE {
 OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
 [0] {
  SEQUENCE {
   INTEGER 1
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
     NULL
     }
    }
   SEQUENCE {
    OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
    }
   [0] {
    SEQUENCE {
     SEQUENCE {
      [0] {
       INTEGER 2
       }
      INTEGER 145
      SEQUENCE {
       OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
       NULL
       }
```

continues

```
SEQUENCE {
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER countryName (2 5 4 6)
   PrintableString 'CH'
   }
  }SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationName (2 5 4 10)
   PrintableString 'Swisskey AG'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
   PrintableString 'Public CA Services'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER localityName (2 5 4 7)
   PrintableString 'Zuerich'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER commonName (2 5 4 3)
   PrintableString 'Swisskey ID CA 1024'
   }
  }
 }
```

Problems with X.509 (ctd)

```
SEQUENCE {
 UTCTime '980929093816Z'
 UTCTime '000929093800Z'
 }
SEQUENCE {
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationName (2 5 4 10)
   PrintableString 'Swisskey AG'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
   PrintableString '0085100000502000000128'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
   PrintableString 'Product Management'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER postalCode (2 5 4 17)
   PrintableString '8008'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER localityName (2 5 4 7)
   PrintableString 'Zuerich'
   }
  }
```

```
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER countryName (2 5 4 6)
   PrintableString 'CH'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER commonName (2 5 4 3)
   PrintableString 'Juerg Spoerndli'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
   IA5String 'jspoerndli@swisskey.ch'
   }
  }
 }
SEQUENCE {
 SEQUENCE {
  OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
  NULL
  }
 BIT STRING 0 unused bits
  30 81 89 02 81 81 00 EE 7B BA 00 A0 1A C2 05 8B
  8F 52 26 E9 01 C4 A3 7A C9 6E C5 4C 2B FD 3A 2A
  44 48 72 29 7E E3 57 03 2A C9 F3 BB 1D C2 12 2D
  E7 7E 8D B3 3C 58 AD D6 8A 29 4D D1 9F 0F 1E 45
  F3 1E 67 39 9D 83 0B 1A 0D 1F 82 35 B0 D7 2A 6E
  35 6B 76 C2 05 9B 67 E4 3F 8B 6A 8F A6 04 85 F7
  56 EB 51 D9 69 D6 C9 23 AF 5E 0A AE D3 90 7F 60
  16 81 CF 1F 20 B6 A5 A5 5E F0 9F 6D B0 40 F9 8D
      [ Another 12 bytes skipped ]
 }
```

continues

Problems with X.509 (ctd)

```
[3] {
 SEQUENCE {
  SEQUENCE {
   OBJECT IDENTIFIER netscape-cert-type (2 16 840 1 113730 1 1)
   OCTET STRING, encapsulates {
    BIT STRING 5 unused bits
     '101'B
    }
   }
  SEQUENCE {
   OBJECT IDENTIFIER netscape-comment (2 16 840 1 113730 1 13)
   OCTET STRING
    16 81 C6 54 68 69 73 20 63 65 72 74 69 66 69 63
    61 74 65 20 68 61 73 20 62 65 6E 20 69 73 73
    75 65 64 20 62 79 20 53 77 69 73 73 6B 65 79 20
    41 47 20 67 6F 76 65 72 6E 65 64 20 62 79 20 69
    74 73 20 43 65 72 74 69 66 69 63 61 74 65 20 50
    72 61 63 74 69 63 65 20 53 74 61 74 65 6D 65 6E
    74 20 28 43 50 53 29 2E 20 43 50 53 20 61 6E 64
    20 66 75 72 74 68 65 72 20 69 6E 66 6F 72 6D 61
        [ Another 73 bytes skipped ]
   }
  SEQUENCE {
   OBJECT IDENTIFIER keyUsage (2 5 29 15)
   OCTET STRING, encapsulates {
    BIT STRING 5 unused bits
     '101'B
    }
   }
```

```
  SEQUENCE {
   OBJECT IDENTIFIER privateKeyUsagePeriod (2 5 29 16)
   OCTET STRING, encapsulates {
    SEQUENCE {
     [0] '19980929093816Z'
     [1] '20000929093800Z'
     }
    }
   }
  }
 }
SEQUENCE {
 OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
 NULL
 }
BIT STRING 0 unused bits
 2A 2A 40 C4 03 48 0B B9 7D 7F B6 85 FD CF A8 D7
 CF 96 D8 55 5D C0 87 4D BE E6 C1 0F 7A 0B 0F 17
 DF 7A 10 49 81 EB A1 6B 8C 16 93 FB 38 37 79 A0
 B6 1F B3 EA F0 AA D5 CA 0A 52 DA D3 19 3A 55 B6
 F6 7F 77 4E 30 15 D4 5C 8C 73 44 62 FF 15 9C 44
 C3 38 F0 D1 58 85 D0 C6 88 55 7C FF D0 67 14 4C
 DE D2 7F F8 00 A8 BC 6E A7 35 BD 51 DD CB 7D F2
 C8 E7 34 61 00 C2 25 51 F0 ED 0B B0 38 93 FC 30
}
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER 5
  SEQUENCE {
   OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
   NULL
   }
```

continues

Problems with X.509 (ctd)

```
SEQUENCE {
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER countryName (2 5 4 6)
   PrintableString 'CH'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationName (2 5 4 10)
   PrintableString 'Swisskey AG'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
   PrintableString 'Public CA Services'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER localityName (2 5 4 7)
   PrintableString 'Zuerich'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER commonName (2 5 4 3)
   PrintableString 'Swisskey Root CA'
  }
 }
}
SEQUENCE {
 UTCTime '9807061348492'
 UTCTime '0512312359002'
}
```

```
SEQUENCE {
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER countryName (2 5 4 6)
   PrintableString 'CH'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationName (2 5 4 10)
   PrintableString 'Swisskey AG'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
   PrintableString 'Public CA Services'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER localityName (2 5 4 7)
   PrintableString 'Zuerich'
  }
 }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER commonName (2 5 4 3)
   PrintableString 'Swisskey ID CA 1024'
  }
 }
}
SEQUENCE {
 SEQUENCE {
  OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
  NULL
 }
```

continues

Problems with X.509 (ctd)

```
 BIT STRING 0 unused bits
  30 81 89 02 81 81 00 AB E9 1F E9 AD FF 53 9F 71
  70 35 6D F8 F8 4C 76 B4 F7 43 E8 19 80 DD A9 0A
  D6 4E 60 C2 FD 48 7B 43 F6 6E BE 53 D0 0E 62 F0
  35 27 6F 2E 55 22 F2 82 40 2E 21 5B 5D 7E 18 16
  CA 87 31 2E 12 71 4C 5F 92 8A AB 36 61 9C 91 38
  BC BD 95 88 BF 7E 0C 4A D7 A0 12 F9 FA FF 0F 84
  F8 57 6E DE AE B4 03 FC 77 CF 7C E5 B3 33 79 61
  31 4E CE 70 03 E7 73 D8 E8 1B D3 EB 15 FF 69 B3
        [ Another 12 bytes skipped ]
 }
[3] {
 SEQUENCE {
  SEQUENCE {
   OBJECT IDENTIFIER basicConstraints (2 5 29 19)
   BOOLEAN TRUE
   OCTET STRING, encapsulates {
    SEQUENCE {
     BOOLEAN TRUE
     INTEGER 0
    }
   }
  }
  SEQUENCE {
   OBJECT IDENTIFIER keyUsage (2 5 29 15)
   BOOLEAN TRUE
   OCTET STRING, encapsulates {
    BIT STRING 1 unused bits
     '1100000'B
   }
  }
 }
}
```

```
SEQUENCE {
 OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
 NULL
}
BIT STRING 0 unused bits
 0E 0F 67 22 AA D2 8A 7B BF 3D 47 AB 1F 5E 8C F3
 2C 32 3E AB D3 48 60 A1 BA 49 FD 81 28 6A 26 69
 83 97 29 1F E8 80 14 96 30 2B C3 18 97 3B 6C F3
 F0 A2 D6 E0 30 EF F6 2C 38 1F C0 37 7E 9E 45 FD
 62 38 67 07 27 BE 81 07 E9 12 60 E8 BE 6B ED 14
 8E 61 17 52 99 C2 FE 33 B7 21 CA 5E FE 6D B4 1E
 B9 8C 54 36 42 55 1E 73 D9 81 DE 5D 25 AD 72 39
 15 AF 68 E9 44 45 55 7F 2E 2E F9 6F EF 44 B0 E0
}
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
  }
  INTEGER 1
  SEQUENCE {
   OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
   NULL
  }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'CH'
    }
   }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationName (2 5 4 10)
     PrintableString 'Swisskey AG'
    }
   }
```

continues

Problems with X.509 (ctd)

```
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
      PrintableString 'Public CA Services'
      }
     }
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER localityName (2 5 4 7)
      PrintableString 'Zuerich'
      }
     }
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'Swisskey Root CA'
      }
     }
    }
   SEQUENCE {
    UTCTime '980706120207Z'
    UTCTime '051231235900Z'
    }
   SEQUENCE {
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER countryName (2 5 4 6)
      PrintableString 'CH'
      }
     }
```

```
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER organizationName (2 5 4 10)
      PrintableString 'Swisskey AG'
      }
     }
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
      PrintableString 'Public CA Services'
      }
     }
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER localityName (2 5 4 7)
      PrintableString 'Zuerich'
      }
     }
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'Swisskey Root CA'
      }
     }
    }
   SEQUENCE {
    SEQUENCE {
     OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
     NULL
     }
```

continues

Problems with X.509 (ctd)

```
    BIT STRING 0 unused bits
     30 81 89 02 81 81 00 AC AB 60 E0 C5 69 FD 07 4E
     97 9B AF 4A 1C 30 D7 68 26 D1 2C 3D 44 F0 D6 AB
     16 34 6F 00 D8 7F D6 3F B9 35 D6 83 28 77 A3 3E
     24 5D A4 D1 C2 FA 04 B3 DB 4D 38 91 23 70 6C 2B
     2D 48 69 D5 15 6F 4A 9F 91 BC E4 83 2F 35 A2 29
     DB 55 66 F8 90 C6 0E 0C 32 75 95 24 E0 8D B7 8E
     AB 13 70 61 1E 01 91 7D 9D 44 37 42 41 C9 C2 01
     DD 26 D8 B9 2C 29 57 A1 54 17 1E AC 1A DE 8C 6C
         [ Another 12 bytes skipped ]
    }
   [3] {
    SEQUENCE {
     SEQUENCE {
      OBJECT IDENTIFIER basicConstraints (2 5 29 19)
      BOOLEAN TRUE
      OCTET STRING, encapsulates {
       SEQUENCE {
        BOOLEAN TRUE
        INTEGER 3
        }
       }
      }
     SEQUENCE {
      OBJECT IDENTIFIER keyUsage (2 5 29 15)
      BOOLEAN TRUE
      OCTET STRING, encapsulates {
       BIT STRING 1 unused bits
        '1100000'B
       }
      }
     }
    }
   }
```

```
   SEQUENCE {
    OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
    NULL
    }
   BIT STRING 0 unused bits
    72 A7 93 A3 CD D7 3A DB 79 50 DB 98 03 52 B0 CD
    AF 0C D2 A6 89 38 52 6C 5C E9 7C B3 37 3C 9E 94
    C4 74 57 D4 BB 78 05 5B B6 B9 31 04 FC 60 33 51
    5F CF 2C 44 55 85 EC 1F 0B CB 89 E7 F0 93 D4 CD
    85 D3 FF B6 B5 99 D3 7C 35 06 11 7B 0E 9F E6 BE
    99 B3 49 D0 5A 85 FA 7C BA 54 9B B9 AF F7 4B E3
    FF DC 83 4A 04 F8 F9 A5 1D EC 37 AE C6 23 4C 9D
    B2 01 1F D4 26 EA E4 4A 7E BE BE 1E 11 1E 27 D1
   }
  }
 SET {
  SEQUENCE {
   INTEGER 1
   SEQUENCE {
    SEQUENCE {
     SET {
      SEQUENCE {
       OBJECT IDENTIFIER countryName (2 5 4 6)
       PrintableString 'CH'
       }
      }
     SET {
      SEQUENCE {
       OBJECT IDENTIFIER organizationName (2 5 4 10)
       PrintableString 'Swisskey AG'
       }
      }
     SET {
      SEQUENCE {
       OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
       PrintableString 'Public CA Services'
       }
      }
```

continues

Problems with X.509 (ctd)

```
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER localityName (2 5 4 7)
      PrintableString 'Zuerich'
     }
    }
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'Swisskey ID CA 1024'
     }
    }
   }
  INTEGER 145
  }
 SEQUENCE {
  OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
  NULL
  }
 [0] {
  SEQUENCE {
   OBJECT IDENTIFIER contentType (1 2 840 113549 1 9 3)
   SET {
    OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
   }
  }
  SEQUENCE {
   OBJECT IDENTIFIER signingTime (1 2 840 113549 1 9 5)
   SET {
    UTCTime '9811130721332'
   }
  }
```

```
  SEQUENCE {
   OBJECT IDENTIFIER messageDigest (1 2 840 113549 1 9 4)
   SET {
    OCTET STRING
     2F 7E 95 9F 34 AC 85 B8 1C 53 9E 5C F8 60 BE 3A
     AA D0 30 B5
    }
   }
  SEQUENCE {
   OBJECT IDENTIFIER sMIMECapabilities (1 2 840 113549 1 9 15)
   SET {
    SEQUENCE {
     SEQUENCE {
      OBJECT IDENTIFIER des-EDE3-CBC (1 2 840 113549 3 7)
     }
     SEQUENCE {
      OBJECT IDENTIFIER rc2CBC (1 2 840 113549 3 2)
      INTEGER 128
     }
     SEQUENCE {
      OBJECT IDENTIFIER desCBC (1 3 14 3 2 7)
     }
     SEQUENCE {
      OBJECT IDENTIFIER rc2CBC (1 2 840 113549 3 2)
      INTEGER 64
     }
     SEQUENCE {
      OBJECT IDENTIFIER rc2CBC (1 2 840 113549 3 2)
      INTEGER 40
     }
    }
   }
  }
 }
```

continues

Problems with X.509 (ctd)

```
  SEQUENCE {
   OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
   NULL
   }
  OCTET STRING
   9F EC C4 B4 B2 5A FE 87 EA 28 22 C2 6A 1F E3 2F
   16 8D 01 EA 2F 35 0E 13 D1 3E BE 1D 92 48 EF F0
   8E BB BC 98 3B 11 44 88 A8 20 AE AB 65 2D 98 E1
   3E 62 E1 47 5F FE 18 39 AF 97 29 7E D1 68 03 F1
   03 78 44 DB A1 BB 9F 3B C9 89 D5 0D 00 B3 0B FA
   98 F8 2E 58 4C E4 4F 73 02 D6 17 41 84 B6 50 A2
   94 F8 E2 6F C3 78 AF 4D 71 CF E7 FF 25 97 B9 00
   CC A5 BE A8 8C 3D 52 43 C9 BB 41 A9 87 5F 85 6F
  }
 }
}
```

All this from a standard S/MIME signature!

Problems with X.509 (ctd)

Hierarchical certification model doesn't fit typical business practices

- Businesses generally rely on bilateral trading arrangements or existing trust relationships
- Third-party certification is an unnecessary inconvenience when an existing relationship is present

Problems with X.509 (ctd)

X.509 PKI model entails building a parallel trust infrastructure alongside the existing, well-established one

- Requires re-engineering business infrastructure with an entirely new security architecture
- In the real world, trust and revocation is handled by closing the account, not with PKIs, CRLs, certificate status checks, and other paraphernalia

Problems with X.509 (ctd)

In a closed system (SWIFT, Identrus)

- Members sign up to the rules of the club
- Only members who will play by the rules and can carry the risk are admitted
- Members are contractually obliged to follow the rules, including obligations for signatures made with their private key

In an open system

- Parties have no previously established network of contracts covering private key use on which they can rely
  - On what basis do you sue someone when they repudiate a signature?
  - Have they published a legally binding promise to the world to stand behind that signature?

Problems with X.509 (ctd)

  - Do they owe a duty of care, actionable in the case of negligence?
- Possible ways to proceed
  - Claim a duty of care where negligence resulted in financial loss (generally negligence claims for pure financial loss won't support this)
  - Claim that publishing the key was a negligent misstatement (unlikely that this will work)
  - Go after the CA (CA won't suffer any loss if the keyholder is negligent, so they can't go after the keyholder)
- On the whiteboard:

  "Alice does something magical/mathematical with Bob's key, and the judge says 'Obviously Bob is guilty'"
- In practice: Would you like to be the test case?

Problems with X.509 (ctd)

Certificates don't model standard authority delegation practices

- Manager can delegate authority/responsibility to an employee
  – "You're in charge of purchasing"
- CA can issue a certificate to an employee, but can't delegate the responsibility which comes with it

Residential certificates are even more problematic

- Noone knows who has the authority to sign these things

Problems with Implementations

Relying parties must, by definition, be able to rely on the handling of certificates

Currently difficult to do because of

- Implementation bugs
- Different interpretations of standards by implementors
- Implementation of different parts of standards
- Implementation of different standards

Problems with Implementations (ctd)

Examples of common problems

- rfc822Name has ambiguous definition/implementation (Assorted standards/implementations)
  - Should be used as `luser@aol.com`
  - Can often get away with `President George W.Bush <luser@aol.com>`
- Name constraints can be avoided through creative name encoding (Problem in standards)
  - Multiple encodings for the same character, zero-width spaces, floating diacritics, etc
  - Can make identical-appearing strings compare as different strings
  - Can also evade name constraints by using altNames

Problems with Implementations (ctd)

- Software crashes when it encounters a Unicode or UTF-8 string (Netscape)
  - Some other software uses Unicode for any non-ASCII characters, guaranteeing a crash
  - At least one digital signature law requires the (unnecessary) use of Unicode for a mandatory certificate field
    - Standards committee must have had MS stockholders on it
- Software produces negative numeric values because the implementors forgot about the sign bit (Microsoft and a few others)
  - Everyone changed their code to be bug-compatible with MS
- Software hardcodes the certificate policy so that any policy is treated as if it were the Verisign one (Microsoft)

Problems with Implementations (ctd)

- Known extensions marked critical are rejected; unknown extensions marked critical are accepted (Microsoft)
  - Due to a reversed flag in the MS certificate handling software
  - Other vendors and CAs broke their certificates in order to be bug-compatible with MS
  - Later certs were broken in order to be bug-compatible with the earlier ones
  - Spot check: If you have a cert from a public CA, check whether the important extensions are marked critical or not

Problems with Implementations (ctd)

- Software ignores the key usage flags and uses the first cert it finds for the purpose it needs (Microsoft)
  - If users have separate encryption and signing certs, the software will grab the first one it finds and use it for both purposes
  - CryptoAPI seems to mostly ignore usage constraints on keys
    - AT_KEYXECHANGE keys (with corresponding certificates) can be used for signing and signature verification without any trouble

Problems with Implementations (ctd)

- Cert chaining by name is ignored (Microsoft)
  - Certificate issued by “Verisign Class 1 Public Primary Certification Authority” could actually be issued by “Honest Joe’s Used Cars and Certificates”
  - “No standard or clause in a standard has a divine right of existence” – MS PKI architect
  - Given the complete chaos in DNs, this isn’t quite the blatantly wrong decision which it seems

Problems with Implementations (ctd)

- Obviously bogus certificates are accepted as valid (Microsoft)

Problems with Implementations (ctd)

- Validity period is actually December 1951 to December 2050
  - At one point MS software was issuing certificates in the 17th century
  - This was deliberate
- Software reports it as December 1950 to December 1950, but accepts it anyway
- Exponent is 1 (bogus key) but cert is accepted as valid

Problems with Implementations (ctd)

- End entity certificates are encoded without the basicConstraints extension to indicate that the certificate is a non-CA cert (PKIX)
  - Some apps treat these certificates as CA certificates for X.509v1 compatibility
  - May be useful as a cryptographically strong RNG
    - Issue 128 certificates without basicConstraints
    - User other app's CA/non-CA interpretation as one bit of a key
    - Produces close to 128 bits of pure entropy
- CRL checking is broken (Microsoft)
  - Older versions of MSIE would grope around blindly for a minute or so, then time out and continue anyway
  - Some newer versions forget to perform certificate validity checks (eg expiry times, CA certs) if CRL checking enabled

Problems with Implementations (ctd)

- Applications enforce arbitrary limits on data elements (GCHQ/CESG interop testing)
  - Size of serial number
    - Supposedly an integer, but traditionally filled with a binary hash value
  - Number/size of DN elements
  - Size of encoded DN
  - Certificate path/chain length
  - Path length constraints
    - Oops, we need to insert one more level of CA into the path due to a company reorg/merger
  - Ordering/non-ordering of DN elements
    - Allow only one attribute type (eg OU) per DN
    - Assume CN is always encoded last

Problems with Implementations (ctd)

- The lunatic fringe: Certs from vendors like Deutsche Telekom/Telesec are so broken they would create a matter/antimatter reaction if placed in the same room as an X.509 spec
  - "Interoperability considerations merely create uncertainty and don't serve any useful purpose. The market for digital signatures is at hand and it's possible to sell products without any interoperability" – Telesec project leader (translated)
  - "People will buy anything as long as you tell them it's X.509" (shorter translation)

Problems with an X.509-style PKI

PKI will solve all your problems

- PKI will make your network secure
- PKI will allow single sign-on
- PKI solves privacy problems
- PKI will allow *<insert requirement which customer will pay money for>*
- PKI makes the sun shine and the grass grow and the birds sing

Problems with an X.509-style PKI (ctd)

Reality vs hype

- Very little interoperability/compatibility
- Lack of expertise in deploying/using a PKI
- No manageability
- Huge up-front infrastructure requirements
  - Few organisations realise just how much time, money and resources will be required
  - Incremental change to legacy systems is easier than starting from scratch with a PKI

Problems with an X.509-style PKI (ctd)

- "PKI will get rid of passwords"
  - Current implementations = password + private key
  - Passwords with a vengeance
- Certificate revocation doesn't really work
  - Locating the certificate in the first place works even less

How Effective are Certificates Really?

Sample high-value transaction: Purchase $1,500 airline ticket from United Airlines

- Site is `http://www.united.com` aka `http://www.ual.com`
- Browser shows the SSL padlock
  - Certificate is verified (transparent to the user)
  - It's safe to submit the $1,500 payment request

How Effective are Certificates Really? (ctd)

But

- Actual site it's being sent to is `itn.net`
- Company is located in Palo Alto, California
  - Who are these people?
  - Site contains links to the Amex web site
    - Anyone can add links to Amex site to their home page though
- Just for comparison
  - Singapore Airlines, British Airways, and Lufthansa have appropriate certificates
  - Air New Zealand also uses `itn.net`
  - American Airlines don't seem to use any security at all
  - Quantas don't even have a web site

How Effective are Certificates Really? (ctd)

This is exactly the type of situation which SSL certificates are intended to prevent

- Browsers don't even warn about this problem because so many sites would break
  - Outsourcing of merchant services results in many sites handling SSL transactions via a completely unrelated site
- Effectively reduces the security to unauthenticated Diffie-Hellman

Most current certificate usage is best understood by replacing all occurrences of the term "trusts" with "relies upon" or "depends upon", generally with an implied "has no choice but to …" at the start

PGP Certificates

Certificates are key-based, not identity-based

- Keys can have one or more free-form names attached
- Key and name(s) are bound through (independent) signatures

Certification model can be hierarchical or based on existing trust relationships

- Parties with existing relationships can use self-signed certificates
  - Self-signed end entity certificates are a logical paradox in X.509v3

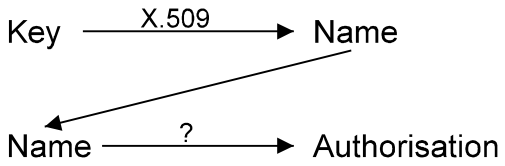
Authentication keys are used to certify confidentiality keys

- Confidentiality keys can be changed at any time, even on a per-message basis

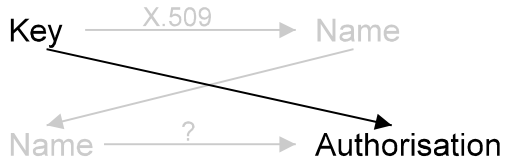
SPKI

Simple Public Key Infrastructure

Identity certificates bind a key to a name, but require a parallel infrastructure to make use of the result



SPKI certificates bind a key to an authorisation or capability



SPKI (ctd)

Certificates may be distributed by direct communications or via a directory

Each certificate contains the minimum information for the job (cf X.509 dossier certificates)

If names are used, they only have to be locally unique

- Global uniqueness is guaranteed by the use of the key as an identifier
- Certificates may be anonymous (eg for balloting)

Authorisation may require *m* of *n* consensus among signers (eg any 2 of 3 company directors may sign)

SPKI Certificate Uses

Typical SPKI uses

- Signing/purchasing authority
- Letter of introduction
- Security clearance
- Software licensing
- Voter registration
- Drug prescription
- Phone/fare card
- Baggage claim check
- Reputation certificate (eg Better Business Bureau rating)
- Access control (eg grant of administrator privileges under certain conditions)

Certificate Structure

SPKI certificates use collections of assertions expressed as LISP-like S-expressions of the form (*type value(s)*)

(name fred) ⇒ Owner name = fred

(name *CA_root CA1 CA2 ... CAn leaf_cert*) ⇒ X.500 DN

(name (hash sha1 |TLCgPLFlGTzyUbcaYLW8kGTEnUk=|) fred) ⇒ Globally unique name with key ID and locally unique name

(ftp (host ftp.warez.org)) ⇒ Keyholder is allowed FTP access to an entire site

(ftp (host ftp.warez.org) (dir /pub/warez)) ⇒ Keyholder is allowed FTP access to only one directory on the site

Certificate Structure (ctd)

```
( cert
    ( issuer ( hash sha1 |TLCgPLFlGTzyUbcaYLW8kGTEnUk=|
      ) )
    ( subject ( hash sha1 |Ve1L/7MqiJcj+LSa/l10fl3tuTQ=l| ) )
    ...
    ( not-before "1998-03-01_12:42:17" )
    ( not-after "2012-01-01_00:00:00" )
  ) ⇒ X.509 certificate
```

Internally, SPKI certificates are represented as 5-tuples <Issuer, Subject, Delegation, Authority, Validity>

- Delegation = Subject has permission to delegate authority
- Authority = Authority granted to certificate subject
- Validity = Validity period and/or online validation test information

Trust Evaluation

5-tuples can be automatically processed using a general-purpose tuple reduction mechanism

<I1, S1, D1, A1, V1> + <I2, S2, D2, A2, V2>
$\Rightarrow$ <I1, S2, D2, intersection(A1, A2), intersection(V1, V2)
if S1 = I2 and D1 = true

Eventually some chains of authorisation statements will reduce to <Trusted Issuer, *x*, D, A, V>

- All others are discarded

Trust Evaluation (ctd)

Example authorisation chain

- A may access resource X. Signed: Service Provider
- B may access resource X. Signed: A
- Service provider, please allow me to access X. Signed: B

Verification

- Service provider checks signatures from B $\rightarrow$ A $\rightarrow$ own key
- Authorisation loop requires no CA, trusted third party, or external intervention
- Trust management decisions can be justified/explained/verified
  - "How was this decision reached?"
  - "What happens if I change this bit?"

X.509 has nothing even remotely like this