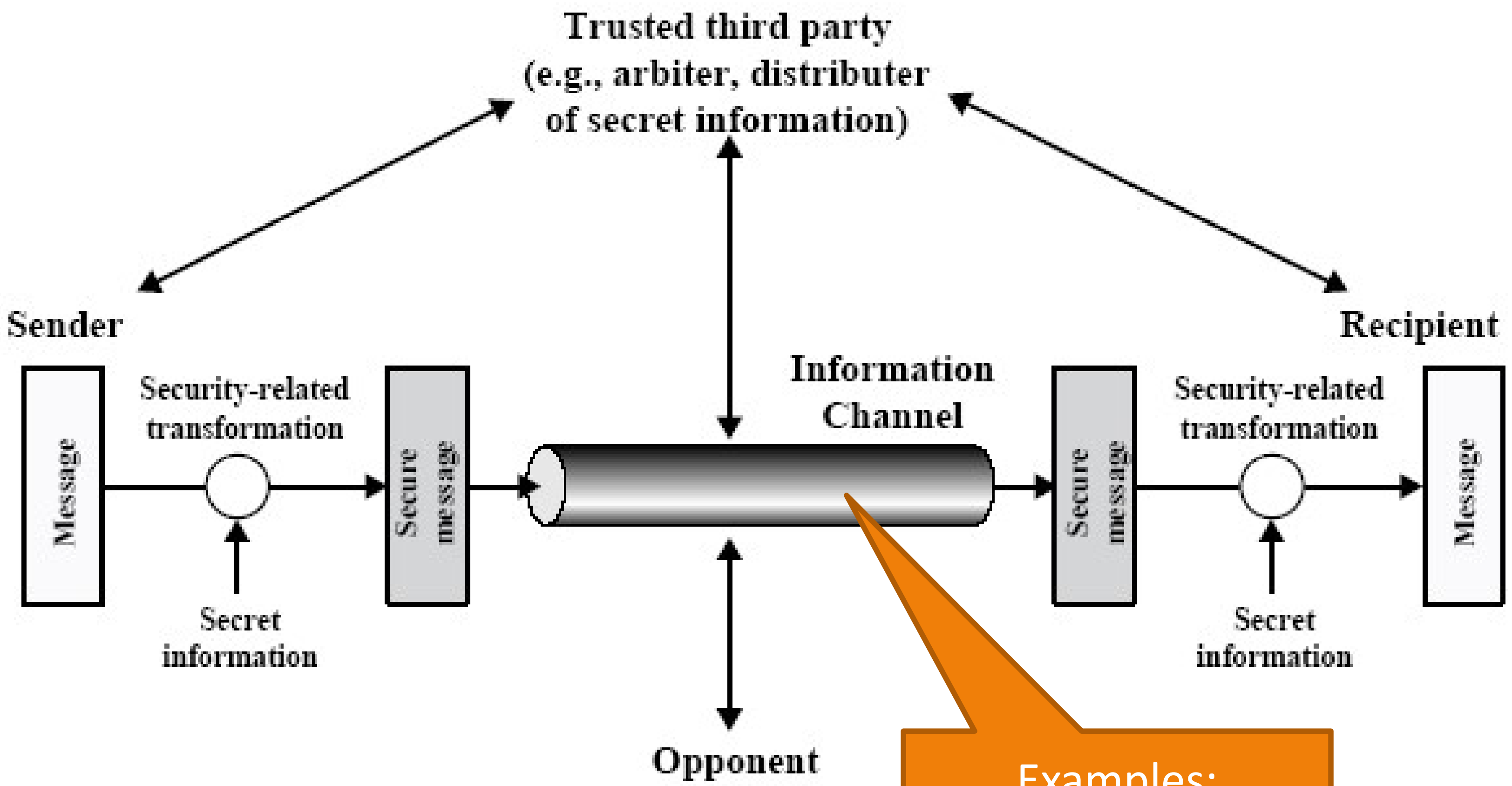
TLS and SSH overview

Segurança Informática em Redes e Sistemas
2022/23

Miguel Pardal

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Ricardo Chaves

Secure communication channel



Examples:

TLS, SSH

Roadmap

- TLS – Transport Layer Security
- SSH – Secure Shell
- Key management

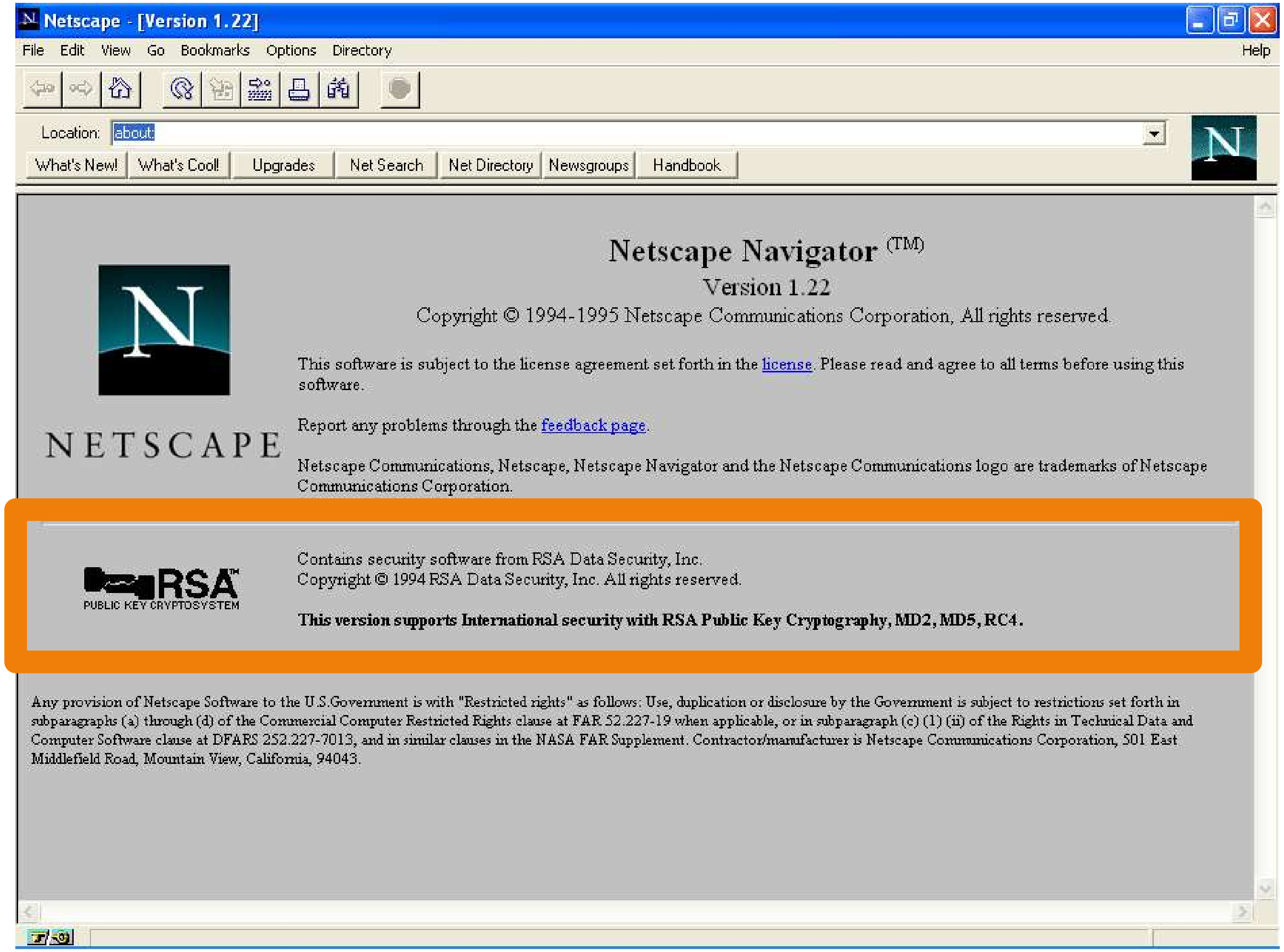
Roadmap

- TLS – Transport Layer Security
  - **HTTPS**
- SSH – Secure Shell
- Key management

HTTPS

- **HTTPS** = **HTTP** over **S**SL (now TL**S**)
  - In fact, SSL was originally designed to be used with HTTP
    - Netscape Navigator (1994)
      - Forefather of Mozilla Firefox
  - Version history
    - SSL 1.0 1994
    - SSL 2.0 1995
    - SSL 3.0 1996
    - TLS 1.0 1999
    - TLS 1.1 2006
    - TLS 1.2 2008
    - TLS 1.3 2018

Netscape Navigator



Netscape Navigator (TM)

Version 1.22

Copyright © 1994-1995 Netscape Communications Corporation, All rights reserved.

This software is subject to the license agreement set forth in the license. Please read and agree to all terms before using this software.

Report any problems through the feedback page.

Netscape Communications, Netscape, Netscape Navigator and the Netscape Communications logo are trademarks of Netscape Communications Corporation.

Contains security software from RSA Data Security, Inc.
Copyright © 1994 RSA Data Security, Inc. All rights reserved.

This version supports International security with RSA Public Key Cryptography, MD2, MD5, RC4.

Any provision of Netscape Software to the U.S.Government is with "Restricted rights" as follows: Use, duplication or disclosure by the Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer Restricted Rights clause at FAR 52.227-19 when applicable, or in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Netscape Communications Corporation, 501 East Middlefield Road, Mountain View, California, 94043.

- A few months later, in July 1995, a computer programmer ordered the first book ever sold by an online bookstore named after a river in South America…

TLS goals

- Secure communication channels over TCP/IP
  - Current version: **TLS 1.3** – august 2018
    - Standard based on the deprecated **SSL (Secure Sockets Layer)**
    - Sometimes called **SSL** for that reason
  - Manages secure sessions over **TCP/IP** per application
    - Initially designed for the HTTP protocol (HTTPS)
    - Currently used by other protocols, e.g., SMTP, IMAP, POP3
- Security mechanisms
  - **Authentication** of the communicating parties
  - **Confidentiality** and **integrity** of the communication
  - **Key distribution**

TLS utilization

- TLS is just a protocol; not a standard API
- Common APIs:
  - Reference API for SSL: SSLref (Netscape)
  - Public implementations: OpenSSL, GnuTLS, SSLeay, Mbed TLS, Java Secure Socket Extension (JSSE)
- Remote interfaces:
  - Conventional: protocol/port
    - e.g., TCP/443 for HTTPS
  - STARTTLS: allows upgrading text connection to TLS
    - Defined for SMTP, IMAP, POP, SMTP, FTP, IRC,...

TLS operation management

- Client-Server model as in TCP
- The applications (e.g., web, mail) define the strategy
- Authentication
  - If it is needed and how it is performed
- Cryptographic algorithms
  - The client presents the **cipher suites** it supports
  - The server selects one
- Session key management
  - Lifetime of the **master secret** = lifetime of the **TLS session**
    - Used whenever the client and the server decide to communicate
    - Maximum of 24 hours recommended
  - Lifetime of the **session keys**: at most lifetime of the TCP connection

TLS Cipher Suites

- **Cipher suite**
  - Public-key algorithm
  - Symmetric encryption algo.
  - MAC algorithm
- TLS supports several

- **Negotiation**:
  - Client offers choice
  - Server picks 1
    - e.g., the most secure

- **Common algorithms**
- Public-key encryption
  - RSA, ECDSA
- Symmetric ciphers
  - AES: block
  - ChaCha20: stream
- Hash functions
  - SHA-2
  - SHA-3

TLS protocols

- **Handshake Protocol**
  - Exchange of identity and supported cipher suites
  - Authentication of the communicating parties
    - Client challenges the server, that proves its identity with certificate(s) *[Optional, but mandatory if the next exists]*
    - Server challenges the client, that proves its identity with certificates *[Optional]*
  - Key distribution
- **Record Protocol**
  - Creation and verification of secure messages
    - ~~Compression~~, cipher, integrity control

Removed in TLS 1.3

TLS handshake (1)

Purpose

1. Server authentication
2. Negotiation: agree on crypto algorithms
3. Establish keys
4. Client authentication (optional)

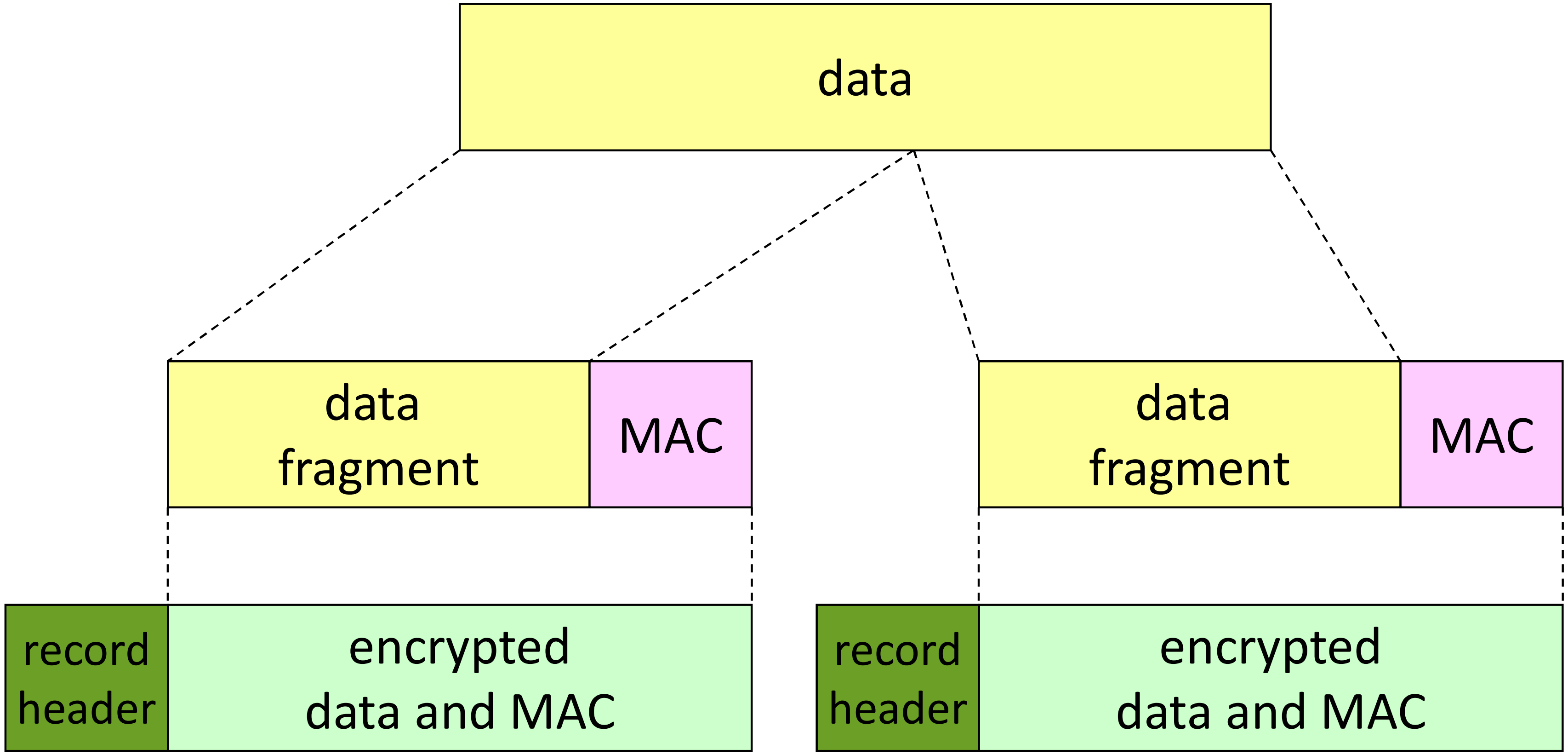
TLS handshake (2)

1. **Client** sends list of algorithms it supports, along with client nonce
2. **Server** chooses algorithms from list; sends back: choice + certificate + server nonce
3. **Client** verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. **Client and server** independently compute encryption and MAC keys from pre_master_secret and nonces
5. **Client** sends a MAC of all the handshake messages
6. **Server** sends a MAC of all the handshake messages

TLS handshake (3)

- Last two steps protect handshake from tampering:
  - Client typically offers range of algorithms, some strong, some weak
  - Man-in-the-middle could delete best algorithms from list
  - Last two message MACs allows detecting such an attack
- Why not simply sign or add MACs to the handshake messages?
  - Not possible: it is the handshake that sets up the keys!

TLS record protocol

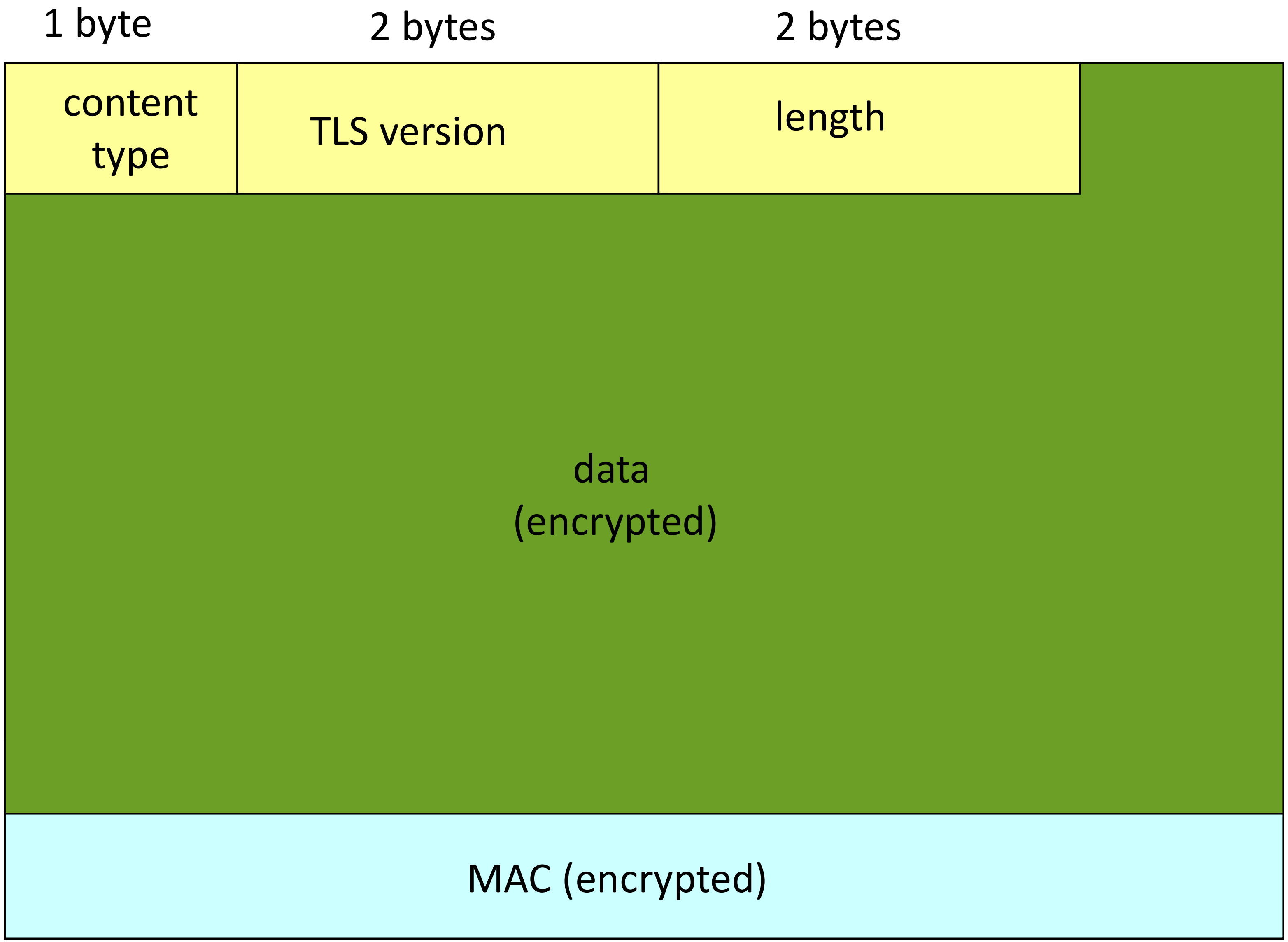


Record header: content type; TLS version; length

MAC: is function of the sequence number and the MAC key M_x

Fragment: each data fragment has up to 2^{14} bytes (~16 Kbytes)

TLS record format



1 byte

2 bytes

2 bytes

content type

TLS version

length

data (encrypted)

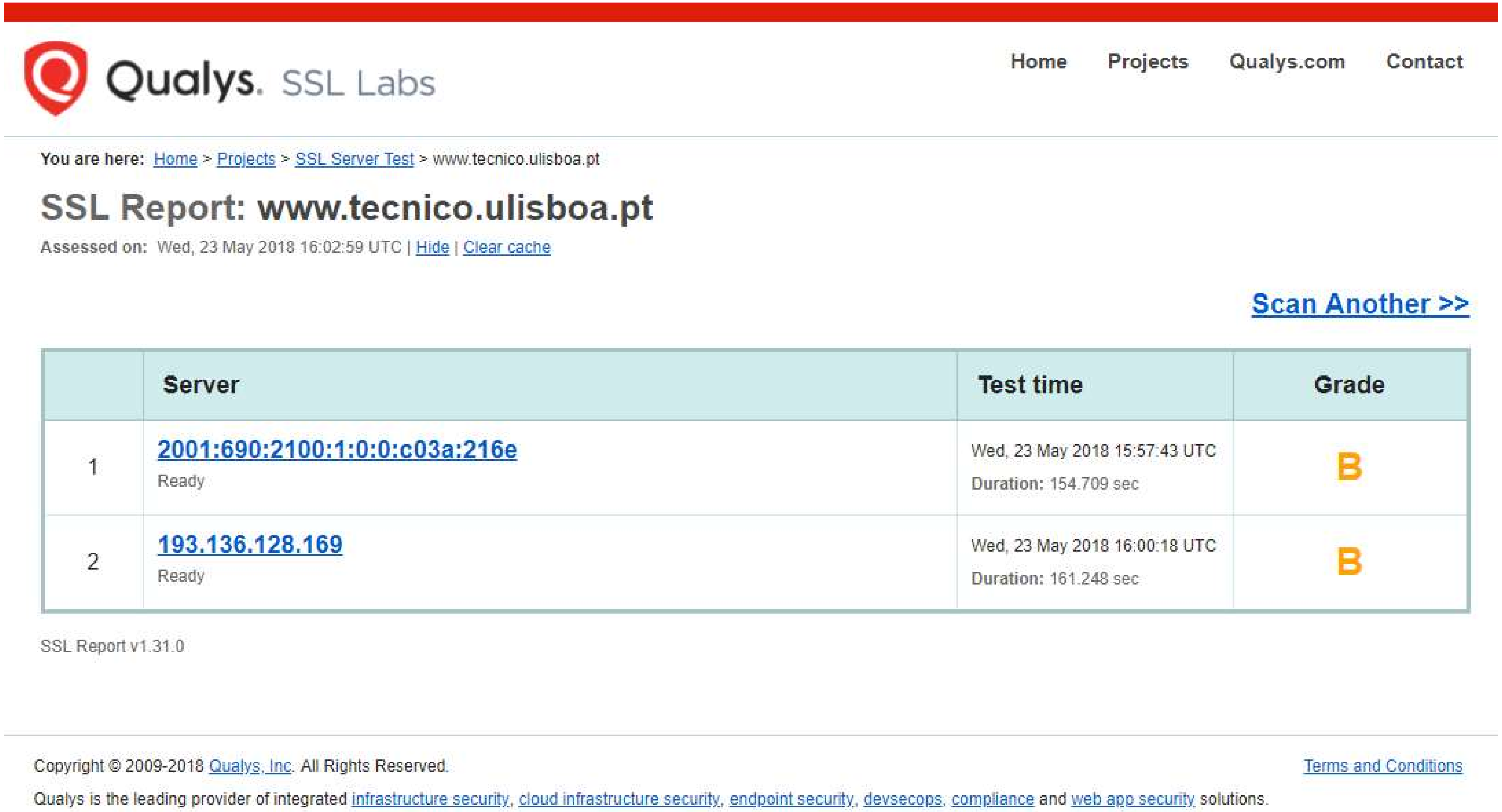
MAC (encrypted)

TLS performance

- First implementations were slow
- Currently, when properly configured, can be fast

- “On our production frontend machines,
  TLS accounts for
  less than 1% of the CPU load,
  less than 10 KB of memory per connection and
  less than 2% of network overhead.”
  - Adam Langley, Google "Overclocking SSL“
  - https://istlsfastyet.com/

TLS tool: SSLTest



You are here: Home > Projects > SSL Server Test > www.tecnico.ulisboa.pt

SSL Report: www.tecnico.ulisboa.pt

Assessed on: Wed, 23 May 2018 16:02:59 UTC | Hide | Clear cache

Scan Another >>

	Server	Test time	Grade
1	2001:690:2100:1:0:0:c03a:216e Ready	Wed, 23 May 2018 15:57:43 UTC Duration: 154.709 sec	B
2	193.136.128.169 Ready	Wed, 23 May 2018 16:00:18 UTC Duration: 161.248 sec	B

SSL Report v1.31.0

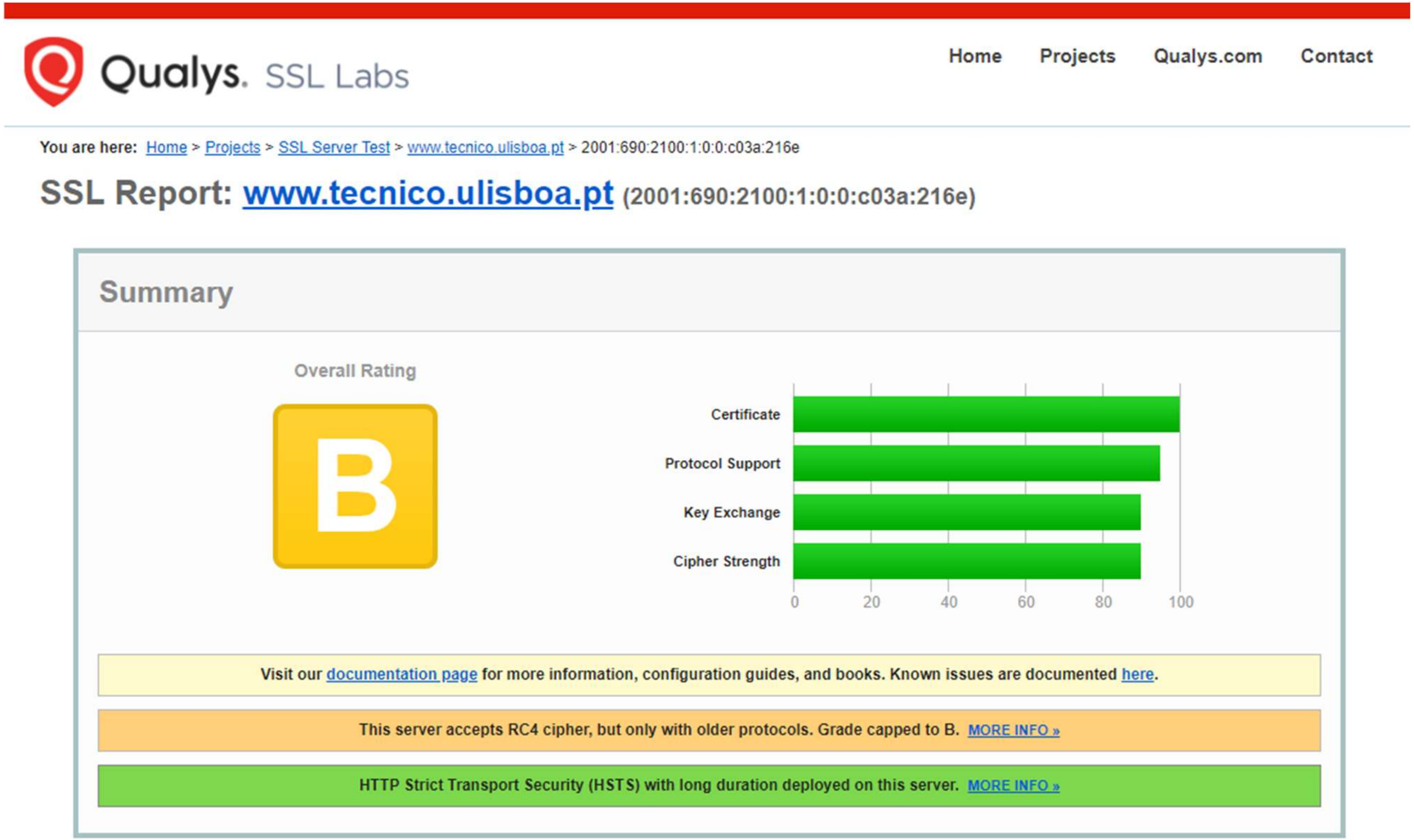
Copyright © 2009-2018 Qualys, Inc. All Rights Reserved.

Terms and Conditions

Qualys is the leading provider of integrated infrastructure security, cloud infrastructure security, endpoint security, devsecops, compliance and web app security solutions.

https://www.ssllabs.com/ssltest/

SSLTest output for Técnico



Home Projects Qualys.com Contact

You are here: Home > Projects > SSL Server Test > www.tecnico.ulisboa.pt > 2001:690:2100:1:0:0:c03a:216e

SSL Report: www.tecnico.ulisboa.pt (2001:690:2100:1:0:0:c03a:216e)

Summary

Overall Rating

B

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server accepts RC4 cipher, but only with older protocols. Grade capped to B. MORE INFO »

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

https://www.ssllabs.com/ssltest/analyze.html?d=tecnico.ulisboa.pt&s=193.136.128.169

Roadmap

- TLS – Transport Layer Security
- **SSH – Secure Shell**
- Key management

SSH

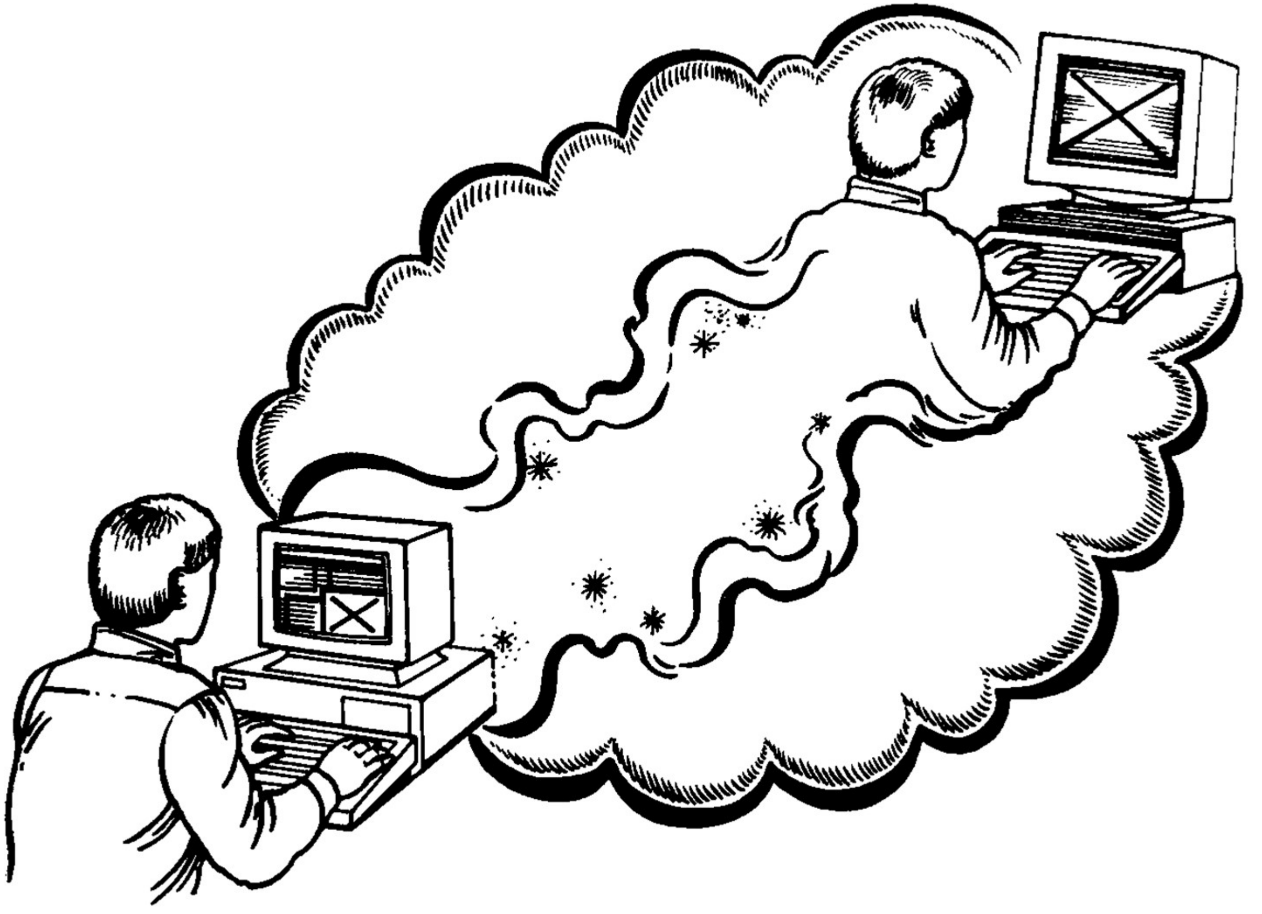


Figure 7.1 Remote login is a lot like astral projection.

SSH goals

- Secure communication application and protocol over TCP
  - Allows **secure remote sessions (~telnet)** and **file transfer (~ftp)**
  - Allows **tunneling** of TCP/IP traffic
- Security mechanisms
  - **Confidentiality** and **integrity** of the communication
  - **Distribution of keys**
  - Communicating parties' **authentication**
    - The server (or, usually, the server machine)
    - The user

SSH protocols (1/2)

- Transport Layer Protocol
  - Server authentication
    - Signature of the exchanged DH ephemeral values
    - Server public key can be transferred at this time, if not already held by the client

      ➔ **TOFU (Trust on First Use) model**
      - Vulnerable to man-in-the-middle
      - No certificates or CA
  - Distribution of keys
    - Diffie-Hellman key exchange
    - The session keys are computed with ephemeral DH values
  - Creation and analysis of secure messages
    - Compression, encryption, integrity control

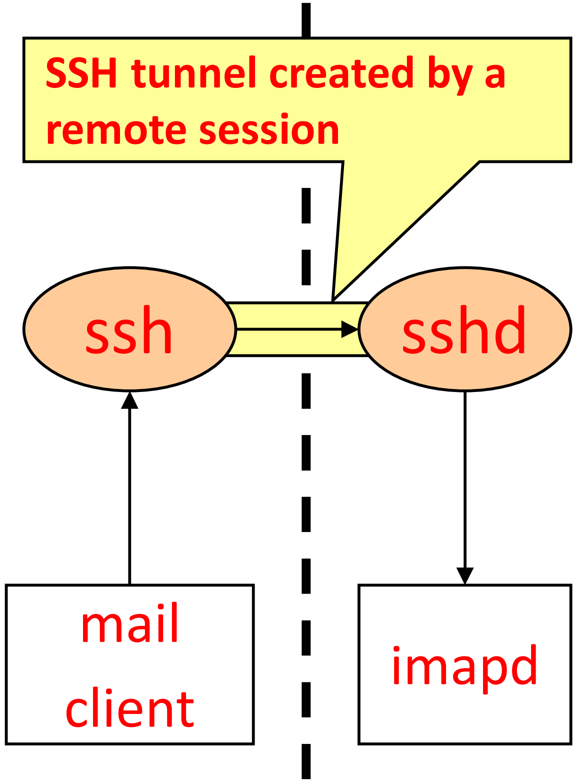
SSH protocols (2/2)

- User authentication towards the remote machine:
  - Password or
  - Signature with private key of client
    - Server knows the public key of client
- Connection Protocol
  - Information/data flow **multiplexing** over a secure session

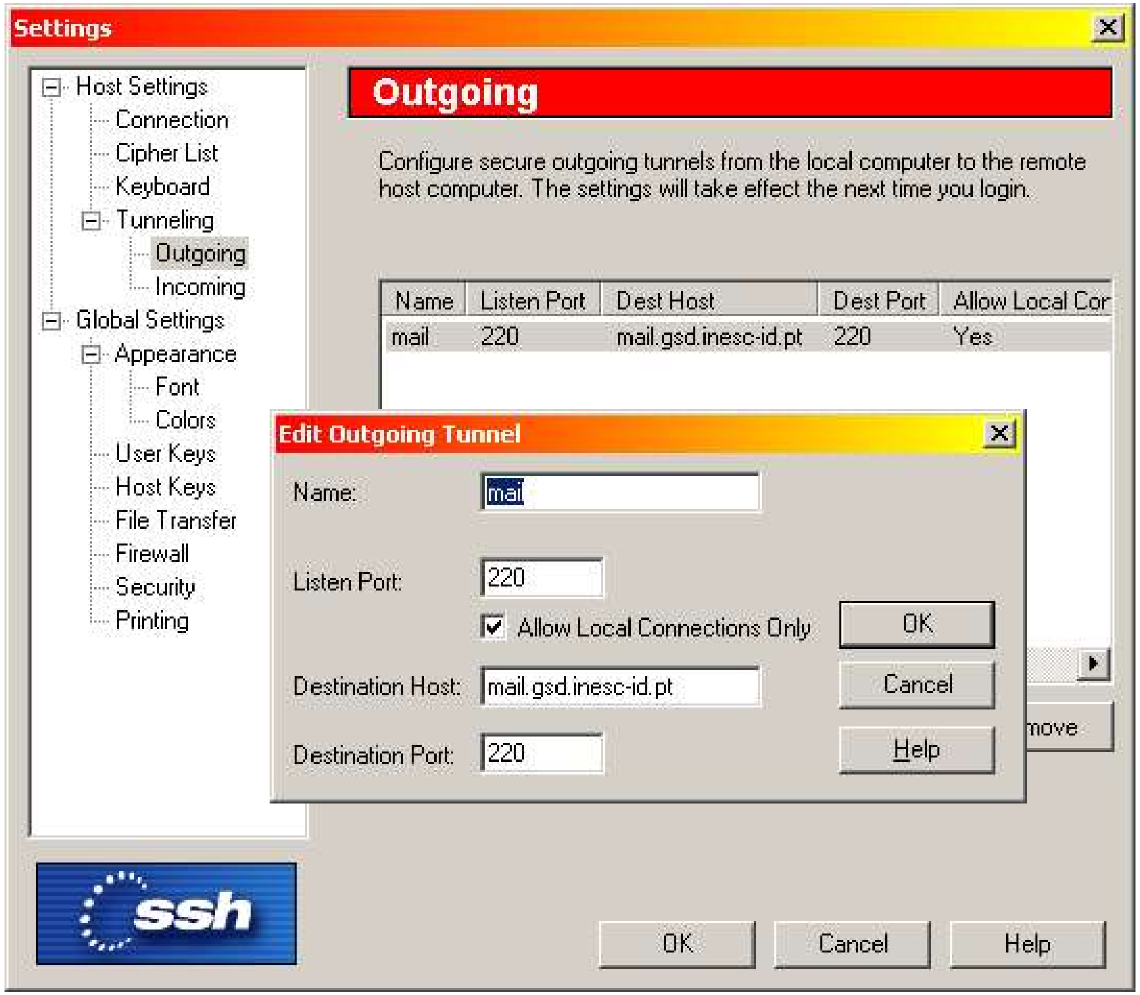
Tunneling with SSH

- In the client machine, a **mapping** is created between a **local TCP port** and a **port in the remote machine**
  - e.g., *localhost:IMAP* → *mail.myorg.pt:IMAP*
- **SSH session / tunnel** is created to mail.myorg.pt
- Client application is configured to use the **local port**
  - When using the port, it will securely interact via SSH with the remote server *mail.myorg.pt*
  - Client SSH and server SSHd operate as a secure relay mechanism

Configuration example of SSH tunnel for IMAP (port 220)



SSH tunnel created by a remote session



Outgoing

Configure secure outgoing tunnels from the local computer to the remote host computer. The settings will take effect the next time you login.

Name	Listen Port	Dest Host	Dest Port	Allow Local Cor
mail	220	mail.gsd.inesc-id.pt	220	Yes

Edit Outgoing Tunnel

Name: mail

Listen Port: 220

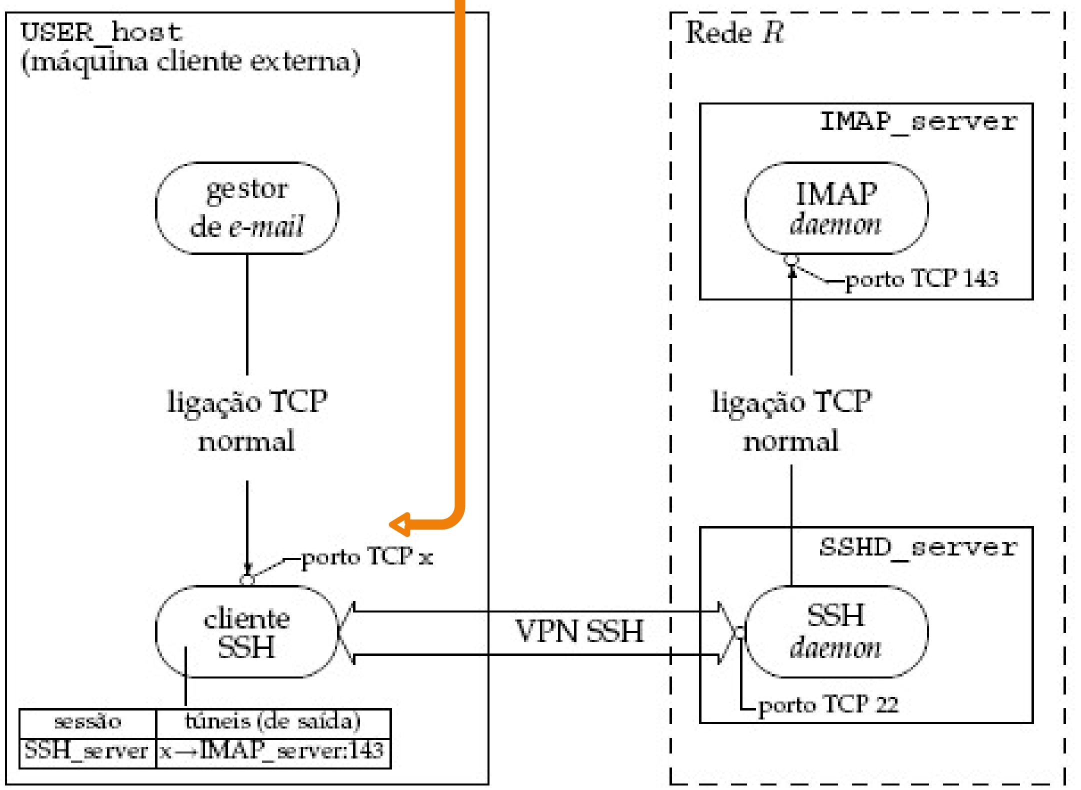
☑ Allow Local Connections Only

Destination Host: mail.gsd.inesc-id.pt

Destination Port: 220

Output tunnel

- TCP connections started at the client, so **outbound**
  - **SSH client** opens **port X** and waits for connections from client application
  - Similar situation to remote login

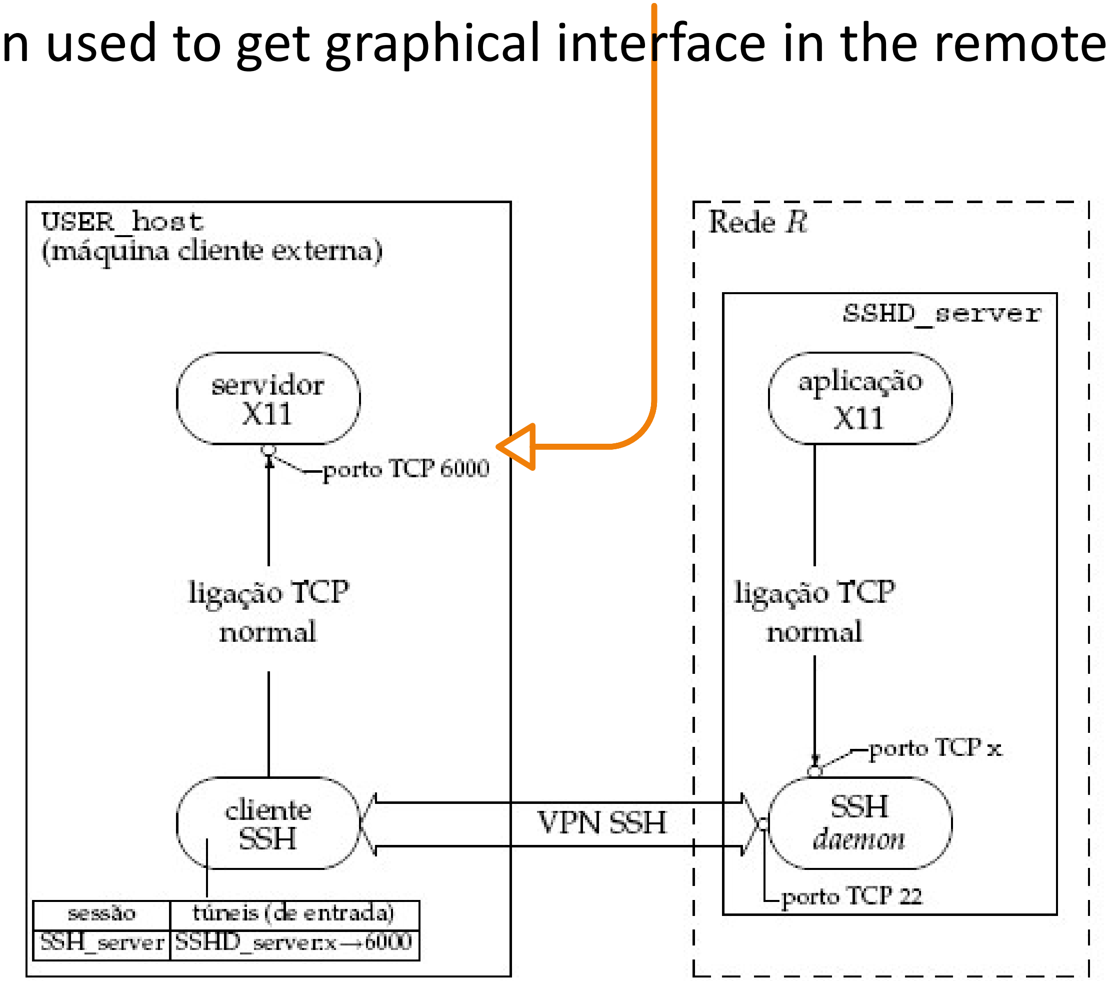


sessão	túneis (de saída)
SSH_server	x→IMAP_server:143

© André Zúquete

Input tunnel

- TCP connection started at the server, so **inbound**
  - **SSH client** connects to port in local server
  - Often used to get graphical interface in the remote host



© André Zúquete

Roadmap

- TLS – Transport Layer Security
- SSH – Secure Shell
- **Key management**

Keys

- TLS and SSH use **public-key cryptography**
  - (We will study this cryptography later, in more detail)
- **Pair of keys**
  - One is **private**, personal, non-transmissible
  - One is **public**, can/should be widely known
- Allow for
  - **Confidentiality** with the exchange of secret keys
  - **Authentication and Integrity** with digital signatures

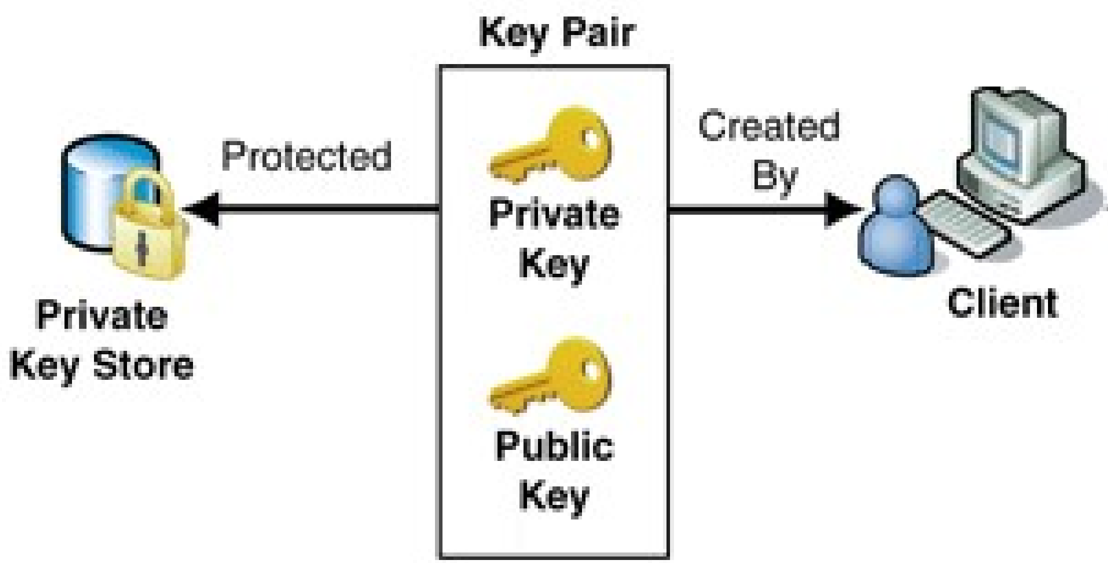
Private key

- The private key represents its owner so:
  - The probability of it being compromised must be minimized
  - Backup copies must be physically secure
- Private key must be protected
  - The access path to the private key must be restricted
  - Password protected, e.g., JKS, PGP
  - Security of applications using the private key must be guaranteed

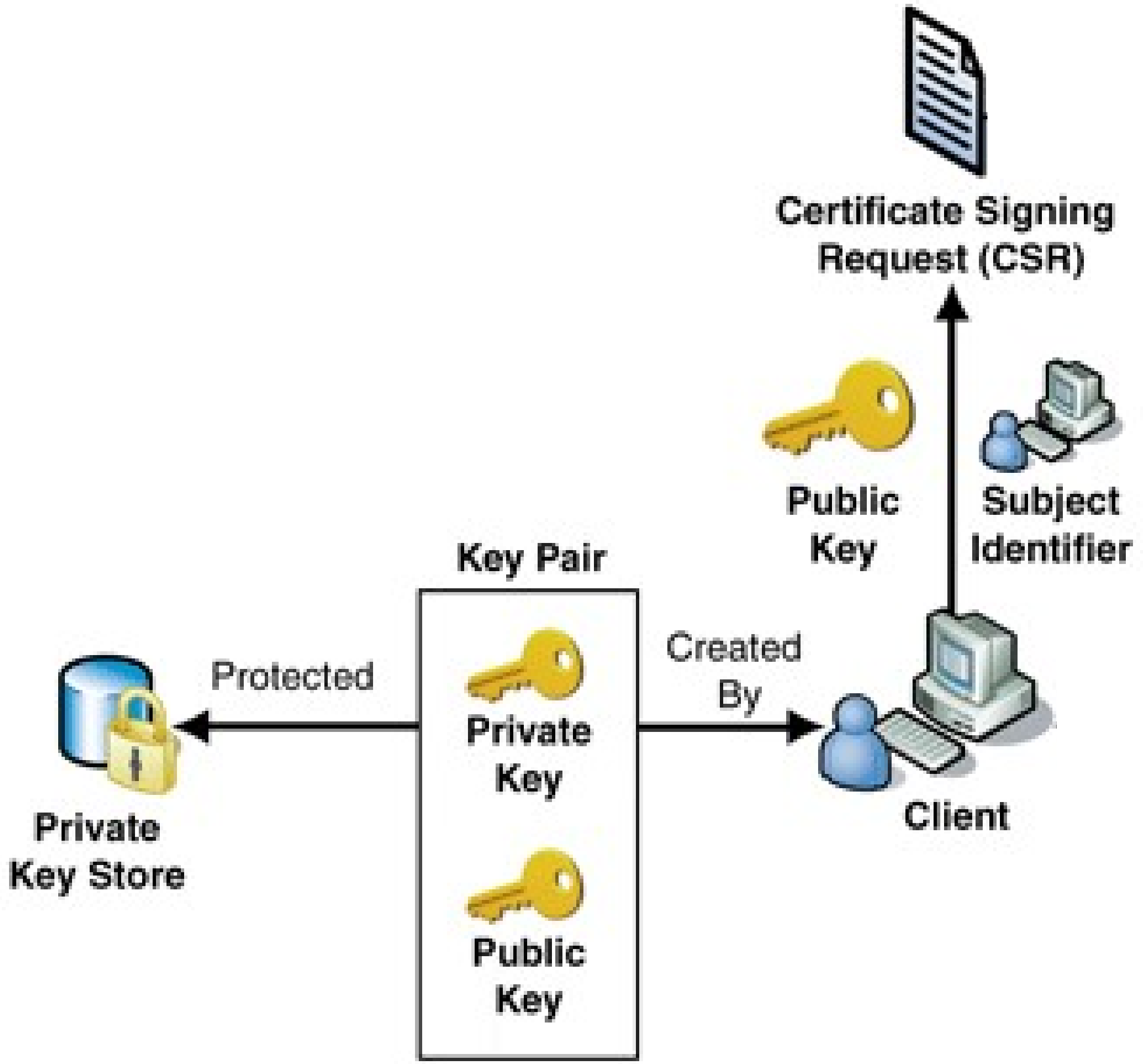
Public key certificate

- **Certificates** are documents signed by a certification entity
  - **Certification Authority (CA)**, public organization or company
  - Certificates are public documents
  - Certificates have a digital signature to assure authenticity
- Can distribute public key through unsecure channel
  - Receiver can validate the certificate signature using the CA public key
  - If it trusts the CA and the signature is valid, then it can trust the public key
- Certificate standard format: X.509 (RFC 3280)

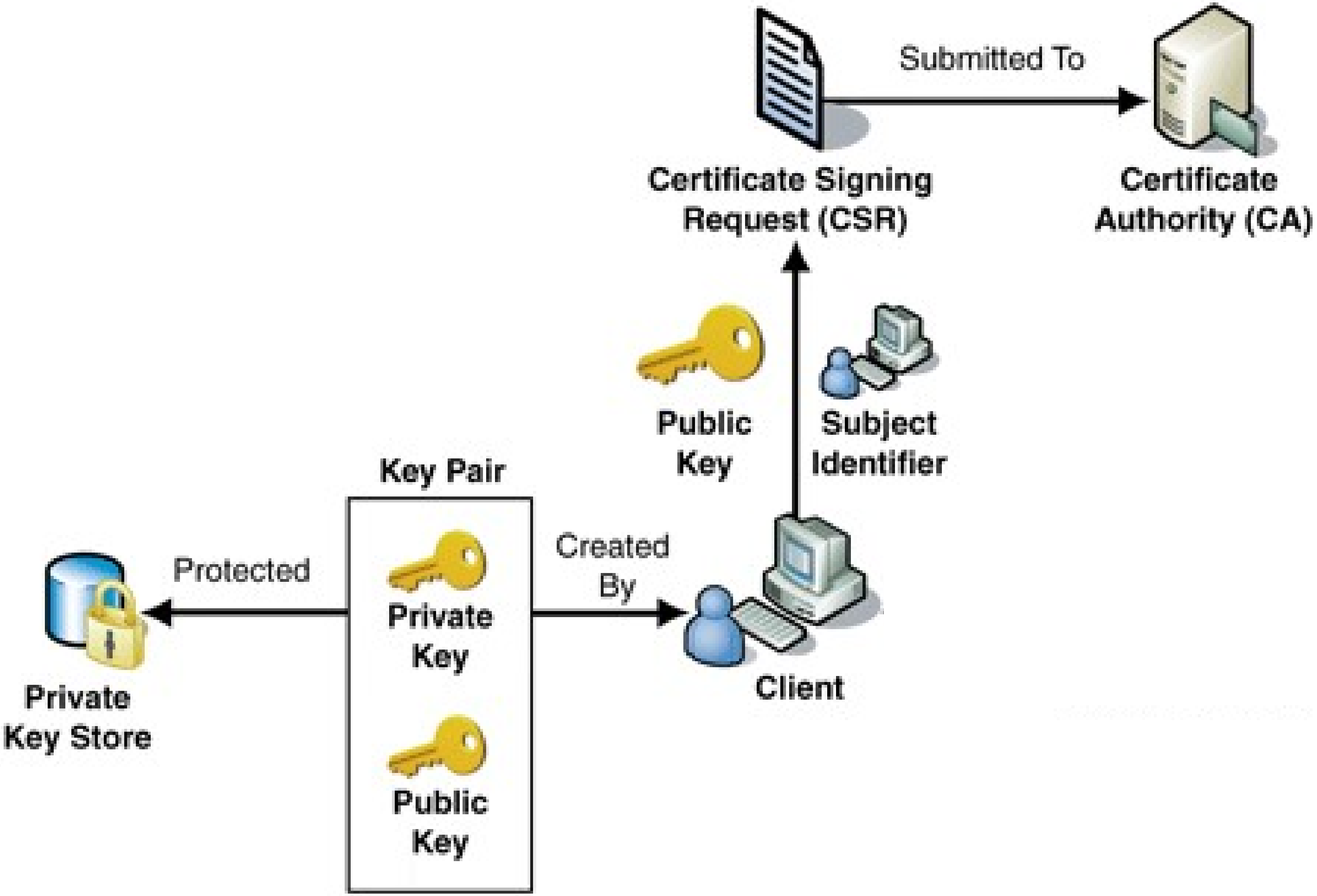
Certificate signing steps 1/4



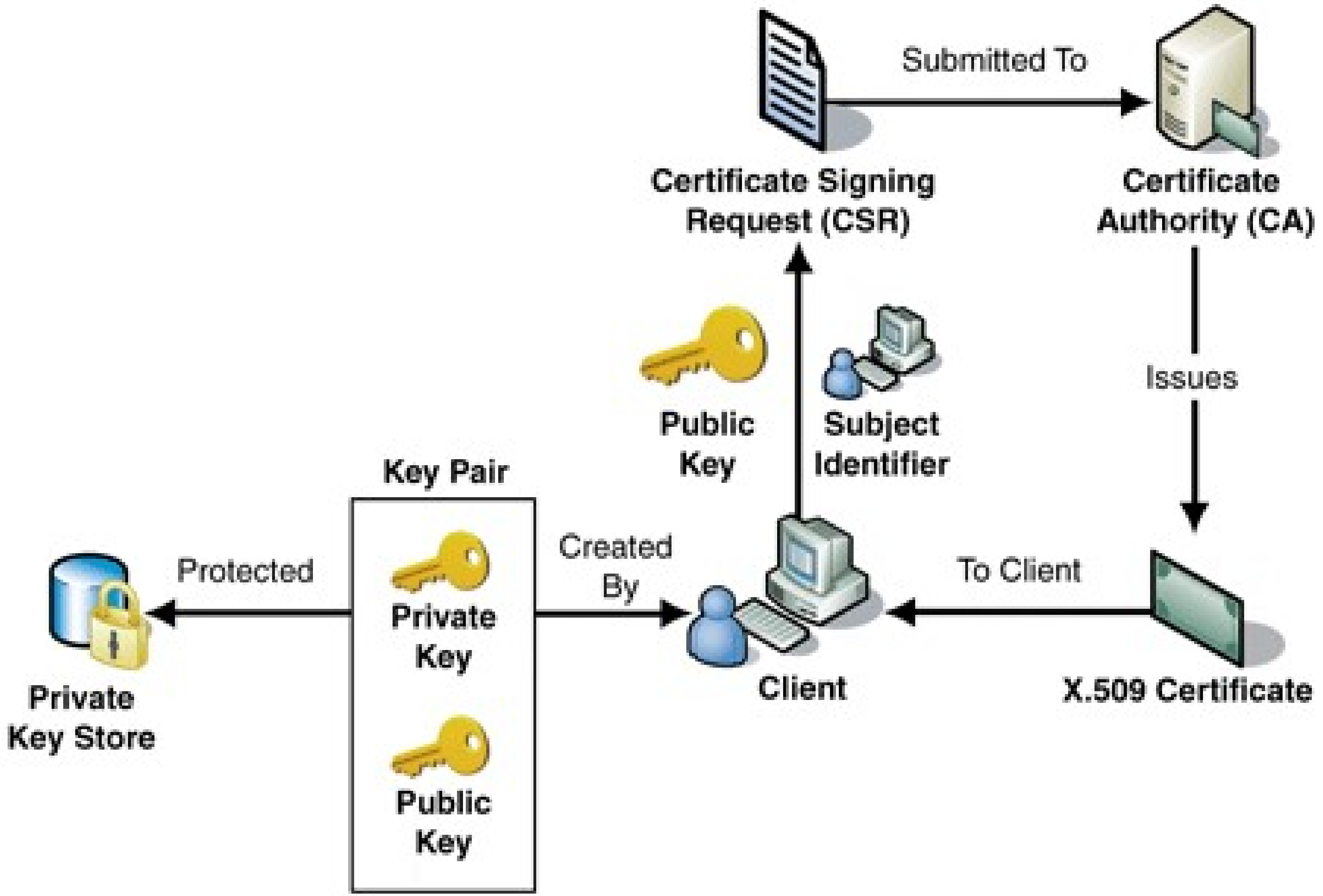
Certificate signing steps 2/4



Certificate signing steps 3/4



Certificate signing steps 4/4



Summary

- TLS – Transport Layer Security
- SSH – Secure Shell
- Key management