

Security protocols and
specifications
(Generalities)

Protocol specifications

“Security by obscurity” hides specifications.

Open design is often superior.

Specifications of security protocols abound. They vary in

- quality
- scope
- purpose
- vocabulary

There are at least three kinds:

- step-by-step narrations (“bits on the wire”)
- properties (e.g., secrecy)
- properties at the interface (what interface?)

Protocol narrations

Typical protocol narrations combine lots of prose, a few bubble diagrams, ad hoc notations, and message sequences like:

Message 3 $A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$

Message 4 $B \rightarrow A : \{N_B\}_{K_{AB}}$

Message 5 $A \rightarrow B : \{N_B - 1\}_{K_{AB}}$

They often include an informal security analysis.

Communication model

What does “Message n $X \rightarrow Y : M$ ” mean?

We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material. [...]

We also assume that each principal has a secure environment in which to compute, such as is provided by a personal computer or would be by a secure shared operating system.

(Needham and Schroeder)

This model is refined by specifying:

- the computation power of intruders
- the collaboration of some inside agents
- some secure communication paths.

Model of cryptographic operations

Sometimes, the description of cryptographic operations is fairly informal or abstract.

Often, encryption is assumed (implicitly or explicitly) to guarantee more than secrecy. In particular:

- $\{M\}_K$ cannot be produced by anyone who does not know M and K .

Thus, encryption is not malleable.

$$\{N_B\}_K \text{ vs. } \{N_B - 1\}_K$$

- Decryption with an incorrect key is evident.

Other ambiguities

- Which pieces of data are known in advance and which are freshly generated.
- Checking of messages.
- Ordering of messages.
- Concurrent executions and roles.
- Properties of nonces.
- Objectives, trust relations, ...

A mundane ambiguity

The simplest fix is to require that a SSL implementation receive a **change cipher spec** message before accepting a finished message. (Indeed, there is some language in the specification which could be interpreted to mandate this restriction, although it is not entirely clear.) [...] at least one implementation has fallen for this pitfall.

(Wagner and Schneier)

We can do better!

Design and analysis methods

There are now several methods for describing and verifying security protocols, based on:

- rigorous but informal frameworks,
- temporal logics,
- process algebras (CSP, the pi calculus),
- theorem-proving tools,
- special-purpose formalisms.

They have resulted in:

- increased confidence in some protocols,
- discovery of many protocol flaws,
- a better understanding of how to design robust protocols.

Protocol objectives

Narrations need not specify the meaning or the purpose of messages.

Some standard protocol properties are:

- authenticity (known message origin)
- secrecy (known message destination)

and also:

- forward secrecy (resilience to future leak)
- anonymity
- service availability
- non-repudiation
- deniability

As usual, there is no precise, unique definition of security.

Common themes (revisited)

- Interaction with an uncertain environment.
(Contrast with mutual exclusion.)
- Some security even against lucky, powerful, or persistent attackers.
 - Even if the attacker controls the network.
 - Even if a session key is compromised.
(See Needham-Schroeder, SSL.)
 - Even if an insider is dishonest.
(See Denning-Sacco, SSL, SSH.)
- Doing without full functional correctness.
E.g., message origin, not message correctness.

Specifying the Denning-Sacco protocol

We expect the following properties:

- **Secrecy:**
Only A and B know data.

- **Authenticity:**

If B is convinced that data is from A ,
then it is.

These depend on auxiliary properties about K_{AB} .

Some good news

Usually, a property is a predicate on behaviors.

Authenticity may be phrased as a property.

We can turn

“*B* is convinced that data is from *A*”

into a statement about *B*’s actions,

e.g., “*B* writes data to a file” .

A difficulty

But this is a property of a system in which there is an active attacker.

The set of all behaviors of the system has transitions for A , B , S , and an attacker.

It is hard to model the attacker (at least without complexity theory):

- It can choose random numbers.
- It cannot get too lucky in guessing keys.

Another difficulty

Secrecy is not a standard property.

Secrecy is not true or false of a behavior, but of a set of behaviors.

The boundaries of protocols

Protocols are parts of systems.

Protocol boundaries are not always well defined.

They give rise to confusion and error.

There are boundaries of several natures.

Up: interfaces and rules for protocol use.

Down: interfaces and assumptions about auxiliary functions and servers.

To the sides: traversals of machine and network boundaries.

Before: preliminary protocol negotiations.

After: communication, error handling.

Error handling

(according to Bleichenbacher)

The SSL documentation does not clearly specify error conditions and the resulting alert messages.

SSL implementations vary in their error handling.

Even sending out an error message may be risky.

The timing of checks in the protocol is crucial.

Protocol use

There is a gap between the protocol properties usually proved and those that clients need.

- Clients do not care about keys and nonces.
- Clients may care about concepts like data integrity and secrecy, identity, authority.

Clients have obligations (e.g., protecting secrets).

APIs with semantics (or even compilers) should bridge this gap.

Negotiation

Systems often include several protocols, and perhaps several versions of each.

Protocol interactions can lead to flaws.

Protocol choice (negotiation) is tricky.

- If A and B are willing to use protocols P and Q , and both prefer P , then an attacker should not get them to use Q .
- But, at the time of negotiation, A and B may not yet have a good channel for negotiating.

Protocols like SSL do a fancy version negotiation with complex, unwritten goals.