Computer and Network Security

Computer & Information Sciences & Engineering
University Of Florida
Gainesville, Florida 32611-6120
nemo@cise.ufl.edu

Cryptographic Protocols (Pfleeger Ch. 4, Stallings Ch. 10)

Distributed Programming and Logic

1 Types of Protocol

1.1 Arbitrated

Trusted third party involved vs. Non-arbitrated - only the principals, mutually suspicious

1.1.1 Advantages

1. serialized
2. documented
3. arbitrator has total knowledge
4. often much easier

1.1.2 Disadvantages

1. Trust?

2. Availability

3. Delay

4. Bottleneck

5. Secrecy

1.2 Adjudicated

Third party can verify what has happened and determine if one of the parties cheated

1.3 Self-enforcing

Either one of the parties can determine and prove that cheating has occurred if it did, as the protocol proceeds

2 What to look for

1. Initial assumptions

2. Trust relationships - who trusts whom, and for what

3. Goals of the protocol

4. Hidden assumptions (trust, keys, etc.)

5. Weaknesses to various forms of attack

6. Requirements on underlying mechanisms (clock, PRNG, crypto)

3 Attacks

3.0.1 Interception

3.0.2 Modification

1. Straight modification

2. Cut & Paste

3.0.3 Fabrication

1. chosen plaintext

2. chosen ciphertext

3.0.4 Replay

1. Simple replay

2. Reflection

3. Delay/deferred delivery

3.0.5 Man-in-the-Middle (Bucket Brigade)

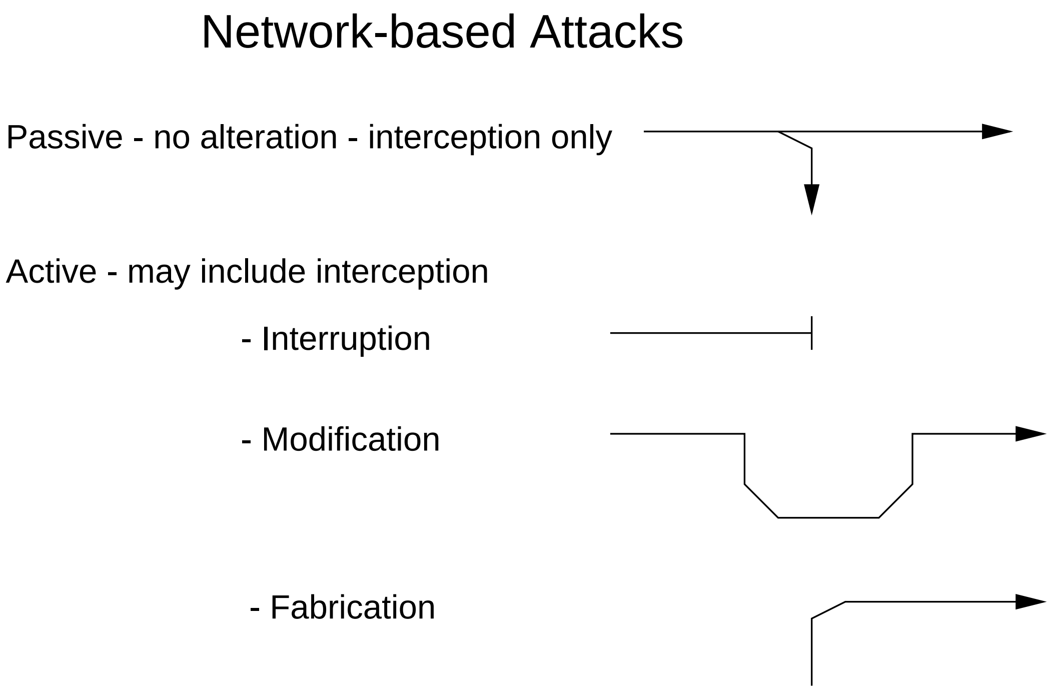
Network-based Attacks
Passive - no alteration - interception only
Active - may include interception
- Interruption
- Modification
- Fabrication

Figure 1: Basic Network Attacks

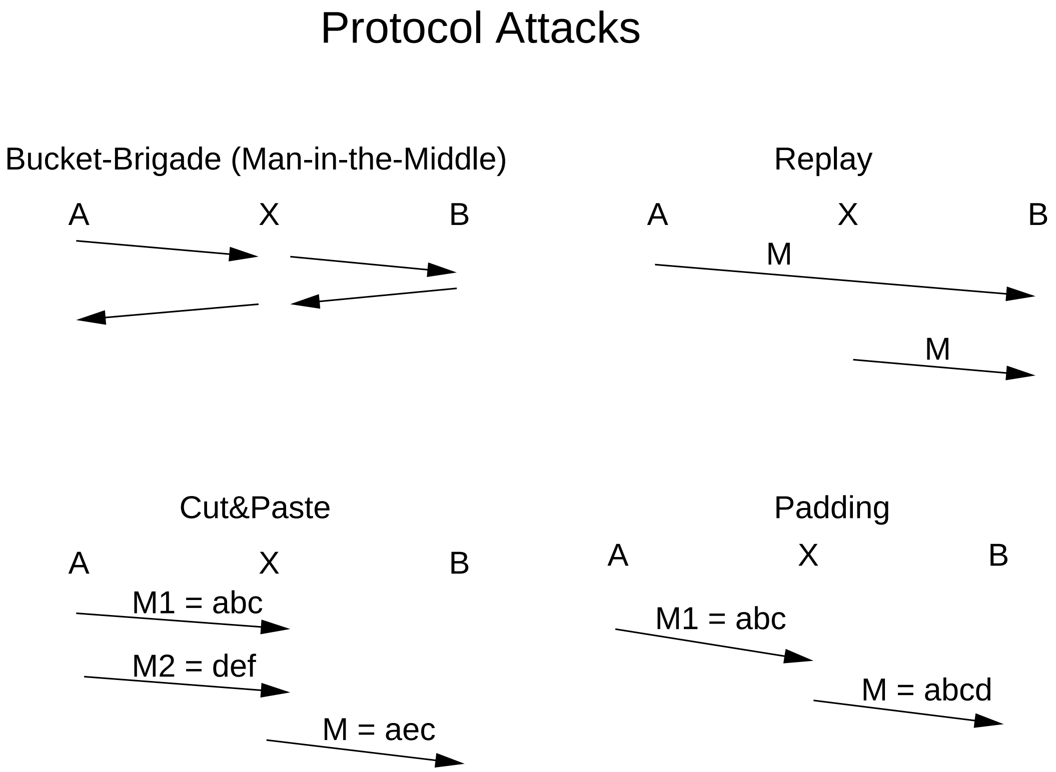

Figure 2: Protocol Attacks

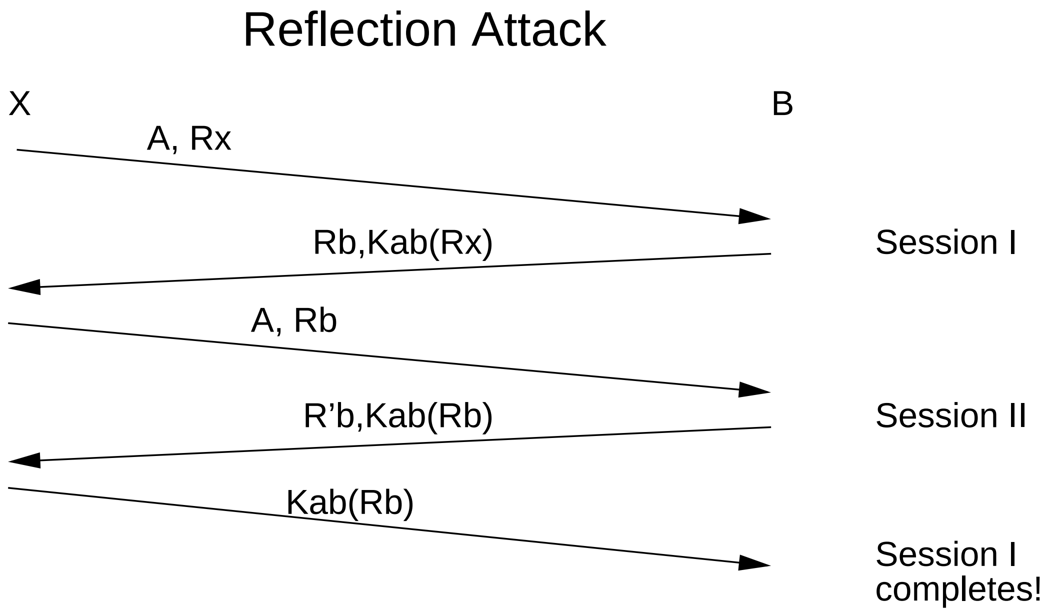

Figure 3: Reflection Attack

4 Tools

4.1 Digital signatures

1. source

2. association - elements of same message

3. authenticity

4. integrity

4.2 Encryption

1. secrecy

2. association - elements of same message

3. authenticity

4. integrity

5. binding encrypted message elements

4.3 Nonces

1. prevent replay

2. allow association of messages in same run

3. must be random

4. must be used only once

5. may also act as confounder

6. may be altered in reply if symmetric key used

4.4 Timestamps

1. prevent replay

2. must protect time service

3. clock skew issues - acceptable bounds on error

4. must remember recent past

5 Protocols

5.1 Notation

1. $\{x|y\}$ is x concatenated with y (often used to randomize an otherwise small set of possible x's)

2. $\{M\}K$ is message M encrypted with key K

3. $\langle M\rangle K$ is message M signed with key K

4. K_{ab} is a symmetric key used by A and B

5. K_a is A's public key

6. K_a^{-1} is A's private key

The following two forms are used when we need to be explicit about encrypting and decrypting

1. $C = E(M, K)$ is also message M encrypted with key K

2. $M' = D(C, K)$ is ciphertext C decrypted using key K (Note that if C is not a message encrypted using key K, then M' is garbage.)

5.2 Simple Authentication

5.2.1 Symmetric Key

- A wants to authenticate herself to B, A and B share a common secret K.

- General Protocol

    1. B sends A a challenge (some random number not used before)

    2. A combines that with K to produce a reply, which B can verify:

- With hashes

    1. $A \rightarrow B$: I'm A

    2. $B \rightarrow A$: Prove it by signing R_b

3. $A \rightarrow B : H(R_b|K)$

- With encryption

  1. $A \rightarrow B :$ I'm A
  2. $B \rightarrow A :$ Prove it by encrypting R_b
  3. $A \rightarrow B : E(R_b, K)$

- Mutual authentication

  1. Run the same protocol twice
  2. A's challenge for B appended to her reply
  3. If encryption is used, then R_a may be included with R_b in the encrypted reply.

- Care must be taken that this does not provide a means for X to obtain chosen plaintext (or perhaps, even known plaintext)

- Vulnerable to reflection attack

  1. Use different keys for each direction,
  2. Separate challenge space

5.2.2 Simple Asymmetric Key

- Using decryption

  1. $A \rightarrow B :$ I'm A
  2. $B \rightarrow A :$ Prove it by decrypting $\{R_b\}K_a$
  3. $A \rightarrow B : R_b$

- Using signatures

  1. $A \rightarrow B :$ I'm A
  2. $B \rightarrow A :$ Prove it by signing R_b
  3. $A \rightarrow B : \langle R_b \rangle K_a^{-1}$

5.3 Key Distribution

5.3.1 Symmetric Key Exchange w/o Server

- If A and B share key K, then it may be used as a master key (a.k.a. key distribution key or key encryption key, KEK)
- Session keys or message encryption keys (MEKs) or temporary keys are exchanged by encrypting them with KEK.
- The KEK is rarely used, and for little text, and for text that is difficult to recognize (usually), so breaking K is harder.
- Loss of a session key only exposes those messages encrypted with it, and only allows false authentication for the duration of the key's lifetime.
- Forward security: exposure of a session key does not reveal subsequent session keys (message contents)
- Backward security: exposure of a session key does not reveal previous session keys (message contents)

5.3.2 Symmetric Key Exchange w/Server

- If A shares a key K_{as} with S and
- B shares a key K_{bs} with S,
- where S is a trusted server, then
- A can ask S for a key to use with B for a session (either directly or indirectly through B),
- A can obtain the key (either directly or indirectly through B),
- B can obtain the key (either directly or indirectly through A),
- then A and B can authenticate each other using the session key

Care must be taken against replay attacks - generally through use of sequence numbers, timestamps or nonces.

Lots of examples - we will see these in the next lecture in detail

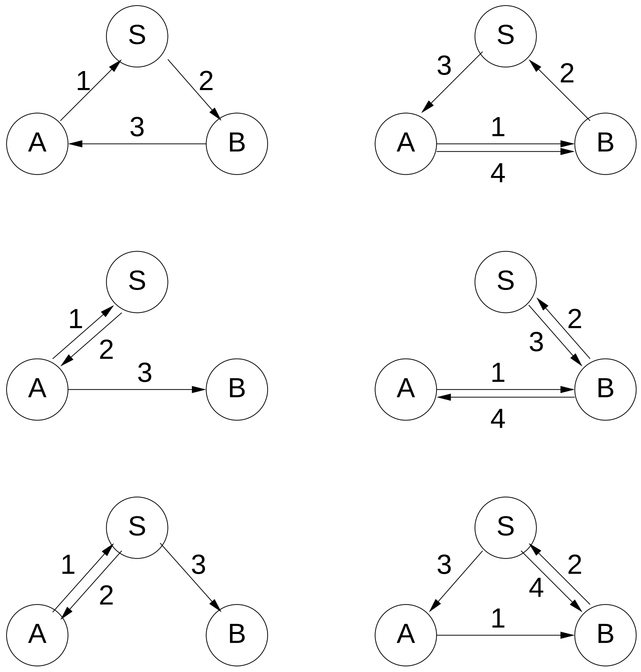

Figure 4: Authentication Message Orders

5.4 Interlock Protocol

5.4.1 Statement

This Rivest and Shamir protocol is intended to defeat a Man-in-the-Middle attack. It works by forcing the possible attacker to commit before it can read messages from either side.

$$
\begin{aligned}
M1: &\quad A \rightarrow B: \quad K_a \\
M2: &\quad B \rightarrow A: \quad K_b \\
M1: &\quad A \rightarrow B: \quad First(\{M_a\}_{K_b)} \\
M2: &\quad B \rightarrow A: \quad First(\{M_b\}_{K_a)} \\
M1: &\quad A \rightarrow B: \quad Last(\{M_a\}_{K_b)} \\
M2: &\quad B \rightarrow A: \quad Last(\{M_b\}_{K_a})
\end{aligned}
$$

where *First* and *Last* are functions that split their arguement into two halves and return the first or last half, respectively.

1. Mallory can substitute her key in msg1 and msg2, but
2. can't decrypt either msg3 or msg4 without the last half of these messages
3. Mallory can substitute her own first half messages in msg3 and msg4, but won't be able to find out what Alice and Bob were saying until msg5 and msg6.
4. The resulting conversation is completely different, unless Mallory can guess what Alice and Bob would say....

For the First and Last functions:

1. For block encryption, these can be even bits and odd bits of blocks
2. For chained encryption, the IV can be sent as Last (or send even blocks and odd blocks)
3. For MIC'ed messages, send MIC first, then the message