TLS

RFC2246: The TLS Protocol
What does it achieve?

• Confidentiality and integrity of the communication
• Server authentication
• Eventually: client authentication

(c) A. Mariën -
What is does not do

• Protect the server
• Protect the client
• Protect stored data
• Provide any liability by itself
• See PKI story
The TLS Handshake Protocol

• Three sub-protocols used to allow peers:
  – To agree upon security parameters for the record layer
  – To authenticate themselves
  – To instantiate negotiated security parameters
  – To report error conditions to each other

• Change cipher spec protocol

• Alert protocol overview

• Handshake Protocol
The Handshake Protocol is responsible for negotiating a session, which consists of the following items:

- session identifier: to identify an active or resumable session state.
- peer certificate: X509v3 [X509] certificate of the peer.
- compression method
- cipher spec: bulk data encryption algorithm and MAC algorithm (MD5/SHA)
- master secret: 48-byte secret shared between the client and server.
- is resumable: flag indicating whether the session can be used to initiate new connections.
Change cipher spec protocol

• to signal transitions in ciphering strategies
• Switch: pending state becomes current state
Alert protocol

• Alert messages
  – the severity of the message
  – A description of the alert

• “fatal” messages: immediate termination of the connection
  – Closure alerts
  – Error alerts
Handshake Protocol overview

Steps:
• Exchange “hello” messages to agree on algorithms
  – exchange random values
  – check for session resumption.
• Exchange cryptographic parameters
• Client and server agree on a premaster secret.
• Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
• Generate a master secret from the premaster secret and exchanged random values.
• Provide security parameters to the record layer.
• Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.
TLS messages

Client hello

ServerHello
Certificate
ServerKeyExchange
CertificateRequest
ServerHelloDone

Certificate
ClientKeyExchange
CertificateVerify
[ChangeCipherSpec]
Finished

[ChangeCipherSpec]
Finished
Starting TLS

• Client: client hello
• Server: server hello
• Info:
  – Protocol Version
  – Session ID
  – Cipher Suite
  – Compression Method
  – ClientHello.random
  – ServerHello.random
TLS key exchange

• Up to four messages:
  – the server certificate
  – the server key exchange
  – the client certificate
  – The client key exchange.
End TLS negotiation

• Server: “hello done” message
• Client: “change cipher spec” message
Client-server

Client

Client Key pair

Server

Server Key pair

(c) A. Mariën -
Client: key store

Client

Client keys

Client

USB

Smartcard

Smartcard reader

(c) A. Mariën -
Client – store

• High-level interfaces:
  – PKCS#11 interface
  – Microsoft: CSP

• Smartcard APIs

• Disk storage:
  – Key encryption (password derived key)
  – Wallets

(c) A. Mariën -
Client: via proxy

Protected communication?
Client: via proxy (1)

Client -> Proxy -> Server

Connect xxx.yyy.org:443

Packet exchange

Virtual connection

(c) A. Mariën -
Client: via proxy

Client

Proxy

Server

GET https://s.org/...

TLS communication

(c) A. Mariën -
Server: key store

Server

Server keys

Smartcard reader

Smartcard

Server

TRD

(c) A. Mariën -
Server: via reverse proxy

- Client
- Reverse Proxy
- Server

Protected communication?
Server: via reverse proxy

Client — Reverse Proxy — Server

TLS 1 — [TLS 2]
openSSL

• Public implementation
• Hooks into Apache (and others)
• All common functionalities
• Vendors of TRD link into it
Public Key Infrastructure

(c) A. Mariën -
Fantastic technology...

- Public/private key technology is a magnificent solution for:
  - Key exchange
  - Digital signatures

- But key management is a nightmare, much like with symmetric keys
Critical element

• To encrypt for someone, use his public key
  – Encdata = enc(data,symkey);
  – enckey = enc(symkey,pubkey)
  – Send encdata, enckey
• To check his signature, use his public key
  – Signdata = enc(#data,privkey);
  – Checksign = dec(signdata,pubkey) == #data
• CRITICAL: link between public key and entity, otherwise no use
• Need to get the public key
• Need to trust the public key
• Intruder can fake it all…

(c) A. Mariën -
Certificates: why

• Certificates are an important concept to make PK manageable
  
• Replace individual links and trust by trust in CA
  – CA signs statements on individual key pairs
  – Need only to trust the CA
Certificates: what

• Signed statement by CA about link between public key and entity
  – Cert = sign((identity, pub key, valid from/to, ...), privkeyCA)

• Assumed:
  – Trust that CA checked possession of private key by the entity
  – Trust in CA itself: the CA can fake anything
PKI: main elements

• Issuing certificates
• Trust models
• Certificate publishing
• Certificate suspension and revocation
CA operation

• CA issues certificates (signs pieces of data)
• CA publishes certificates (to allow customers to find certificates, without prior contact)
• CA publishes revocation lists (LDAP server)
• CA uses Registration Authority(s) (RA) to establish link between key and entity
RA – CA

• CA
  – CA has certificate practice statement (CPS): rules of operation
  – CA assign RA and gives guidelines
  – RA may have multiple Local RA (LRA)
  – (L)RA establishes and registers entity/public key link

• RA checking
  – Physical verification
  – Paper trail checks
  – Round trip checks
    • Send mail with code: check address (access)
    • Send email with code: check email address (access)
  – Key possession checking
    • Challenge response: generate challenge, request must be encrypted with matching private key

(c) A. Mariën -
Trust models

• Top level CAs
• Subordinate CAs
  – People certificates
    • Level 1, level 2, level 3
  – Server certificates (SSL)
  – Code signing certificates (activeX, java jars)
  – VPN certificates
  – CRL/OCSP signing

(c) A. Mariën -
Certificate suspension and revocation

• LDAP
  – Globalsign:
    • Host: directory.globalsign.net
      – port LDAP: 389
      – port LDAPS: 636
      – Base DN: dc= globalsign, dc=net
  • CRL: ou=Class 1 CA
    – Attribute: certificateRevocationList
eID – Belgium

• Electronic identity card
• Cards: company a
• Certificates: company b
• Important role: ‘rijksregister’
Information

• Name, first name(s), birthdate, gender, hash of photo
• RRN, Nationality
• Card number, validity, issuing municipality
• Keys, certificates
  – Authentication
  – Signing
CA

- Belgian root CA
  - Card CA
  - Citizen CA
    - Authentication
    - Signing
- Government CA
  - Code sign
  - Server sign
  - RRN

(c) A. Mariën -
Revocation

• CRLs
• Delta CRLs
  – Making CRL practical
  – Alternative for OCSP

(c) A. Mariën -
Operations

• RA: Rijksregister
• LRA: municipality
• Card production
• Certificate production
• CRL and delta CRL production

(c) A. Mariën -
Revocation issues

• Problem with certificates
  – Once given, always validating correctly
  – Infinite replay attack (within limits of valid dates)
  – Similar to signed email message

• Need a way to flag certificates as no longer valid
  – Two methods
    • Check with issuer before trusting
      – OCSP protocol
    • Get periodically list from issuer with revoked certificates
      – CRL and delta-CRL lists
Problem with the solution for the revocation problem

• Principle: certificates allow a direct connection between user and server without third party connectivity
  – User gets a certificate
  – Server can check it with his stored CA certificates
  – So:
    • Server needs to get CA certificate only once/CA
    • No need to contact CA for every CA user
Certificates

1a: I am John
2a: Cert(John, key, validated 14/2/2014 by CA)
3: Cert(John, key, validated 14/2/2014 by CA)
2b: Cert(CA, key, validated 11/11/2011 by CA)
4: Hi John

1a,2a,1b: once
3,4: repeated
Alternative to certificates: trusted third party

1: I am John

2,3: This is John, validated 14/2/2014 by TTP

4: Hi John

TTP involved each and every time
Certificates with OCSP

1: Cert(John, key, validated 14/2/2014 by CA)
2: sign(certjohn, still valid)
3: repeated
4: Hi John