Plug-and-play PKI:
A PKI your Mother can use

Peter Gutmann

University of Auckland

Why is PKI failing?

The usual suspects...

- Difficult to deploy
- Expensive
- Lack of interoperability
- Poor match to pressing real-world problems

> 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

From the end-user perspective: It's too hard to use

- How can we make it easier to use for end users?

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

- 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 passport, 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
- Filled out more fields in eleven further web pages
  - Much of the contents were incomprehensible to the user: "certificate Distinguished Name", "X.509 SubjectAltName"
  - User guessed and clicked "Next"

How bad is it really? (ctd)

- Web page announced that a certificate had been issued, but none seemed available
- 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

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

What should it 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
- Not supported in Win'95/98/ME
- Those who need it most don't have it
  - Expecting your mum 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

- 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
  - 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 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

PKIBoot: Obtaining Initial Certificates (ctd)

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
- Eliminates the need for secure service discovery
  - Authenticate the service *after* it's been discovered
  - Security and transparent service discovery are mutually exclusive

PKIBoot: Obtaining Initial Certificates (ctd)

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

PKIBoot: Obtaining Initial Certificates (ctd)

- Since only the legitimate service can generate the MAC, certificate spoofing isn't possible
- Eliminates the need for secure service discovery
  - Authenticate the service after it's been discovered
  - Security and transparent service discovery are mutually exclusive

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

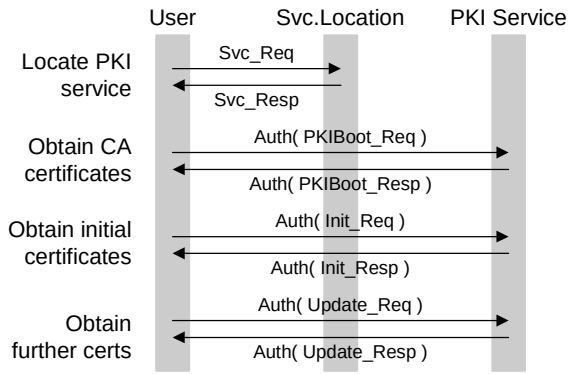
Obtaining User Certificates (ctd)

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

PKIBoot for Internet-enabled Toasters

Design so far assumes a user sitting in front of a PC

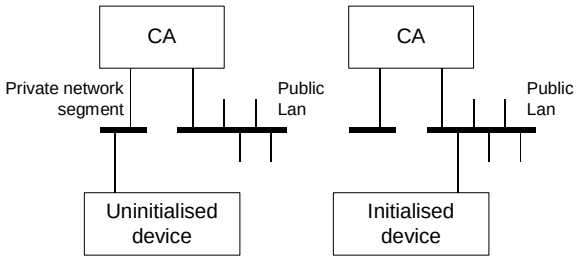
- Embedded devices don't have a user to drive things, and often no UI

Use the baby duck security model

- Device initialisation removes any existing state
- Newly-initialised device imprints upon the first thing it sees
- Device trusts it to issue certificates

PKIBoot for Internet-enabled Toasters (ctd)

Security is physical, not based on MACs/sigs



- PKIBoot and signing certificate issue are done on a private/secure network
  - Physically isolated LAN segment (factory setup)
  - Plugged directly into a laptop (field setup)
- Further operations (e.g. firmware download) are done on public network, secured by PKIBoot certificates

Implementation details

Store-bought or home-grown?

- Use existing standards: Nothing terribly appropriate available
- Home-grown design: Not yet another PKI RFC!
  - 21 RFCs, 30 RFC drafts
  - 1,600 pages of PKI RFCs
  - At least that many more non-IETF PKI standards

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 — Me

Implementation details (ctd)

Underlying transport protocol is PKIX CMP

- Incredibly complex
- Ambiguous specification
  - No two implementations can agree on what to do
- See the paper for more information — too horrible to go into further here

PnP PKI in action

User

- Enters username + password
  - 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

Future work

Use of non-CMP transport

- SCEP (Simple Cert Enrolment Protocol), used in IPsec routers
- Somewhat quirky: Cert messaging is done via secured messages, but cert fetch is done via (insecure) HTTP GET
  - PKIBoot is difficult
  - Overload GetCert message: NULL issuer name → get all trusted certificates
  - No provision for MAC'd messages
- SCEP uses all-in-one certificates, so separation of signing and encryption certificates isn't possible

Future work / Availability

PnP for other certificate operations

- Certificate update/renewal
- Mostly taken care of by CMP anyway

Available in cryptlib, `http://www.cs.auckland.ac.nz/~pgut001/cryptlib`