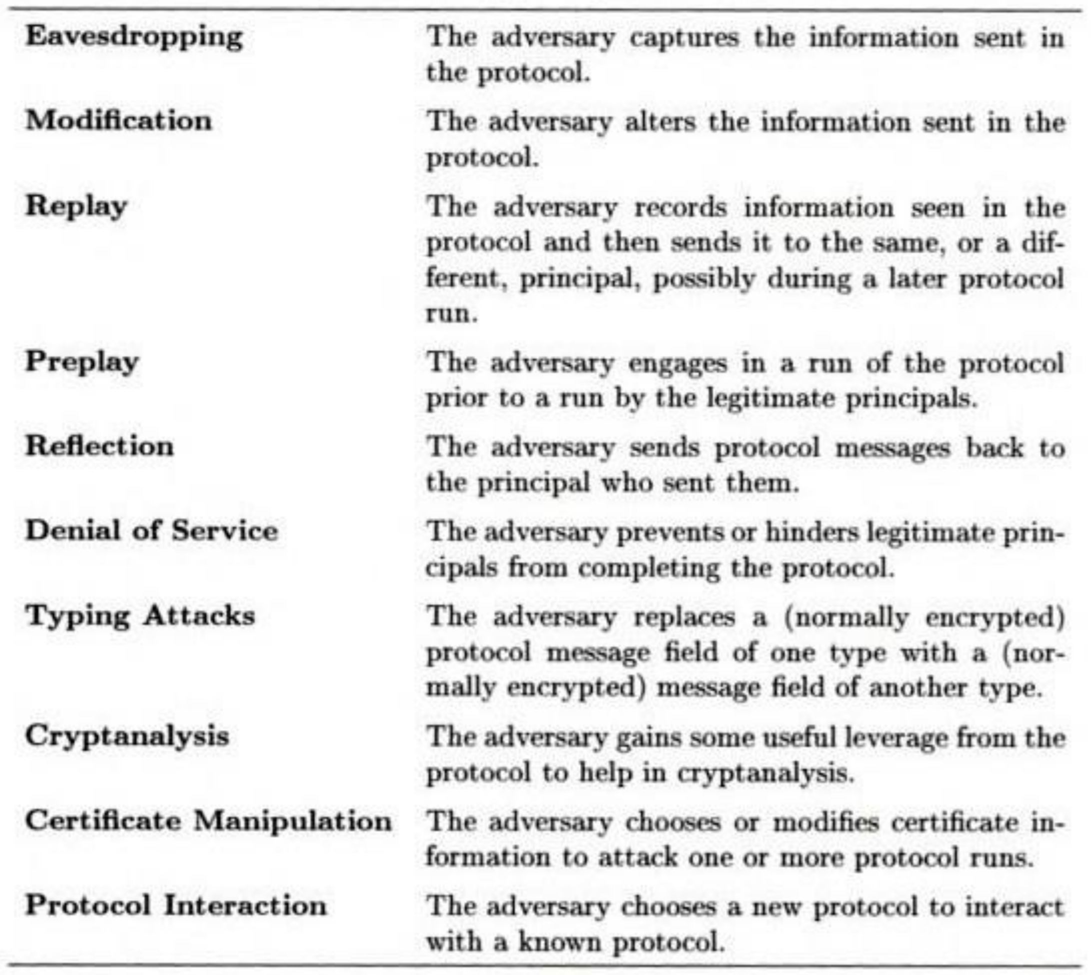
Introduction to Crypto Protocols : Features and Attacks

Types of attacks on a protocol

Summary of attacks

- Summary of attacks

Table 1.3. Types of protocol attack



Eavesdropping	The adversary captures the information sent in the protocol.
Modification	The adversary alters the information sent in the protocol.
Replay	The adversary records information seen in the protocol and then sends it to the same, or a different, principal, possibly during a later protocol run.
Preplay	The adversary engages in a run of the protocol prior to a run by the legitimate principals.
Reflection	The adversary sends protocol messages back to the principal who sent them.
Denial of Service	The adversary prevents or hinders legitimate principals from completing the protocol.
Typing Attacks	The adversary replaces a (normally encrypted) protocol message field of one type with a (normally encrypted) message field of another type.
Cryptanalysis	The adversary gains some useful leverage from the protocol to help in cryptanalysis.
Certificate Manipulation	The adversary chooses or modifies certificate information to attack one or more protocol runs.
Protocol Interaction	The adversary chooses a new protocol to interact with a known protocol.

1. Eavesdropping

- Most basic attack
- Passive form of attack (all others are active)
- Addressed by encryption
  - What elements to encrypt may be tricky
  - May not solve the problem even if all elements are encrypted

- Still useful as part of other attacks

2. Modification

- If a message field is not integrity protected, it's a potential target of a modification attack

- Use integrity mechanisms (hash/MAC/signature)
  - Encryption is not sufficient
  - But authenticated encryption will do

- Some attacks simply split and re-assemble fields from different messages (no need to alter them)
  - Must cover all parts of the message that must be kept together

3. Replay attack

- Attackers insert a message or part of a message, that has been used in a previous protocol run

- Replayed messages could be copied from:
  - a past run
  - the current run

- Defense: freshness items

- Related attacks: Preplay, Reflection, Relay

4. Preplay attack

- Attackers extract/calculate/guess some items of a future protocol run from current/past runs, and use those values to compromise a protocol

- Possibly not much useful on its own – generally used with replay attacks
- See HAC Note 10.7 (page 397)
- Defense: challenge-response / freshness

5. Reflection attack

- Special case of replay
- The adversary sends the protocol messages back to the originating party in a shared key protocol
- May require concurrent protocol runs
- Security assumption:
  - The attacker may initiate any number of parallel protocol runs between any principals – with the same or different roles

- Now, consider the following protocol (K is the shared secret between A and B):
  1. A B: $\{N_A\}_K$
  2. A B: $\{N_B\}_K$, N_A
  3. A B: N_B

- What is achieved in this protocol?

Reflection attack on the previous protocol

- C starts two parallel session with A
  - labeled as (1,2,3) and (1′,2′,3′) – see below
- Note that C does not know K
- A acts as a decryption oracle – sometimes these attacks are called “oracle attacks”

- Protocol messages:

  1. A ⍰ C: $\{N_A\}_K$
  1′. C ⍰ A: $\{N_A\}_K$
  2′. A ⍰ C: $\{N'_A\}_K$, N_A
  2. C ⍰ A: $\{N'_A\}_K$, N_A
  3. A ⍰ C: N'_A
  3′. C ⍰ A: N'_A

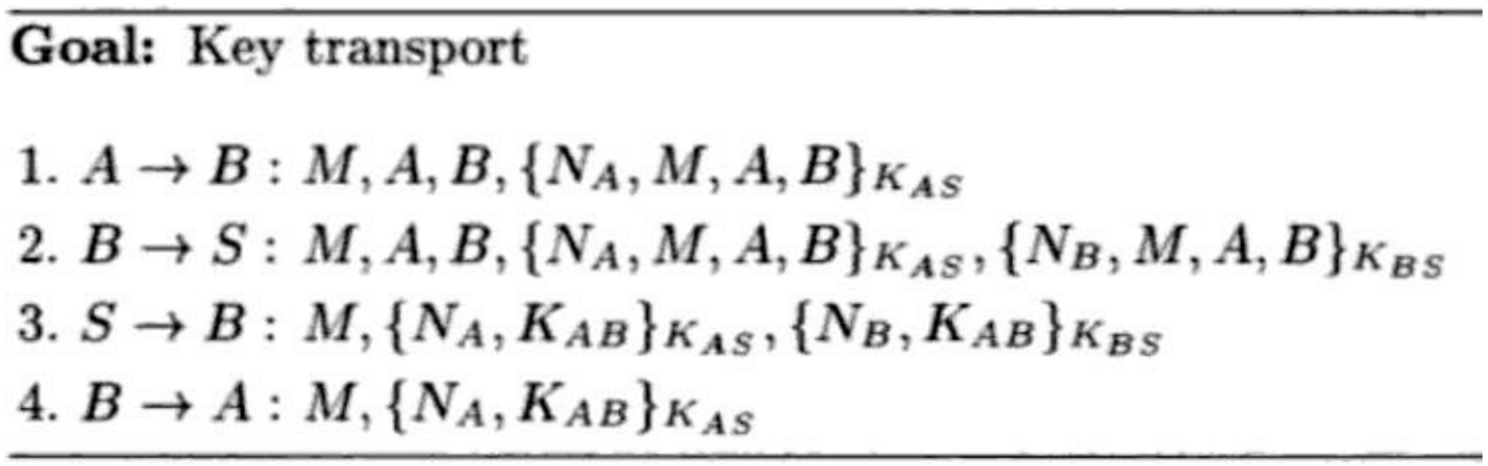
6. Denial of service (DoS)

- Adversary prevents legitimate users from completing a protocol
  - Client side: Drop a message/connection
  - Server side:
    - Resource depletion attacks
    - Connection/bandwidth depletion attacks

- Defense: no complete defense, but you can still restrict attacks
  - Use cookies (stateless connection), client puzzles, distributed servers, a “big” network pipe

7. Typing attacks

- Manipulate how string of bits are interpreted
- Recipient accepts one protocol element as another (i.e., one type as another)
- Occurs when an adversary successfully replaces a message field of one type with a message field of another type---encrypted or not
- Consider the Otway-Rees protocol (M, N_A: nonces chosen by A; N_B: nonce chosen by B)



Goal: Key transport

1. $A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$
2. $B \rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$
3. $S \rightarrow B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
4. $B \rightarrow A : M, \{N_A, K_{AB}\}_{K_{AS}}$

Typing attack on the previous protocol

- Assume the length of K_{AB} = length of (M, A, B)
- E.g.: M = 64 bits; A, B = 32 bits; K_{AB} = 128 bits

- C can masquerade as B for the whole session (i.e., C knows the session key – how?)

- The attack:

$$1.\ A \rightarrow C_B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$$
$$4.\ C_B \rightarrow A : \{N_A, M, A, B\}_{K_{AS}}$$

8. Cryptanalysis

- Here we consider the cryptanalysis of a protocol, rather than a crypto primitive
- More opportunities for cryptanalyst
  - Primitives + messages
  - The use of weak secret in a message

9. Certificate manipulation

- Assume: the issuing party does not verify ownership of a private key corresponding to the public key it certifies

- The protocol of Matsumoto et al. (A's public key g^a, private key: a, certificate with g^a: $Cert_A$)

  1. $A \rightarrow B$: g^x, $Cert_A$
  2. $B \rightarrow A$: g^y, $Cert_B$

- The shared key $K_{AB} = g^{ay+bx}$ (how is this calculated?)

Cert. manipulation attack on the previous protocol

- Assume C collects g^a and calculates g^{ac} (c is a random value), and gets $Cert_C$ with g^{ac}; C does not know a
- C masquerades as B and completes two parallel runs (one with A, another with B)

- The attack:

  1. $A \rightarrow C_B$: g^x, $Cert_A$

  1'. $C \rightarrow B$: g^x, $Cert_C$

  2'. $B \rightarrow C$: g^y, $Cert_B$

  2. $C_B \rightarrow A$: g^{yc}, $Cert_B$

- After the runs: A calculates $K_{AB} = (g^{yc})^a(g^b)^x = g^{acy+bx}$; B calculates $K_{CB} = (g^{ac})^y(g^x)^b = g^{acy+bx}$

- A believes K_{AB} is shared only with A & B; B believes K_{CB} is shared only with B & C
  - Is it bad?

10. Protocol interaction attack

- Happens when the same long-term key is used in multiple protocols (of different types)
  - Designer's fault
  - Deliberate choice – due to resource constraints (e.g., smart cards)

- Example:
  - One protocol may require decryption to prove possession of an authenticating key
  - Attackers can now decrypt arbitrary messages if the same key is used

11. Binding attacks

- Principals in charge of distributing public keys must ensure that there exists a verifiable binding between a public key and the corresponding agent

- Otherwise, binding attacks may be feasible

- Example (S distributes public keys among users; here A requests B's public key K_B from S; *Sig* is a signature scheme with message recovery):

1. $A \rightarrow S$: A, B, N_A
2. $S \rightarrow A$: $S, Sig_S(S, A, N_A, K_B)$

Binding attack – example

- Attacker's public key is accepted by A as B's valid key

    1. A $\rightarrow$ C_S: A, B, N_A

    1'. C_A $\rightarrow$ S: A, C, N_A

    2'. S $\rightarrow$ C_A: S, Sig_S(S, A, N_A, K_C)

    2. C_A $\rightarrow$ A: S, Sig_S(S, A, N_A, K_C)

- How to fix this problem?
    - Stronger binding of public keys with corresponding identities

Some protocol design principles

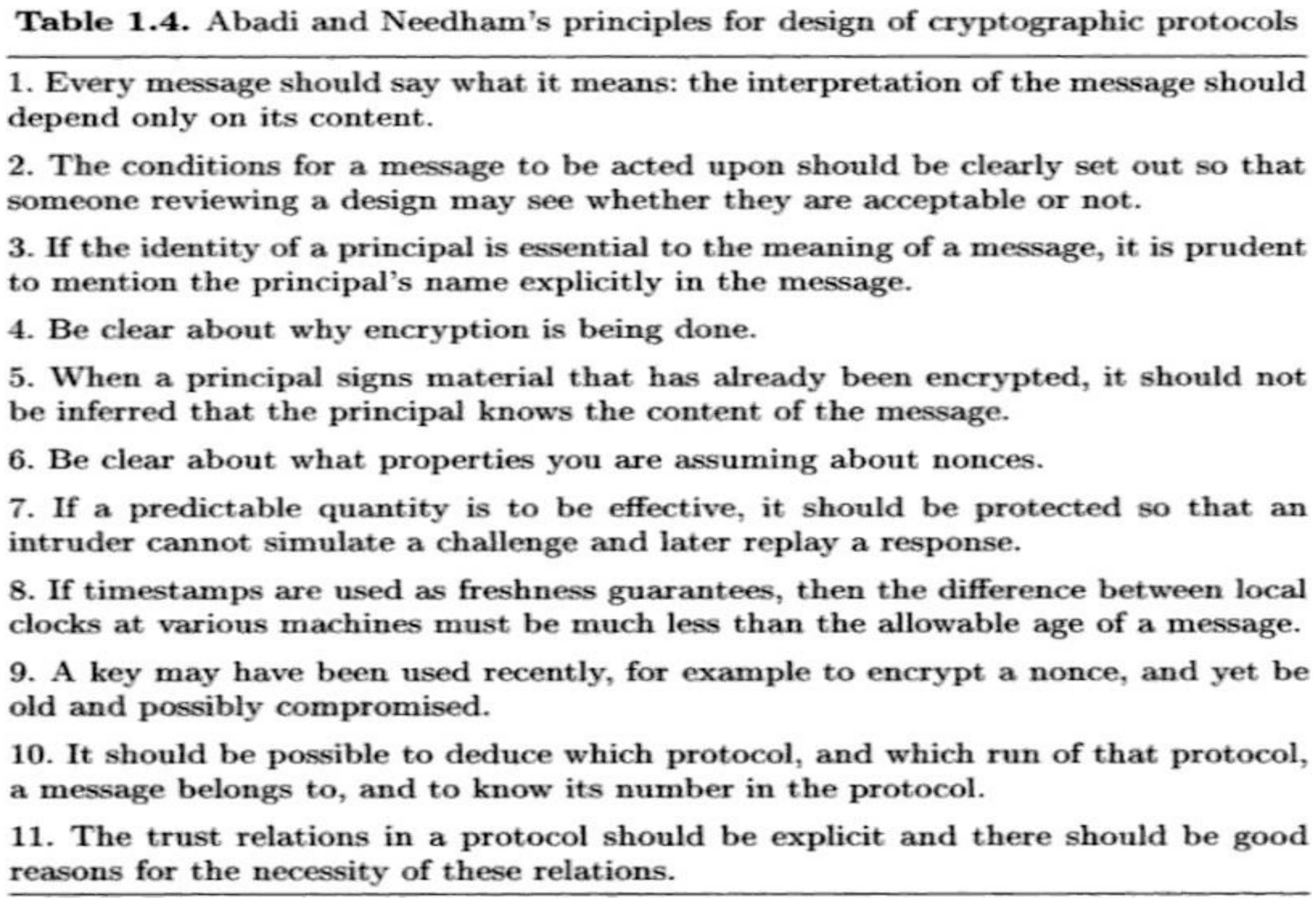


Table 1.4. Abadi and Needham's principles for design of cryptographic protocols

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

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.

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.

4. Be clear about why encryption is being done.

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.

6. Be clear about what properties you are assuming about nonces.

7. 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.

8. If timestamps are used as freshness guarantees, then the difference between local clocks at various machines must be much less than the allowable age of a message.

9. A key may have been used recently, for example to encrypt a nonce, and yet be old and possibly compromised.

10. It should be possible to deduce which protocol, and which run of that protocol, a message belongs to, and to know its number in the protocol.

11. The trust relations in a protocol should be explicit and there should be good reasons for the necessity of these relations.

Prudent Engineering Practice [AN’95]

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

2. 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.

3. Be clear about why encryption is being done.

4. When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message.

5. A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised.

6. The protocol designer should know which trust relations his protocol depends on, and why the dependence is necessary.