

Security protocols

Topics

- Informal descriptions of some protocols and of their properties.
- Some design principles.
- Some methods for protocol analysis.

Security protocols

Security protocols are concerned with properties such as integrity and secrecy.

- Primary examples are protocols that establish communication channels with authentication and confidentiality.
- Other examples include protocols for commerce and electronic voting.

In distributed systems, security protocols invariably rely on cryptography.

They have to be right, but they often aren't.

Communication with public-key cryptography

To send a message confidentially,

- the sender encrypts it with the recipient's public encryption key,
- the recipient decrypts it with its secret decryption key.

To send a message with integrity,

- the sender signs it with its secret signature key,
- the recipient checks it with the corresponding public key.

Note:

- For both, sign before encrypting.
- Encryption keys and signature keys should be different.

Communication with shared-key cryptography

If the sender and recipient share a key K :

To send a message confidentially,

- the sender encrypts it with K ,
- the recipient decrypts it with K .

To send a message with integrity,

- the sender includes a MAC with K ,
- the recipient checks the MAC with K .

Note:

- For both, the order matters, but is less clear.
- Encryption keys and MAC keys should be different.
- Each direction of communication may have its own keys.

Another twist

Informal descriptions often assume that encryption guarantees integrity.

In practice, encryption often does guarantee integrity (in particular, because of plaintext redundancies).

In theory, there is much progress on authenticated encryption schemes.

Complications

We may need:

- key identifiers,
- timestamps,
- sequence numbers,
- various other tags,
- compression,
- padding.

Problems

Remaining issues:

- performance
(particularly for public-key methods),
- associating keys with principals
(particularly for shared-key methods),
- access control, auditing, . . .

Authentication protocols (or channel-establishment protocols)

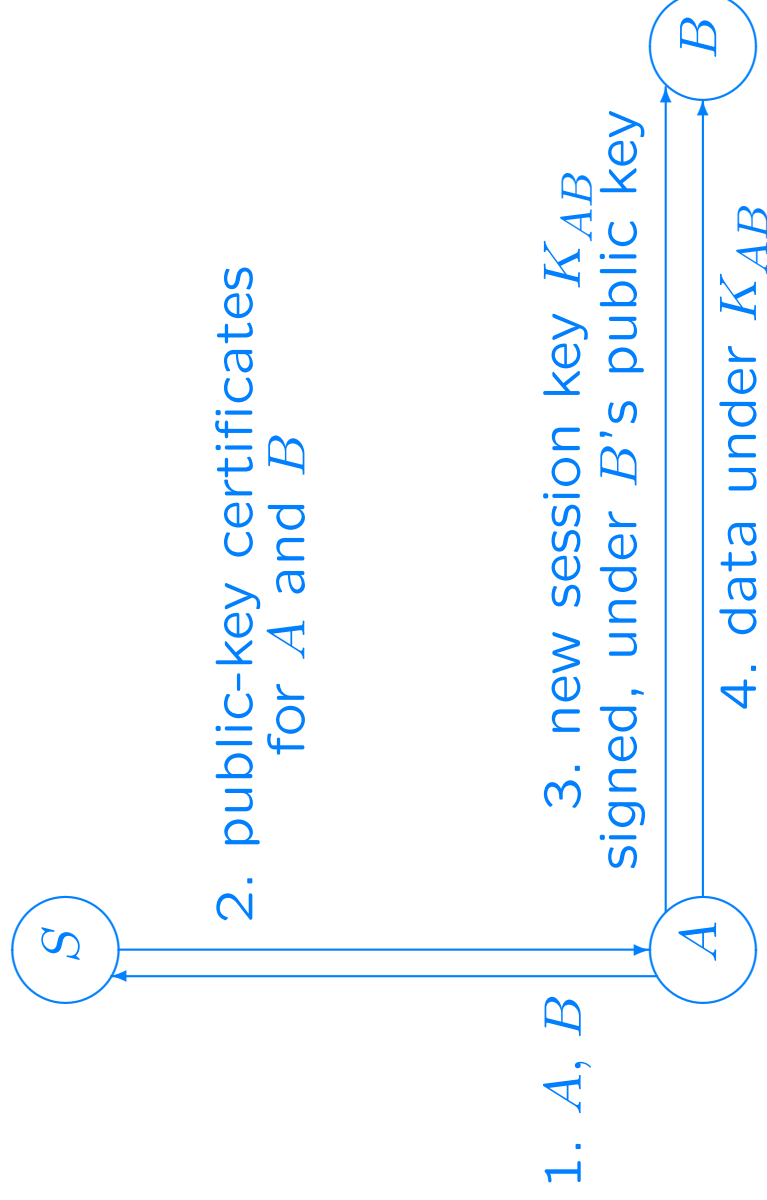
There are many authentication protocols.

They typically involve:

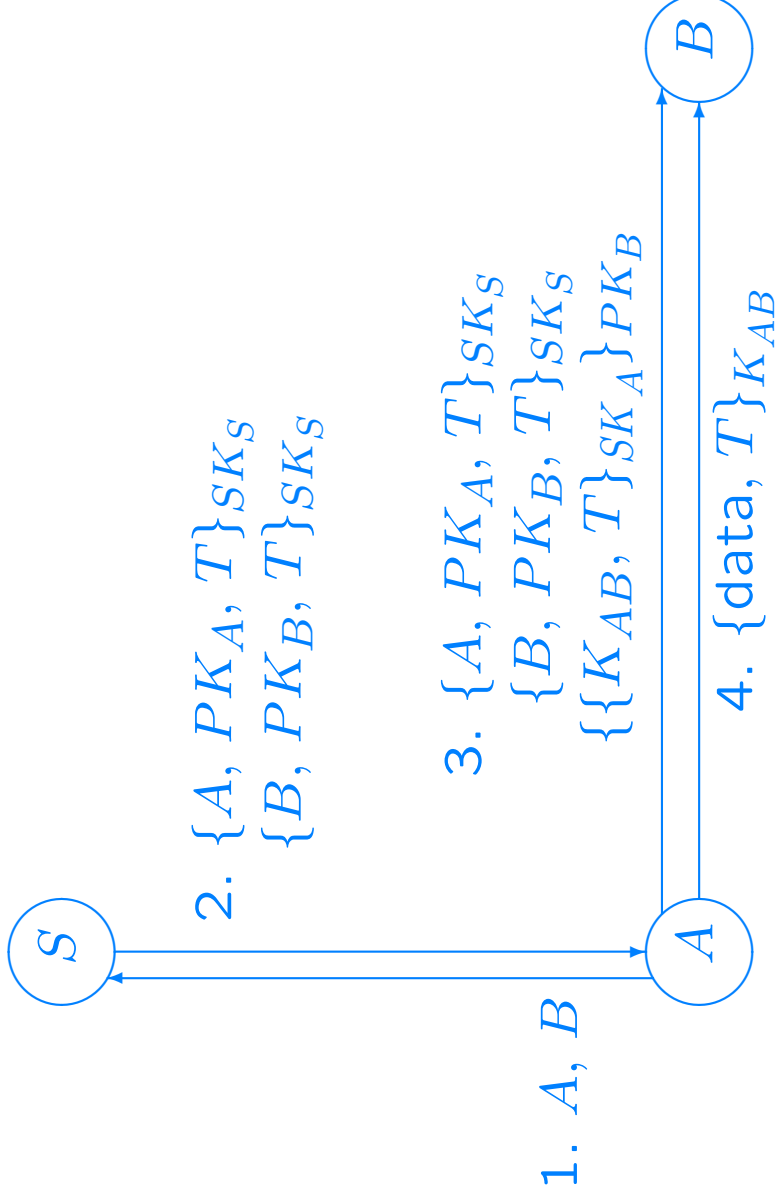
- two principals
 - hosts, users, services
- secrets (possibly shared)
 - usually encryption keys
- encryption
 - shared-key or public-key
- trusted servers
- proofs of timeliness
 - nonces, timestamps

A typical protocol

- Each principal has a public-key pair.
- S is a **certification authority** for public keys.
- A and B agree on a **session key** K_{AB} and send data under K_{AB} :



The Denning-Sacco protocol



$\{X\}_{PK_B}$: X encrypted with B 's public key, for secrecy

$\{X\}_{SK_A}$: X signed with A 's secret key, for integrity

T : a timestamp

Questions

Does the protocol work?

What assumptions are we making?

Can we do better?

No single protocol will do.

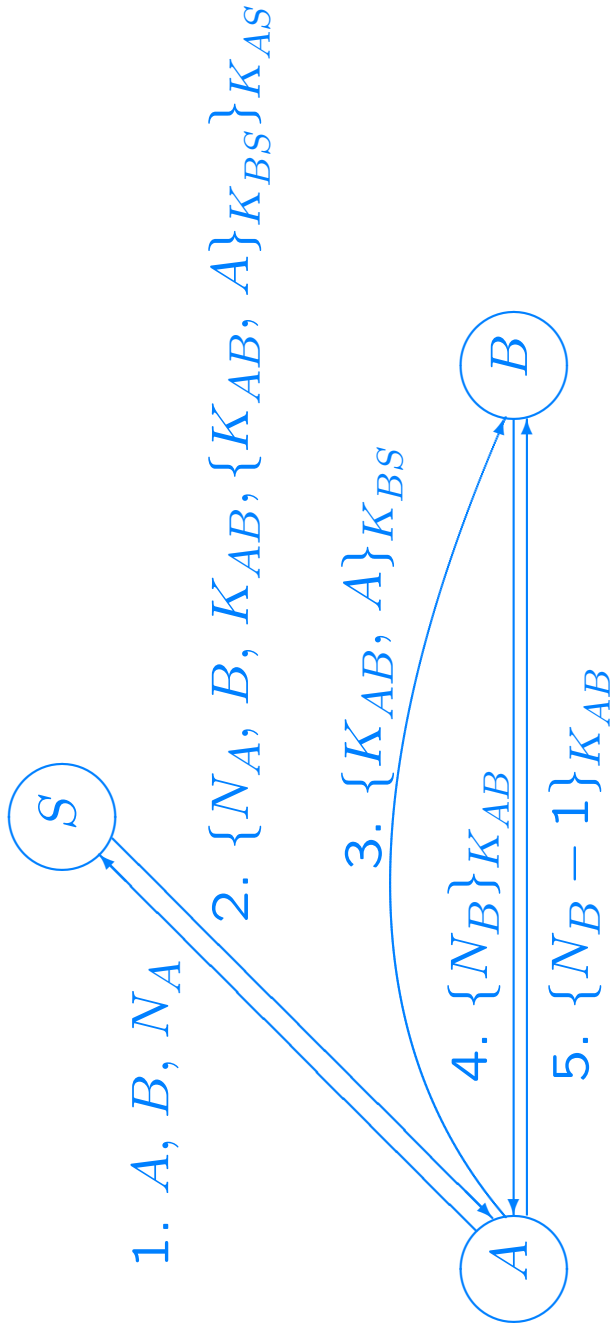
We may want:

- few messages,
- little encryption,
- little trust of other machines.

and also:

- asynchronous checking (for e-mail),
- different cryptosystems,
- little server state,
- one-way or two-way authentication,
- client anonymity.

The Needham-Schroeder shared-key protocol



$\{X\}_K$: X encrypted with the shared key K

N_A, N_B : nonces

K_{AB} : a session key for A and B , for encrypting subsequent communications

A criticism

Long after a run, an attacker may

- discover K_{AB} ,
- replay $\{K_{AB}, A\}_{K_{BS}}$ to B ,
- conduct a handshake with B ,
- send arbitrary data to B under K_{AB} ,
impersonating A . (Denning & Sacco)

Solutions:

- make K_{AB} strong, change K_{BS} often
- let B and S interact (Needham & Schroeder),
- use timestamps (Denning & Sacco, Kerberos).

Some observations

Most security protocols have subtleties and flaws.

Many of these have to do with cryptography.

Many of these don't have to do with the details of cryptography.

For design, implementation, and analysis, a fairly abstract view of cryptography is often practical.

Needham-Schroeder descendants

Initially, the Kerberos protocol was based on the Needham-Schroeder shared-key protocol plus timestamps. (And an early version did not even check the timestamps properly!)

Over time, it incorporated improvements and extensions.

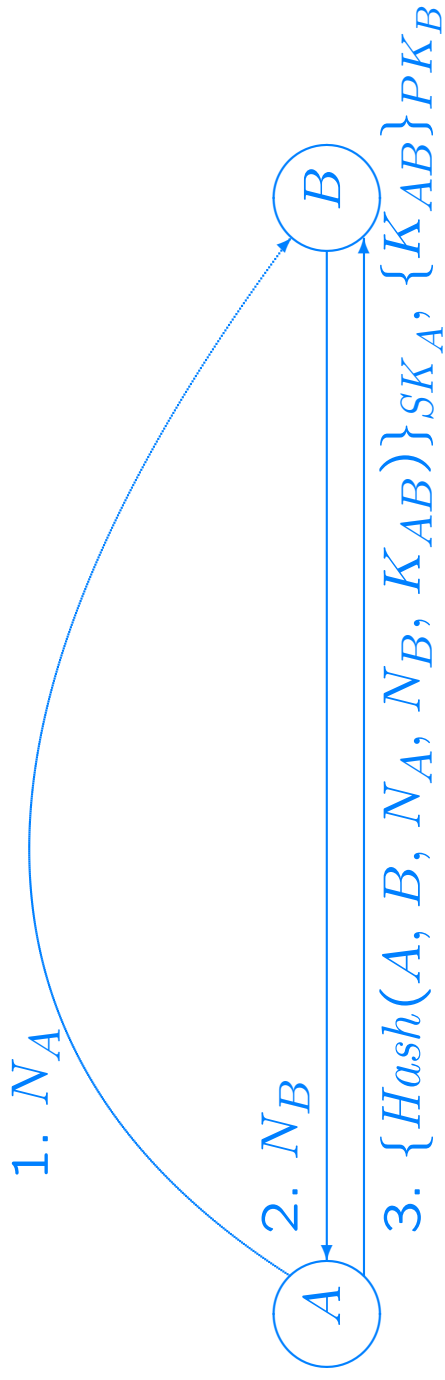
For example:

- bootstrapping from passwords,
- cross-domain authentication (that is, several *S*'s).

A variant is used in Windows 2000.

Netscape's SSL v3 protocol

(fragment, simplified)



$\{X\}_{PK_B}$: X encrypted with B 's public key, for secrecy

$\{X\}_{SK_A}$: X signed with A 's secret key, for integrity

K_{AB} , N_A , N_B are used in computing a session key

The full SSL (or TLS)

A more complete and complex protocol with a bulkier specification:

- Specifics of cryptographic operations.
- Several options:
 - various cryptographic algorithms,
 - protocol versions,
 - one-way or two-way authentication,
 - a fast mode.
- Option negotiation.
- Message transmission (“the record layer”).
- Alert messages.

Reading

A paper by Needham and Schroeder:
“Using encryption for authentication in large networks of
computers”
in the ACM Digital Library.

The SSL protocol:

<http://wp.netscape.com/eng/ssl3/draft302.txt>

and/or

the TLS protocol:

<http://www.ietf.org/rfc/rfc2246.txt>

<http://www.ietf.org/rfc/rfc4346.txt>

<http://www.ietf.org/internet-drafts/>

[draft-ietf-tls-rfc4346-bis-00.txt](#)

Homework 6 (due on May 18)

This homework consists of three exercises.

Exercise 1:

Suppose that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a secure shared-key encryption scheme. Suppose that the set of plaintexts Plaintext consists of all strings, for simplicity, and that encryption is length-preserving.

Let $\mathcal{K}' = \mathcal{K}$, $\mathcal{E}'_k(M) = \mathcal{E}_k(\mathcal{E}_k(M))$, $\mathcal{D}'_k(M) = \mathcal{D}_k(\mathcal{D}_k(M))$.

Is $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ a secure encryption scheme? (Yes/No)

(If you followed the lectures closely, you may be able to give a proof for your answer, for extra credit.)

Exercise 2:

Suppose that A and B exchange a key K_{AB} using either the shared-key Needham-Schroeder protocol or the public-key Denning-Sacco protocol, then A sends to B some data M encrypted under K_{AB} . Suppose that an attacker C sees every message exchanged and records it. Later, C somehow compromises the server S and finds the long-term secrets in S , including the shared keys K_{AS} and K_{BS} in the case of Needham-Schroeder and the secret key SK_S in the case of Denning-Sacco (but not K_{AB} or M). Will C be able to recover M

- a) in the case of the shared-key Needham-Schroeder protocol? (Yes/No)
- b) in the case of the public-key Denning-Sacco protocol? (Yes/No)

Exercise 3:

Suppose that A is a computer whose clock has not been initialized, and which wants to set the clock correctly (to within the accuracy permitted by network delays) with the help of a publically available and trusted time server B . An attacker should not be able to persuade A to set its clock to a future time or, to the extent permitted by network delays, to a past time. For this purpose, we may let A and B establish a secure communication channel, with one of the protocols discussed in the lectures, and B use this channel to tell the time to A .

- a) Can we rely on the shared-key Needham-Schroeder protocol? (Yes/No)
- b) Can we rely on the public-key Denning-Sacco protocol? (Yes/No)