

Design principles for protocols

Reading

A paper by Abadi and Needham:

- “Prudent engineering practice for cryptographic protocols”
reachable from www.soe.ucsc.edu/~abadi/allpapers.html.

A paper by Anderson and Needham:

- “Robustness principles for public key protocols”
reachable from www.cl.cam.ac.uk/~rja14/.

And while you are there you might as well read:

- “Why cryptosystems fail”

Design principles

(joint work with Roger Needham)

We collected:

- some common-sense rules (\simeq ten)
 - all fairly obvious
 - some general, some concrete
- published examples
 - largely wrong
 - as a proof of applicability

The principles seem generally useful for cryptographic protocols

- for authentication,
- for electronic commerce
- ∴

Principle 1

Every message should say what it means: the interpretation of the message should depend only on its content.

It should be possible to write down an English sentence describing the content—though if there is a suitable formalism available that is good too.

An attack

The core of the Denning-Sacco protocol is:

$$A \rightarrow B : \{\{K_{AB}, T\}_{SK_A}\}_{PK_B}$$

A tells B that K_{AB} is a good key for A and B at time T .

An attack goes:

$$A \rightarrow C : \{\{K_{AC}, T\}_{SK_A}\}_{PK_C}$$

$$C \rightarrow B : \{\{K_{AC}, T\}_{SK_A}\}_{PK_B}$$

so any C may have the key that B believes shared with A .

Then:

$$C \rightarrow B : \{\text{data}, T\}_{K_{AC}}$$

and B would believe that A sent data.

Diagnosis

Optimistic use of encryption.

Names are missing.

⇒ It is not possible to parse the message into the statement that represents its meaning.

Solution

$A \rightarrow B : \{\{A \text{ says } K_{AB} \dots B \dots T\}_{SK_A}\}_{PK_B}$

or

$A \rightarrow B : \{\{A, B, K_{AB}, T\}_{SK_A}\}_{PK_B}$

or

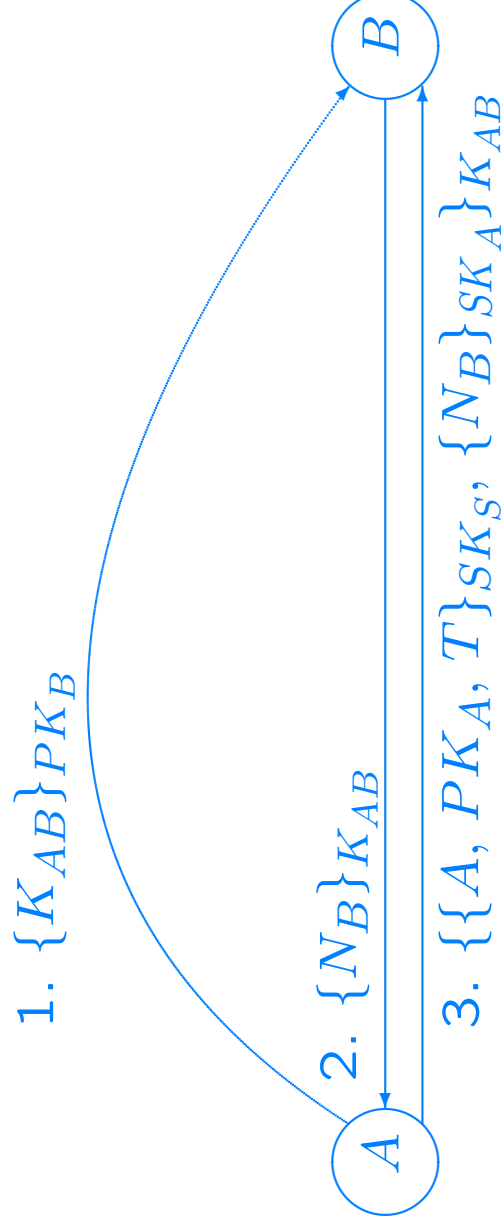
any other unambiguous encoding of the meaning of the message.

Principle 2

If the identity of a principal is important for the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

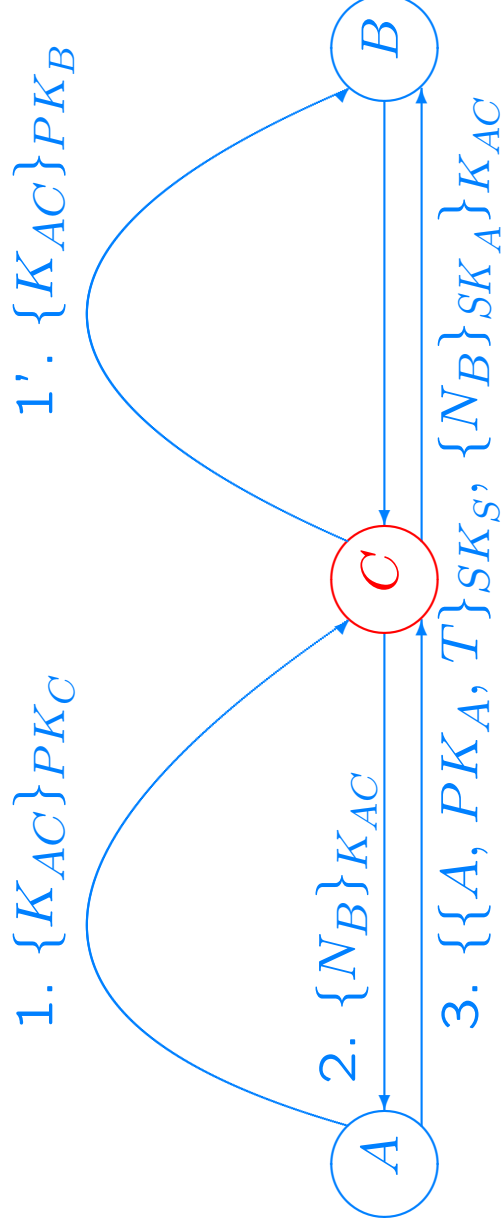
Netscape's SSL v1 protocol

RFC of Oct. 31, 1994 (simplified):

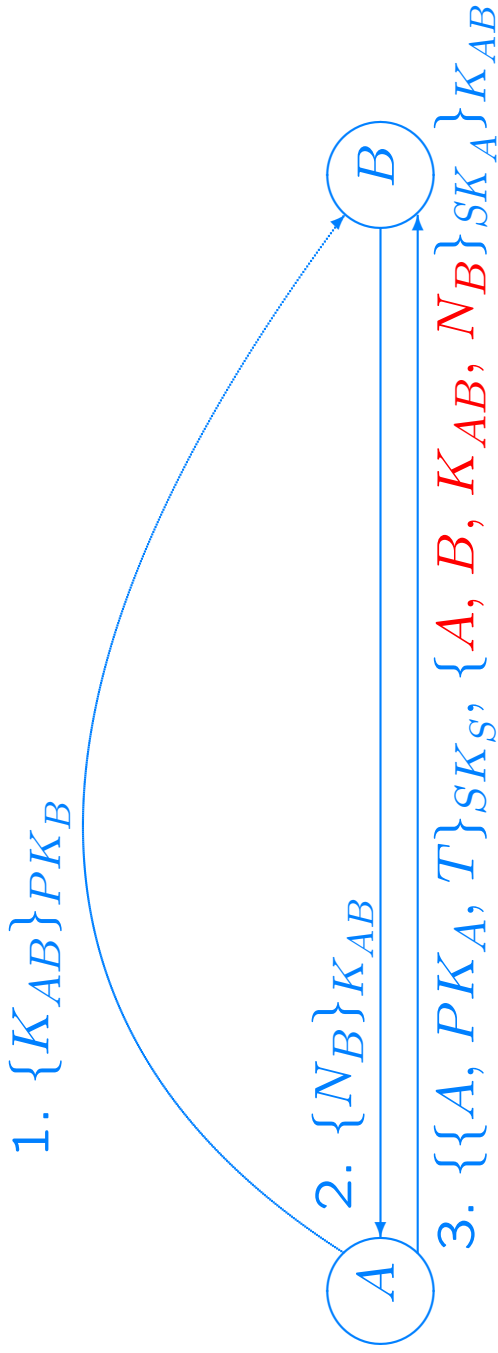


An attack

If A wants to talk to C , then C can impersonate A :

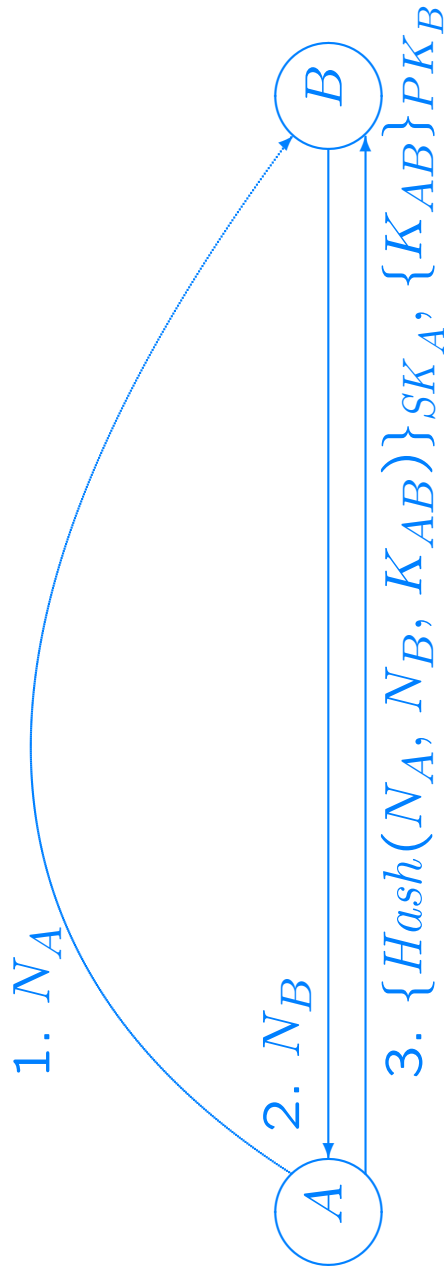


Solution



Netscape's SSL v3 protocol

Specification of Nov. 10, 1995 (simplified):



An analogous attack works.

SSH v2 protocol

(Internet draft of June 1996)

Option with public-key client authentication:

$A \rightarrow B : N_A$

$B \rightarrow A : N_B$

$B \rightarrow A : PK_{Bh}, PK_{Bs}$

$A \rightarrow B : \{\{H(\text{previous msgs.}), K\}PK_{Bs}\}PK_{Bh}$

$A \rightarrow B : \{A, PK_A, \{H(A, N_A, N_B)\}SK_A\}K'$

PK_{Bh} is a long-term host key.

PK_{Bs} is a shorter-term server key.

K' is a shared key derived from K , N_A , and N_B .

An analogous attack works again.

The Woo-Lam protocol

$A \rightarrow B: A$

$B \rightarrow A: N_B$

$A \rightarrow B: \{N_B\}_{K_{AS}}$

$B \rightarrow S: \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$

$S \rightarrow B: \{N_B\}_{K_{BS}}$

A wants to talk to B .

B sends a random challenge N_B .

A responds with N_B encrypted with key K_{AS} shared by A and the server S .

B queries S , who translates to key K_{BS} between B and S .

An attack

Imagine a concurrent execution:

$$C \rightarrow B : A$$
$$C \rightarrow B : C$$
$$B \rightarrow A : N_B$$
$$B \rightarrow C : N'_B$$
$$C \rightarrow B : \{N_B\}_{K_{CS}}$$
$$C \rightarrow B : \{N_B\}_{K_{CS}}$$
$$B \rightarrow S : \{A, \{N_B\}_{K_{CS}}\}_{K_{BS}}$$
$$B \rightarrow S : \{C, \{N_B\}_{K_{CS}}\}_{K_{BS}}$$
$$S \rightarrow B : \{N_B\}_{K_{BS}}$$
$$S \rightarrow B : \text{error or junk}$$

so any C can convince B that A is present.

Diagnosis

It is not possible to parse S 's message into the statement that represents its meaning:

A 's name is missing.

Nonces are good for ensuring freshness
but not always association.

Double encryption is no cause for optimism.

Solution

$A \rightarrow B : A$

$B \rightarrow A : N_B$

$A \rightarrow B : \{N_B\}_{K_{AS}}$

$B \rightarrow S : A, \{N_B\}_{K_{AS}}$

$S \rightarrow B : \{A \text{ said } N_B\}_{K_{BS}}$

S is explicit.

N_B is used only for freshness.

No double encryption.

Other principles concern

- encryption
- timeliness
- trust
- secrecy

The principles serve to

- simplify protocols
- simplify formal analysis
(see Paulson's work with Isabelle,
Blanchet's with ProVerif)
- avoid many mistakes

Principle 3

Be clear as to why encryption is being done.

Encryption is not synonymous with security.

Principle 4

Be clear as to what properties you assume of nonces.

What may do for ensuring timeliness may not do for ensuring association—and perhaps association is best established by other means.

Principle 5

The protocol designer should know which trust relations the protocol depends on.

Principles for secrecy

Information-flow ideas lead to other principles for secrecy:

- Classify data, channels, and keys into levels.
 - Secret data encrypted under secret keys may be made public.
- Do not send secret data on public channels (and also avoid implicit information flows).
- Distinguish secret inputs from public inputs.

Principles for secrecy (fuzzier)

- Secrets should be strong enough for the data that they protect.
 - Beware of weak passwords.
 - Change and diversify keys.
- Do not expect attackers to obey rules.
- Do not underestimate the initial knowledge or the observations of attackers.

Staying out of trouble

Keep your protocols simple.

Be suspicious of clever optimizations.

Be explicit:

- include sufficient proofs of freshness,
- include sufficient names,
- do not count on context,
- use evident classifications.

Interpreting a message should be a simple matter of parsing.

Staying out of trouble (cont.)

Cryptography helps, but it is not the whole story.

We can and should go beyond patching security holes one by one.