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

Peter Gutmann

University of Auckland

What is Public Key Infrastructure

Public-key encryption is used for encryption and digital signatures

The public key is a string of bits

- Whose bits are they?
- What can they be used for?
- Are they still valid?
- Examples
  - Is this really the key for `foo.com`?
  - Was the key used to sign this valid at the time of signing?
  - Fetch me the key of Alfredo Garcia

The purpose of a PKI is to answer these questions (and more)

Certificate History

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

Why does X.509 do otherwise straightforward things in such a weird way?

> [The] standards have been written by little green monsters from outer space in order to confuse normal human beings and prepare them for the big invasion
>
> — comp.std.internat

- Someone tried to explain public-key-based authentication to aliens. Their universal translators were broken and they had to gesture a lot
- They were created by the e-commerce division of the Ministry of Silly Walks

Certificate History (ctd)

Original paper on public-key encryption proposed the Public File

- Public-key white pages
- Key present → key valid
- Communications with users were protected by a signature from the Public File

A very sensible, straightforward approach…

- … today
- Not so good in 1976

Certificate History (ctd)

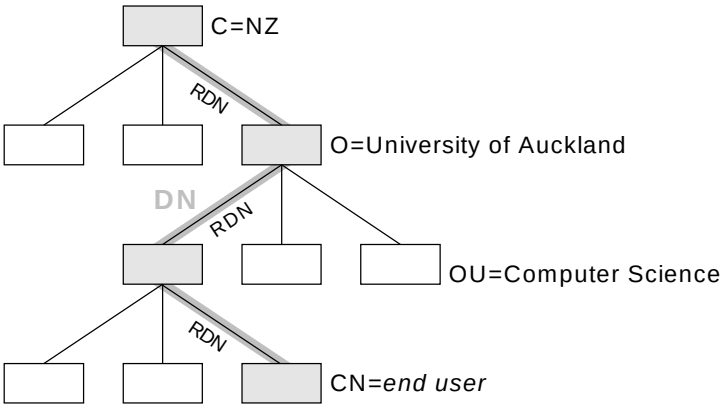
Adapted for offline operation by Kohnfelder in 1978

- Offline CA signs name + key to bind the two in a certificate
- Online directory distributes certificates

OSI proposed (among many other things) X.500, an all-encompassing global directory run by monopoly telcos

- Hierarchical database (or data organisation, or both)
- Path through the directory/database to keys is defined by a series of relative distinguished names (RDNs)
- Collection of RDNs form a distinguished name (DN)
- Data being looked up is found at the end of the RDN path

Certificate History (ctd)



Search key is C=NZ, O=University of Auckland, OU = Computer Science, CN = foo

- Complex way of saying `SELECT data WHERE key = 'foo'`

Certificate History (ctd)

Concerns about misuse of the directory

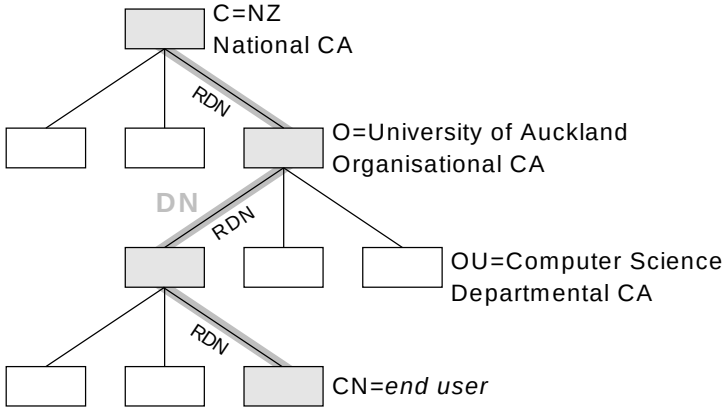
- Companies don't like making their internal structure public
  - Directory for corporate headhunters
- Privacy concerns
  - Directory of single women
  - Directory of teenage children

X.500 proposed various access control mechanisms

- Passwords
- Hashed passwords
- Digital signatures

Certificate History (ctd)

For signature-based access control, each portion of the directory has a certification authority (CA) attached to it



Top-level CA is called the root CA, a.k.a. "the single point of failure"

Certificate History (ctd)

X.509v1 clearly shows these origins

- Issuer and subject DN to place a cert in the directory
- Validity period
- Public key

No indication of…

- CA vs. end entity certs
  - Implicit from position in directory
- Key usage
  - Only one usage, directory authentication
- Cert policy
  - Only one policy, directory authentication
- Any of the other X.509v3 paraphernalia

Certificate History (ctd)

No directories of this type were ever seriously deployed

- We've had to live with the legacy of this approach ever since

This model turns certificates into capabilities

- Tickets which can be used for authorisation/access control purposes
- Capabilities can be passed around freely
- Revocation is very hard

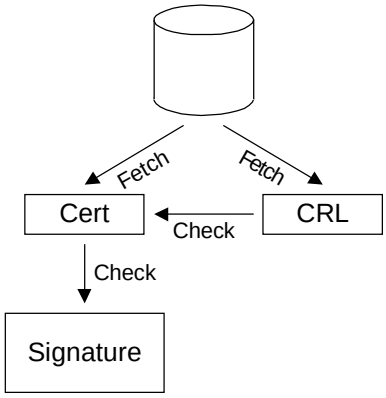
X.500 tried to address revocation with…

- Replacing the cert with a new one
- Notifying the owner "by some off-line procedure"
- Certificate revocation lists (CRLs), a blacklist of revoked certs
- Assorted handwaving

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



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

Identity in Certificates

No-one understands X.500 DNs

- Locality? Organisational unit? Administrative domain?
- Don't fit any real-world domain
- Require extensive versing in X.500 theology to comprehend

Solution: Users cram anything they feel like into the DN

- DN becomes a meaningless blob
- In some cases privacy requirements (e.g. FERPA) result in the creation of DNs containing random noise (a full DN reveals too much info)
- Only useful components are the common name (CN) + email address, or server URL

Other PKI designs use a more pragmatic approach

Identity in Certificates (ctd)

PGP: Used for email encryption

- Identity is name + email address

SPKI: Used for authorisation/access control

- Identity is a name meaningful within the domain of application
  - Account name on a server
  - Credit card number
  - Merchant ID

PGP and SPKI also use the public key as a unique ID

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
  - These were based on even earlier cheque blacklists
- 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 that re-fetch the CRL each time will recognise the cert as expired
  - Clients that cache CRLs won't
  - Users must choose between huge bandwidth consumption/ processing delays or missed revocations

Certificate Revocation (ctd)

Many applications require prompt revocation

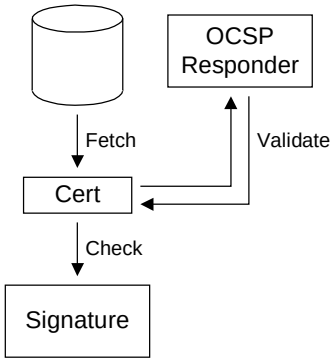
- CAs (and X.509) don't really support this
- CAs are inherently an offline operation

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 CRLs
  - 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
  - Requires a signing operation for every query

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"
- If asked "Is this a valid cert" and fed…
  - A freshly-issued cert, can't say "Yes"
  - An MPEG of a cat, can't say "No"
- Compare this with the credit card authorisation model
  - Response is "Authorised" or "Declined" (with optional reasons)

Online Status Checking (ctd)

- Problems arise to some extent from 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 implementers 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: No-one 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

CRLs are useful in special-case situations where there exists a statutory or contractual obligation to use them

- Relying party needs to be able to claim CRL use for due diligence purposes or to avoid liability

Alternative: Use Online Authorisation Check

Simple Public Key Infrastructure (SPKI)

- Prefers online authorisation/validation checks
  - This is a true online authorisation check, not the OCSP silly-walk
- Positive assertions are more tractable than negative ones
  - Compare "Aliens exist" vs. "Aliens don't exist"
- Cert renewal interval is based on risk analysis of potential losses
  - X.509 renewal interval is usually one year, motivated by billing concerns
    - Treated like a domain name: Once a year, re-certify the same key
- Provides for one-time renewal
  - Cert is valid for a single transaction

Alternative: Design Around the Problem

Use existing mechanisms to design around the revocation problem

SET

- Certs are tied to credit cards
- Cardholder certs are revoked by cancelling the card
- Merchant certs are revoked by removing them from the acquiring bank's database
- Payment gateway certs are short-lived and quickly replaced

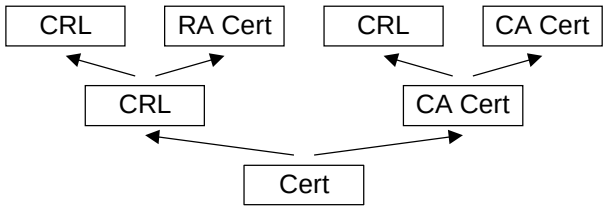
Account Authority Digital Signatures (AADS/X9.59), ssh

- Ties keys to accounts
- Revocation is handled by removing the key/closing the account

Certificate Chains

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

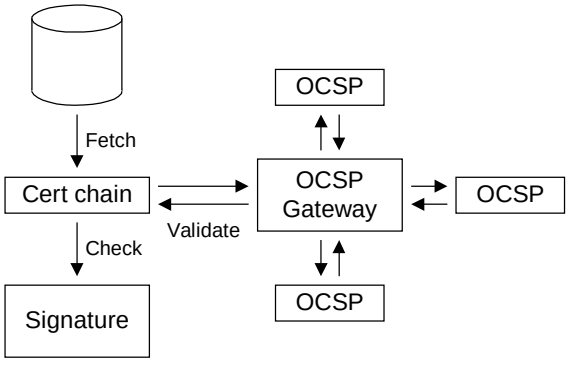
- All previous problems are multiplied by the length of the chain



- Complexity of certificate 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

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

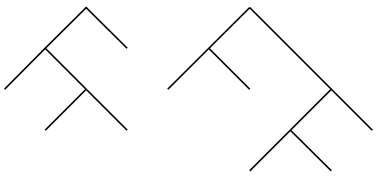
- A signs B's certificate
- B signs A's certificate

Cross-Certification (ctd)

Problem: Each certificate now has *two* issuers

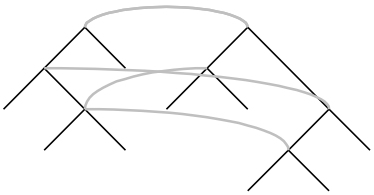
- All of X.509 is based on the fact that there's a unique issuer
- Toto, I don't think we're in X.509 any more

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

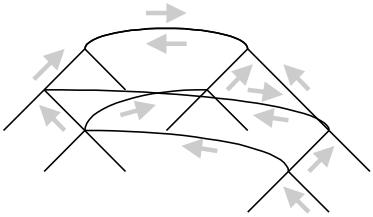


Cross-Certification (ctd)

…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
- No-one knows what it's like on the other side

Cross-Certification (ctd)

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

- "If it does occur, we can handle it via nameConstraints, policyConstraints, etc etc"

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
- Support for name and policy constraints is dubious to nonexistant
  - Playing Russian roulette with your security

Cross-Certification in Browsers

Hard-coded into browsers

- Implicitly trusted
- Totally unknown CAs
  - CA keys have been on-sold to third parties when the original CA went out of business
- Moribund web sites
- 512-bit keys
- 40-year cert lifetime (!!)
- How much would you trust a "NO LIABILITY ACCEPTED" CA?

All CA certs are trusted equally

- Implicit universal cross-certification

Cross-Certification in Browsers (cont)

Any CA can usurp a certificate issued by any other CA

- Overall security is that of the least trustworthy CA

Anyone who selects a public CA on a factor other than price fails to understand the trust models that underlie today's use of CAs

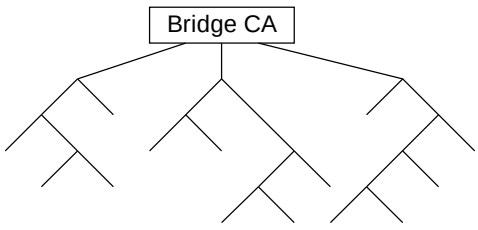
— Lucky Green

Disabling all of these certificates…

- Netscape 6: ~600 mouse clicks
- MSIE 6: ~700 mouse clicks

Bridge CAs

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



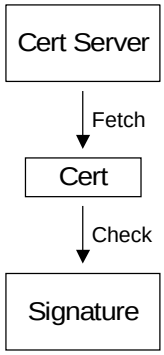
Still has problems

- PKIn root has different semantics than bridge root
- What if PKI1 = CIA, PKI2 = KGB, PKI3 = Mossad?

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 the previous revocation-checking problems
- Simplify further: Submit a hash of the certificate on hand
  - "It's good, go ahead and use it"
  - "It's no good, use this one instead"



Closing the Circle (ctd)

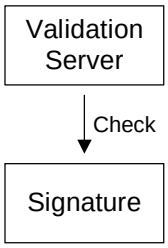
All we really care about is the key

- Issuer/subject DN, etc are historical artifacts/baggage
- "Bring me the key of Alfredo Garcia"
- 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 1970s Davies and Price model
  - Arbitrator 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
- No-one has really figured out a PKI business model yet

CA Business Model

Free email certs

- No-one will pay for them
- Clown suit certificates

SSL certificates run as a protection racket

- Buy our certs at US$500/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

Getting your CA Key into Browsers

Total cost: $0.5M per browser

- Netscape: Hand over the cash and a floppy
- MSIE: No special charge, but you must pass an SAS70 electronic data security audit
  - US CPA Statement on Auditing Standards 70
  - Lengthy (up to 6 months), expensive, and painful
  - Infrastructure, policy, staff, and auditing costs run to $0.5M

CA keys are bought and sold on the secondary market

- Equifax's certificates are actually owned by Geotrust
- Cheaper to buy another CAs HSM than to have your own key added

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
  - Will never fly for exactly that reason
- PGP uses this model

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 967650145
      SEQUENCE {
       OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1
5)
       NULL
       }
      SEQUENCE {
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationName (2 5 4 10)
         PrintableString 'AlphaTrust'
         }
        }
```

continues

```
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
         PrintableString '000-001-0002'
         }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER commonName (2 5 4 3)
         PrintableString 'AlphaTrust CA1'
         }
        }
       }
      SEQUENCE {
       UTCTime 30/08/2000 16:43:22 GMT
       UTCTime 31/08/2001 23:59:59 GMT
       }
      SEQUENCE {
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationName (2 5 4 10)
         PrintableString 'AlphaTrust'
         }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER countryName (2 5 4 6)
         PrintableString 'US'
         }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
         PrintableString '650-517-2672'
         }
        }
```

Problems with X.509 (ctd)

```
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER commonName (2 5 4 3)
   PrintableString 'Bill Brice Jr (Pro-S:650-517-2672:A)'
    }
   }
  }
SEQUENCE {
 SEQUENCE {
  OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
  NULL
   }
 BIT STRING, encapsulates {
  SEQUENCE {
   INTEGER
    00 C9 22 1D C0 E9 41 74 C6 35 D9 9E 37 BD A2 AF
    13 F7 04 F0 F9 53 DA 57 F1 90 9E 1F 63 7E EA C3
    1C 14 37 59 4D E9 43 2B 11 D3 6C 9C DC 2A 84 F9
    43 D5 E5 01 F0 28 F7 84 58 C1 6E 56 C3 95 85 6B
    2C 9E 36 46 02 3E 8C 45 C9 DE F8 27 EC A5 DB 4A
    57 C5 6D 53 26 25 0D D8 5A FC C8 CD C0 C1 0A D6
    3B 2F 3E A4 AB A8 CE 1E B9 C8 F3 DB 93 3F CC 94
    F7 A9 76 8B 4B FD 9A BA 3C 06 11 DD B7 4E D4 9D
      [ Another 1 bytes skipped ]
   INTEGER 65537
    }
   }
  }
[3] {
 SEQUENCE {
  SEQUENCE {
   OBJECT IDENTIFIER keyUsage (2 5 29 15)
   BOOLEAN TRUE
   OCTET STRING, encapsulates {
    BIT STRING 7 unused bits
     '1'B (bit 0)
    }
   }
```

```
  SEQUENCE {
   OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
   OCTET STRING, encapsulates {
    SEQUENCE {
     OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
     OBJECT IDENTIFIER emailProtection (1 3 6 1 5 5 7 3 4)
      }
    }
   }
  SEQUENCE {
   OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
   OCTET STRING, encapsulates {
    SEQUENCE {
     SEQUENCE {
      [0] {
       [0] {
        [6]
         'http://crl.alphatrust.com/crl/at1-crl.crl'
        }
       }
      }
     }
    }
   }
  SEQUENCE {
   OBJECT IDENTIFIER subjectAltName (2 5 29 17)
   OCTET STRING, encapsulates {
    SEQUENCE {
     [1] 'bill.brice@alphatrust.com'
      }
    }
   }
```

continues

Problems with X.509 (ctd)

```
  SEQUENCE {
   OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
   OCTET STRING, encapsulates {
    SEQUENCE {
     SEQUENCE {
      OBJECT IDENTIFIER '2 16 840 1 114003 1 1 1 4 1 1'
      SEQUENCE {
       SEQUENCE {
        OBJECT IDENTIFIER unotice (1 3 6 1 5 5 7 2 2)
        SEQUENCE {
         VisibleString
          'This certificate may only be used by authorized '
          'AlphaTrust Members. ALL LIABILITY TO OTHERS IS D'
          'ISCLAIMED. (c) 2000 AlphaTrust Corporation.'
         }
        }
       SEQUENCE {
        OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
        IA5String
         'https://www.alphatrust.com/rep/policy.htm'
        }
       }
      }
     }
    }
   }
  SEQUENCE {
   OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
   OCTET STRING, encapsulates {
    SEQUENCE {
     SEQUENCE {
      OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
      [6] 'http://validate.alphatrust.com'
       }
      }
     }
    }
```

```
SEQUENCE {
       OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
       OCTET STRING, encapsulates {
        SEQUENCE {
         [0]
          26 5E BF 83 E6 CA 4B 31 79 62 A9 0B 79 F5 27 F5
          13 8D 6A AA
         }
        }
       }
      SEQUENCE {
       OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
       OCTET STRING, encapsulates {
        OCTET STRING
         D0 3C C8 1F BE 60 59 68 2A 55 BD AA F8 A0 0A 30
         57 F9 7B 15
        }
       }
      SEQUENCE {
       OBJECT IDENTIFIER basicConstraints (2 5 29 19)
       OCTET STRING, encapsulates {
        SEQUENCE {}
         }
        }
       }
      }
     }
    SEQUENCE {
     OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
     NULL
      }
    BIT STRING
     6E 64 5C C6 7D A2 10 30 49 D1 24 DA EB 7D A4 5B
     A7 D5 17 AD 66 97 E4 47 FA BC 0A BE 49 C8 1A 70
     D9 1A 53 F1 04 B8 EC 88 65 A0 46 DA 8C 02 BF 15
     07 23 C3 58 0A 92 9E 34 44 18 0E CA FF 1A FD 5E
     92 DE 63 4D 34 22 33 7C EC 34 96 3A 39 08 75 B9
     4D 2A 9F 7C 8F D1 E5 31 A9 02 F9 1F F0 AD BF E7
     C5 EB 50 46 64 D3 B4 56 9A 1D 31 0F BC E3 AA 67
     C7 42 4A 2F E9 D0 50 30 2A 92 B7 23 92 FE 6C 30
         [ Another 128 bytes skipped ]
    }
```

continues

Problems with X.509 (ctd)

```
    SEQUENCE {
     SEQUENCE {
      [0] {
       INTEGER 2
        }
      INTEGER 967650191
      SEQUENCE {
       OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1
5)
       NULL
        }
      SEQUENCE {
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationName (2 5 4 10)
         PrintableString 'AlphaTrust'
          }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
         PrintableString '000-001-0002'
          }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER commonName (2 5 4 3)
         PrintableString 'AlphaTrust CA1'
          }
        }
       }
      SEQUENCE {
       UTCTime 30/08/2000 16:44:09 GMT
       UTCTime 31/08/2001 23:59:59 GMT
       }
      SEQUENCE {
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationName (2 5 4 10)
         PrintableString 'AlphaTrust'
          }
        }
```

```
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER countryName (2 5 4 6)
         PrintableString 'US'
          }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
         PrintableString '650-517-2672'
          }
        }
       SET {
        SEQUENCE {
         OBJECT IDENTIFIER commonName (2 5 4 3)
         PrintableString 'Bill Brice Jr (Pro-E:650-517-2672:B)'
          }
        }
       }
      SEQUENCE {
       SEQUENCE {
        OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
        NULL
         }
       BIT STRING, encapsulates {
        SEQUENCE {
         INTEGER
          00 C1 CB 0F 56 FD 4D A7 16 5F 40 41 BB A6 9B A2
          E3 7F 30 33 36 E3 5A 9C EE 3B 1E 48 0E FD 4D 09
          4F D0 60 3A 49 74 E3 E6 BC 78 8A DC EB A7 36 EC
          93 D9 A6 4C C3 FB F8 B6 2F D1 DA 59 E4 E3 6F 2C
          38 D5 6E 44 EC 8F 82 B2 C4 FD 1E 38 39 38 97 1A
          7E 74 A4 0F E2 A3 67 81 D4 60 14 23 19 9A B4 22
          A4 BD B4 2C 29 C1 5C BD 3E E2 68 A2 99 E2 B5 34
          64 06 C2 E7 F3 90 12 F7 34 7E 5F 33 B3 8B F3 9B
              [ Another 1 bytes skipped ]
         INTEGER 65537
          }
        }
       }
```

continues

Problems with X.509 (ctd)

```
      [3] {
       SEQUENCE {
        SEQUENCE {
         OBJECT IDENTIFIER keyUsage (2 5 29 15)
         BOOLEAN TRUE
         OCTET STRING, encapsulates {
          BIT STRING 5 unused bits
           '100'B (bit 2)
           }
          }
        SEQUENCE {
         OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
         OCTET STRING, encapsulates {
          SEQUENCE {
           OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
           OBJECT IDENTIFIER emailProtection (1 3 6 1 5 5 7 3 4)
            }
           }
          }
        SEQUENCE {
         OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
         OCTET STRING, encapsulates {
          SEQUENCE {
           SEQUENCE {
            [0] {
             [0] {
              [6]
               'http://crl.alphatrust.com/crl/at1-crl.crl'
               }
              }
             }
            }
           }
          }
        SEQUENCE {
         OBJECT IDENTIFIER subjectAltName (2 5 29 17)
         OCTET STRING, encapsulates {
          SEQUENCE {
           [1] 'bill.brice@alphatrust.com'
            }
           }
          }
```

```
        SEQUENCE {
         OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
         OCTET STRING, encapsulates {
          SEQUENCE {
           SEQUENCE {
            OBJECT IDENTIFIER '2 16 840 1 114003 1 1 1 4 1 1'
            SEQUENCE {
             SEQUENCE {
              OBJECT IDENTIFIER unotice (1 3 6 1 5 5 7 2 2)
              SEQUENCE {
               VisibleString
                'This certificate may only be used by authorized '
                'AlphaTrust Members. ALL LIABILITY TO OTHERS IS D'
                'ISCLAIMED. (c) 2000 AlphaTrust Corporation.'
               }
              }
             SEQUENCE {
              OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
              IA5String
               'https://www.alphatrust.com/rep/policy.htm'
              }
             }
            }
           }
          }
         SEQUENCE {
          OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
          OCTET STRING, encapsulates {
           SEQUENCE {
            SEQUENCE {
             OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
             [6] 'http://validate.alphatrust.com'
              }
             }
            }
           }
```

continues

Problems with X.509 (ctd)

```
    SEQUENCE {
     OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
     OCTET STRING, encapsulates {
      SEQUENCE {
       [0]
        26 5E BF 83 E6 CA 4B 31 79 62 A9 0B 79 F5 27 F5
        13 8D 6A AA
       }
      }
     }
    SEQUENCE {
     OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
     OCTET STRING, encapsulates {
      OCTET STRING
       83 DF 8A B8 27 C4 27 8C B1 AC F9 E2 83 B2 B5 19
       89 45 66 68
      }
     }
    SEQUENCE {
     OBJECT IDENTIFIER basicConstraints (2 5 29 19)
     OCTET STRING, encapsulates {
      SEQUENCE {}
     }
    }
   }
  }
 SEQUENCE {
  OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
  }
 BIT STRING
  69 18 D4 0F 5B 01 34 60 A8 2A 68 D0 ED D5 B8 48
  D1 8A E5 DF 79 51 E7 09 AB E0 90 73 44 E6 E0 F1
  80 2E 1E 5F 79 49 8D CF F3 CE 3D A7 EB 24 F1 FD
  B8 99 3C BC EA 08 1E 1A 58 81 09 4C 93 B0 04 64
  E9 0D 24 93 D0 C5 4A 6A 7B 93 7E 86 B6 90 17 5E
  BB FD 7F BA 7D A8 5C 33 29 9C 66 E6 38 04 0A E3
  63 48 38 21 A3 7D 61 DE 1B 8E 06 C3 D0 7D 57 DA
  48 20 92 19 67 EB E7 E0 2C 9A CC B3 C7 62 57 2F
      [ Another 128 bytes skipped ]
 }
```

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER 966820115
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1
5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationName (2 5 4 10)
     PrintableString 'AlphaTrust'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
     PrintableString '000-001-0001'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER commonName (2 5 4 3)
     PrintableString 'AlphaTrust Global Root Authority'
     }
    }
   }
  SEQUENCE {
   UTCTime 21/08/2000 01:08:35 GMT
   UTCTime 31/12/2020 01:08:35 GMT
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationName (2 5 4 10)
     PrintableString 'AlphaTrust'
     }
    }
```

continues

Problems with X.509 (ctd)

```
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
     PrintableString '000-001-0002'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER commonName (2 5 4 3)
     PrintableString 'AlphaTrust CA1'
     }
    }
   }
  SEQUENCE {
   SEQUENCE {
    OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
    NULL
    }
   BIT STRING, encapsulates {
    SEQUENCE {
     INTEGER
      00 CA B0 08 FA FE A8 A5 4B 5C 16 D6 B6 0D 7D 6E
      C6 7D 21 B1 9F 49 1D 37 41 12 42 D5 B4 03 59 E9
      F4 41 94 D8 D7 6C A8 A5 93 3D EA C7 FE 24 13 FF
      42 04 49 B7 D2 27 57 AB C6 55 9E 37 93 67 F6 18
      E3 65 A1 40 80 F4 80 D3 5D A7 23 FA C5 D8 68 42
      D2 61 9C 98 D1 7B 9F 79 E1 6D E0 9C 17 90 9D 66
      34 52 7C 51 A2 97 2C 52 C6 08 AF 06 C6 DE 09 D5
      1E 5C 63 6F 7E 5A A0 74 98 66 B0 04 B7 0E DE 53
          [ Another 129 bytes skipped ]
     INTEGER 65537
     }
    }
   }
```

```
  [3] {
   SEQUENCE {
    SEQUENCE {
     OBJECT IDENTIFIER basicConstraints (2 5 29 19)
     BOOLEAN TRUE
     OCTET STRING, encapsulates {
      SEQUENCE {
       BOOLEAN TRUE
       }
      }
     }
    SEQUENCE {
     OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
     OCTET STRING, encapsulates {
      SEQUENCE {
       SEQUENCE {
        OBJECT IDENTIFIER '2 16 840 1 114003 1 1 1 101 1 1'
        SEQUENCE {
         SEQUENCE {
          OBJECT IDENTIFIER unotice (1 3 6 1 5 5 7 2 2)
          SEQUENCE {
           VisibleString
            'This certificate may only be used by authorized '
            'AlphaTrust Members. ALL LIABILITY TO OTHERS IS D'
            'ISCLAIMED. (c) 2000 AlphaTrust Corporation.'
           }
          }
         SEQUENCE {
          OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
          IA5String
           'https://www.alphatrust.com/rep/policy.htm'
          }
         }
        }
       }
      }
     }
    }
```

continues

Problems with X.509 (ctd)

```
SEQUENCE {
 OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
 OCTET STRING, encapsulates {
  SEQUENCE {
   SEQUENCE {
    OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
    [6] 'http://validate.alphatrust.com'
     }
    }
   }
  }
 SEQUENCE {
  OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
  OCTET STRING, encapsulates {
   SEQUENCE {
    SEQUENCE {
     [0] {
      [0] {
       [6]
        'http://crl.alphatrust.com/crl/atr-arl.crl'
       }
      }
     }
    }
   }
  }
 SEQUENCE {
  OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
  OCTET STRING, encapsulates {
   SEQUENCE {
    [0]
     19 6B 5F 94 3A 36 94 06 C4 69 27 EE F1 51 E7 09
     C6 17 63 DA
    }
   }
  }
```

```
   SEQUENCE {
    OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
    OCTET STRING, encapsulates {
     OCTET STRING
      26 5E BF 83 E6 CA 4B 31 79 62 A9 0B 79 F5 27 F5
      13 8D 6A AA
     }
    }
   }
  }
 SEQUENCE {
  OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
   }
 BIT STRING
  B1 E2 1F 4F 60 44 F6 A2 07 53 4A F7 A3 6E 52 2F
  DB 33 AA 9E 09 DE FF 78 52 D2 F9 FD A4 BC 6C 5E
  AA 06 EA B6 B3 8A 26 F6 50 E8 40 4B 97 F2 82 40
  7A 07 B6 EA E2 C0 DB 54 CE FD 0F 85 CB 74 B2 55
  1E 00 CA AC B7 AA 84 B3 A4 F9 05 C9 C4 6F 22 3D
  17 04 3B 35 EE 87 19 1E E1 86 5B CA EC ED 69 D5
  F9 95 40 FD FE AB 62 93 23 A1 60 16 04 E5 40 B7
  10 EF 5D 73 9D 25 34 BA 70 65 EF 9A 30 50 D1 77
     [ Another 128 bytes skipped ]
  }
 }
SET {
 SEQUENCE {
  INTEGER 1
  SEQUENCE {
   SEQUENCE {
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER organizationName (2 5 4 10)
      PrintableString 'AlphaTrust'
       }
      }
```

continues

Problems with X.509 (ctd)

```
    SET {
     SEQUENCE {
      OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
      PrintableString '000-001-0002'
       }
      }
     SET {
      SEQUENCE {
       OBJECT IDENTIFIER commonName (2 5 4 3)
       PrintableString 'AlphaTrust CA1'
        }
       }
      }
     INTEGER 967650145
      }
    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 23/09/2000 02:48:58 GMT
        }
       }
     SEQUENCE {
      OBJECT IDENTIFIER messageDigest (1 2 840 113549 1 9 4)
      SET {
       OCTET STRING
        24 F4 24 92 EB 0A 18 75 76 3C 5F 4F FD B0 FB C2
        20 FF B2 69
       }
      }
```

```
     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 40
          }
        SEQUENCE {
         OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
          }
        SEQUENCE {
         OBJECT IDENTIFIER md5 (1 2 840 113549 2 5)
          }
         }
        }
       }
      SEQUENCE {
       OBJECT IDENTIFIER microsoftRecipientInfo (1 3 6 1 4 1 311 16
4)
       SET {
        SEQUENCE {
         SEQUENCE {
          SET {
           SEQUENCE {
            OBJECT IDENTIFIER organizationName (2 5 4 10)
            PrintableString 'AlphaTrust'
             }
            }
```

continues

Problems with X.509 (ctd)

```
          SET {
           SEQUENCE {
            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
            PrintableString '000-001-0002'
             }
           }
          SET {
           SEQUENCE {
            OBJECT IDENTIFIER commonName (2 5 4 3)
            PrintableString 'AlphaTrust CA1'
             }
           }
          }
         INTEGER 967650191
          }
         }
        }
       }
      SEQUENCE {
       OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
       NULL
        }
      OCTET STRING
       5A 3D BD 6C EF 70 21 E6 05 42 C4 4A 6B 6C F5 80
       AC CE 17 56 FD 3A 55 55 4A 36 0D 8D CF 77 C4 81
       D1 3B FC F1 F0 C7 DA 0B A9 4F AD E9 84 D8 3D 3D
       FB A4 53 C4 9E 0C 89 19 2E 93 D7 B7 22 3D 80 B6
       51 7D AC 23 4C D0 AD 3D D3 B1 4C BF 67 05 A1 D7
       4F 4F 9B E3 4C 55 81 FD D5 B0 60 25 54 F7 1A 5D
       D2 58 A3 89 95 0C 38 92 B8 37 86 08 41 4F 9A 68
       D7 7C C4 D4 55 39 40 58 A9 5C 1D 00 79 C5 40 5B
       }
      }
     }
    }
   }
```

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, ACH)

- 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
- Design can be frozen at some point when members sign off on it
  - Continuous flow of standards, drafts, modifications, and proposals is impossible to keep up with
  - PKIX has become a standing committee that will standardise anything with an ASN.1 syntax
    — from ietf-pkix

Problems with X.509 (ctd)

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?
  - Do they owe a duty of care, actionable in the case of negligence?

Problems with X.509 (ctd)

- 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?
  - Current digital signature legislation won't help

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)

- CA flag in certificates is ignored (Microsoft, several Mozilla-derived browsers)
  - Anyone can act as a CA
  - *You* (or Honest Al down at the diner) can issue Verisign certificates
- 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 as blatantly wrong as it seems

Problems with Implementations (ctd)

- Obviously bogus certificates are accepted as valid (Microsoft)


```
-----BEGIN CERTIFICATE-----
MIIQojCCCIoCAQAwDQYJKoZIhvcNAQEEBQAwGDEWMBQGA1UEAxMNS29tcGxleCBM
YWJzLjAeFw01MTAxMDEwMDAwMDBaFw01MDEyMzEyMzU5NTlaMBgxFjAUBgNVBAMT
DUtvbXBsZXggTGFicy4wggggMA0GCSqGSIb3DQEBAQUAA4IIDQAwgggIAoIIAQCA
A++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+////////////////////////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///++++HELLO+THERE++++////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/welcome/to/the/base64/coded/x509/pem/certificate/of///+
+////////////////////////////////////////////////////////////////+
+///KOMPLEX/MEDIA/LABS//////////////////////////////////////////+
+///www/dot/komplex/dot/org//////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///created/by/Markku+Juhani/Saarinen///////////////////////////+
+///22/June/2000///dw3z/at/komplex/dot/org/////////////////////+
+////////////////////////////////////////////////////////////////+
+///You/are/currently/reading/the/public/RSA/modulus//////////+
+///of/our/root/certification/authority/certificate///////////+
+////////////////////////////////////////////////////////////////+
+///Which/happens/to/be/16386/bits/long//////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/fully/working/and/shit///////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/totally/insecure/////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///You/can/save/this/text/to/a/file/called/foo/dot/crt////////+
+///Then/click/on/it/with/your/explorer/and/you/can/see////////+
+///that/your/system/doesn+t/quite/trust/the/komplex/root//////+
+///CA/yet+/////////////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///But/that+s/all/right///////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///Just/install/it////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/you+re/happily/part/of/our/16386/bit/public/key////////+
+///infrastructure//////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///One/more/thing//////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///Don+t/try/read/this/with/other/PKI/or/S/MIME/software/////+
```


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
- CA certs are marked as being invalid for the purpose of issuing certificates (Several large CAs)
  - No-one even noticed

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 (e.g. cert expiry, 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 (e.g. 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)

Implementation Problem Redux

Certified for use with Windows

- Microsoft owns the trademark
- Submit software to Microsoft, who perform extensive testing
- Passing software can use the certification mark
- Reasonable (given the size of the deployed base) interoperability among tested products

S/MIME

- RSADSI owns (owned) the trademark
- Simple interoperability test for signing and encryption
  – Anyone could participate, at no cost
- Passing software can use the certification mark
- Good interoperability among tested products

Implementation Problem Redux (ctd)

X.509

- No quality control
- You cannot build software so broken than it can't claim to be X.509v3

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)
    - (Ignoring implementation bugs for now)
  - It's safe to submit the $1,500 payment request
- Some merchants actually put padlock GIFs on their pages to reassure users

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
    - They do if you spell their name Quantas (!!)

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
  - Treat them as meaningless blobs

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
- One-size-fits-all approach is only useful to verify well-known entities
  - amazon.com et al
  - Banks
  - Government departments
- Application-specific approaches work better for everything else
  - Use the same channels to verify Bob's key as you use to verify other transactions with Bob
- Third-party CAs merely get in the way

PKI Design Guidelines (ctd)

Application-specific PKIs

- 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 (e.g. tax filing)
  - The authority knows who its users/clients are; everyone knows who the authority is
  - 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
  - (US) Business Records Exception allows standard business records to be treated as evidence (rather than hearsay) in court
    - Following standard legal precedent is easier than becoming a test case for PKI

PKI Design Guidelines (ctd)

- Example: Business transactions
  - Ask Citibank about certificate validity

  Vs.
  - Ask Citibank to authorise the transaction directly

    → Use an online authorisation
  - Well-established mechanisms (and much legal precedent) for online authorisation
  - Strong consumer protection via Reg.E and Reg.Z
    - Report loss within 2 days: No liability
    - Report loss within 2-60 days (time to get a bank statement): Liability of $50
  - Enacted when ATM/credit cards were introduced to keep the banks honest
    - Highly effective (c.f. UK banks' card security)

PKI Design Guidelines (ctd)

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

- Provides broad toolkit and crypto token support without tying you to X.509 peculiarities
- 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

"PKI: It's not dead, just resting", IEEE Computer, August 2002.