X.509 Certificates

PKI: The OSI of a new generation

Certificate Structure

Version (marked as X.509v3, even if v4 or v5)
Serial number Issuer name (DN)
Validity (start and end time)
Subject name (DN)
Subject public key
Extensions (added in X.509v3) Extra identification information, usage constraints, policies, and general kitchen-sink area

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

Validity field indicates when certificate renewal fee is due

Certificate Structure (ctd)

Typical certificate

- Serial Number = 177545
- Issuer Name = Verisign
- ValidFrom = 12/09/05
- ValidTo = 12/09/06
- Subject Name = John Doe
- Public Key = RSA public key

Certificate Extensions

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

- X.509v3 added certificate extensions to augment X.509v1/v2 certificates
- X.509v4 and X.509v5 just add more extensions to X.509v3
- All indicators are that this will continue in perpetuity

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

Certificate Extensions (ctd)

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

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

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

Certificate Extensions (ctd)

Actual definitions of critical flag usage are extremely vague

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

Certificate Extensions (ctd)

Extensions come in two types

Usage/informational extensions

- Provide extra information on the certificate and its owner

Constraint extensions

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

Usage Extension: Key Usage

Defines the purpose of the key in the certificate

digitalSignature

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

Usage Extension: Key Usage (ctd)

nonRepudiation

- Binding long-term signature (performed consciously)
- Another school of thought holds that nonRepudiation acts as an additional service on top of digitalSignature
- Certificate profiles are split roughly 50:50 on this
- Later versions of X.509 renamed the flag "contentCommitment" to resolve the ambiguity
  - Doesn't help because now you can't tell whether an implementation uses nonRepudiation interpretation #1, nonRepudiation interpretation #2, or contentCommitment

Usage Extension: Key Usage (ctd)

keyEncipherment

- Exchange of encrypted session keys (RSA)

keyAgreement

- Key agreement (DH)

keyCertSign/cRLSign

- Signature bits used by CA's

Usage Extension: Key Usage (ctd)

No-one really knows what the nonRepudiation bit signifies

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

Possible definition: "Nonrepudiation is anything that fails to go away when you stop believing in it"

Usage Extension: Key Usage (ctd)

If you can convince someone that it's not worth repudiating a signature then you have nonrepudiation

- Have them sign a legal agreement promising not to do it
- Convince them that the smart card they used is infallible and it's not worth going to court over ("Chip and PIN")
- Threaten to kill their kids

The only definitive statement that can be made upon seeing the NR bit set is "The subscriber asked the issuing CA to set this bit"

- Suggestion that CAs set this bit at random just to prevent people from arguing that its presence has a meaning

Usage Extension: Extended Key Usage

Extended forms of the basic key usage fields

- serverAuthentication
- clientAuthentication
- codeSigning
- emailProtection
- timeStamping
- … any many more

Usage Extension: Extended Key Usage (ctd)

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

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

As usual, PKI people are split 50:50 on this

Usage Extension: Netscape Key Usage

Netscape cert-type

- An older Netscape-specific extension that performed the same role as keyUsage, extKeyUsage, and basicConstraints
- Still used even today by some CAs (!!), ten years after it was discontinued

Usage Extension: Private Key Usage Period

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

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

Disallowed by the PKIX profile, although no-one can provide a reason for this

Usage Extension: Alternative Names

Everything that doesn't fit in a DN

- rfc822Name
  - email address, `dave@wetaburgers.com`
- dNSName
  - DNS name for a machine, `ftp.wetaburgers.com`
- uniformResourceIdentifier
  - URL, `http://www.wetaburgers.com`
- iPAddress
  - 202.197.22.1 (encoded as `0xCAC51601`)
- x400Address, ediPartyName
  - X.400 and EDI information

Usage Extension: Alternative Names (ctd)

- directoryName
  - Another DN, but containing stuff that you wouldn't expect to find in the main certificate DN
  - Exists because the alternative name uses a data type called a GeneralName, of which a DN is a little-used subset
- otherName
  - Type-and-value pairs (type=MPEG, value=MPEG-of-cat)

Usage Extension: Certificate Policies

Originally implicit in PEM (X.509v1) certificates

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

Information on the CA policy under which the certificate is issued

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

Usage Extension: Certificate Policies (ctd)

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

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

Usage Extension: Certificate Policies (ctd)

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

- Many policies serve mainly to protect the CA from liability

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

  — Verisign certificate policy
- Effectively these certificates have null semantics
- If CAs didn't do this, their potential liability would be enormous
  - Universal ID certs → universal liability
  - Closed PKIs restrict this problem to manageable levels

Usage Extension: Policy Mappings (ctd)

Maps one CA's policy to another CA

- Allows verification of certificates issued under other CA policies
  - "For verification purposes we consider our CA policy to be equivalent to the policy of CA *x*"

Mapping of constraints is left hanging

Constraint Extension: Basic Constraints

Whether the certificate is a CA certificate or not

- Prevents users from acting as CAs and issuing their own certificates
- Redundant, since keyUsage specifies the same thing in a more precise manner
  - Occurred because various trial-and-error proto-X.509v3 extension ideas have lingered on into current versions

Much confusion over its use in non-CA certificates

- German ISIS profile mandates its use
- Italian profile forbids its use

Constraint Extension: Name Constraints

Constrain the DN subtree under which a CA can issue certificates

- Constraint of C=NZ, O=University of Auckland would enable a CA to issue certificates only for the University of Auckland

Main use is to balkanize the namespace so that a CA can buy or license the right to issue certificates in a particular area

- Constraints can also be applied to email addresses, DNS names, and URLs

Constraint Extension: Policy Constraints

Can be used to disable certificate policy mappings

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

Other Extensions

Many, many more extensions

- See earlier slide on the perpetuum mobile of X.509 standards work
- Most really belong to the problem set rather than the solution set

Too obscure to cover

- No-one but the vendor who proposed the extension (if that) bothers to implement it

Certificate Profiles

X.509 is extremely vague and nonspecific in many areas

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

PKIX: Internet PKI profile

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

Certificate Profiles (ctd)

FPKI: (US) Federal PKI profile

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

MISSI: US DoD profile

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

Certificate Profiles (ctd)

ISO 15782: Banking — Certificate Management Part 1: Public Key Certificates

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

Canada

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

Certificate Profiles (ctd)

SEIS: Secured Electronic Information in Society

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

Certificate Profiles (ctd)

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

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

ISIS: German Industrial Signature Interoperability Spec

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

Certificate Profiles (ctd)

Australian PKAF Profile

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

German Profile: German Digital Signature Law profile

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

Certificate Profiles (ctd)

SIRCA Profile: (US) Securities Industry Association

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

Microsoft Profile (de facto profile)

- Rejects certificates with critical extensions (eventually fixed)
- Always seems to set the nonRepudiation flag when the digitalSignature flag is set
- Ignores the keyUsage flags
- Many, many other peculiarities, see the implementation problems section

Certificate Profiles (ctd)

Many, many more

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

— Frank Zappa

And an X.509 profile.

— Peter Gutmann

CA Policies

Serve three functions

- Provides a CA-specific mini-profile of X.509
- Defines the CA terms and conditions/indemnifies the CA
- Hides kludges for PKI problem areas

CA policy may define

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

CA Policies (ctd)

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

CA Policies (ctd)

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

CA Policies (ctd)

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

CA Policies (ctd)

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

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

Problems with X.509 (ctd)

Standards groups are working on protocols to fix this

- Progress is extremely slow
- Instead of fixing one standard when it doesn't work, create a new one
  - Better yet, create three new ones

Most of the world has stopped caring about new PKI developments, so most of the new standards are being ignored by implementers

Problems with X.509 (ctd)

Certificates are based on owner identities, not keys

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

Problems with X.509 (ctd)

Revocation should revoke the capability, not identities

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

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

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

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

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

Problems with X.509 (ctd)

Aggregation of attributes shortens the overall certificate lifetime

- Steve's Rule of Revocation: Frequency of certificate change is proportional to the square of the number of attributes
- Inflexibility of certificates 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 that 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'
          }
         }
```

```
       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'
          }
         }
```

continues

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

Problems with X.509 (ctd)

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

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

Problems with X.509 (ctd)

In a closed system (SWIFT, Identrus, 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 was created to do one thing but 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?

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 that comes with it

Residential certificates (certificates for private individuals) 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: The Trivial

rfc822Name has an ambiguous definition and implementations (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: The Trivial (ctd)

CAs / PKI apps get subjectKeyID / authKeyID wrong (too many to list)

- CA copies subjKID into authKID field
  - Fields have a completely different structure
  - Undetected by Eudora, Mulberry, Netscape 4.x – 6.x, OpenSSL, OS X Mail, Windows
- Major CA stores binary garbage as authKID
  - No-one noticed
- European national CA encodes empty authKID

```
666   9:       SEQUENCE {
668   3:         OBJECT IDENTIFIER authKeyID
673   2:         OCTET STRING, encapsulates {
675   0:           SEQUENCE {}
       :           }
```

Problems: The Trivial (ctd)

- CAs create circular references for authKID / subjKID
  - AIA / altNames can also contain circular references (URLs)
  - "Processing" this extension presumably requires an infinite loop
- Not a big problem, most apps seem to ignore these values anyway (obviously)

The other CA didn't populate the [field] at all, justifying it with "Everything ignores those anyway, so it doesn't matter what you put in there"

— Peter Gutmann on ietf-pkix

Problems: The Trivial (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

Problems: The Trivial (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

... continues...

Problems: The Trivial (ctd)

... continued...

- Ordering/non-ordering of DN elements
  - Allow only one attribute type (e.g. OU) per DN
  - Assume CN is always encoded last

(And so on ad nauseum, there are simply too many examples to list)

Problems: The Serious

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

Software hard-codes the certificate policy so that any policy is treated as if it was the Verisign one (Microsoft)

- Some implementations hardcode checks for obscure cert constraints
- c.f. Dhrystone detectors in compilers

Problems: The Serious

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 will forget to perform any certificate validity checks (e.g. expiry times, CA certs) if CRL checking is enabled

(As before, there is too much of this to list)

Problems: The Scary

CA flag in certificates is ignored (Microsoft, Konqueror/ KDE, Lynx, Baltimore's S/MIME plugin, various others)

- Anyone can act as a CA
- *You* (or Honest Al down at the diner) can issue Verisign certificates
- This was known among PKI developers for *five years* before widespread publicity forced a fix

CA certs have basicConstraints CA = false (Several large CAs, PKIX RFC (!!))

- No-one noticed

Problems: The Scary (ctd)

Survey of CA certs in MSIE by Laurence Lundblade found:

- 34 had basicConstraints present and critical
- 28 had basicConstraints present and not critical
- 40 did not have basicConstraints present
  - Some of these were X.509v1

So have CAs also issued EE certs with basicConstraints CA = true?

- Yes
  - Consider the interaction of this with the implicit universal cross-certification model

Problems: The Scary (ctd)

Toxic co-dependency of broken certs and broken implementations

- Programmer has a pile of broken certs from big-name CAs/the PKIX RFC
- Ignoring basicConstraints makes them "work"
- CAs can continue issuing broken certs; implementations can continue ignoring basicConstraints

  There is a fine line between tolerant and oblivious. A lot of security software which is built around highly complex concepts like PKI works mostly because it's the latter

  — Peter Gutmann on ietf-pkix

Problems: The Scary (ctd)

Software ignores the key usage flags and uses the first cert it finds for whatever purpose it currently needs (a who's who of PKI vendors)

- If Windows users have separate encryption and signing certs, the software will grab the first one it finds and use it for both purposes
  - This makes things less confusing for users

Problems: The Scary (ctd)

- CryptoAPI ignores usage constraints on keys for user convenience purposes
  - AT_KEYXCHANGE keys (with corresponding certificates) can be used for signing and signature verification without any trouble

When I use our CSP to logon to a Windows 2000 domain, the functions SignHash AND ImportKey are both called with the AT_EXCHAGE !! Key. The certificates […] only requires the keyusage DS bit to be true. So the public key in the certificate can only be used to verify a signature. But again: […] the key is also used to Import a Session key. This is NOT allowed because the keyusage keyenc is not defined.

— Erik Veer on the CryptoAPI list

Problems: The Scary (ctd)

- Large PKI vendor ran an interop test server
  - Successfully tested against a who's who of other PKI vendors
  - After 2 years of operation, I pointed out that the certs' critical key usage didn't allow this
- European govt. organisation marked signature keys as encryption-only
  - No-one noticed
- European CA marked signature key as non-signature key
- Another CA marked their root cert as invalid for cert signing
  - Other CAs mark keys as invalid for their intended (or any) usage
- CA reversed bits in keyUsage

Problems: The Scary (ctd)

- The self-invalidating cert
  - Policy text: Must be used strictly as specified in keyUsage
  - Key usage: keyAgreement (for an RSA key)

What happens when you force the issue with sig-only algo?

> I did interop testing with outlook, netscape mail, and outlook with entrust s/mime plugin […] at that time I could elicit a blue screen and crypto library internal error from outlook and netscape mail respectively by giving them a DSA cert (marked with key usage of sig only). (How I came to this discovery was I tried imposing key usage restrictions and they were ignoring key usage = sign only on RSA keys, encrypting to them anyway, so I figured well let's see them try to encrypt with DSA, and lo they actually did try and boom!)
>
> — PKI app developer

Problems: The Scary (ctd)

Hi. My name is Peter and I have a keyUsage problem. Initially it was just small things, a little digitalSignature after lunch, maybe a dataEncipherment after dinner and keyAgreement as a nightcap. Then I started combining digitalSignature and keyEncipherment in the same certificate. It just got worse and worse. In the end I was experimenting with mixing digitalSignature and nonRepudiation, and even freebasing keyCertSigns. One morning I woke up in bed next to a giant lizard wearing a Mozilla t-shirt, and knew I had a problem.

It's now been six weeks since my last nonRepudiation…

— Peter Gutmann on ietx-pkix

Problems: The Scary (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

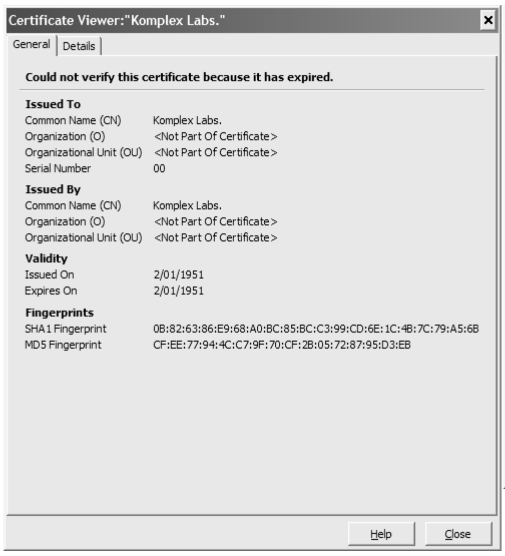
  — MS PKI architect
- Given the complete chaos in DNs, this isn't quite the blatantly wrong decision that it may seem to be

Problems: The Scary (ctd)

Obviously bogus certificates are accepted as valid (MS)

```
-----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: The Scary (ctd)



Could not verify this certificate because it has expired.

Issued To

Common Name (CN)	Komplex Labs.
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	00

Issued By

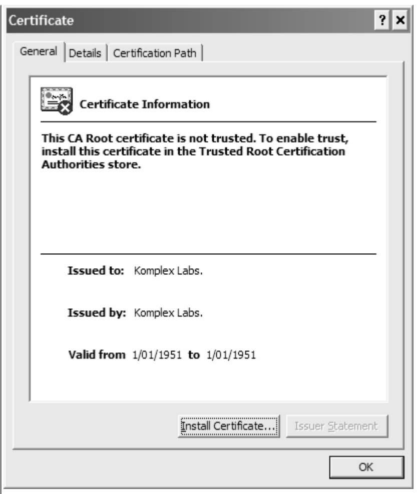
Common Name (CN)	Komplex Labs.
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Validity

Issued On	2/01/1951
Expires On	2/01/1951

Fingerprints

SHA1 Fingerprint	0B:82:63:86:E9:68:A0:BC:85:BC:C3:99:CD:6E:1C:4B:7C:79:A5:6B
MD5 Fingerprint	CF:EE:77:94:4C:C7:9F:70:CF:2B:05:72:87:95:D3:EB



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: Komplex Labs.

Issued by: Komplex Labs.

Valid from 1/01/1951 **to** 1/01/1951

Problems: The Scary (ctd)

- Validity period is actually January 1951 to December 2050
  - At one point MS software was issuing certificates in the 17th century
    - This was deliberate

the text should be changed to address the question of dates prior to 1950

— MS PKI architect on ietf-pkix

I agree with this. Every time I load one of these pre-1950 certs into the ENIAC in the basement it blows half a dozen tubes trying to handle the date, and it takes me all afternoon to replace the fried circuits. My Difference Engine handles it even more poorly, the lack of extra positions in the date field breaks the teeth of several of the gears

— Peter Gutmann, in response

Problems: The Scary (ctd)

- Software reports validity as 1 January 1951 to 1 January 1951, but accepts it anyway
  - It actually has a negative validity period (–1 second)
- Certificate is unsigned but cert is accepted as valid

```
30 20 30 0C 06 08 2A 86
48 86 F7 0D 02 05 05 00
04 10 A1 A1 1C 22 90 61
AF 58 8C E6 5D 40 48 BF
4D 21
```

  - RSA key has exponent 1, “signing” = no-op

PGP implementations performed key validity checks after the Klima-Rosa attack

- Most X.509 implementations don’t seem to do this

Problems: The Scary (ctd)

CAs issue certificates with e = 1

- Problem was only noticed when Mozilla was modified to detect some obviously bogus RSA keys

  Both of these certs have the same problem: The RSA public key has a public exponent value that is the number 1 !! [...] I'm surprised to find certs in actual use with such a public key, especially in certs issued by well known public CAs!

  — Comment on Bugzilla
- Consider the interaction of this with the universal implicit cross-certification employed in browsers
- CryptoAPI uses e = 1 keys as a standard (documented) technique for plaintext key export

Problems: The Scary (ctd)

CRL checking is broken (Microsoft)

- Hard-codes a Verisign URL for CRL checks
- Older versions of MSIE, Outlook 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
- If outgoing LDAP (CRL fetch) is blocked, the software hangs until it times out, then continues anyway
- Outlook 2000 doesn't check for a CRL and always reports a cert as not revoked (requires registry hack to turn on)

  continues…

Problems: The Scary (ctd)

> Today I noticed that the CRLs all have a “Next Update” date of 1/29/04, and since today is 3/26/04, I can’t understand how these CRLs could still be working [...] I have been able to test that even when the “Next Update” date on CRLs has passed, IIS is still processing connection requests normally [...] Since the last post, I’ve been continuing to try all manner of things to try to get Windows 2000 AS to actually "care" about the validity period of the CRL, but unfortunately, have failed [...] This may be a nuance with IIS 5.0, but many applications treat no CDP in certs as an indicator that revocation does not need to be checked.
>
> — Excerpts from a thread in MS security groups

- Outlook 2002 checks for a CRL but can’t determine whether a cert is revoked or not (CRLDP-related bug)

Behaviour is representative of other PKI apps

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)

How Far Can You Trust a PKI App?

After a decade of effort, we've almost made it to the first step beyond X.509v1 (basicConstraints)

> There's not a single X.509v3 extension defined in PKIX a PKI designer can really rely on. For each and every extension somebody planning/deploying a PKI has to check each and every implementation if and how this implementation interpretes this extension. This is WEIRD!
>
> – Michael Ströder on ietf-pkix

- When asked about this, PKI people just change the subject: Not my problem

We're expecting banks to protect funds with this stuff?

> Having worked with PKI software, I wouldn't trust it to control access to our beer fridge.
>
> – Developer, international software company

Implementation Problem Redux

Certified for use with Windows / WHQL

- 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
- Certified software provides certain guarantees
  - UI style
  - Install / uninstall procedures
  - Interaction with other components
  - Use of registry, correct directories, per-user data, etc etc

Implementation Problem Redux (ctd)

S/MIME

- RSADSI owns (owned) the trademark
- Simple interoperability test for signing and encryption
  - Anyone could participate, at no cost
  - Send signed + encrypted message to interop site
  - Process returned signed + encrypted message
- 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 that it can’t claim to be X.509v3
- (Deeply-buried) PGP has been sold as X.509
- “The other side is using a different version of X.509” explained away interop problems

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 that 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

- Little interoperability/compatibility above basic X.509v1 levels
- Absolutely no quality control
- Lack of expertise in deploying/using a PKI
- No manageability
- Huge up-front infrastructure requirements
  - Few organisations realise just how much time, money and resources will be required
  - Incremental change to legacy systems is easier than starting from scratch with a PKI

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

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

How Effective are Certificates Really?

Ten years after they were introduced, HCI researchers evaluated certificate use with browsers

- No-one had ever thought of doing this before

The results were scary

- The effect of certificates, demonstrated across multiple studies, was indistinguishable from placebo

This is a complex topic, see the security HCI material for details