

Everything you Never Wanted to Know about PKI but were Forced to Find Out

Peter Gutmann
University of Auckland

Certificate History

To understand the X.509 PKI, it's necessary to understand the history behind it

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

Problems with Naming/Identity Certificates

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

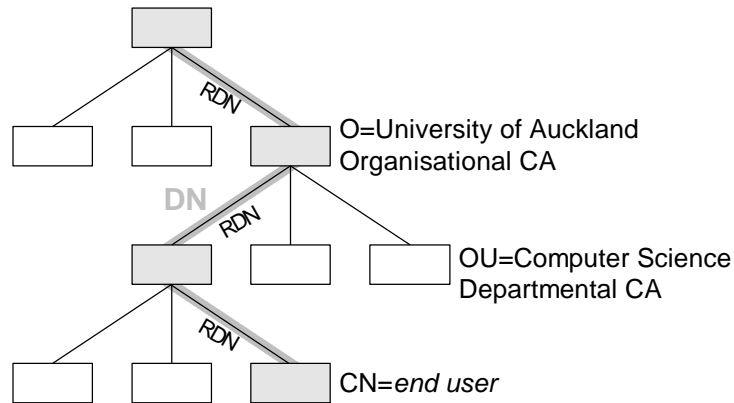
- Which directory?
- Which John Smith?

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

CA Hierarchy in Theory

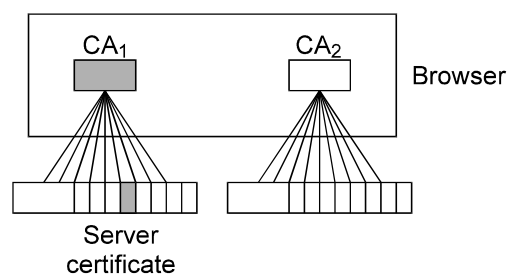
Portions of the X.500 hierarchy have CAs attached to them



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

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

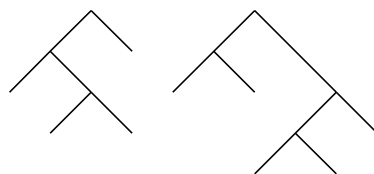
- A signs B's certificate
- B signs A's certificate

Cross-Certification (ctd)

Problem: Each certificate now has *two* issuers

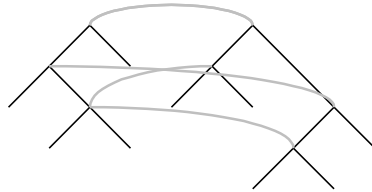
- All of X.509 is based on the fact that there's a unique issuer
- Toto, I don't think we're in X.509 any more

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

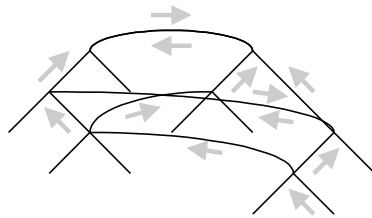


Cross-Certification (ctd)

...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)

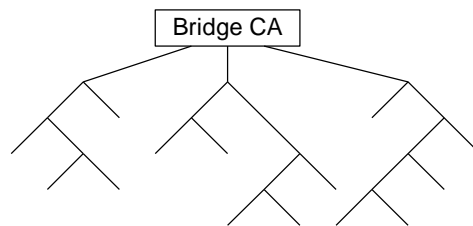
The theory: A well-managed PKI will never end up like this

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

Bridge CAs

Attempt to solve the cross-certification chaos by unifying disparate PKIs with a super-root



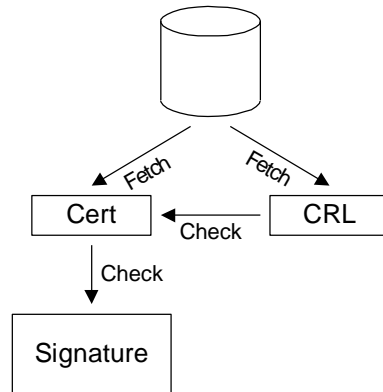
Still has problems

- PKIn root has different semantics than bridge root
- What if PKI1 = CIA, PKI2 = KGB, PKI3 = Mossad?
 - Trust issues are discussed elsewhere

X.509 Certificate Usage Model

Relying party wants to verify
a signature

- Fetch certificate
- Fetch certificate revocation list (CRL)
- Check certificate against CRL
- Check signature using certificate



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)

Revoking self-signed certificates is hairy

- 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

Certificate Revocation (ctd)

Many applications require prompt revocation

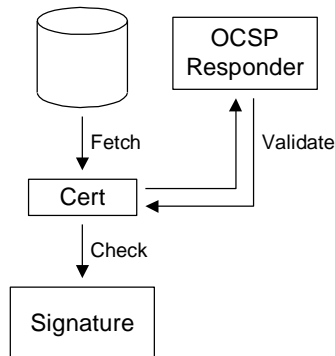
- CA's (and X.509) don't really support this
- CA's are inherently an offline operation

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

Online Status Checking (ctd)

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”
- 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”
⇒ “not good”, so any other status = “good”
 - Much debate among implementors about OCSP semantics

Cost of Revocation Checking

CAs charge fees to issue a certificate

- Most expensive collection of bits in the world

Revocation checks are expected to be free

- CA can’t tell how often or how many checks will be made
- CRLs require
 - Processor time
 - Multiple servers (many clients can fetch them)
 - Network bandwidth (CRLs can get large)
- Active disincentive for CAs to provide real revocation checking capabilities

Cost of Revocation Checking (ctd)

Example: ActiveX

- Relatively cheap cert can sign huge numbers of ActiveX controls
- Controls are deployed across hundreds of millions of Windows machines
- Any kind of useful revocation checking would be astronomically expensive

Example: email certificate

- Must be made cheap (or free) or users won't use them
- Revocation handling isn't financially feasible

Cost of Revocation Checking (ctd)

Revocation checking in these cases is, quite literally, worthless

- Leave an infrequently-issued CRL at some semi-documented location and hope few people find it

Charge for revocation checks

- Allows certain guarantees to be associated with the check
- Identrus charges for every revocation check (i.e. certificate use)
- GSA cost was 40¢...\$1.20 each time a certificate was used

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)

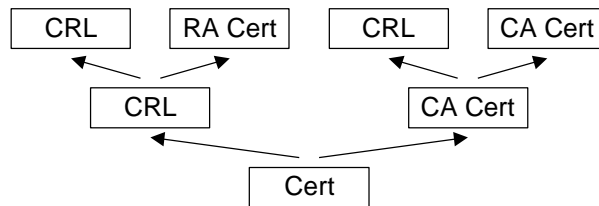
Certificate of health provides better proof than CRLs

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

Certificate Chains

Collection of certificates from a leaf up to a root or trust anchor

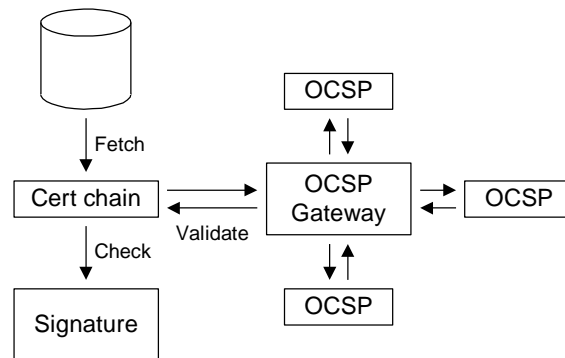
- All previous problems are multiplied by the length of the chain



- Complexity of revocation checking is proportional to the square of the depth of the issuance hierarchy

Certificate Chains (ctd)

Use OCSP with an access concentrator

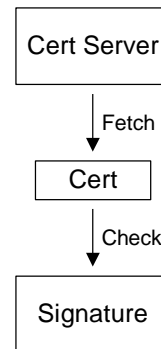


- Gateway does all the work
- Requests can be forwarded to further gateways
- User is billed once at the access concentrator

Closing the Circle

Fetching a cert and then immediately having to perform a second fetch to determine whether it's any good is silly

- Fetch a known-good cert (no revocation check necessary)
- Solves previous revocation-checking problems
- Simplify further: Submit hash of certificate on hand
 - “It's good, go ahead and use it”
 - “It's no good, use this one instead”



Closing the Circle (ctd)

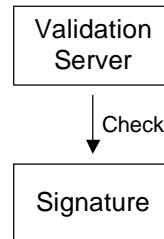
All we really care about is the key

- Issuer/subject DN, etc are historical artifacts/baggage
- “Give me the key for John Smith”
- This operation is currently performed locally when the key is fetched from a certificate store/Windows registry/flat file
- Moving from a local to a remote query allows centralised administration

Closing the Circle (ctd)

Key-fetch is still an unnecessary step

- Validation server performs the check directly
- Similar to the 1970's Davies and Price model
 - CA provides a dispute resolution mechanism via a one-time interactive certificate for the transaction
- Fits the banking/online settlement transaction model



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

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)
 - When you check into a hotel, buy goods in a store, you're asked for a payment instrument, not a passport
- Revoking a key requires revoking the identity of the owner
- Renewal/replacement of identity certificates is nontrivial

Problems with X.509 (ctd)

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 {
        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
        }
      }
    }
  }
  SEQUENCE {
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER countryName (2 5 4 6)
        PrintableString 'CH'
      }
      [SET {
        SEQUENCE {
          OBJECT IDENTIFIER organizationName (2 5 4 10)
          PrintableString 'Swiskey 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 'Swiskey ID CA 1024'
    }
  }
}
```

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 '980706134849Z'
    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 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 ED
}
SEQUENCE {
  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 '98070612020Z'
  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 ]
}
[31] {
  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 {
  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 '981113072133Z'
    }
  }
}
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

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

- 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

- No one 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_KEYEXCHANGE 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)

- 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
- “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
 - Qantas 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

PKI Design Guidelines

Identity

- Use a locally meaningful identifier
 - User name
 - email address
 - Account number
- Don't try and do anything meaningful with DNs

PKI Design Guidelines (ctd)

Revocation

- If possible, design your PKI so that revocation isn't required
 - SET
 - AADS/X9.59
 - ssh
 - SSL
- If that isn't possible, use a mechanism which provides freshness guarantees
- If that isn't possible, use an online status query mechanism
 - Valid/not valid responder
 - OCSP
- If the revocation is of no value, use CRLs

PKI Design Guidelines (ctd)

Application-specific PKIs

- PKIs designed to solve a particular problem are easier to work with than a one-size-(mis)fits all approach
- SPKI
 - Binds a key to an authorisation
 - X.509 binds a key to an (often irrelevant) identity which must then somehow be mapped to an authorisation
- PGP
 - Designed to secure email
 - Laissez-faire key management tied to email address solves "Which directory" and "Which John Doe" problems

PKI Design Guidelines (ctd)

In many situations no PKI of any kind is needed

- Example: Authority-to-individual communications (eg tax filing)
 - Obvious solution: S/MIME or PGP
 - Practical solution: SSL web server with access control
 - Revocation = disable user access
 - Instantaneous
 - Consistently applied
 - Administered by the organisation involved, not some third party

PKI Design Guidelines (ctd)

- Example: AADS/X9.59
 - Ties keys to existing accounts
 - Handled via standard business mechanisms
 - Revocation = remove key/close account
- Example: Business transactions
 - Ask Citibank about certificate validity
 - vs
 - ask Citibank to authorise the transaction directly
 - Use an online authorisation
 - (US) Business Records Exception allows standard business records to be treated as evidence in court
 - Following standard legal precedent is easier than becoming a test case for PKI

PKI Design Guidelines (ctd)

There's nothing which says you have to use X.509 as anything more than a complex bit-bagging scheme

- If you have a cert management scheme which works, use it

Be careful about holding your business processes hostage to your PKI (or lack thereof)

Phew!

More information in part 2 of the godzilla crypto tutorial,
<http://www.cs.auckland.ac.nz/~pgut001/tutorial/index.html>