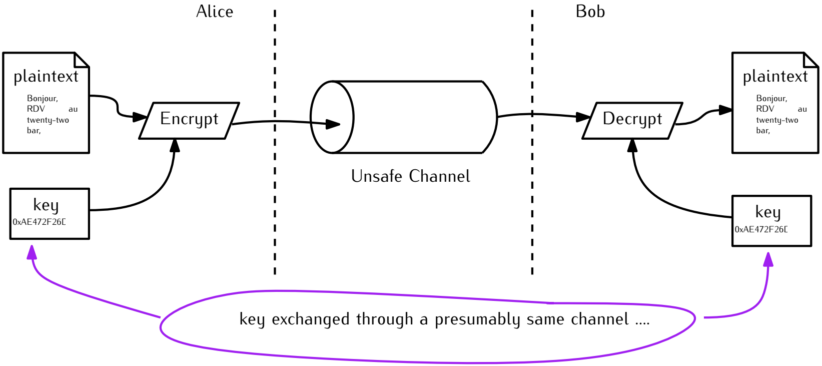
Secured Communication Protocols –Cryptographie et Sécurité des Communications–

Lionel Morel

Telecommunications - INSA Lyon

Fall-Winter 2021-22

Symmetric Cryptography



- Encryption/Decryption is cheap
- State-of-the-Art: AES
- Limitation: requires a key-sharing mechanism

Diffie-Hellman Key Exchange

Alice

let's agree on :
p = 23 (prime)
and g = 5
(primitive root modulo 23)

Bob

Alice chooses a secret key P_A

Bob chooses a secret key P_B

$g^{P_A} \bmod p$

$g^{P_B} \bmod p$

Alice computes
$(g^{P_B} \bmod p)^{P_A} \bmod p$
$= g^{P_B P_A} \bmod p$

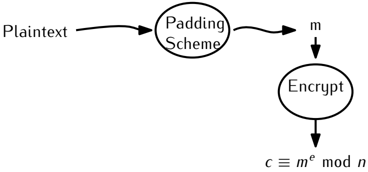
Bob computes
$(g^{P_A} \bmod p)^{P_B} \bmod p$
$= g^{P_A P_B} \bmod p$
$= g^{P_B P_A} \bmod p$

Alice and Bob now share a common key $g^{P_A P_B} \bmod p$

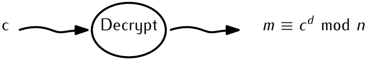
Asymmetric Cryptography

- Each participant builds a $(Pub_k, Priv_k)$ pair of keys

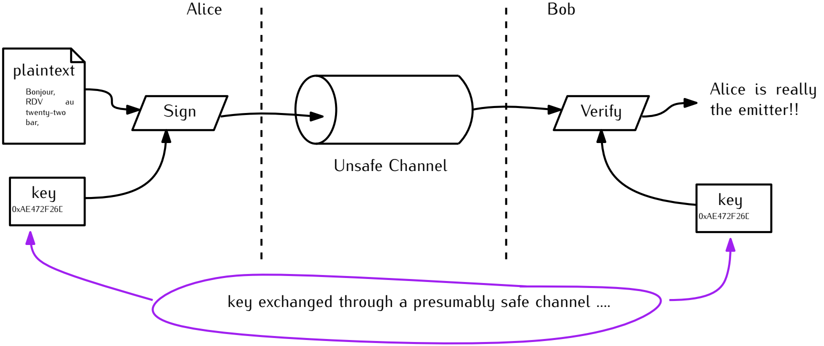
Encryption



Decryption



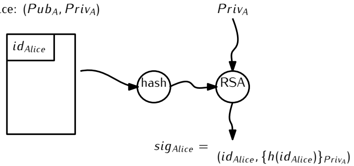
Message-Authentication-Codes (hashes)



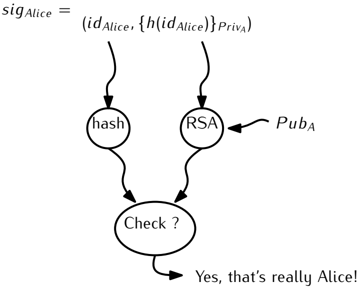
Self-signed Certificates

Sign

Alice: $(Pub_A, Priv_A)$



Verify



- Problem: Man-in-the-Middle

Man-in-the-Middle

- Bob and Alice want to communicate
- Bob $\longrightarrow \boxed{(B, Pub_B)} \longrightarrow$ Alice
- Hypothesis: Charlie can read and modify messages between Bob and Alice.
- Charlie falsifies Key definition of Bob:
  Bob $\longrightarrow \boxed{(B, Pub_B)} \longrightarrow$ Charlie $\longrightarrow \boxed{(B, Pub_C)} \longrightarrow$ Alice
  (he also keeps Pub_B for later)
- Now when Alice writes to Bob, she actually uses Charlie's public key
- Alice $\longrightarrow \boxed{\{m\}_{Pub_C}} \longrightarrow$ Bob
- Charlie can then eavesdrop all messages:
  Alice $\longrightarrow \boxed{\{m\}_{Pub_C}} \longrightarrow$ Charlie $\rightarrow \{\{m\}_{Pub_C}\}_{Priv_C} = m \longrightarrow$
  $\boxed{\{m\}_{Pub_B}} \longrightarrow$ Bob

Public-Key Infrastructure and Certificate Authorities

A PKI consists of:

- A **Certificate Authority** (CA) - stores, issues and signs digital certificates
- A **Registration Authority** (RA) - verifies identity of entities requesting their certificates to be stored at the CA.
- A **Central Directory** - secure location to store keys
- CAs are “Trusted Third Parties”

Certificate Authorities

A
$[Pub_A, Priv_A]$
Pub_C

Init State

- Each participant has its own set of pub/priv keys
- Everybody has C's public key

C (Certificate Authority) $[Pub_C, Priv_C]$

B (insa-lyon.fr)
$[Pub_B, Priv_B]$
Pub_C

Certificate Authorities

A
[$Pub_A, Priv_A$]
Pub_C

Init State

- Each participant has its own set of pub/priv keys
- Everybody has C's public key

C (Certificate Authority) [$Pub_C, Priv_C$]

1. B registers itself to C:
 - (id_B, Pub_B)

B (insa-lyon.fr)
[$Pub_B, Priv_B$]
Pub_C

Certificate Authorities

A
$[Pub_A, Priv_A]$
Pub_C

Init State

- Each participant has its own set of pub/priv keys
- Everybody has C's public key

C (Certificate Authority) $[Pub_C, Priv_C]$
2. C computes:
  - $(id_B, Pub_B).\{h((id_B, Pub_B))\}_{Priv_C}$

1. B registers itself to C:
  - (id_B, Pub_B)

B (insa-lyon.fr)
$[Pub_B, Priv_B]$
Pub_C

Certificate Authorities

A
$[Pub_A, Priv_A]$
Pub_C

Init State

- Each participant has its own set of pub/priv keys
- Everybody has C's public key

C (Certificate Authority) $[Pub_C, Priv_C]$
2. C computes:
   - $(id_B, Pub_B).\{h((id_B, Pub_B))\}_{Priv_C}$

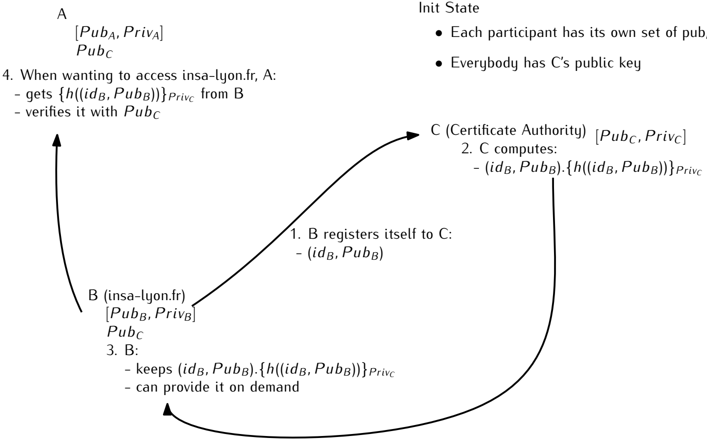
1. B registers itself to C:
   - (id_B, Pub_B)

B (insa-lyon.fr)
$[Pub_B, Priv_B]$
Pub_C
3. B:
   - keeps $(id_B, Pub_B).\{h((id_B, Pub_B))\}_{Priv_C}$
   - can provide it on demand

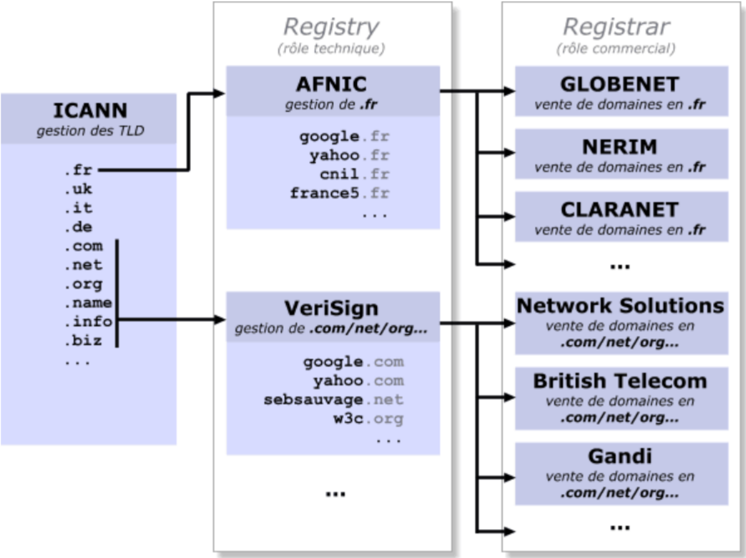
Certificate Authorities

Init State

- Each participant has its own set of pub/priv keys
- Everybody has C's public key



DNS

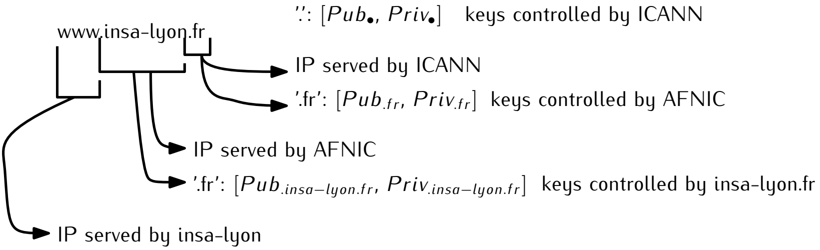


- The **ICANN** (Internet Corporation for Assigned Names and Numbers) manages a list of **Top-Level Domains**

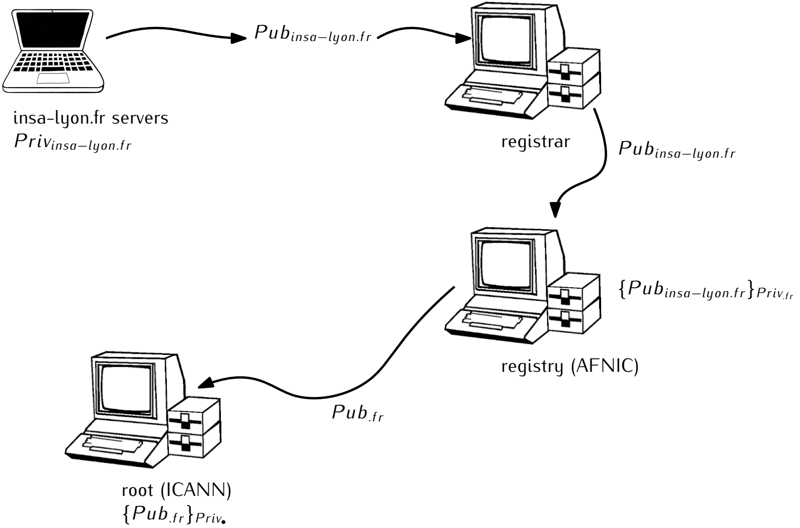
```
whois insa-lyon.fr
%%
%% This is the AFNIC Whois server.
%%
%% complete date format : YYYY-MM-DDThh:mm:ssZ
%% short date format    : DD/MM
%% version              : FRNIC-2.5
%%
%% Rights restricted by copyright.
%% See https://www.afnic.fr/en/products-and-services/services/whois/whois-special-notice/
%%
%% Use '-h' option to obtain more information about this service.
%%
%% [77.134.1.180 REQUEST] >> -V Md5.5.10 insa-lyon.fr
%%
%% RL Net [##########] - RL IP [#########.]
%%

domain:      insa-lyon.fr
status:      ACTIVE
hold:        NO
holder-c:    INSA12-FRNIC
admin-c:     SJ7971-FRNIC
tech-c:      GRST1-FRNIC
tech-c:      LM19215-FRNIC
tech-c:      TP630-FRNIC
zone-c:      NFC1-FRNIC
nsl-id:      NSL1519-FRNIC
registrar:   GIP RENATER
Expiry Date: 2022-12-31T23:00:00Z
created:     1994-12-31T23:00:00Z
last-update: 2021-12-31T23:36:35Z
source:      FRNIC
```

```
ns-list:     NSL1519-FRNIC
nserver:     dns.univ-lyon1.fr
nserver:     dns2.univ-lyon1.fr
source:      FRNIC

registrar:   GIP RENATER
type:        Isp Option 1
address:     Array
address:     75013 PARIS
country:     FR
phone:       +33 1 53 94 20 30
fax-no:      +33 1 53 94 20 31
e-mail:      domaine@renater.fr
website:     http://www.renater.fr
anonymous:   NO
registered:  1998-01-01T12:00:00Z
source:      FRNIC

nic-hdl:     INSA12-FRNIC
type:        ORGANIZATION
contact:     INSTITUT NAT SCIENCES APPLIQUEES LYON
address:     INSA LYON
address:     20, avenue Albert Einstein
address:     69621 Villeurbanne
country:     FR
phone:       +33 4 72 43 81 14
fax-no:      +33 4 72 43 85 00
e-mail:      webmaster@insa-lyon.fr
registrar:   GIP RENATER
changed:     2016-06-07T11:59:36Z nic@nic.fr
anonymous:   NO
obsoleted:   NO
eligstatus:  not identified
reachstatus: not identified
source:      FRNIC
```

DNSSEC key management (1)



DNSSEC key management (2)



TLS

- History:
  - SSL proposed in 1995 by Netscape (RIP)
  - TLS proposed by the IETF, starting 1999
  - Current version 1.3 (2018)
- TLS stands for Transport Layer Security
- It supersedes SSL
- Works on top of TCP/IP
- Application-independent, eg: HTTPs = HTTP+TLS, IMAPS = IMAP+TLS

TLS (cont'd)

Confidentiality

- Exchanged data is encrypted using **Symmetric-key cryptography**
- Encryption keys are re-newed for every session, ie TLS uses **session keys**

Authenticity

- Identity of communicating parties is authenticated using **Public-key cryptography**

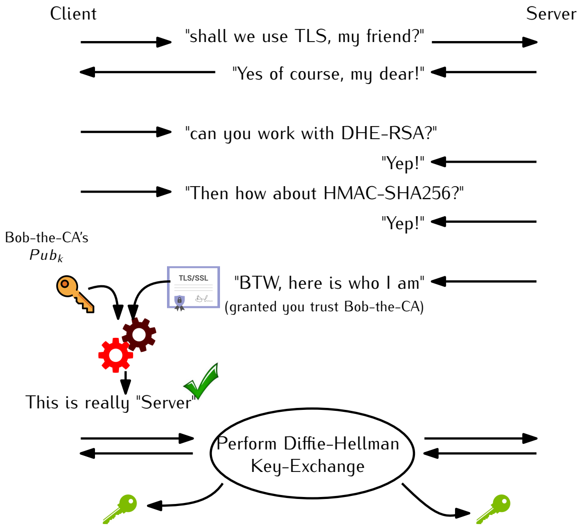
Integrity

- Each message includes a **Message Authentication Code** to prevent alteration

TLS (cont'd)

- Client and server **agree to use TLS**
- They perform a **handshake procedure** together (see next)
- Out of this handshake, they get (secret) encryption keys they can then use to perform **symmetric encryption**

TLS handshake

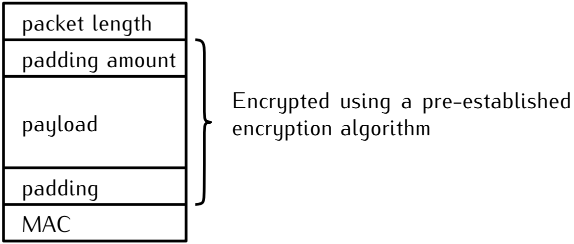


SSH

- SSH stands for Secured Shell
- Provides a way to connect (shell) to a distant computer while encrypting all data exchanged
- Before SSH:
  - data was sent unencrypted over the network
  - not so bad as internet was not there :)
- 1995: Tatu Ylönen proposed a secured version of a remote shell

Sending Data Through SSH

- Consider a TCP connexion between 2 machines
- SSH breaks data into a series of packets



Algorithms available

- EdDSA, ECDSA, RSA and DSA for public-key cryptography.
- ECDH and Diffie–Hellman for key exchange.
- HMAC, AEAD and UMAC for MAC.
- AES (and deprecated RC4, 3DES, DES[29]) for symmetric encryption.
- AES-GCM] and ChaCha20-Poly1305 for AEAD encryption.
- SHA (and deprecated MD5) for key fingerprint.

SSH properties

- Channel multiplexing: You can open a number of different channels to send different data to/from different partners
- Tunneling: on can redirect a (eg) TCP flow into an ssh tunnel
- Distant shell
- File transfer
- Port redirection

SSH Authentication

Client-side

- password
- public/private key

Server-side

- at 1st connection, server stores client's key footprint
- for follow-up connection, server checks footprint
- if key is invalid: connection refused

NB

No guarantee on 1st connection $\Longrightarrow$ "*opportunistic security*"

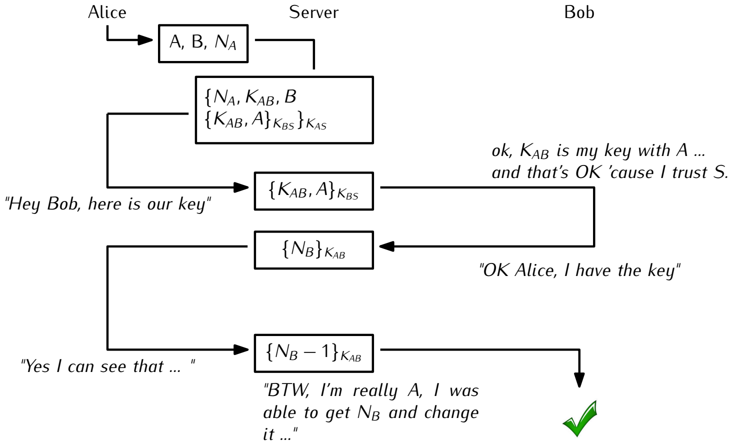
OpenSSH

OpenSSH is a software suite that includes command-line utilities and daemons:

- `scp` (replacement for `rcp`) to copy files through encryted channel
- `sftp`, replacement for `ftp`
- `ssh` itself
- `ssh-agent` to hold keys and ease authentication
- `ssh-keygen` to generate RSA, DSA or elliptic-curve keys
- `sshd` the ssh server daemon

Needham-Schroeder Symmetric Protocol (1978)

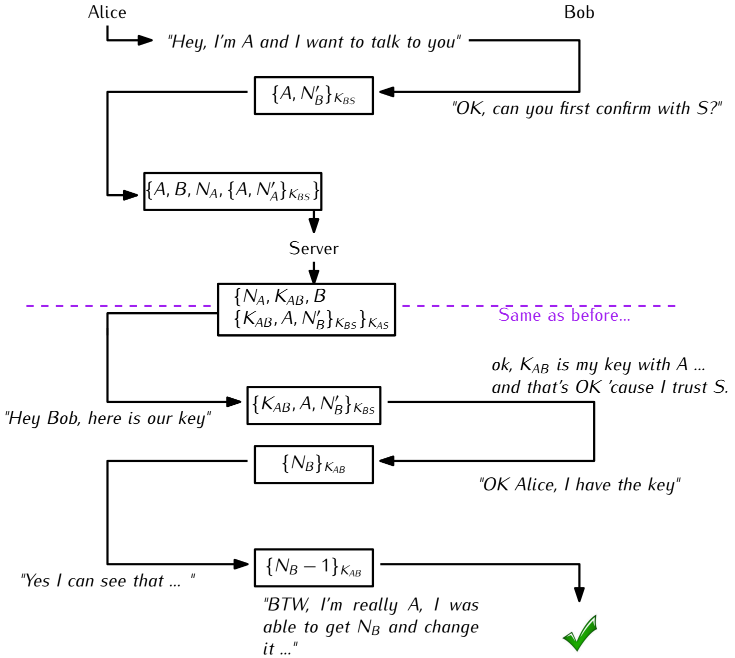
- A and B want to talk
- A (resp B) has a private key with server S, k_{AS} (resp k_{BS})



NB: Vulnerable to replay attack:

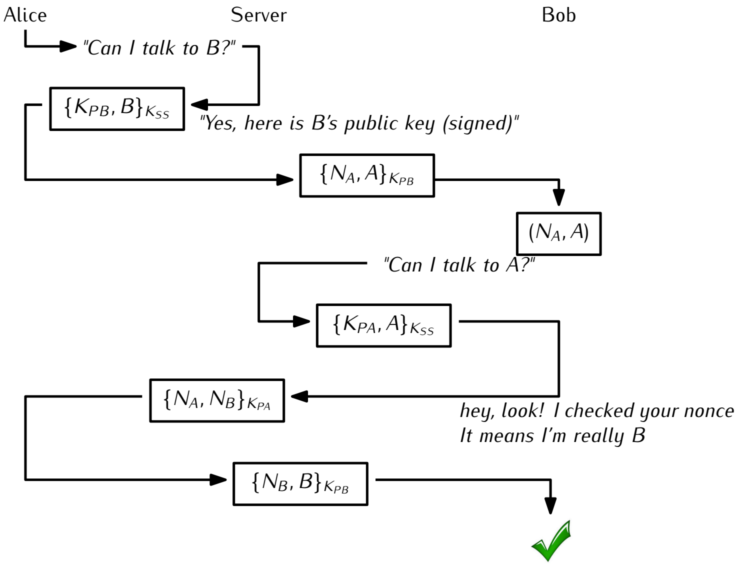
- Charlie gets an old compromised key K_{AB} and pretends to be A by sending $\{K_{AB}, A\}_{K_{BS}}$ to B

Needham-Schroeder: fixing the Replay Attack



Needham-Schroeder Public-Key (Asymetric) Protocol

A: (k_{PA}, k_{SA}) B: (k_{PB}, k_{SB}) S: (k_{PS}, k_{SS})

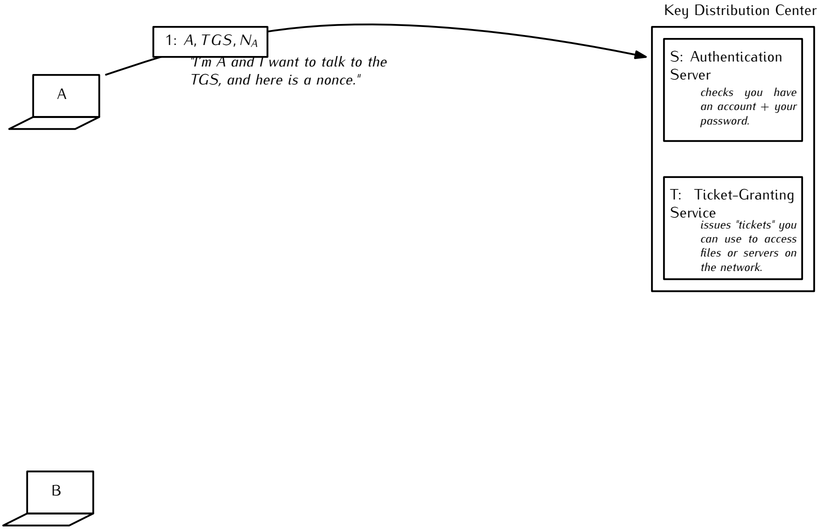


Kerberos

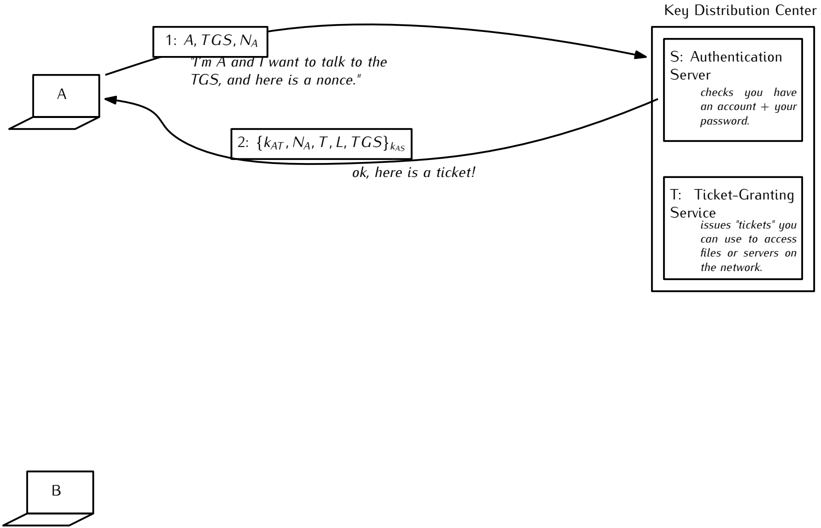
- Active Directory's main authentication protocol
- No public key (symmetric keys only)
- How to share keys ?
  - No key exchange protocol (eg Diffie-Hellman)
  - No certificate verification
- Idea: Use passwords and long-term keys to derive symmetric session keys
- Trust a server to provide session keys
- The connection with the server relies on pre-established long-term keys

Kerberos architecture (1)

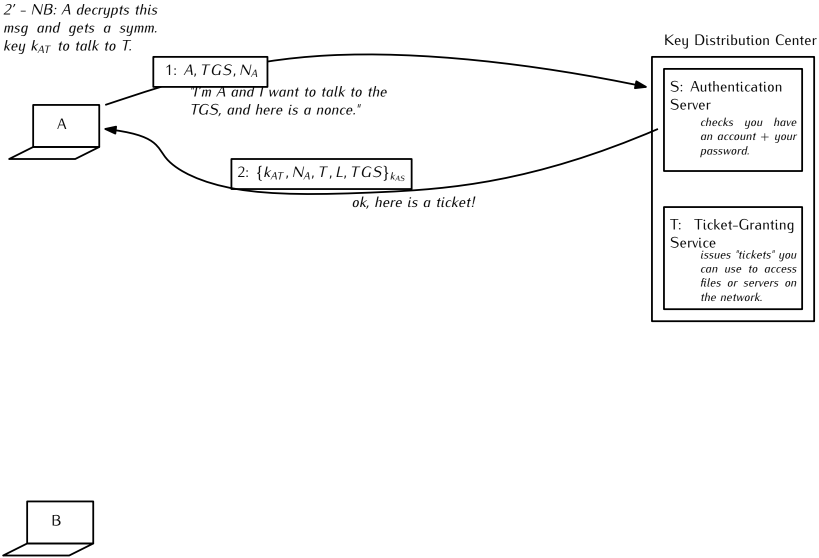
Kerberos architecture (1)



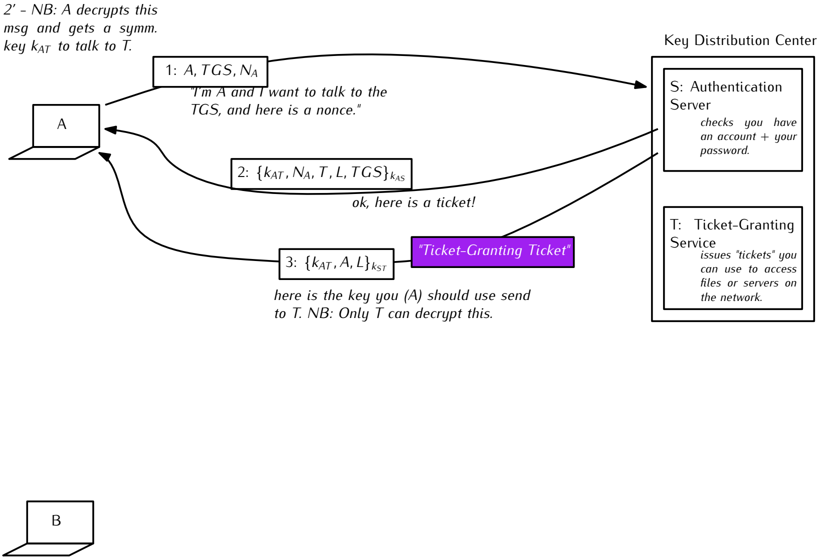
Kerberos architecture (1)



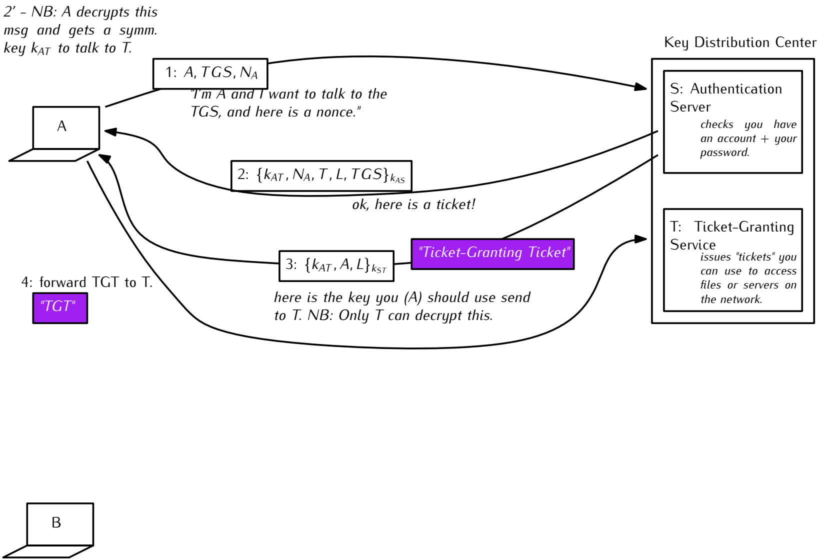
Kerberos architecture (1)



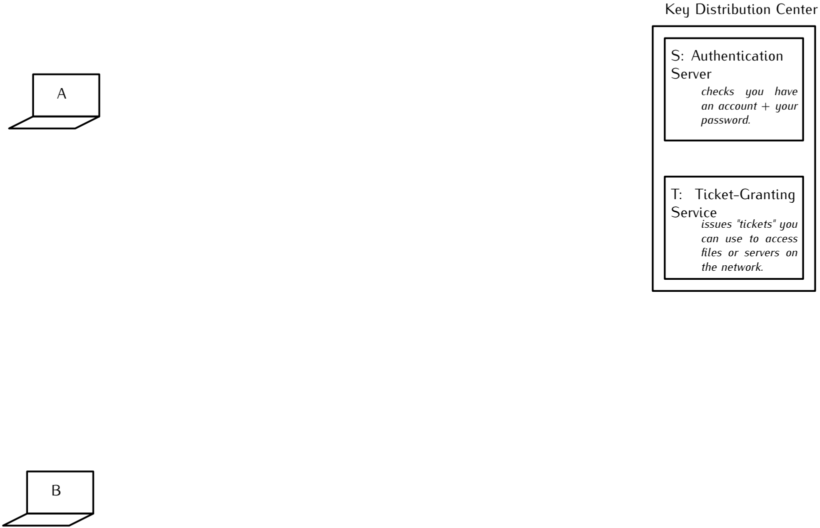
Kerberos architecture (1)



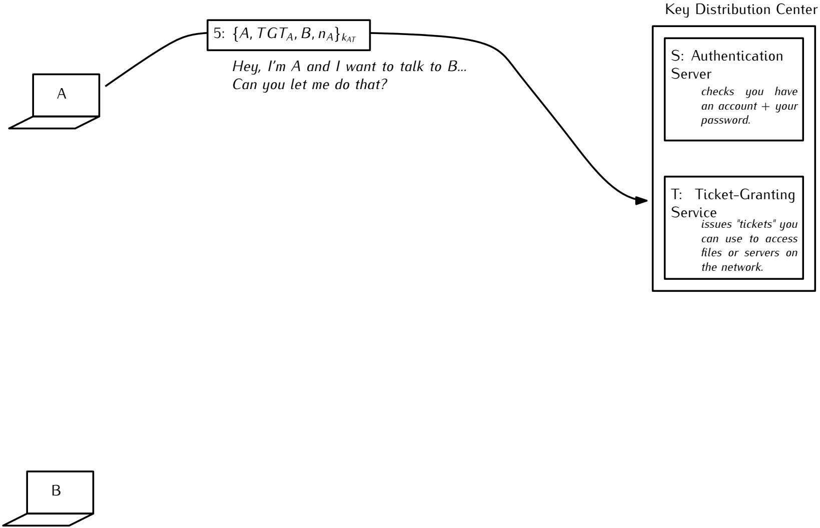
Kerberos architecture (1)



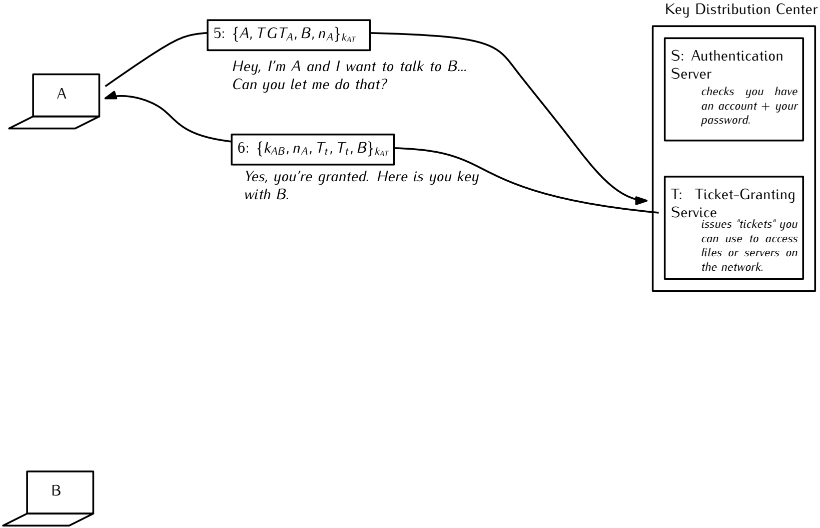
Kerberos architecture (2)



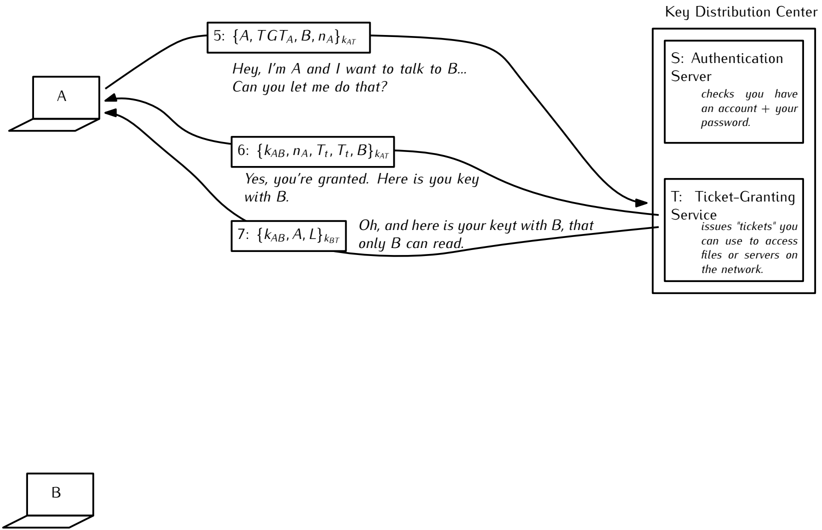
Kerberos architecture (2)



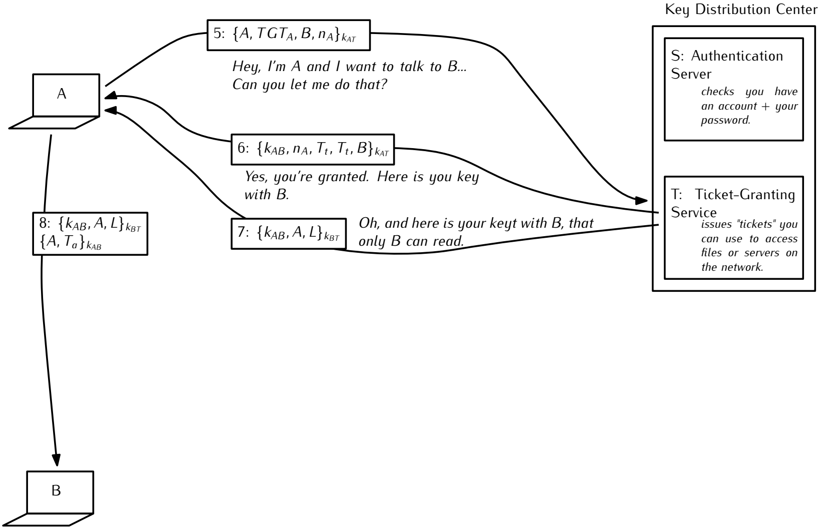
Kerberos architecture (2)



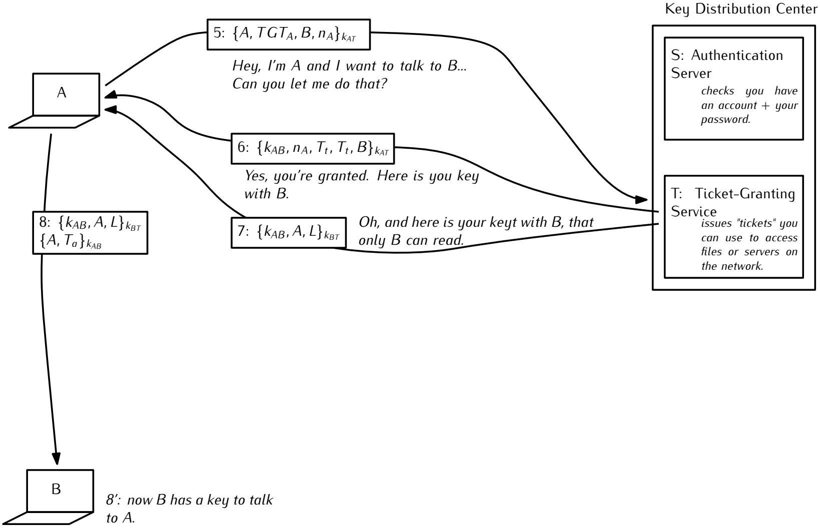
Kerberos architecture (2)



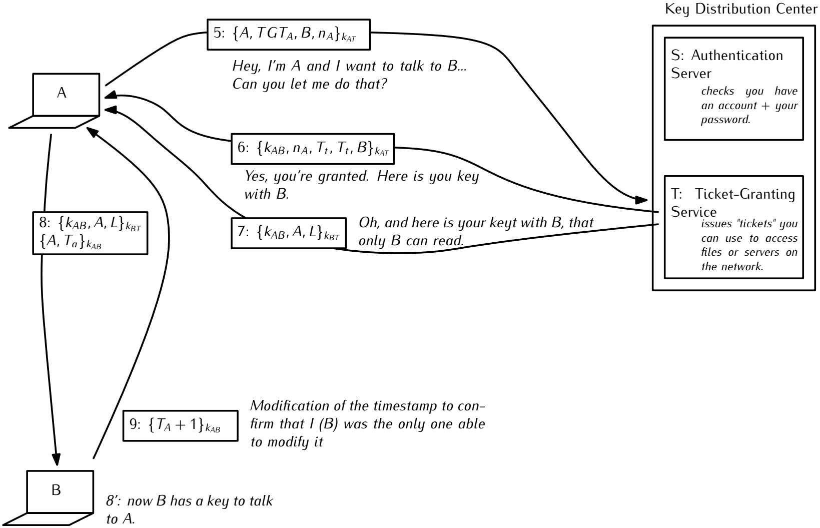
Kerberos architecture (2)



Kerberos architecture (2)



Kerberos architecture (2)



Conclusion (1/2)

- Today, cryptography is ahead of (civilian) cryptanalysis: that has not always been the case (and may not be the case in the future)
- Cryptography is a complex world
- It's based on math!
- The security of cryptographic algorithms relies on a great number of implementation details
- Cryptography is useful only if used ...
  - correctly
  - wisely

Conclusion (2/2)

Your living minimum is to:

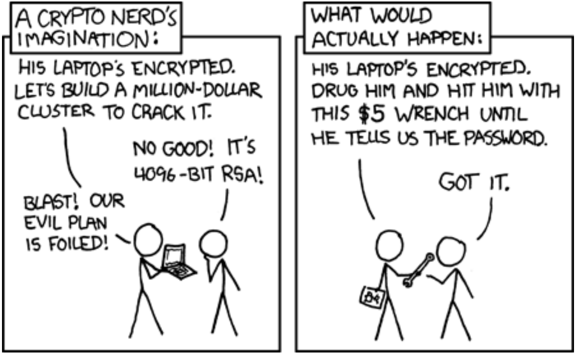
- Know what cryptography can and cannot do for you
- Know when cryptography is adequate
- Use known libraries (eg OpenSSL) and not your own home-made reimplementations

Key management is the crux

- Without the correct key, cryptography is useless!
- With the correct key, cryptography only garanties the security of the connexion (not outsiders' truthfulness). . .)

- The whole security of internet relies on encryption keys
- Good, that was the goal
- But key management is still a major issue

In practice. . .



http://xkcd.com/538/