Practical Aspects of Modern Cryptography

Josh Benaloh & Brian LaMacchia

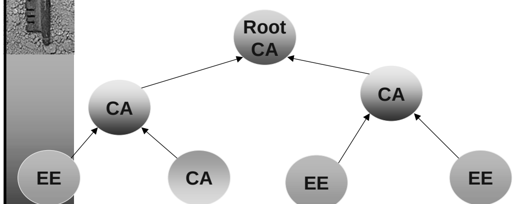
Lecture 6: Certificates & Trust Part II -- X.509 vs PGP

Last time on CSE 590...

- Certificates
- Certificate Authorities
  - "Certificate Enrollment" -- acquiring a cert from a CA
- Trusted Root CAs
- CA Hierarchies
- Expiration & Revocation

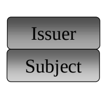
CA Hierarchies

- CAs can certify other CAs or "end entities"
- Certificates are links in a tree of EEs & CAs



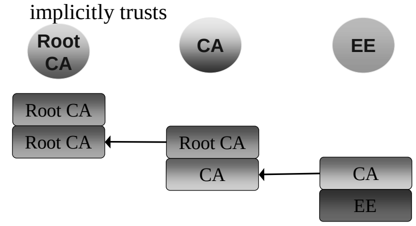
BAL's No-Frills Certs

- Certificates can contain all sorts of information inside them
- In abstract, they're just statements by an issuer about a subject



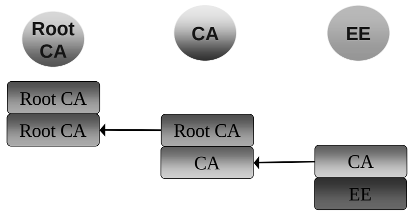
Does Alice trust Bob's Key?

- Alice trusts Bob's key if there is a chain of certificates from Bob's key to a root CA that Alice implicitly trusts



Chain Building & Validation

- "Given an end-entity certificate, does there exist a cryptographically valid chain of certificates linking it to a trusted root certificate?"



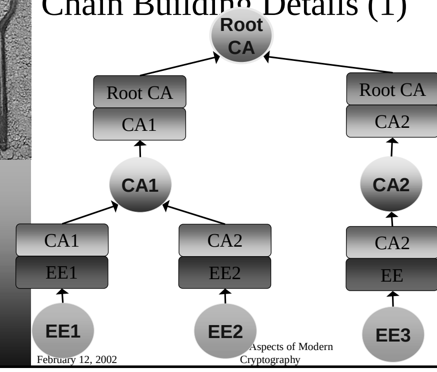
Today on CSE 590...

- Building Certificate Chains
- The innards of an X.509v3 Certificate
  - Distinguished Names
  - A plethora of extensions
- PGP – Phil's Pretty Good Privacy
  - PGP certificates
  - PGP keyservers (certificate directories)

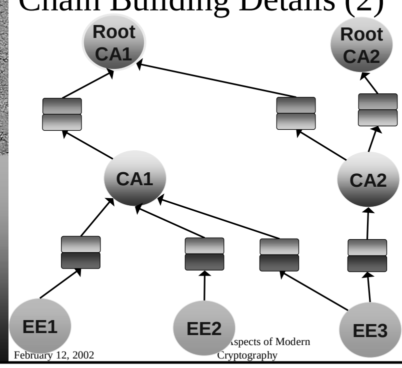
Chaining Certificates

- In theory, building chains of certificates should be easy
  - "Just link them together like dominos"
- In practice, it's a lot more complicated...

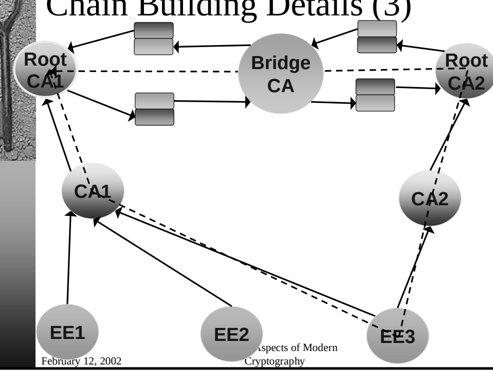
Chain Building Details (1)



Chain Building Details (2)



Chain Building Details (3)



Chaining Certificates

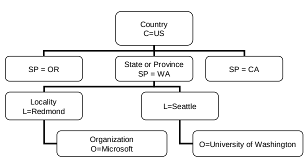
- How do we determine whether two certificates chain together?
  - You'd think this was an easy problem...
  - But it's actually a question with religious significance in the security community
  - *"Are you a believer in names, or in keys?"*
- In order to understand the schism, we need to digress for a bit and talk about names and some history

The X.500 Directory Model

- The model SSL/TLS uses, the X.509 certificate model, is based on names
  - *Names as principles*
- Specifically, X.509 is based on the X.500 directory model
- X.500 defined a global, all-encompassing directory, to be run by the telcos
  - *One directory to rule them all, one directory to define them...*

X.500 Distinguished Names

- In the X.500 model, everything has a single, unique, global, assigned name
  - There is a worldwide hierarchy, and you're in it!



X.500 DNs

- Typical X.500 DN
  - C=US/
L=Area 51/
O=Hanger 18/
OU=X.500 credential acquisition for extra-terrestrial visitors/
CN=John Whorfin
- *When the X.500 revolution comes, your DN will be lined up against the wall and shot*

Problems with X.500 DNs

- No one ever figured out how to make them work
  - No clear plan on how to organize the one global hierarchy
  - People couldn't even agree on the meaning of "localities"
- Hierarchical naming model fits the military & governments real well, but doesn't work well for businesses & individuals

Problems with X.500 DNs (2)

- Consider the following simple cases
  - Communal living (jails, college dormitories)
  - Nomadic peoples
  - Merchant ships
  - Quasi-permanent non-continental structures
    - Oil drilling platforms
  - US APO addresses

Problems with X.500 DNs (3)

- What is C, SP, L for a corporation?
  - Location of headquarters?
  - Location of office where the CA is located?
  - Location of incorporation?
- What is C, SP, L for a person?
  - Current residence?
  - Place of birth?
  - Place of work?
- Solution: Define in the certificate practice statement (CPS), incorporated by reference in the cert, which no one but lawyers ever reads.

DNs in Practice

- Name is unique within the scope of the CA's name
- Public CAs (e.g. Verisign) typically set
  - C = CA Country
  - O = CA Name
  - OU = Certificate type/class
  - CN = User name
  - E= email address

Private-label DNs

- If you own the CA, you get to decide what fields go in the DN
  - Really varies on what the software supports
- Can get really strange as people try to guess values for fields that are required by software
  - Software requires an OU, we don't have OUs, so I better make something up!

DNs in X.509 Certificates

- The X.509 certificate standard began as a way to associate a certificate with a node in the directory.
- How is the subject of a cert identified?
  - By its DN.
- How is the issuer of a cert identified?
  - By its DN.
- How are certificates linked together?
  - By DNs.

Key fields in a certificate

- The core fields of an X.509 certificate are
  - The subject public key
  - The subject Distinguished Name
  - The issuer Distinguished Name
- What's missing here?
  - The issuer's public key is <u>not</u> present in the certificate.
  - You can't verify the signature on the cert without finding a parent cert!

Religion

- X.509 certificates are part of the "names as principles" camp
  - "The important thing in an X.509 cert is the DN, everything else is along for the ride."
- The X.509 assumption is that you always have access to the global directory
  - Need to find the issuer's public key? Use the issuer DN to query the global directory, find the user object, and find his one & only certificate

Certificate Extensions

- An extension consists of three things
  - A “critical” flag (boolean)
  - A type identifier
  - A value

Critical Flags

- The “critical flag” on an extension is used to protect the issuing CA from assumptions made by software that doesn’t understand (implement support for) a particular extension
  - If the flag is set, relying parties must process the extension if they recognize it, or reject the certificate
  - If the flag is not set, the extension may be ignored

Critical Flags (2)

- Some questions you might be asking yourself right now...
- What does “must process the extension if they recognize it” mean?
  - What does “recognize” mean?
  - What does “process” mean?
  - You’ve got me....
  - The IETF standards folks didn’t know either...

Critical Flags (3)

- Actual definitions of flag usage are vague:
  - X.509: Non-critical 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 constrain extensions” but “support” is never defined.
  - S/MIME: Implementations should “correctly handle” certain extensions
  - Verisign: “All persons shall process the extension...or else ignore the extension”

Types of Extensions

- There are two flavors of extensions
  - Usage/informational extensions, which provide additional info about the subject of the certificate
  - Constraint extensions, which place restrictions on one or more of:
    - Use of the certificate
    - The user of the certificate
    - The keys associated with the certificate

Some common extensions

- Key Usage
  - digitalSignature
    - “Sign things that don’t look like certs”
  - keyEncipherment
    - Exchange encrypted session keys
  - keyAgreement
    - Diffie-Hellman
  - keyCertSign/keyCRLSign
    - “Sign things that look like certs”
  - nonRepidiation

NonRepudiation

- The nonRepudiation bit is the black hole of PKIX
  - It absorbs infinite about of argument time on the mailing list without making any progress toward understanding what it means
  - What does it mean? How do you enforce that?
  - No one knows...
- “Nonrepudiation is anything which fails to go away when you stop believing in it”

More common extensions

- Extended Key Usage
  - Because Key Usage wasn’t confusing enough!
- Private Key Usage Period
  - CA attempt to limit key validity period
- Alternative names
  - Everything which doesn’t fit in a DN
  - RFC822 names, DNS names, URIs
  - IP addresses, X.400 names, EDI, etc.

More certificate extensions

- Certificate policies
  - Information identifying the CA policy that was in effect when the cert was issued
  - Policy identifier
  - Policy qualifier
  - Explicit text
  - Hash reference (hash + URI) to a document
- X.509 defers cert semantics to the CA’s issuing policy
- Most CA policies disclaim liability

Even more extensions

- Policy mappings
  - Convert one policy ID into another
- Basic constraints
  - Is the cert a CA cert?’
  - Limits on path length beneath this cert
- Name constraints
  - Limits on types of certs this key can issue
- Policy constraints
  - Anti-matter for policy mappings

Exploring inside an X.509 Cert

Exploring inside an X.509 Cert

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```

Inside an X.509 Cert (2)

```
SET {
  SEQUENCE {
   OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
   PrintableString 'WA'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER localityName (2 5 4 7)
   PrintableString 'Redmond'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationName (2 5 4 10)
   PrintableString 'Microsoft'
   }
  }
 SET {
  SEQUENCE {
   OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
   PrintableString 'ITG'
   }
  }
```


Inside an X.509 Cert (3)

```
SET {
   SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'Microsoft Intranet FTE User CA 2'
    }
   }
  }
 SEQUENCE {
  UTCTime '011019235011Z'
  UTCTime '021019235011Z'
  }
 SEQUENCE {
  SET {
   SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'Brian LaMacchia'
    }
   }
  }
```


Inside an X.509 Cert (4)

```
SEQUENCE {
  SEQUENCE {
   OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
   NULL
   }
  BIT STRING 0 unused bits, encapsulates {
   SEQUENCE {
    INTEGER
     00 C5 8F 7C 84 CD 23 BC FA F7 1C 1C BD 26 EB 8B
     B7 5C A6 0F B7 19 4D 02 FF F5 95 31 6E 4A CE 92
     82 B2 0B E7 90 DC 7D 5A F7 E6 8F BE B1 C5 41 76
     04 4F 7C 5F 29 76 07 71 06 2D A8 6A EB 33 7E 3E
     78 0D 44 27 7F FC 62 A0 52 3F AD 05 CD 72 21 49
     A5 96 7D C5 6B 89 1C 24 43 54 DB 75 A5 A0 BE E0
     20 27 FA F2 2B AA 65 76 B1 61 B6 EB C2 63 53 24
     C0 F9 64 2D BD 16 CD 36 12 5A CE E7 EB 1B 6E FD
             [ Another 1 bytes skipped ]
    INTEGER 65537
    }
   }
  }
```


Inside an X.509 Cert (5)

```
[3] {
  SEQUENCE {
   SEQUENCE {
    OBJECT IDENTIFIER keyUsage (2 5 29 15)
    OCTET STRING, encapsulates {
      BIT STRING 7 unused bits
       '1'B (bit 0)
      }
    }
   SEQUENCE {
    OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
    OCTET STRING, encapsulates {
      OCTET STRING
       B6 DF 93 F1 85 8B 7D EF 1D 39 6A C4 C9 A6 30 98
       E0 69 0B A5
      }
    }
   SEQUENCE {
    OBJECT IDENTIFIER '1 3 6 1 4 1 311 20 2'
    OCTET STRING, encapsulates {
      BMPString 'ClientAuth'
      }
    }
```


Inside an X.509 Cert (6)

```
SEQUENCE {
     OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
     OCTET STRING, encapsulates {
       SEQUENCE {
        [0]
         30 3C 1D 78 16 7C AD 8B 89 85 97 D4 E8 3E 7F 85
         E3 41 B8 89
        }
       }
     }
    SEQUENCE {
     OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
     OCTET STRING, encapsulates {
       SEQUENCE {
        SEQUENCE {
         [0] {
          [0] {
           [6]
            'ldap:///CN=Microsoft%20Intranet%20FTE%20User%20C'
            'A%202,CN=reditgcac03,CN=CDP,CN=Public%20Key%20Se'
            'rvices,CN=Services,CN=Configuration,DC=corp,DC=m'
            'icrosoft,DC=com?certificateRevocationList?base?o'
            'bjectclass=cRLDistributionPoint'
           }
          }
         }
```


Inside an X.509 Cert (7)

```
SEQUENCE {
         [0] {
          [0] {
           [6]
            'http://reditgcac03.redmond.corp.microsoft.com/Ce'
            'rtEnroll/Microsoft%20Intranet%20FTE%20User%20CA%'
            '202.crl'
           }
          }
         }
        }
       }
      }
     SEQUENCE {
      OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
      OCTET STRING, encapsulates {
        SEQUENCE {
         SEQUENCE {
          OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
          [6]
           'ldap:///CN=Microsoft%20Intranet%20FTE%20User%20C'
           'A%202,CN=AIA,CN=Public%20Key%20Services,CN=Servi'
           'ces,CN=Configuration,DC=corp,DC=microsoft,DC=com'
           '?cACertificate?base?objectclass=certificationAut'
           'hority'
          }
```

Inside an X.509 Cert (8)

```
SEQUENCE {
        OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
        [6]
          'http://reditgcac03.redmond.corp.microsoft.com/Ce'
          'rtEnroll/reditgcac03.redmond.corp.microsoft.com_'
          'Microsoft%20Intranet%20FTE%20User%20CA%202(3).cr'
          't'
         }
       }
     }
   }
  SEQUENCE {
   OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
   OCTET STRING, encapsulates {
     SEQUENCE {
      OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
      }
     }
   }
```


Inside an X.509 Cert (9)

```
SEQUENCE {
    OBJECT IDENTIFIER subjectAltName (2 5 29 17)
    OCTET STRING, encapsulates {
      SEQUENCE {
        [0] {
          OBJECT IDENTIFIER '1 3 6 1 4 1 311 20 2 3'
          [0] {
            UTF8String
              'bal@redmond.corp.microsoft.com..edmond,DC=corp,D'
              'C=microsoft,DC=com'
            }
          }
        }
      }
    }
  }
}
```


Inside an X.509 Cert (10)

```
SEQUENCE {
  OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
  }
BIT STRING 0 unused bits
  78 AE FB D9 B9 3C 0A 5C 99 DE 86 BB BE DB 72 2E
  E7 E4 AC 8D 8B F8 53 5E EC B1 73 43 2F 21 89 CC
  59 DD 4E C1 77 C1 F5 9C 08 35 68 C7 51 B7 05 93
  5A 26 E5 6E D8 F9 C3 2C C3 A4 D0 7F FB 52 57 B1
  6D 1C FC 3C 4D 1F F6 CF 0C 57 00 8B 20 DA 43 13
  35 A2 5F C4 EC 0E 72 98 97 06 70 5D 34 F0 43 0B
  B7 62 CD EC C4 F4 33 81 FB 0C 9B C0 68 EC FF FA
  B3 31 D6 07 C9 93 F2 BA 68 92 5A 4E 3E B0 2F 14
      [ Another 128 bytes skipped ]
}
```


Wow...

- And that's just one certificate!
  - We usually need a chain of 3 or 4 of those to make a trust decision.
- Let's go back, take a look at each field, and understand why it's there and what role it plays in building & evaluating cert chains...

ASN.1 Structures

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2

  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```


Certificate Version

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2

  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```

Serial Number

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```


Signing Algorithm

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```


Start of the Issuer DN

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```


Issuer DN – emailAddress

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```


Issuer DN – Country

```
SEQUENCE {
 SEQUENCE {
  [0] {
   INTEGER 2
   }
  INTEGER
   58 90 44 5F 00 03 00 00 E1 60
  SEQUENCE {
   OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
   NULL
   }
  SEQUENCE {
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)
     IA5String 'pkit@microsoft.com'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER countryName (2 5 4 6)
     PrintableString 'US'
     }
    }
```


Issuer DN – State

```
SET {
    SEQUENCE {
     OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
     PrintableString 'WA'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER localityName (2 5 4 7)
     PrintableString 'Redmond'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationName (2 5 4 10)
     PrintableString 'Microsoft'
     }
    }
   SET {
    SEQUENCE {
     OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
     PrintableString 'ITG'
     }
    }
```

Issuer DN – Locality

```
SET {
  SEQUENCE {
    OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
    PrintableString 'WA'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER localityName (2 5 4 7)
    PrintableString 'Redmond'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationName (2 5 4 10)
    PrintableString 'Microsoft'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
    PrintableString 'ITG'
    }
  }
```


Issuer DN – Organization

```
SET {
  SEQUENCE {
    OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
    PrintableString 'WA'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER localityName (2 5 4 7)
    PrintableString 'Redmond'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationName (2 5 4 10)
    PrintableString 'Microsoft'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
    PrintableString 'ITG'
    }
  }
```


Issuer DN – Organizational Unit

```
SET {
  SEQUENCE {
    OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
    PrintableString 'WA'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER localityName (2 5 4 7)
    PrintableString 'Redmond'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationName (2 5 4 10)
    PrintableString 'Microsoft'
    }
  }
SET {
  SEQUENCE {
    OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
    PrintableString 'ITG'
    }
  }
```


Issuer DN – commonName

```
SET {
  SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'Microsoft Intranet FTE User CA 2'
    }
  }
}
SEQUENCE {
  UTCTime '011019235011Z'
  UTCTime '021019235011Z'
  }
SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'Brian LaMacchia'
      }
    }
  }
```


Validity Period

```
SET {
  SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'Microsoft Intranet FTE User CA 2'
    }
  }
}
SEQUENCE {
  UTCTime '011019235011Z'
  UTCTime '021019235011Z'
  }
SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'Brian LaMacchia'
      }
    }
  }
```


Subject DN – commonName

```
SET {
  SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'Microsoft Intranet FTE User CA 2'
    }
  }
}
SEQUENCE {
  UTCTime '011019235011Z'
  UTCTime '021019235011Z'
  }
SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'Brian LaMacchia'
      }
    }
  }
```

Subject Public Key – Algorithm

```
SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
    NULL
    }
  BIT STRING 0 unused bits, encapsulates {
    SEQUENCE {
      INTEGER
        00 C5 8F 7C 84 CD 23 BC FA F7 1C 1C BD 26 EB 8B
        B7 5C A6 0F B7 19 4D 02 FF F5 95 31 6E 4A CE 92
        82 B2 0B E7 90 DC 7D 5A F7 E6 8F BE B1 C5 41 76
        04 4F 7C 5F 29 76 07 71 06 2D A8 6A EB 33 7E 3E
        78 0D 44 27 7F FC 62 A0 52 3F AD 05 CD 72 21 49
        A5 96 7D C5 6B 89 1C 24 43 54 DB 75 A5 A0 BE E0
        20 27 FA F2 2B AA 65 76 B1 61 B6 EB C2 63 53 24
        C0 F9 64 2D BD 16 CD 36 12 5A CE E7 EB 1B 6E FD
                [ Another 1 bytes skipped ]
      INTEGER 65537
      }
    }
  }
```


Subject Public Key – Key Bits


```
SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
    NULL
    }
  BIT STRING 0 unused bits, encapsulates {
    SEQUENCE {
      INTEGER
        00 C5 8F 7C 84 CD 23 BC FA F7 1C 1C BD 26 EB 8B
        B7 5C A6 0F B7 19 4D 02 FF F5 95 31 6E 4A CE 92
        82 B2 0B E7 90 DC 7D 5A F7 E6 8F BE B1 C5 41 76
        04 4F 7C 5F 29 76 07 71 06 2D A8 6A EB 33 7E 3E
        78 0D 44 27 7F FC 62 A0 52 3F AD 05 CD 72 21 49
        A5 96 7D C5 6B 89 1C 24 43 54 DB 75 A5 A0 BE E0
        20 27 FA F2 2B AA 65 76 B1 61 B6 EB C2 63 53 24
        C0 F9 64 2D BD 16 CD 36 12 5A CE E7 EB 1B 6E FD
                [ Another 1 bytes skipped ]
      INTEGER 65537
      }
    }
  }
```


Extensions – Key Usage

```
[3] {
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER keyUsage (2 5 29 15)
      OCTET STRING, encapsulates {
        BIT STRING 7 unused bits
          '1'B (bit 0)
        }
      }
    SEQUENCE {
      OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
      OCTET STRING, encapsulates {
        OCTET STRING
          B6 DF 93 F1 85 8B 7D EF 1D 39 6A C4 C9 A6 30 98
          E0 69 0B A5
        }
      }
    SEQUENCE {
      OBJECT IDENTIFIER '1 3 6 1 4 1 311 20 2'
      OCTET STRING, encapsulates {
        BMPString 'ClientAuth'
        }
      }
```


Extensions – Subject Key ID


```
[3] {
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER keyUsage (2 5 29 15)
      OCTET STRING, encapsulates {
        BIT STRING 7 unused bits
          '1'B (bit 0)
        }
      }
    SEQUENCE {
      OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
      OCTET STRING, encapsulates {
        OCTET STRING
          B6 DF 93 F1 85 8B 7D EF 1D 39 6A C4 C9 A6 30 98
          E0 69 0B A5
        }
      }
    SEQUENCE {
      OBJECT IDENTIFIER '1 3 6 1 4 1 311 20 2'
      OCTET STRING, encapsulates {
        BMPString 'ClientAuth'
        }
      }
```


Extensions – Certificate Template (MS-specific extension)


```
[3] {
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER keyUsage (2 5 29 15)
      OCTET STRING, encapsulates {
        BIT STRING 7 unused bits
          '1'B (bit 0)
        }
      }
    SEQUENCE {
      OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
      OCTET STRING, encapsulates {
        OCTET STRING
          B6 DF 93 F1 85 8B 7D EF 1D 39 6A C4 C9 A6 30 98
          E0 69 0B A5
        }
      }
    SEQUENCE {
      OBJECT IDENTIFIER '1 3 6 1 4 1 311 20 2'
      OCTET STRING, encapsulates {
        BMPString 'ClientAuth'
        }
      }
```


Extensions – Authority Key ID

```
SEQUENCE {
    OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
    OCTET STRING, encapsulates {
      SEQUENCE {
        [0]
          30 3C 1D 78 16 7C AD 8B 89 85 97 D4 E8 3E 7F 85
          E3 41 B8 89
        }
      }
    }
  SEQUENCE {
    OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
    OCTET STRING, encapsulates {
      SEQUENCE {
        SEQUENCE {
          [0] {
            [0] {
              [6]
                'ldap:///CN=Microsoft%20Intranet%20FTE%20User%20C'
                'A%202,CN=reditgcac03,CN=CDP,CN=Public%20Key%20Se'
                'rvices,CN=Services,CN=Configuration,DC=corp,DC=m'
                'icrosoft,DC=com?certificateRevocationList?base?o'
                'bjectclass=cRLDistributionPoint'
              }
            }
          }
```

Extensions – CRL Dist. Points

```
SEQUENCE {
  OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
  OCTET STRING, encapsulates {
    SEQUENCE {
      [0]
        30 3C 1D 78 16 7C AD 8B 89 85 97 D4 E8 3E 7F 85
        E3 41 B8 89
      }
    }
  }
SEQUENCE {
  OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
  OCTET STRING, encapsulates {
    SEQUENCE {
      SEQUENCE {
        [0] {
          [0] {
            [6]
              'ldap:///CN=Microsoft%20Intranet%20FTE%20User%20C'
              'A%202,CN=reditgcac03,CN=CDP,CN=Public%20Key%20Se'
              'rvices,CN=Services,CN=Configuration,DC=corp,DC=m'
              'icrosoft,DC=com?certificateRevocationList?base?o'
              'bjectclass=cRLDistributionPoint'
            }
          }
        }
```


Extensions – more CDPs

```
SEQUENCE {
        [0] {
          [0] {
            [6]
              'http://reditgcac03.redmond.corp.microsoft.com/Ce'
              'rtEnroll/Microsoft%20Intranet%20FTE%20User%20CA%'
              '202.crl'
            }
          }
        }
      }
    }
  }
SEQUENCE {
  OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
  OCTET STRING, encapsulates {
    SEQUENCE {
      SEQUENCE {
        OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
        [6]
          'ldap:///CN=Microsoft%20Intranet%20FTE%20User%20C'
          'A%202,CN=AIA,CN=Public%20Key%20Services,CN=Servi'
          'ces,CN=Configuration,DC=corp,DC=microsoft,DC=com'
          '?cACertificate?base?objectclass=certificationAut'
          'hority'
        }
```


Extensions – Authority Info Access

```
SEQUENCE {
        [0] {
          [0] {
            [6]
              'http://reditgcac03.redmond.corp.microsoft.com/Ce'
              'rtEnroll/Microsoft%20Intranet%20FTE%20User%20CA%'
              '202.crl'
            }
          }
        }
      }
    }
  }
SEQUENCE {
  OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
  OCTET STRING, encapsulates {
    SEQUENCE {
      SEQUENCE {
        OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
        [6]
          'ldap:///CN=Microsoft%20Intranet%20FTE%20User%20C'
          'A%202,CN=AIA,CN=Public%20Key%20Services,CN=Servi'
          'ces,CN=Configuration,DC=corp,DC=microsoft,DC=com'
          '?cACertificate?base?objectclass=certificationAut'
          'hority'
        }
```


Extensions – more AIA

```
SEQUENCE {
        OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
        [6]
          'http://reditgcac03.redmond.corp.microsoft.com/Ce'
          'rtEnroll/reditgcac03.redmond.corp.microsoft.com_'
          'Microsoft%20Intranet%20FTE%20User%20CA%202(3).cr'
          't'
        }
      }
    }
  }
SEQUENCE {
  OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
  OCTET STRING, encapsulates {
    SEQUENCE {
      OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
      }
    }
  }
```


Extensions – Extended Key Usage

```
SEQUENCE {
        OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
        [6]
          'http://reditgcac03.redmond.corp.microsoft.com/Ce'
          'rtEnroll/reditgcac03.redmond.corp.microsoft.com_'
          'Microsoft%20Intranet%20FTE%20User%20CA%202(3).cr'
          't'
        }
      }
    }
  }
SEQUENCE {
  OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
  OCTET STRING, encapsulates {
    SEQUENCE {
      OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
      }
    }
  }
```


Extensions – Subject Alt. Name

```
SEQUENCE {
  OBJECT IDENTIFIER subjectAltName (2 5 29 17)
  OCTET STRING, encapsulates {
    SEQUENCE {
      [0] {
        OBJECT IDENTIFIER '1 3 6 1 4 1 311 20 2 3'
        [0] {
          UTF8String
            'bal@redmond.corp.microsoft.com..edmond,DC=corp,D'
            'C=microsoft,DC=com'
          }
        }
      }
    }
  }
  }
  }
  }
```

Signature Algorithm

```
SEQUENCE {
  OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
  }
BIT STRING 0 unused bits
  78 AE FB D9 B9 3C 0A 5C 99 DE 86 BB BE DB 72 2E
  E7 E4 AC 8D 8B F8 53 5E EC B1 73 43 2F 21 89 CC
  59 DD 4E C1 77 C1 F5 9C 08 35 68 C7 51 B7 05 93
  5A 26 E5 6E D8 F9 C3 2C C3 A4 D0 7F FB 52 57 B1
  6D 1C FC 3C 4D 1F F6 CF 0C 57 00 8B 20 DA 43 13
  35 A2 5F C4 EC 0E 72 98 97 06 70 5D 34 F0 43 0B
  B7 62 CD EC C4 F4 33 81 FB 0C 9B C0 68 EC FF FA
  B3 31 D6 07 C9 93 F2 BA 68 92 5A 4E 3E B0 2F 14
        [ Another 128 bytes skipped ]
  }
```


Signature Bits


```
SEQUENCE {
  OBJECT IDENTIFIER sha1withRSAEncryption (1 2 840 113549 1 1 5)
  NULL
  }
BIT STRING 0 unused bits
  78 AE FB D9 B9 3C 0A 5C 99 DE 86 BB BE DB 72 2E
  E7 E4 AC 8D 8B F8 53 5E EC B1 73 43 2F 21 89 CC
  59 DD 4E C1 77 C1 F5 9C 08 35 68 C7 51 B7 05 93
  5A 26 E5 6E D8 F9 C3 2C C3 A4 D0 7F FB 52 57 B1
  6D 1C FC 3C 4D 1F F6 CF 0C 57 00 8B 20 DA 43 13
  35 A2 5F C4 EC 0E 72 98 97 06 70 5D 34 F0 43 0B
  B7 62 CD EC C4 F4 33 81 FB 0C 9B C0 68 EC FF FA
  B3 31 D6 07 C9 93 F2 BA 68 92 5A 4E 3E B0 2F 14
        [ Another 128 bytes skipped ]
  }
```


Whew!

- There’s a lot of stuff packed into that cert
  - 1.6KB of data…
  - … and there could have been more extensions

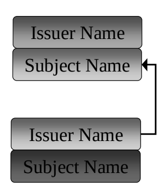
Back to chain building

- OK, assume we’re a “relying party application” -- something that received an end-entity certificate and wants to verify it.
  - Our task is to build a cert chain from that end-entity cert to one of our trusted roots
- How do we do that?
  - We start with our EE cert, and using the information contained within we look for possible parent certificates.

Parent certs

- What’s a valid parent certificate?
  - In the raw X.509 model, parent-child relationships are determined solely by matching Issuer DN in the child to Subject DN in the parent
  - Recall that there’s an assumption that you have a big directory handy to find certs.
- If you don’t have a directory handy, you need to do the matching yourself
  - This is not as easy as you might think…

Name matching



Name

Matching Names

- How do we determine if two DNs match?
  - "Use directory name matching rules!"
  - Try to be mildly smart about it
    - Remove spaces, case-fold, etc.
    - Disaster…
  - Try to be really dumb about it
    - Exact binary match
    - Less of a disaster, but there are still problems we can't work around…

Unicode Names

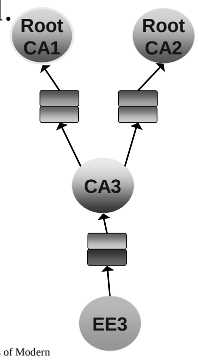
- Are these two character equal?

é é

- They look equal…
- …but may not be
- In Unicode, you can compose characters, so:
  - "é" as one character
  - "é" as two characters – "e" followed by non-spacing accent
  - "é" as two characters – non-spacing accent followed by "e"
- Ick!

So we're screwed.

- Yup, and it gets worse…
- Imagine you have a CA3 that is certified by CA1 and now wants to also be certified by CA2
- Under name chaining, CA1 & CA2 must call CA3 by the same name!

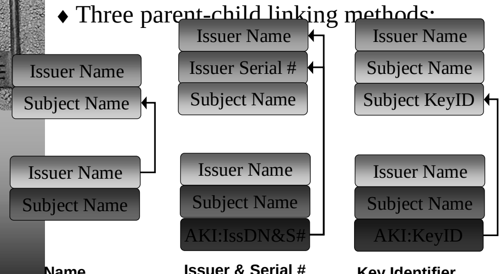


First issuer wins naming rights

- Once you're "named" by someone (e.g. the government at birth), everyone has to call you by the same name if name-chaining is to succeed
- What's the solution?
  - Use another chaining method
- Use key-based chaining
  - It's the keys that matter, since that's what signs & makes the statement.

Chain Building Details (4)

- Three parent-child linking methods:



Three ways to chain

- Name matching
- "Exact matching"
  - Exactly one parent, rigidly defined
  - When parent cert expires, link always dies, need to reissue
- "KeyID matching"
  - Match off keys, not names

We need something else…

- "X.509 is just nuts…there must be a simpler way to do this"
- Something that puts the user in charge, not a Certificate Authority
- Something with free-form names
- Something that's key-based…

Pretty Good Privacy (PGP)

- We need "Phil's Pretty Good Privacy"
  - Created in 1991 by Phil Zimmermann
  - One of the focal points of crypto politics
    - Patent politics ("guerrilla-ware") w/ RSA
    - Crypto export politics with the US Gov't.

PGP Certificates

- PGP certificates are key-based, not name-based
  - Keys can have one or more free-form names attached to them
  - Keys and name(s) are bound together by one or more signatures from other keys
- PGP certs are unidirectional links between keys
  - "I sign a key/name binding you have. Maybe you sign one of mine."

PGP Cert Models

- Certification model can be hierarchical, "mesh", based on an existing trust relationship, etc.
  - Whatever you want
- "Web of Trust"
  - As the holder of a private key, you get to decide
    - What keys you explicitly trust (by signing them)
    - Whether those keys are allowed to introduce other keys to you.

More on PGP Certs

- PGP was designed for secure e-mail
  - Especially secure e-mail among a close circle of friends.
- Originally, keys were always exchanged directly with correspondents
  - But users really wanted to be able to look up someone's public key in a directory & send them mail (perhaps unsolicited)
  - Need a directory, but no common CA or set of CAs to run it

PGP Keyservers

- Answer: The PGP keyserver network
  - A distributed network of key/cert databases that can be queried by anyone.
- Started as a mail-based server network
  - Mail in your key, it gets added to the server's keyring.
  - Mail in a request for a key, you get sent back a subring containing every key that matched your query.
  - Servers updated each other by passing around "add key" requests.

Web-based keyservers

1. Slap a web-based front-end on top of the keyserver to allow real-time lookups.
2. Watch web server performance slow to a crawl because PGP’s key management routines were O(n^2)!
3. Re-implement the keyserver on top of a real database engine
4. Standardize the protocol to allow automated queries from other programs

The network is still going…

- http://wwwkeys.us.pgp.net
  - Random US-based keyserver
- http://wwwkeys.pgp.net
  - Random world-wide keyserver
- (DNS can do tricks like this, yes!)