

PKI in practice

Željko Vrba

March 16, 2008

Abstract

This lecture has three main parts: in the first part, public key cryptography and the topic of trust are introduced. The second part introduces technical details of the X.509 public key infrastructure (PKI) and shows basic usage of the OpenSSL tool to accomplish most common tasks. The last part of the lecture covers experiences from real-world deployment (the author has worked as one of the chief administrators of the Croatian National CA).

- 1 Introduction
- 2 X.509
- 3 Experiences from the real world
- 4 Conclusion

Cryptography: basic definitions

Cryptography (Greek): hidden writing

- cryptanalysis / cryptology
- plaintext / ciphertext
- source of randomness / key
- hashes / symmetric / asymmetric cryptography
- primitives / protocols

Books: The wikipedia article has a decent list of external links.

<http://en.wikipedia.org/wiki/Cryptography>

WARNING: never, ever, design your own crypto algorithm.

Asymmetric cryptography

- Key: public + private part (key-pair)
- Encryption and digital signature (Hash: 1-way function of suitable length (≥ 160 bits; *birthday paradox!*))
- Public key: encryption / signature verification
- Private key: decryption / signature generation
- Algorithms: RSA (S+E), ElGamal (E), DSA (S; variant of ElG), DH key agreement
- Key problem: *trust* (i.e. how to ensure that key really belongs to the listed *entity*).

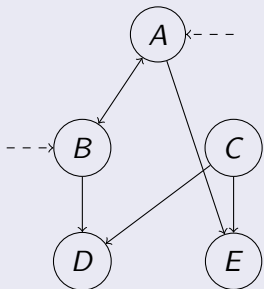
Trust (1)

- Public key: (Key, Id)
- PROBLEM Identifiers can be faked:
 - Mallory publishes (K, Alice) and can read all communications destined to her!
 - Could be discovered relatively quickly, but the damage is already done.
 - First-time contact is critical!
- SOLUTION Introduce *trusted* third party (TPP) to sign public keys and *revocation lists*.
- Hierarchical model (X.509) or web of trust (OpenPGP)

Trust (2)

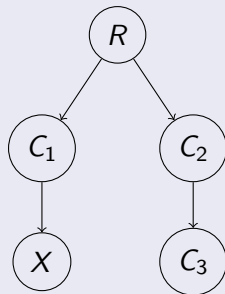
WoT (OpenPGP)

Self-signed key + signatures of others.



Hierarchy (X.509)

Certificate = Signature(Key, Id).



Problem

Neither model really replaces personal contact.

Subset of most relevant standards

- Certificate and CRL profile (RFC3280)
- Qualified certificate profile (RFC3739) [EU directive on ES]
- OCSP (RFC2560) [slightly better than CRLs]
- CMP (RFC4210)
- Timestamp protocol (RFC3161, policy requirements: RFC3628)
- A number of PKCS# standards (we will touch PKCS#1, PKCS#7, PKCS#10, PKCS#12). <http://www.rsa.com/rsalabs>

All of them are defined using ASN.1 notation and specify DER encoding rules.

Abstract Syntax Notation

- Abstract *syntax* which defined *structure*, and
- set of *encoding rules* (BER, DER, PER, XER).
- Encoding has self-describing structure, and
- several ways to encode the same data (BER).
- X.509: mostly a (complicated) hierarchy of key-value pairs.
- Keys: encoded using OIDs <http://www.oid-info.com/>
- All objects in PKI are encoded using DER.

OpenSSL ASN.1 dump

```
openssl asn1parse -dump -i -in /tmp/z.cer (add -inform der  
for binary [pem format is default])
```

X.509 certificate: what's inside?

X.509 contents

version, signature algorithm, *unique (for CA)* serial number, validity, issuer DN, *subject* information (DN, public key, extensions [e.g. key usage])

Distinguished name (DN)

A set of attribute/value pairs *intended* to uniquely identify the subject and the CA. Not so in practice.

How to view it?

- In the browser
- OpenSSL: `openssl x509 -in /tmp/z.cer -text`

X.509 certificate: how to get one?

Personal certificate

- Follow procedures of your CA. <http://www.cacert.org>
- Mostly doable from the browser.

Server certificate: PKCS#10

CA will require that you generate a PKCS#10 request:

```
openssl req -new -out request.p10
```

- Outputs pw-protected `privkey.pem` and `request.p10`.
- The latter contains entered data, public key, all *signed*.
- Entered data mostly irrelevant: replaced by CA.
- Private key and certificate must be put on the server.

PKCS#12 (or PFX)

- Storage format for certificates and private keys, protected by the export password.
- Commonly used by web browsers and mail clients.
- Badly designed:
<http://www.cs.auckland.ac.nz/~pgut001/pubs/pfx.html>

Creating PKCS#12 file

Input: Certificate chain, own certificate, private key.

```
openssl pkcs12 -export -out z.p12 -in zvrba.0.pem -inkey  
privkey.0.pem -certfile cacert-root.crt
```

- The author has been one of the chief administrators of Croatian National CA (hosted within FINA – Financial Agency).
- A *lot* of information covered by NDA. The rest can be inferred from public tenders :-) [see “Narodne Novine” ; <http://www.nn.hr>]
- Deployment scenario: private keys on smart-cards, strict identity verification, normalized + qualified certificates.
- Strict and confusing laws.

- Politics and law make *easy* things *unbelievably hard*.
- CA software has to be certified (FIPS level 3) – law.
- Compatibility matrices.
- A lot of (proprietary) SW that needs to be integrated.
- Even firewalls can break stuff (e.g. killing inactive TCP connections).
- Expensive support, unexpected bugs, long fix periods.
- **Use a single SW vendor if you can!** (MS CA)

Experiences: front-end (RA)

- RA = registration authority (id checking, data entry, issuance)
- People of older generation (aged 45 upwards), no computer education, no English knowledge.
- Some of RA apps are non-localized (expensive!) – use cookbooks.
- Same problem as users: overwhelmed with new concepts.
- + issuance on smart-cards may fail (causes confusion).
- Receive many questions that they're not qualified to answer (procedures, technical details, etc.)
- Stressful: only one of assignments; work with people of all temperaments.

- Mostly users uneducated in IT just hoping to make their business more efficient (document signing, payment, etc).
- See “RA” for typical profile.
- May have a smart-card from before: SW/HW compatibility problems.
- Complicated procedure (requirement of physical appearance twice).
- Hard to persuade that digital certificate is worth something (CA is a business, afterall).
- Lack of applications – little incentive to get a certificate (only e-payment).
- Locked cards.
- Running in insecure environment (OS).

The End: PK cryptography beyond X.509

- OpenPGP standard has merits (more flexible), but more user-demanding (in the author's opinion – I have switched to CACert/X.509 after many years of PGP).
- OpenSSH and manual key verification (no TTP).
- RSA algorithm is getting weaker (more computing power): ECC crypto.
- Legally binding digital signatures and non-repudiation.

The End: conclusion

- Public-key cryptography: rather complex technology.
- A lot of concepts to learn even for the most basic use.
- Complicated by existence of several (competing) “standards”.
- Real security gains: questionable (but see MAC (mandatory access control) frameworks).
- As strong as the weakest link (most often: the password!).