How to build a PKI that works

Peter Gutmann

University of Auckland

How to build an X.509 PKI that works

Peter Gutmann

University of Auckland

Preliminaries

Whose PKI are we talking about here?

- Not SSL certs
  - Certificate manufacturing, not PKI

    It's just an expensive way of doing authenticated DNS lookups with a TTL of one year. Plenty of PK, precious little I — Peter Gutmann on the crypto list
- Not PGP, SPKI, *ML, etc
  - Doing fairly well in their (low-I) area
- Not government PKI initiatives
  - Government IT project reality distortion field, keep pumping in money until it cries Uncle
  - Even then, the reality distortion has failed in parts of Europe, Australia

Preliminaries (ctd)

This is PKI for the rest of us

- Businesses, individuals, etc

Talk covers exclusively technical issues

- Policies are someone else's problem

  Ted says that whenever he gets asked a religious question he doesn't understand he always responds with "Ah, that must be an ecumenical matter" which universally produces nods of admiration at the profound wisdom of the statement. It seems that that the PKIX list equivalent is "Ah, that must be a policy matter"

  — Father Ted (via Anon)
- Some religion may sneak in

Preliminaries (ctd)

Microsoft bashing: An apology in advance

- Their PKI software is the most widespread, and features prominently in examples because of this
- There is no indication that other software is any better, it just gets less publicity

It may be a little controversial…

56th IETF agenda item, submitted as a joke when someone pointed out that PKIX didn't have any agenda

What needs to be done to make PKI work?

This forum will be open to all PKIX members, and will constitute a large pool filled knee-deep with custard. Marquis of Queensberry Rules, but with pies substituted for gloves. Participants are expected to provide appropriate clothing. Remaining IETF members will look on in amusement or dismay, depending on their views on PKI

Meeting minutes at
`http://www.cs.auckland.ac.nz/~pgut001/misc/minutes.txt`

Why do we need “a PKI that works”?

PKI is in trouble

PKI is ‘Not Working’ (Government Computing, UK)

“Trust and authentication has been a huge problem for us. We haven’t got a solution for authentication. We’ve been trying with PKI for about 10 years now and its not working because it’s a pain to implement and to use”.

Billion Dollar Boondoggle (InfoSecurity Mag, US)

A recent General Accounting Office report says the federal government’s $1 billion PKI investment isn’t paying off. […] The GAO says widespread adoption is hindered by ill-defined or nonexistent technical standards and poor interoperability […] Despite stagnant participation, federal officials are continuing to promote the [PKI].

PKI is in trouble (ctd)

Gatekeeper goes Missing (The Australian)

> Five years after then finance minister John Fahey launched Gatekeeper to drive public and business confidence in e-commerce, government department and agency interest in PKI is almost zero.
>
> A spokesperson for the Attorney-General's Department said: "I am very grateful for the fact that none of my colleagues has come up with a good use for it. When they do, I will have to do something about it".

End of the line for Ireland's dotcom Star (Reuters)

> The company would have done better to concentrate on making its core PKI technology easier to deploy, a shortcoming that became a key reason Baltimore's UniCERT PKI technology never went mainstream.

PKI is in trouble (ctd)

International and New Zealand PKI experiences across government (NZ State Services Commission)

> Based upon overseas [Australia, Finland, Germany, Hong Kong, US] and New Zealand experiences, it is obvious that a PKI implementation project must be approached with caution. Implementers should ensure their risk analysis truly shows PKI is the most appropriate security mechanism and wherever possible consider alternative methods.

PKI's Image Problem

The message to potential users from mainstream media coverage: PKI doesn't work

> …as computer security professionals, we feel that it is our duty to advise the legislature of the critical importance of requiring the use of a PKI for this system, preferably with multiple root CAs and online certificate revocation.
>
> — Cryptographer John Kelsey proposing a means of killing a DRM initiative by the Copyright Policy Branch of Canadian Heritage

Why is PKI in trouble?

The usual suspects...

- Difficult to deploy
- Expensive
- Hard to use
- Lack of interoperability
- Poor match to pressing real-world problems
- Etc etc etc

The PKI Grand Challenge

Get the basic infrastructure in place before we worry about chrome tailfins, fuzzy dice, certificate warranty permanent qualifier policy logotype extensions, …

- I can add theme music to my certificate if I want, but the only way to publish it is to stick it on my home page
- There'll be plenty of time to add the fuzzy dice once the basic infrastructure is in place

  I think a lot of purists would rather have PKI be useless to anyone in any practical terms than to have it made simple enough to use, but potentially "flawed"

  — Chris Zimman

I still can't use PKI to authenticate myself for the PKI Workshop…

PKI Grand Challenges

Challenge #1: Key lookup

- Original PKI was Diffie and Hellman's "Public File" in 1976
- In 1976, I couldn't look up your public key online
- After thirty years' work, I still can't look up your public key online

Challenge #2: Enrolment

- A torture test for users to see how badly they really want a cert
- Pain of enrolment leads to terrible key hygiene

Challenge #3: Validity checking

- Real-time check to match expectations of online banking, share-trading, bill payment, etc etc

PKI Grand Challenges (ctd)

Challenge #4: User identification

- X.500 DNs (enough said)
- Mostly solved in a de facto manner

Challenge #5: No quality control

- You cannot build a product so broken that it can’t claim to be X.509
- Users *notice* that things don’t work → PKI image problem (see challenge #6)

PKI Grand Challenges (ctd)

Challenge #6: Implementor / user apathy (HCI)

- Complexity / lack of understanding ↔ lack of motivation to do things right
  - Example: Re-checking certificate against an old CRL on disk meets requirements for a revocation check
- Current designs make it too easy to just go through the motions

Well, that's a nice theory, but…

It's practice, not theory

- Based on extensive user feedback / usability testing
- Refined over many years
- Designed to maximise ease of use, correct functionality
  - You have to really work hard to get it wrong
- Designed to minimise implementer pain

This is not just a gedanken experiment / unproven hypothesis

Challenge #1

Key Lookup

Pre-history of Key Lookup (and Certs)

Original 1976 paper on public-key encryption proposed the Public File

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

Not very practical in 1976

- Key lookup over X.25?
  - Having to interrupt a circuit-switched connection to do a Public File lookup was the original motivation for offline certificates (1978)
- A very sensible, straightforward approach now that there's a WWW

The Key Lookup Problem

The problem

- Get me `joe@foo.com`'s key(s)
- Get me `foo.com`'s key(s)

Clayton's solutions: S/MIME, SSL

- Send out all your certificates with each message
- Lazy-update distributed key management

The Web as the Public File

We have a Public File

- It’s called the WWW

  We have a system, it is called the Web, everyone else lost, get over it

  — Phillip Hallam-Baker

Quick-n-dirty solution: Google

- Stick a base64-encoded certificate on your home page
- Add a standardised string for search engines,
  `certificate joe@foo.com`
- Google, cut & paste
- Clunky, but simple and effective
  - Better than anything else we have today

The Web as the Public File (ctd)

Proper solution: Use HTTP to fetch keys

- GET *uri*?*attrib*=*value*
  `GET /search-cgi?email=joe@foo.com`

ID types required

- S/MIME, SSL/TLS, IPsec, PGP, SIP, etc
  - Email, domain name, URI
- Cert chaining
  - Issuer DN, keyID
- S/MIME
  - issuerAndSerialNumber
- PGP
  - PGP keyID

Implementation

HTTP glue + anything you want

- Berkeley DB
  - Lightweight { key, value } lookup
- RDBMS
  - ODBC is built into every copy of Windows
    - ODBC glue for most Unix systems
  - MySQL or Postgres is built into most copies of Linux
  - JDBC for Java
  - Ties into existing corporate databases (SQL Server, Oracle)
- ISAM
- Flat files
  - c.f. PGP's HKP servers
- X.500 / LDAP if you insist

Implementation (ctd)

Implementation effort

- MySQL (server): 30 minutes
  - Every database on the planet is already web-enabled
  - This is what many web servers do all day long
- Java (server): A few hours
- Visual Basic (client): About 5 minutes

Lightweight client

- ~100 lines of code on top of TCP/IP stack in an embedded network device

Other Features

Pre-construct URLs for certificates

- Print on business cards
- Help-desk can mail to users who can’t find their certificates
- Enforce privacy by perturbing the search key
  `x-encryptedSearchKey`=…
- Enforce access controls by authenticating the search key
  `x-macSearchKey`=…

Other Features (ctd)

Standard techniques used to manage high loads

- It’s a standard web server with static pages
  - Web101
- If Amazon / CNN.com can handle this…

More details / rationale in “Certificate Store Access via HTTP”

But what about X.500 / LDAP?

If you can’t be a good example then at least you can be a horrible warning

But what about X.500 / LDAP?

So far, LDAP has not done a great job of supporting PKI requirements.

— Steve Kent, PKIX WG chair

The X.500 linkage […] has led to more failed PKI deployments in my experience than any other. For PKI deployment to succeed you have to take X.500 and LDAP deployment out of the critical path.

— Phillip Hallam-Baker, Verisign principal scientist

- If you really want to, you can always use X.500 / LDAP as another backend for the HTTP certstore — it’s not picky

The most effective way I’ve found to search an X.500 directory to locate a certificate is by Internet email address

— PKI developer

Challenge #2

Enrolment

What it should be like: The DHCP Model

User wants to use TCP/IP / email / WWW

- DHCP client automatically discovers the server
- Client requests all necessary information from the server
- Auto-configures itself using returned information
- User is online without even knowing that the DHCP exchange happened

What it is like: The X.25 Model

User is required to use X.25

- Dozens of parameters to manually configure
- Different vendors use different terms for the same thing
- Get one parameter wrong and nothing works
- Problem diagnosis: Find an X.25 expert and ask for help

The vast majority of users detest anything they must configure and tweak. Any really mass-appeal tool must allow an essentially transparent functionality as default behaviour; anything else will necessarily have limited adoption.

— Bo Leuf, *Peer to Peer*

How bad is it really?

Obtaining a certificate from a large public CA

- User had to ask where to get the certificate
- Filled out eight (!!) browser pages of information
- Several retries due to values being rejected, had to ask for help several times, searched for documentation such as a passport, etc etc
- Cut & pasted data from emailed message to web page
  - Multiple random strings had to be manually copied over
  - Emailed cookies: Only one should be necessary

How bad is it really? (ctd)

- Filled out more fields in eleven further web pages
  - Much of the contents were incomprehensible to the user: “certificate Distinguished Name”, “X.509 SubjectAltName”

    My grandmother just won’t understand the meaning of “initial-policy-mapping-inhibit” no matter how much she loves me.

    — David Cross on ietf-pkix
  - User guessed and clicked “Next”
- Web page announced that a certificate had been issued, but none seemed available

How bad is it really? (ctd)

- Emailed message provided a link to click on
- More web pages to fill out
- Switch to another browser to download file
- Clicking on the file had no effect

At this point the user gave up

How bad is it really? (ctd)

Time taken: > 1 hour (with outside assistance)

- Usenet posts/email suggest that most skilled technical users take between 30 minutes and 4 hours to get a certificate

  "There's a myth […] that the issuance of a public certificate is a remarkably heavyweight operation. You know, you must need steam-powered equipment in the basement of your facility in order to stamp out those certificates, which have to be made out of titanium or what have you"

  — Matt Blaze, Security Protocols Workshop

  The Machine that Issues Certificates,
  `http://www.cs.auckland.ac.nz/~pgut001/misc/certificates.txt`

Consequences of enrolment difficulties

Pain of enrolment encourages poor key hygiene

- Company spends $495 and several hours' work creating a key and getting a Verisign certificate for it
- Most practical (in terms of time and money) application of this is to re-use it everywhere
  - "It cost us $*xxx*/*yyy* hours' effort to get this key, we're not going through all that again"

Much of the problem is social/financial

- Certificates are expensive to obtain
- Certificates are troublesome to obtain
- Users are given a considerable incentive to re-use certs/keys

Consequences of enrolment difficulties (ctd)

CAs generate private keys for users and mail them out as PKCS #12 files

- Password is sent as separate mail or is easily guessed (8 characters, uppercase-only)
- This is standard practice for many, many CAs

I didn’t generate PKCS #10. My CA does not support this request [...] CA sends me two files – private key and certificate.

the certificates and the key pairs are centrally generated and send to the user as PKCS#12 files. The user imports this file in his Internet Explorer and can use it for SSL client authentification. This works successfully.

continues…

Consequences of enrolment difficulties (ctd)

CA generates only PKCS12 key files [...] I can not find an exact explanation how to read a PKCS12 private key form such a file.

Plus, they attach your certificate AND _private key_ to the bottom of the message. The idea is that you copy and paste the cert + private key into a file for the client API to use when it connects. Basically, they are sending all of the information [...] through plain, unencrypted, email.

I have two files from CA – private key and certificate.

what is the format to use for sending me a private key sertificate when the CA does the whole process themselves - and want to send me a pin code and a PKCS#12 cert

continues…

Consequences of enrolment difficulties (ctd)

> The CA generates an encryption key pair for the client and issues a certificate for the public key. The CA sends the private key.
>
> import pkcs#12 files (including private key) onto the smartcard [...] Sometimes they let you even generate keypair(s) on the card and have the public part certified by the CA's, which is not always a good idea…
>
> — Representative sampling from newsgroups and mailing lists

- One development group took to referring to the private key as "the lesser-known public key"

Consequences of enrolment difficulties (ctd)

CAs distribute their own private keys as PKCS #12 files

- The theory is that once installed, it makes the CA key trusted
- This "solution" is so common that it's warned about in the OpenSSL FAQ
- At least one computer security book contains step-by-step instructions on how to distribute your CA's private key to all users

Application developers send PKI software developers their private keys during debugging

- Verisign Authenticode code-signing keys, banking keys, etc etc

Consequences of enrolment difficulties (ctd)

Smart cards store private keys internally and don’t reveal them

- “How can I use a smart card if I can’t get at the key?”

  what is the point in jailing the private key for life in a single smart card? This argument is totally contrary to logical thinking.

  — Anon on ietf-pkix

Consequences of enrolment difficulties (ctd)

- Attempted fixes are to…
  - Construct mechanisms for sharing cards across multiple machines
  - Generate the key externally and keep a copy after it’s loaded onto the card
    - Exacerbated by the mail-a-PKCS12 approach to certification
- Maybe the inconvenient fact that they keep private keys private is why crypto smart cards aren’t taking off

What should enrolment be like?

The mom test: Could your mother use this?

The ISP model

- Call ISP with credit card
- ISP provides username and password
- Enter username and password, click OK
- DHCP does the rest

PKI enrolment should be similar

- Others have debugged the process for us
- Users have been conditioned to do this
- Most users can handle this

Assumptions

Basic networking services are present

- The user has a net connection, IP address, etc etc (DHCP at work)

Assumptions (ctd)

The user has some existing relationship with the certificate-issuing authority

- Issuing identity certificates to strangers doesn't make much sense
- Online banking / tax filing / loyalty program sign-up is usually handled by
  - In-person communications
  - (Snail) mailed authenticator
  - Phone authorisation
- Follows existing practice
  - People are used to it
  - Established legal precedent

Assumptions (ctd)

We're not designing a system to handle nuclear weapons launch codes

- The system need only be as secure as the equivalent non-PKI alternative
  - Techies tend to go overboard when designing authentication systems
- Operations where a cert might be used (online banking, shopping, tax filing) all get by with a username and password
- If it's good enough when used without certificates, it's equally good with them

Cumbersome technology will be deployed and operated incorrectly and insecurely, or perhaps not at all

— Ravi Sandhu, *IEEE Internet Computing*

PKI Service Location

DHCP

- Limited to local subnet
- Would require modifying all existing DHCP servers
- Unnecessarily low-level: Higher-level network infrastructure is already in place

DNS SRV

- Easily added to existing servers
  - Single line in a config file
- Not supported in Win'95/98/ME
- Those who need it most don't have it
  - Expecting Auntie Ethel to install `bind` is probably a bit much

PKI Service Location (ctd)

SLP

- Service Location Protocol, specialised service-location mechanism
- Rarely used, requires configuring and maintaining yet another server/service

UPnP

- Very complex
- Requires XML (SOAP), HTML GUI interface, etc etc
- Many sites block UPnP for the same reason that they block NetBIOS

PKI Service Location (ctd)

Jini

- Very complex
- Tied to Java-specific mechanisms (RMI, code downloading, etc etc)

Others: Salutation, Rendezvous, …

- See SLP

PKI Service Location (ctd)

Faking it

- Use of "well-known" locations for services
- Full IP service (e.g. PC): Use "pkiboot" at start of domain name
  - `foo.domain.com` → `pkiboot.domain.com`
  - Example: Corporate/organisational CA certifying users
- Partial IP service (e.g. web-enabled embedded device): Append "pkiboot" to device's IP address or location:
  - 192.0.0.1 → `http://192.0.0.1/pkiboot/`
  - Example: Print server certifying printers
- Use HTTP redirects if necessary
- Somewhat clunky, but can be done automatically/transparently

PKIBoot: Obtaining Initial Certificates

Establishing the initial trusted certificate set (PKI TCB)

- Browsers contain over 100 hardcoded certificates
  - Unknown CAs
  - Moribund web sites
  - 512-bit keys
  - No-liability certificates
  - Keys on-sold to third parties
- Any one of these CAs can usurp any other CA
  - Implicit universal cross-certification
  - Certificate from "Verisign Class 1 Public Primary Certification Authority" could be issued by "Honest Al's Used Cars and Certificates"
  - Browser trusts Verisign and Honest Al equally

PKIBoot: Obtaining Initial Certificates (ctd)

Why do browsers do this?

- Prime directive: Don't expose the users to scary warning dialogs
- One-size-fits-all browser can't know in advance which entities the user has a trust relationship with
  - Need to include as many certificates as possible to minimise the chances of users getting scary warning dialogs
  - The ideal user-friendly situation would be to automatically trust all certificates

Goal: User should only have to trust certificates of relevance to them (minimised TCB)

PKIBoot: Obtaining Initial Certificates

Initial state: No certificates

Use username + password to authenticate download of known-good/trusted certs (PKIBoot)

- Messages are protected using a cryptographic message authentication code (MAC) derived from the password
- User → PKI service: Send known-good certificates
  - User request is authenticated with MAC
- PKI service → user: Known-good certificates
  - PKI service response is authenticated with MAC
- Since only the legitimate service can generate the MAC, certificate spoofing isn't possible

Obtaining User Certificates

Initial state: CA certificates

Use MAC to authenticate the request for a signing certificate

- User → PKI service: Sign this for me
  - User request is authenticated with a MAC
- PKI service → user: Signed certificate
  - PKI service response is authenticated with a signature from the PKIBoot cert

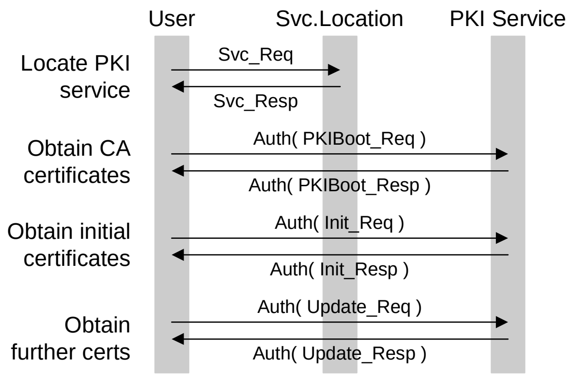
Obtaining User Certificates (ctd)

Initial state: CA certificates, signing certificate

Use signing certificate to authenticate the request for an encryption certificate

- User → PKI service: Sign this for me
  - User request is authenticated with the signing cert
- PKI service → user: Signed certificate
  - PKI service response authenticated with a signature from the PKIBoot cert

Sequence of Operations



Multi-phase bootstrap

- MAC → CA cert, signing cert request
- CA cert → response
- Signing cert → encryption cert

PnP PKI in action

User

- Enters username + password (identifier + authenticator)
  - No need to even mention certificates

Software developer

- Creates PnP PKI session
- Adds file/smart card for key storage
  - Card can be pre-personalised with enrolment information
- Adds username + password

PnP PKI in action (ctd)

PnP PKI session

- Performs PKIBoot using username + password
- Generates signing key
- Requests signing certificate using username + password
- Generates encryption key
- Requests encryption certificate using signing certificate
- Updates file/smart card with signing, encryption keys and user and CA certificates

User/Software developer

- Has keys and certificates ready for use

HCI Aspects of PnP PKI

Minimalist enrolment (with pre-personalised smart card)

- Insert card
- Enter PIN to unlock/access card
- Wait a few seconds
- Done

Enforces best practices by default

- Minimal set of trusted certificates (TCB)
- Locally-generated private keys
  - Keys can be generated inside crypto hardware
- Distinct encryption and signing keys

Details / rationale in "Plug-and-play PKI: A PKI your mother can use"

Challenge #3

Validity Checking

Current Approaches

Ignore it entirely

Go through the motions

- Repeatedly re-check a day / week-old CRL

OCSP

- If fed a freshly-issue cert, can't say "It's valid"
- If fed an Excel spreadsheet (or a forged cert), can't say "It's not valid"
- No scalability
  - Vendors eliminate replay-attack protection in order to get usable performance

    The changes we are making to scale our OCSP responder will result in the discontinuation of the nonce extension
    — Verisign

What's Needed

The web has conditioned users to expect live, real-time status updates

- ebay bidding
- Amazon.com et al
- Stock trading
- Online bill payment
- Travel booking
- Paypal

Certificate validation checking should be no less functional than these systems

What’s Needed (ctd)

The target: Yes/no response in as close to real-time as possible

Learning in 80 ms that the cert was good as of a week ago and to not hope for fresher information for another week seems of limited, if any, utility to us or our customers.

— PKI architect

Implementation

Query: hash(cert)

- Cert fingerprint / thumbprint recognised by any PKI software

Response: CMS(yes | no)

- Signed response (slow)
- MAC’d response (fast)
- Plain response (over secure link, very fast)

Totally unambiguous response, in real time

- It’s valid right now
- It’s not valid right now
  - Can be embellished with reasons, dates, etc etc

Performance

A single PC can saturate a 100Mbps link

- Connectionless (UDP) queries
  - Both queries and responses are tiny
- O(1) hash table / CAM lookup
  - Query is pre-hashed by the client
- memcpy result data
  - ASN.1, but fixed format, requires no en/decoding
- Drop MAC or sig. into fixed location

You cannot build a faster validity checking mechanism

Performance (ctd)

Performance options

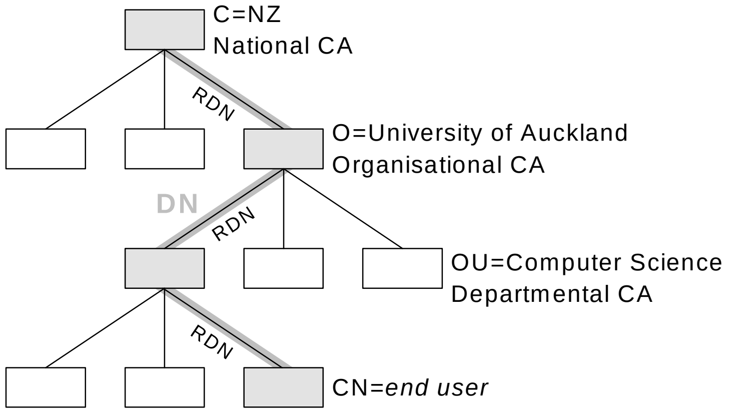
- Software-only, MAC'd response
  - Can saturate 100Mbps link
  - CMS can bootstrap MAC keys via PKC exchange
  - Key exchange can be initiated by the server to reduce load
- Broadcom 582x, scatter/gather operation
  - 4K signed responses/sec (10Mbps)
- Cavium Nitrox, all ops done on-chip
  - 40K signed responses/sec, (100Mbps)

Challenge #4

User Identification

The X.500 DN

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



X.500 Naming

Typical DN components

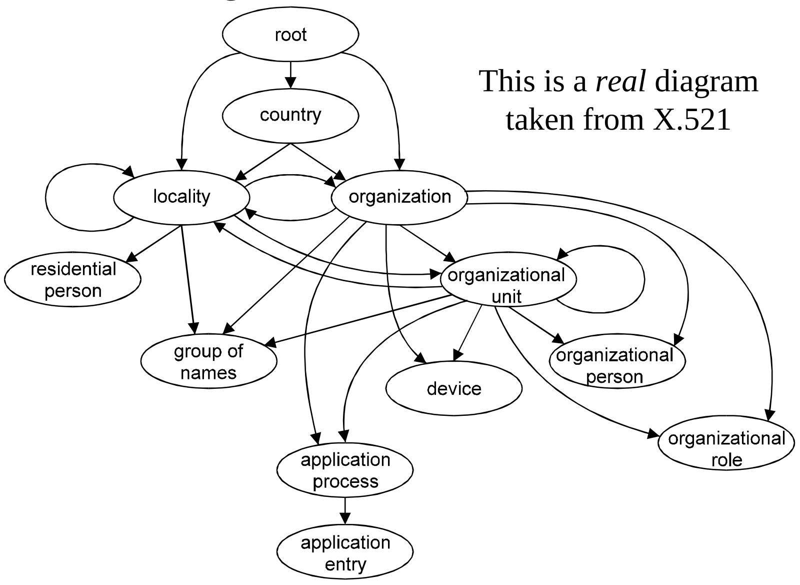
- Country C
- State or province SP
- Locality L
- Organisation O
- Organisational unit OU
- Common name CN

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

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

Problems with X.500 Names

No-one ever figured out how to make DNs work



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

Problems with X.500 Names (ctd)

No clear plan on how to organise the hierarchy

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

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

Problems with X.500 Names (ctd)

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

Sample problem cases

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

Problems with X.500 Names (ctd)

For a corporation, is C, SP, L

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

For a person, is C, SP, L

- Place of birth?
- Place of residence/domicile?
  - Dual citizenship
  - US military personnel can choose “resident” state for tax purposes
  - Stateless persons
  - Nomads
- Place of work?

DNs in Practice

Public CAs typically set

C = CA country or something creative (“Internet”)

O = CA name

OU = Certificate type / class / legal disclaimer

CN = User name or URI

email = User email address

- Some European CAs add oddball components required by local signature laws
  - Italy adds IDs like BNFGRB46R69A944C

DNs in Practice (ctd)

- Some CAs modify the DN with a nonce to try and guarantee uniqueness
  - – Armed services CA adds last 4 digits of SSN
  - – Another CA encodes random CA/RA-specific data

    The disambiguating factor will be variable length alphanumeric […] for example: XYZ221234 […] or, for example ABC00087654321.

    — GTE Government Systems Federal PKI pilot

Some DNs are deliberately mangled

> For educational institutions here in the US, FERPA regulations apply. The way we do this here at Wisconsin is to only include a bunch of random gibberish in the DN as an identifier.
>
> — Eric Norman on ietf-pkix

DNs in Practice (ctd)

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

- Some software requires that all of { C, O, OU, SP, L, CN } be set
  - – "Invent random values to fill these boxes in order to continue"
- Resulting certificates contain strange or meaningless entries as people try and guess values, or use dummy values
  - – "… a bunch of random gibberish in the DN…"

DNs in Practice (ctd)

The goal of a cert is to identify the holder of the corresponding private key, in a fashion meaningful to relying parties.

— Steve Kent

- Minimalist DNs
  - “Fred’s Certificate”
  - “My key”
  - “202.125.47.110”

DN Encodings

Encoding of DNs is more or less random

- Arbitrary grouping of AVAs, ordering and number of RDNs, etc etc

DNs may be encoded backwards

- A side-effect of the RFC 1779 string representation
- Java-created certs often have backwards DNs because of this
- Some .NET DN orders are forwards, some backwards
  - GetIssuerName / GetSerialNumber vs. MMC snap-in
- One European national CA encodes DNs backwards and forwards at random
  - Other CAs are more consistent in getting DNs backwards

DN Encodings (ctd)

Applications enforce arbitrary limits on data elements (GCHQ/CESG interop testing)

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

The real DN encoding / name comp.rules

There is no name comparison rule but binary compare, and `memcmp()` is its implementation

- Originator encodes the DN any way they want
- Further “re-encoding” is done via `memcpy`
- Comparisons are done via `memcmp`

While technically there’s this DN compare algorithm in RFC2459 or the evil X.500 version anyone with any sense ignores it completely and treats DNs as equal only if they have the same encoding.

— PKI developer

We treat DNs as opaque blocks of binary data […] we yank the exact binary blob out of the certificate and combine that with the exact binary blob of the serial number.

— S/MIME developer

The real DN encoding / name comp.rules (ctd

These are the only rules that always work

- No matter how garbled the DN, they'll handle it
- Performing a bit-for-bit copy ensures that other apps get to see exactly what they need to see

  We are testing signing and encryption in S/MIME software […] It seems that all the software we have tested (eg. MSoft, Utimaco) tend to do somekind of binary comparison on the certificate.

  — Saku Vainikainen on ietf-pkix

The real DN encoding / name comp.rules (ctd

Application developers learn these rules fairly quickly

- Client submits cert request with PrintableString
- CA returns cert with UTF-8 String
- Client app rejects the cert because the DN doesn't match

  "Don't user Master Documents in MS Word"
  "Don't change the monitor frequency settings in XF86Config"
  "Don't rewrite DNs in certificates"

  — Peter Gutmann on ietf-pkix

Challenge #5

Quality Control

Quality Control: The absence thereof

You can’t build an app so broken that it can’t claim to be X.509

- Any old rubbish can claim to be X.509, and frequently does

The X.509 brand has been diluted to the point of worthlessness

- (Deeply-buried) PGP has been sold as X.509
- “The other side is using a different version of X.509” explained away interop problems

QC Examples: The Trivial

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 implementers forgot about the sign bit (Microsoft and a few others)

- Everyone changed their code to be bug-compatible with MS

QC Examples: 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 {}
         :             }
```

QC Examples: 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

QC Examples: 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

QC Examples: 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

QC Examples: 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

QC Examples: 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

QC Examples: The Scary (ctd)

Software ignores the key usage flags and uses the first cert it finds for the purpose it 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

QC Examples: The Scary (ctd)

- CryptoAPI ignores usage constraints on keys for user convenience purposes
  - AT_KEYXECHANGE 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

QC Examples: 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

QC Examples: 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

QC Examples: The Scary (ctd)

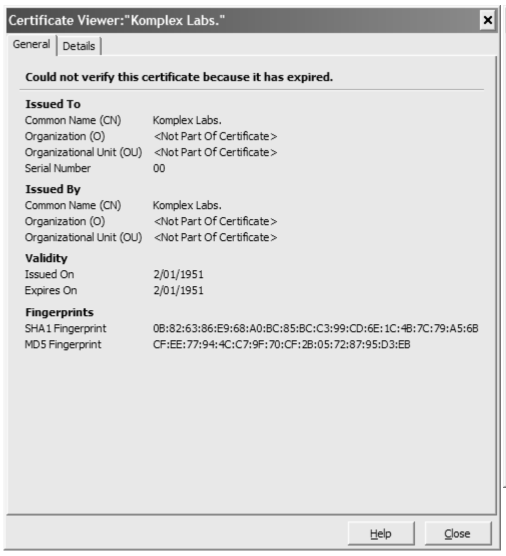
> Hi. My name is Peter and I have a keyUsage problem. Initially it was just small things, a little digitalSignature after lunch, maybe a dataEncipherment after dinner and keyAgreement as a nightcap. Then I started combining digitalSignature and keyEncipherment in the same certificate. It just got worse 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

QC Examples: 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//////+
```

QC Examples: 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

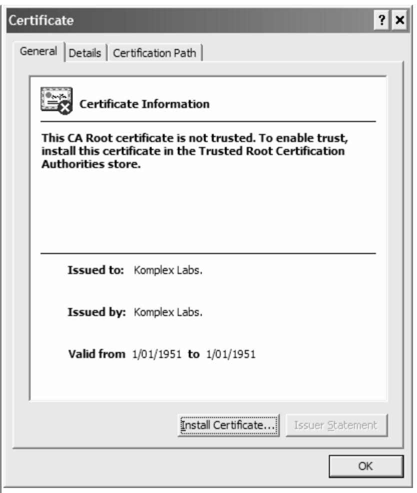
Common Name (CN)	Komplex Labs.
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

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

QC Examples: 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

QC Examples: 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

QC Examples: The Scary (ctd)

CAs issue certificates with e = 1

- Problem was only noticed when Mozilla was modified to detect 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

QC Examples: 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…

QC Examples: 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

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

Fixing the Quality Problem

1. Create a brand (WHQL, S/MIME, …)
2. Certify software to the brand
3. Tell users that if it has the brand, it's OK
    - (If it doesn't have the brand, it could do absolutely anything)

How not to Test

Not another industry consortium

"You've-never-heard-of-us consortium plans to have a test plan ready for X.509v7"

Not another comprehensive test suite

- Test as many obscure and rarely-used features as possible
    - Vulnerable to implementation tuning / Dhrystone detectors
- X.509 is far too complex to ever test properly
    - Follow any 2-300 message PKIX thread for examples
    - Continuous flow of new extensions and updates make all cert semantics highly mutable
    - What constitutes a pass? (nonRepudiation, anyone?)

How to Test

Just get the basics right

- Cert fetch
- Validity check
- basicConstraints / keyUsage enforcement

Simple enough that there's a single unambiguous pass / fail measure

Tests are designed to quickly catch common bugs

Lookup

App can locate the certificate it needs for an email address (S/MIME), domain name (IPsec), web server (SSL/TLS)

- Checks usability with standard Internet security protocols

App can handle multiple returned certificates

- Choose encryption cert for encryption
- Choose signing cert for signing
  - Catches lack of keyUsage enforcement

Verification

CA-issued cert contains online check URL

- CA server can be contacted at this URL

App reports valid cert † as valid

App reports invalid cert as invalid

App reports forged (manufactured) certificate as invalid

- Catches implicit universal cross-certification problems, any CA in the TCB can spoof any other CA

Verification (ctd)

App reports now-invalid cert † as invalid

- Catches the all-too-common re-read the old CRL trick
- Use blinding to detect cheating

App warns of inability to contact validation server

- Catches apps that time out and continue anyway

CA-side Cert Handling

CA cert handling

- CA cert
  - basicConstraints true
  - keyCertSign set
- EE cert
  - basicConstraints false
  - keyCertSign not set
  - digitalSignature or keyEncipherment set
    - Some CAs create lamp test keyUsage entries
  - Key is valid (e.g. no e = 1)

Catches broken CAs

Client-side Cert Handling

Client-side / application cert handling: CA certs

- basicConstraints set, keyCertSign set $\rightarrow$ accept
- basicConstraints not set or keyCertSign not set $\rightarrow$ reject
  - Catches lack of basicConstraints, keyUsage enforcement
- Rejects CA certs with invalid keys (e.g. e = 1)

Client-side / application cert handling: EE certs

- Can encrypt/decrypt with encryption cert
- Can’t sign/verify with encryption-only cert
- Can sign/verify with signature cert
- Can’t encrypt/decrypt with signature-only cert
  - Catches lack of basicConstraints, keyUsage enforcement
- Rejects EE certs with invalid keys (e.g. e = 1)

Challenge #6

Implementer / User Apathy (HCI)

Users find PKI incomprehensible

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

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

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

Consequences of lack of user understanding

PKI-enabling an app is just a side-job for developers

- Motivation: The boss said do it

  I don't need to pay Verisign a million bucks a year for keys that expire and expire. I just need to turn off the friggen [browser warning] messages.

  — Mark Bondurant, alt.computer.security
- Get it out of the way as quickly as possible
  - CA generates key and mails it out
  - Private key is shared across as much of the org. as possible
  - "Revocation check" repeatedly re-checks against the same old CRL stored on disk
- Meets all PKI checkbox requirements without having to go to the effort of getting it right

Default-to-Secure Design

Make the right way the only way to do it

- PnP PKI makes it very hard to not do local key generation, distinct signature and encryption keys, minimised TCB (trusted CA certs), keys generated in hardware, etc etc
- Realtime validity check makes it very hard to just go through the motions of performing the check

KISS

Simple design discourages homebrew (= insecure) mechanisms

```
cryptCreateSession session
cryptSetAttribute session, _
  CRYPT_SESSINFO_SERVER_NAME, "[Autodetect]"
cryptSetAttribute session, _
  CRYPT_SESSINFO_USERNAME, userName
cryptSetAttribute session, _
  CRYPT_SESSINFO_PASSWORD, password
cryptSetAttribute session, _
  CRYPT_SESSINFO_PRIVKEYSET, keyset
cryptSetAttribute session, _
  CRYPT_SESSINFO_ACTIVE, true
```

This is the entire PnP PKI (Challenge #2) interface

KISS (ctd)

Other operations are similarly idiot-proof

```
crypt.CreateSession( session );
status = crypt.CryptCheckCert( certificate,
  session );
```

This is the complete real-time validity checking (Challenge #3) interface

Conclusion

Certificate lookup

- Simple HTTP interface uses the web as a Public File

Enrolment

- PnP PKI eliminates enrolment pain, makes it easy to do the right thing

Certificate validity check

- Real-time online check matches requirements for online banking, etc

Quality control

- Core functionality checked through simple, unambiguous tests

Postscript: Implementation Availability

Available as the cryptlib security toolkit, `http://www.cs.auckland.ac.nz/~pgut001/cryptlib/`

Implementation and usage details

- ANSI C, runs on anything: BeOS, DOS, eCOS, µITRON, Mac OS X, MVS, QNX Neutrino, RTEMS, Stratus OS, Tandem NSK, Unix (any variant), Win16, Win32, WinCE/PocketPC, VxWorks, VM/CMS, no OS (runs on the bare metal)
  - Minimum RAM requirement: ~128K (may run in 64K)
  - Please contact the author if using one of the more obscure embedded/RTOS systems with special considerations
- Open-source implementation, dual-licence
  - GPL or standard commercial license, your choice