Computer and Network Security

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

Cryptographic Protocols (Pfleeger Ch. 4)

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. Goals of the protocol
3. Hidden assumptions
4. Trust relationships - who trusts whom, and for what
5. Weaknesses to various forms of attack

3 Attacks

3.0.1 Interception

3.0.2 Modification

1. Straight modification
2. Cut & Paste

3.0.3 Fabrication

3.0.4 Replay

1. Simple replay
2. Reflection
3. Delay/deferred delivery

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

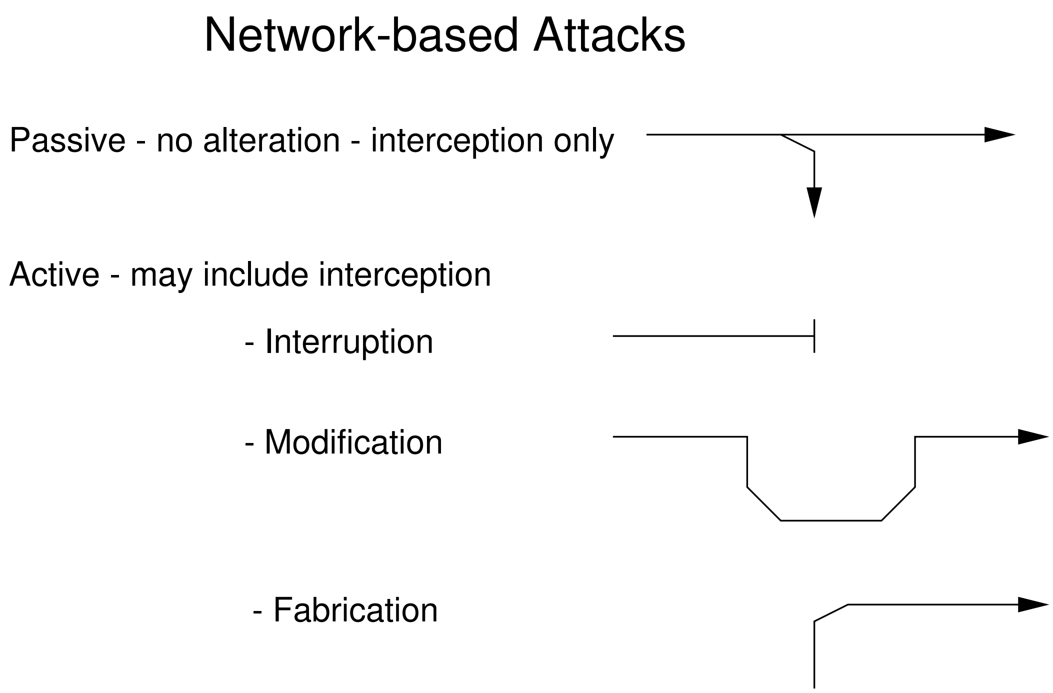

Figure 1: Basic Network Attacks

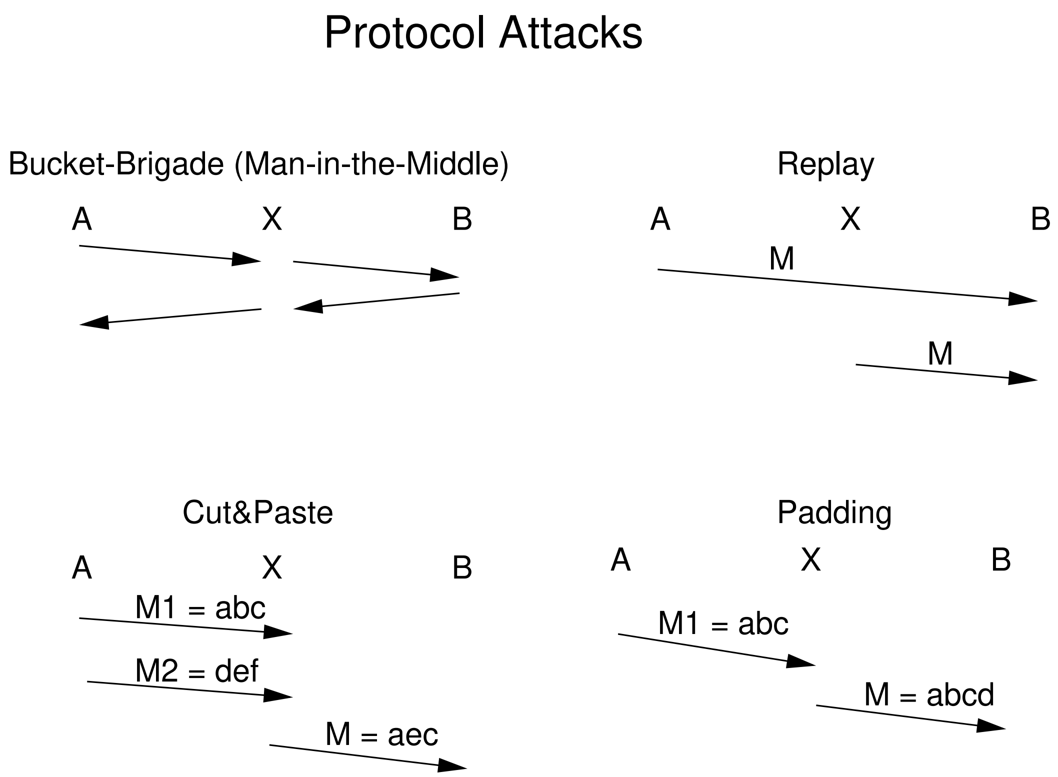

Figure 2: Protocol Attacks

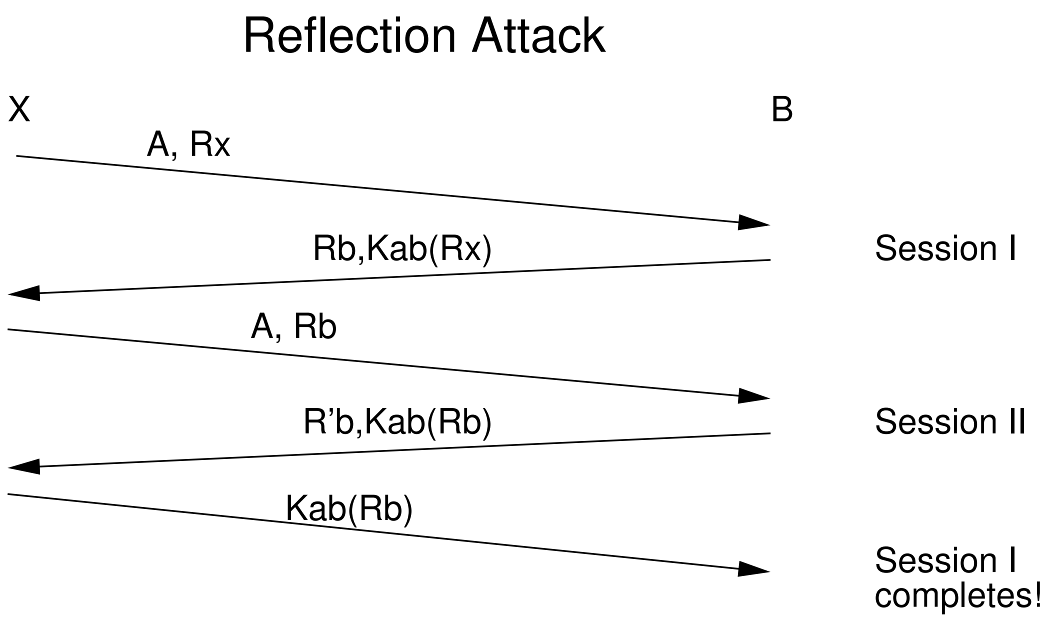

Figure 3: Reflection Attack

4 Tools

4.1 Digital signatures

1. source
2. association
3. authenticity
4. integrity

4.2 Encryption

1. secrecy
2. association
3. integrity

4.3 Nonces

1. prevent replay
2. must be random
3. must be used only once
4. may also act as confounder
5. may be altered in reply if symmetric key used

4.4 Timestamps

1. prevent replay
2. must protect time service
3. clock skew issues
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. MK is message M encrypted with key K
3. $\langle M\rangle K$ is message M signed with key K

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. $A \to B$: I'm A
  2. $B \to A$: Prove it by signing R_b
  3. $A \to B$: $H(R_b|K)$
- With encryption
  1. $A \to B$: I'm A
  2. $B \to A$: Prove it by encrypting R_b
  3. $A \to B$: $E(R_b, K)$

- 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
- 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.

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

5.4 Mental Poker

5.4.1 Statement

Fairly and secretly distribute to each of N parties, K items chosen from a total of M, such that no two parties has the same item in their set.

Two person version ($N = 2, K = 52$):
In this protocol, Alice and Bob "deal" a five-card poker hand to each other in a distributed manner. K_x is a key only known to X, and $C_1, C_2, ..., C_{52}$ are the digital versions of the 52 cards.

$$
\begin{aligned}
M1:\ A \rightarrow B:&\ \{C_{i1}\}_{Ka}, \{C_{i2}\}_{Ka}, ...\{C_{i52}\}_{Ka} \\
M2:\ B \rightarrow A:&\ \{\{C_{j1}\}_{Ka}\}_{Kb}, \{\{C_{j2}\}_{Ka}\}_{Kb}, ...\{\{C_{j47}\}_{Ka}\}_{Kb}, \\
&\ \{C_{j'1}\}_{Ka}, \{C_{j'2}\}_{Ka}, ...\{C_{j'5}\}_{Ka} \\
M3:\ A \rightarrow B:&\ \{C_{i'1}\}_{Kb}, \{C_{i'2}\}_{Kb}, ...\{C_{i'5}\}_{Kb}
\end{aligned}
$$

where the i indices are a permutation of [1..52], the j and j' indices partition [1..52], and the i' indices are a subset of the j indices.

In essence,

1. Alice masks the "cards" for Bob
2. Bob chooses Alice's hand by not encrypting those five, randomly chosen quantities.
3. Alice can decrypt and see only these five, while the other 47 are hidden by Bob's encryption.
4. Alice then chooses Bob's hand blindly by removing her encryption from five of the doubly encrypted quantities and sending these to Bob
5. Bob can then remove his encryption and see his hand.

Nota bene: This assumes that encryption with the two keys commutes! i.e., $D(E(E(M, K_A), K_B), K_A) = E(M, K_B)$, which may not be so.

5.5 A Variant - Secret Item Distribution

1. A sends a sequence of items encrypted with K_A
2. B selects one (can't see what it is yet) and encrypts with K_B, sends back to A
3. A decrypts the returned doubly encrypted item using K_A, so that it is now only encrypted using K_B, and returns to B
4. B decrypts using K_B and gets item.

Possible Problem - A knows that whatever B got it had to be one of the items that A had sent out before receiving B's selection....

5.6 Secure Voting

5.6.1 Statement

N voters must be able to cast a ballot such that every voter knows

1. their vote counted,
2. every other voter voted just once,
3. nobody else knows how they voted, and
4. the final results (all the vote contents, but without IDs)

Let E_A and D_A be encryption and decryption (using public key system) for user A. Let $R_i(m, r)$ be a randomizing encryption, in which user U_i embeds random string r in message m and encrypts (so that two identical messages will look different). Only U_i knows R_i or R_i^{-1} (how to decrypt and extract m from $R_i(m, r)$.

5.6.2 Protocol (Original 3 voter version)

1. Each user U_i chooses a vote v_i,
2. encrypts it using the public keys (in order),
3. and then applies randomizing encryptions (again in order), producing

$$R_1(R_2(R_3(E_1(E_2(E_3(v_i))))))$$

4. and sends this secretly to U_1.

Note: Each voter can recognize any of the partial results in this chain for their own vote.

- Phase I - Shuffling the votes U_1 can tell who sent what, but can't tell what each is (due to the randomizing encryptions).
  1. U_1 verifies that U_1's vote is there, then produces for each i
  $$R_2(R_3(E_1(E_2(E_3(v_i)))))$$
  and sends these secretly to U_2.
  2. U_2 verifies that U_2's vote is there, then produces for each i
  $$R_3(E_1(E_2(E_3((v_i)))))$$
  and sends these secretly to U_3.
  3. U_3 verifies that U_3's vote is there, then produces for each i
  $$E_1(E_2(E_3(v_i)))$$
  and sends these secretly to U_1.

- Phase II - Revealing the results
  1. U_1 then decrypts, sends
  $$E_2(E_3(v_i))$$
  to U_2 and signatures to U_2 and U_3.
  2. U_1 then decrypts, sends
  $$E_3(v_i)$$
  to U_3 and signatures to U_1 and U_3.
  3. U_3 then decrypts, sends
  $$v_i$$
  and signatures to all.

5.7 Probabilistic Transfer (a.k.a.Oblivious Transfer)

5.7.1 Statement:

A wants to transfer some preset message M to B with probability one half. A must be able to verify that B did get it if B claims to have gotten it, and B must be able to verify that A did not cheat if B does not get it.

5.7.2 Original Version

Here, Nancy and Pete flip a fair coin in a distributed manner.

- K is a symmetric key generated by Nancy,
- K_i and K_j are public keys generated by Pete, and
- K_i^{-1} and K_j^{-1} are their inverses, respectively.
- M is a message stating that the coin has value *heads*.
- $\{X\}_K$ is a message X that has been processed using key K, which will either encrypt X or decrypt it, depending on the application (i.e., $\{X\}_K$ encrypts X with symmetric key K, but if $X = \{Y\}_K$, then $\{X\}_K = \{\{Y\}_K\}_K = Y$).
- x is either i or j (Nancy's choice),
- and y is either i or j (Pete's choice).

$$
\begin{aligned}
M1 &: \ P \rightarrow N : \ K_i, K_j \\
M2 &: \ N \rightarrow P : \ \{K\}_{Kx} \\
M3 &: \ P \rightarrow N : \ \{M\}_{\{\{K\}_{Kx}\}_{Ky}^{-1}} \\
M4 &: \ N \rightarrow P : \ \{\{M\}_{\{\{K\}_{Kx}\}_{Ky}^{-1}}\}_K, K \\
M5 &: \ P \rightarrow N : \ M, K_i^{-1}, K_j^{-1}
\end{aligned}
$$

Discussion
If $x = y$, then Pete uses K to encrypt M, and Nancy sees M when she decrypts with K. Otherwise, Nancy sees garbage, since a different "key" was used to encrypt than was used to decrypt M. Nancy "wins" if she sees M, otherwise Pete "wins." In the end, Pete can use Nancy's K (sent in message $M4$) to verify that Nancy did not cheat at $M2$ or $M4$ (she used the public key corresponding to the private key Pete used to decrypt K to obtain the encryption key for M used in $M3$), and satisfy himself that Nancy won. Nancy can verify that Pete won by using the private key corresponding to the public key that she did not use in $M2$ to generate the same key that Pete used to encrypt M in $M3$, then decrypting that message and seeing that M really was encrypted using the "wrong" key.