PKI and x509

Dirk-Willem van Gulik / dirkx@apache.org

@semantics - roma - san francisco - oslo - leiden

Overview

- Basics
- Trust
- Technical options
- Conclusion
- Questions

Basics 1/3

- Cyphers - symmetric encryption (DES, AES)
  - hard to break
  - does not leak
  - computationally light; longer keys ok
  - problem: shared secret
  - feature: shared secret

Basics 2/3

- Public key cryptography
  - public (91) and private (7, 13) pair
  - lockbox/valve function
  - expensive to calculate
  - can leak

Basics 3/3

- Typical use (SSL, PGP, S/MIME)
  - Use PK to establish trust
    - Exchange a session key
    - Use symmetric crypt with session key
- Compromise: avoids shared secret penalty - yet allows trust, avoids leaking and intensive calculations.

Trust

- Prelude to a transaction
  - How much trust is needed ?
  - Who needs to trust whom ?
- Fortified by a receipt after the transaction
  - Non-repudiation
- Business transactions are NOT symmetric.

Practical Trust

- Burden/penalty for "leaking" often lopsided
- Shared secrets too painful to manage
  - distribute, logistics
- PK: verify possesion of private key matching a public key
  - keep your own list or trust a third party

More Practical Trust

- Keep your own list
  - It's a pain
  - No blame-game when you get it wrong
  - *(but you may need to do it anyway)*
- Let a third party do that: Certificates
  - Need to trust them; biiig waiver
  - *(but it may be needed anyway (D&B,KvK))*

Certificate

- Descriptive metadata (name, email)
- Validity/use “rules”
- The Public key
- Perhaps some signatures of others
- Conveniently packaged (PGP, x509)

Technology

- x509
  - binary file format to conveniently pack public keys and some metadata
- SSL, S/MIME
  - Proof possession of private key of certificate shown to each other
  - Agree session key and Encrypt payload

Baseline Options

- No certificate at all
- Just a certificate (plain/self-signed)
  - proof that you a have a private key
- A certificate signed by “someone else/CA”.
  - proof that you have a private key
  - and also showed that fact to the ‘CA’

RIPE / RIR

- Prove that you are talking to RIPE
  - why ?
- Prove ‘who’ you are
  - why ?
- Implicit non repudiation
- **IF** the private keys are ‘a key’.

Options

server / client	3rd Party signed	RIPE self signed	NOT signed
3rd Party signed			
RIPE signed			
Customer self signed			
No Certificate			

Options

server / client	3rd Party signed	RIPE self signed	NOT signed
3rd Party signed			
RIPE signed			
Customer self signed			
No Certificate			

(reality check)

- Each option is available in COTS
- “Trust” is configurable
  - as explicit user decision pain.
  - as ‘hidden’ and accepted.
- And if you care enough in ‘real life’ (£€$)
  - Set aside dedicated hardware/room

other options

- Hardware tokens
  - Chipcards
  - iButton's
  - RSA SecurID
- Paperware
  - s/key, list of one time tokens

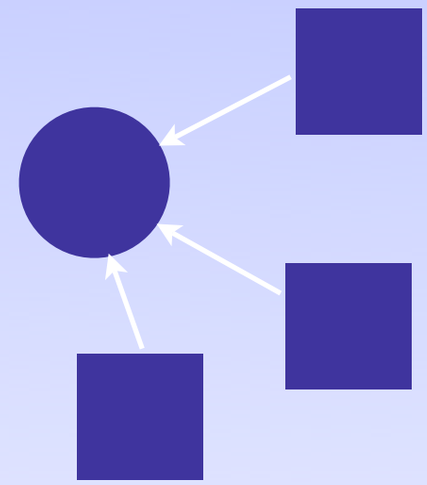
Client / Server

- One ‘Server’
- Many ‘Clients’
- Asymmetric business relation:

  Clients **want** something from the server

  Server **acceptable** to the client

  Basis for a transaction



Server 3rd Party Signed

- Clients just trusts a certain third party
- This third party has established the servers identity to its satisfaction.
- That is ‘enough’ for the client to satisfy the business related trust needed by the client for the transaction with the server.
- ‘enough’: generally yes...

Client 3rd Party Signed

- Server just trusts a certain third party
- This third party has established the clients identity to its satisfaction.
- That is ‘enough’ for to satisfy the business related trust needed by the server for the transaction with the client.
- ‘enough’: generally no...

Server: RIPE signed

- i.e. a Self signed certificate
- Clients have to trust that the Server can manage their own keys and identity
  - (and propably a whole lot more)
- Thus: axiomatically good enough for the business transaction

Client: RIPE signed

- i.e. client generated certificate (public key) is signed by RIPE.
- Client does not care if RIPE signs carelessly
  - except if they sign a key with the same metadata as their own ? non-issue
- Server has to trust its own keys

Client: self signed

- Server needs to keep track of client certificates of customers.
  - Really - this is just keeping a ‘list’
- Server has to trust that the client did not leak their key.

Roundup

- You are going to trust something
  - Your own list of keys
    - just RIPE's / just your customers
  - Your own 'CA'
  - Or some third party CA
- **Premisse: that they key is 'key'.**

“CA” role

- Publish your ‘root’ certificate
  - CommonName (CN), Validity range
  - Fingerprints
  - Rollover procedure
- *or alternatively*
  - have it signed by a “well known” CA

Problem: Revocation

- Certificates go ‘bad’
  - Short Time to live
  - Revocation list
  - “Backoffice” check

Revocation: Short TTL

- Need to re-issue a lot
- Inherently less leaky - more secure
- Inherently more automated - and thus easier to subvert.
- Easy to stop very early in the SSL exchange

Revocation Lists 1/2

- List of invalid certificates
- kept at the server
- ***Must*** be distributed:
  - if third parties rely on your trust statement/signature.
  - signed by the same private root-ish key which originally vouched for the validity.

Revocation Lists 2/2

- Issue versus recall of certificates
  - not symmetric in terms of biz/legal meaning
  - always err on the safe side
  - very different admin roles.
  - yet both need access to sensitive key.

Revocation: backstop

- First Check for validity (signature, dates)
  - easy, during SSL exchange
    - no valuable info on the web server
- Then check with the backend
  - Lot of resources in motion
  - But you may need to do it anyway.

Bottom Line

- What trust is needed ?
- Burden of trust on whose side ?
- Who is weak, who is strong ?
- Who gets blamed if it went wrong ?
- Who can make sure it does not go wrong ?
- Who is not penalized when he fails ?

Conclusion

- At a minumum
  - Self signed RIPE cert for the server must be acceptable.
  - RIPE signed client certs ought to be acceptable
  - Self signed certs of clients may be quite acceptable due to workflow/biz-process.

Questions ?

Dirk-Willem van Gulik <dirkx@apache.org>

@semantics - roma - san francisco - oslo - leiden