What is protocol?

- A protocol, for present purposes, is a set of rules or conventions defining an exchange of messages between a set of two or more partners.
Explicit communication

• Principle 1

Every message should say it means: the interpretation of the message should depend only on its content. It should be possible to write down a straightforward English sentence describing the content—though if there is suitable formalism available.

All elements of this meaning should be explicitly represented in the message, so that a recipient can recover the meaning without any context.
Appropriate Action

- Principle 2

The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not.

Example

If someone believes that choosing session keys should be done by a suitable trusted server rather than by one of the participants in a session, then he will not wish to use a protocol whose session key is generated by participants.
Naming

• **Principle 3**

If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal’s name explicitly in the message.

• The names relevant for a message can sometimes be deduced from other data and from what encryption keys have been applied.
Naming Example

- **Message 1**  
  A→S: A, B

- **Message 2**  
  S→A: CA, CB

- **Message 3**  
  A→B: CA, CB, \{\{K_{ab}, T_a\}K_a^{-1}\}K_b

- There is no guarantee that B should know that the message was intended for B (because of the use of $K_b$).

- **Attack:**
  - **Message 3'**  
    B→C: CA, CC, \{\{K_{ab}, T_a\}K_a^{-1}\}K_c

- **Solution:**
  - **Message 3**  
    A→B: CA, CB, \{\{A, B, K_{ab}, T_a\}K_a^{-1}\}K_b
Encryption

- Encryption is used for a variety of purposes:
  - Confidentiality, Authentication, Binding, Producing random numbers

- Principle 4

Be clear about why encryption is being done. Encryption is not wholly cheap, and not asking precisely why it is being done can lead to redundancy. Encryption is not synonymous with security, and its improper use can lead to errors.
The use of encryption examples

• The Kerberos protocol

Message 1  \[ A \rightarrow S: \quad A, B \]

Encryption is not essential

Message 2  \[ S \rightarrow A: \quad \{T_s, L, K_{ab}, B, \{T_s, L, K_{ab}, A\}K_{bs}\} K_{as} \]

\( K_{ab} \) should keep confidential, \( S \) should sign as proof of authenticity

Message 3  \[ A \rightarrow B: \quad \{T_s, L, K_{ab}, A\}K_{bs}, \{A, T_a\} K_{ab} \]

First part: The proof of extraction
Second part: prove knowledge of \( K_{ab} \) near time \( T_a \)

Message 4  \[ B \rightarrow A: \quad \{T_a + 1\} K_{ab} \]
Signing encrypted data

• Principle 5

When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of the message.

• Example

Message 1  A→B:  \{X\}K_b, \{H(X)\}K_a^{-1}

X may be password, H(X) may be get from other ways.
Timelines

• An important part of the meaning of a message is made up of temporal information.

• One common precondition for action upon a message is that there is reason to believe that the message is fresh and hence not a replay of an old one.

• It should be bound together with the rest of the message.
Timestamps, sequence numbers and other nonces

• When guarding against replay of messages from an earlier run of the same protocol, it is common to use nonces as part of challenge-response exchange.

• Principle 6

Be clear what properties you are assuming about nonces. What may do for ensuring temporal succession may not do for ensuring association—and perhaps association is best established by other means.

• Example

Message 1  A→B:  A
Message 2  B→A:  \(N_b\)
Message 3  A→B:  \(\{N_b\}K_{as}\)
Message 4  B→S:  A, B, \(\{N_b\}K_{as}\)
Message 5  S→B:  \(\{A, N_b\}K_{bs}\)
Timestamps, sequence numbers and other nonces

- Principle 7

The use of a predictable quantity (such as the value of counter) can serve in guaranteeing newness, though a challenge-response exchange. But if a predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a challenge and later replay a response.

- Example (synchronized clocks)

Message 1 \( A \rightarrow S : \quad A, N_a \)
Message 2 \( S \rightarrow A : \quad \{T_s, N_a\}K_{as} \)

Attack:
Enemy modify message 1 in which \( N_a \) is set to a future value.
Timestamps, sequence numbers and other nonces

- Principal 8

If timestamps are used as freshness guarantees by reference to absolute time, then the difference between local clocks at various machines must be much less than the allowable age of a message deemed to be valid.

Furthermore, the time maintenance mechanism everywhere becomes part of the trusted computing base.
What is fresh: use vs. generation

• Principle 9

A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised. Recent use does not make the key look any better than it would otherwise.

• Example (Needham-Schroeder protocol, similar to Kerberos)

Message 1  A → S : A, B, Na
Message 2  S → A : {Na, B, Kab, {Kab, A}Kbs}Kas
Message 3  A → B : {Kab, A}Kbs
Message 4  B → A : {Nb} Kab
Message 5  A → B : {Nb + 1} Kab
Recognizing messages and encodings

- Principle 10

If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol.

Example (Needham-Schroeder protocol):

Message 4  B → A:  \{N_b\} K_{ab}
Message 5  A → B:  \{N_b+1\} K_{ab}

The message would be clearer if they were rewritten:

Message 4  B → A:  \{N-S Message 4: N_b\} K_{ab}
Message 5  A → B:  \{N-S Message 5: N_b\} K_{ab}
Trust

• We may simply say that A trusts B in regard to some function if a loss of security to A could follow from B not behaving in the specified way; it is usually difficult or impossible for A to verify B’s good behavior.

• Principal 11

The protocol designer should know which trust relations his protocol depends on, and why the dependence is necessary. The reasons for particular trust relations being acceptable should be explicit though they will be founded on judgment and policy rather than on logic.

Example:
Principals associate public keys with other principals by consulting public-key certificates.
Conclusions

• The design of good security protocol is hard.

• The principles and examples described in the paper are useful.
Thanks very much