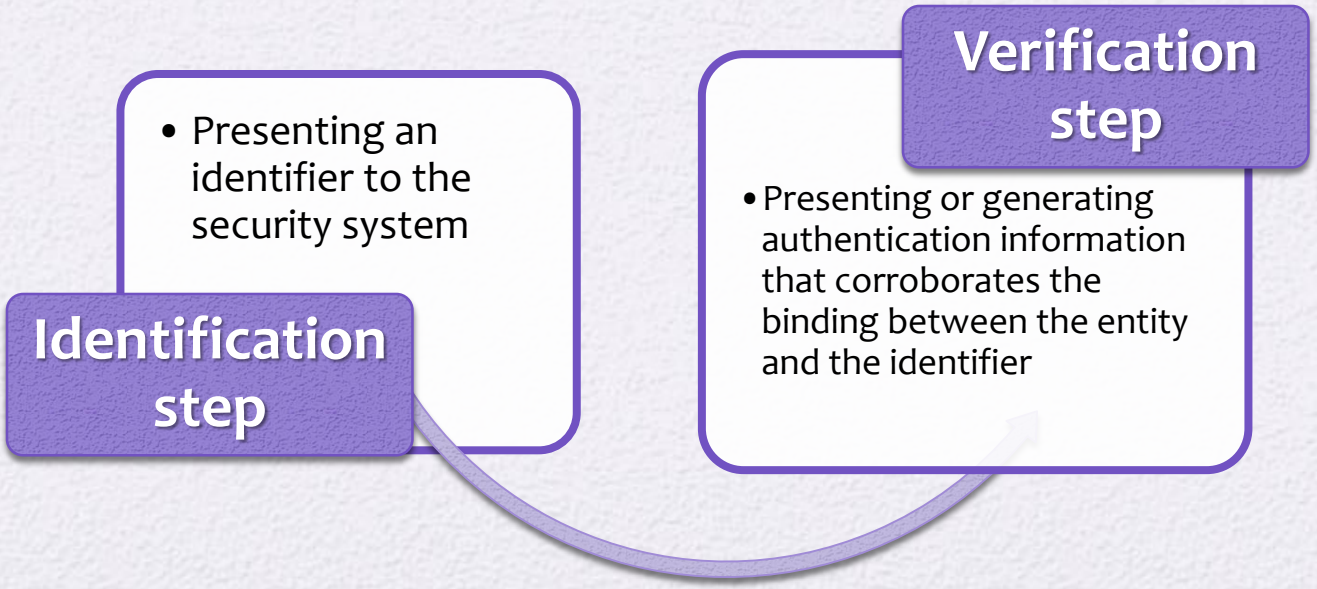
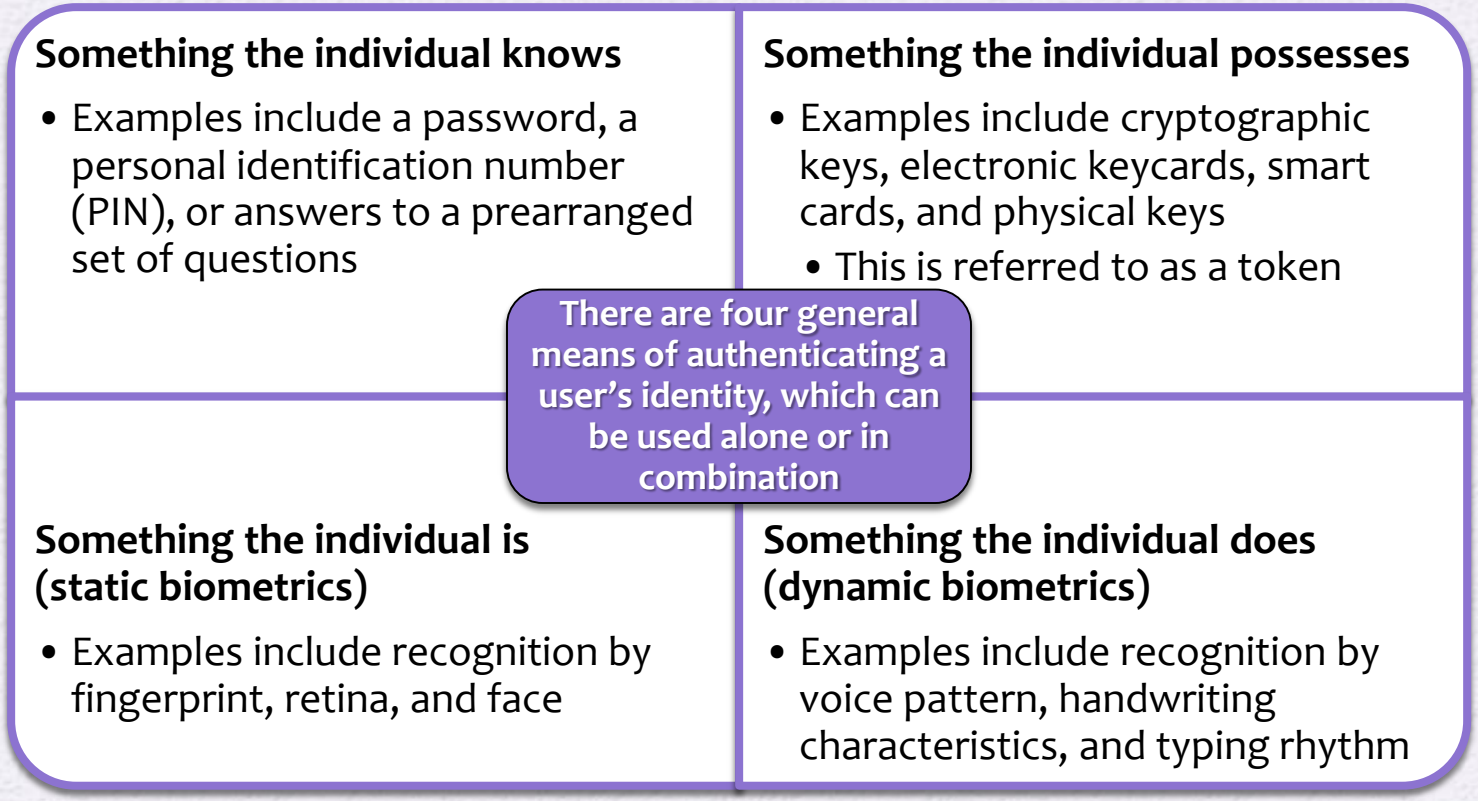
Remote User-Authentication Principles

- The process of verifying an identity claimed by or for a system entity
- An authentication process consists of two steps:



- User authentication is the basis for most types
- of access control and for user accountability. RFC 4949

Means of User Authentication



- For network-based user authentication, the most important methods involve cryptographic keys and something the individual knows, such as a password

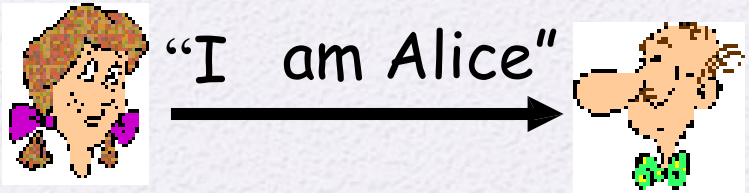
- An adversary may be able to guess or steal a password. Similarly, an adversary may be able to forge or steal a token.
- A user may forget a password or lose a token.
- Furthermore, there is a significant administrative overhead for managing password and token information on systems

- With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.
- For network-based user authentication, the most important methods involve cryptographic keys and something the individual knows, such as a password.

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

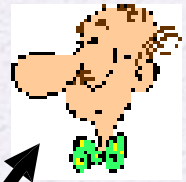


Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

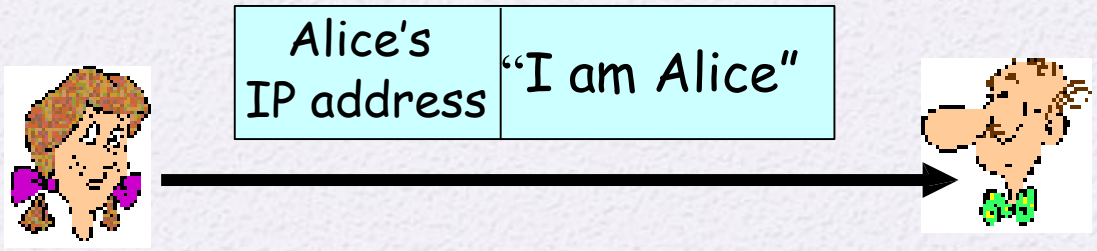


"I am Alice"

in a network,
Bob can not "see"
Alice, so Trudy simply
declares
herself to be Alice

Authentication: another try

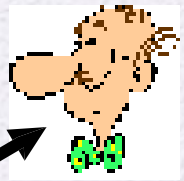
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Failure scenario??

Authentication: another try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

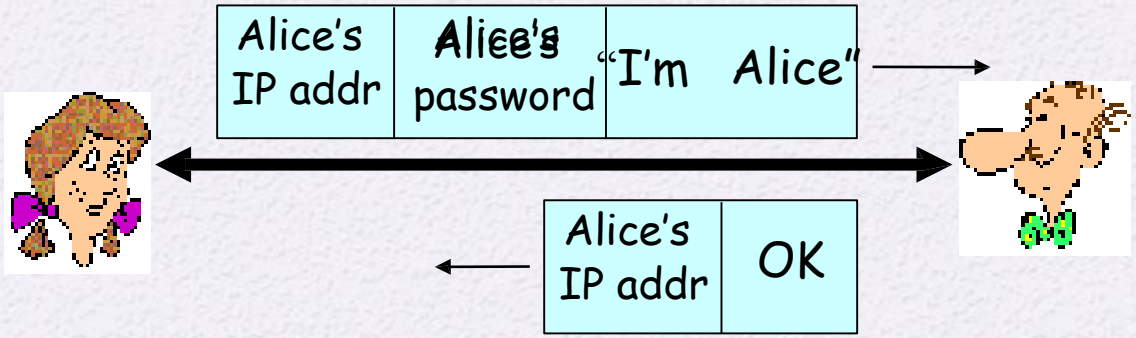


Alice's IP address	"I am Alice"

Trudy can create a packet "spoofing" Alice's address

Authentication: another try

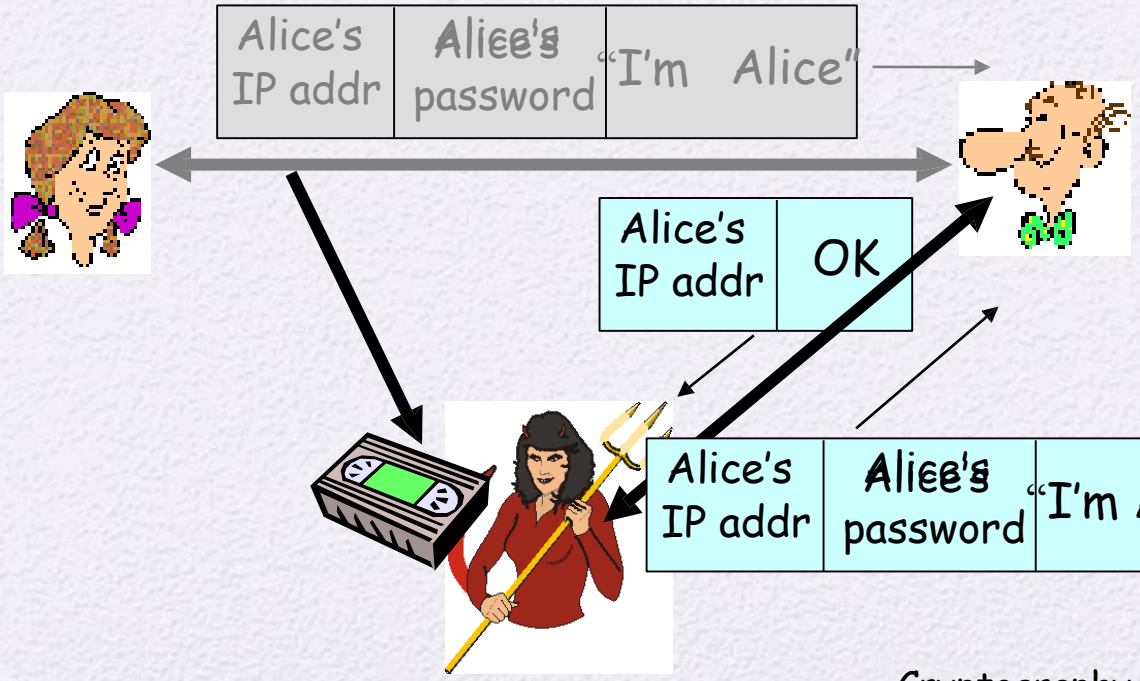
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

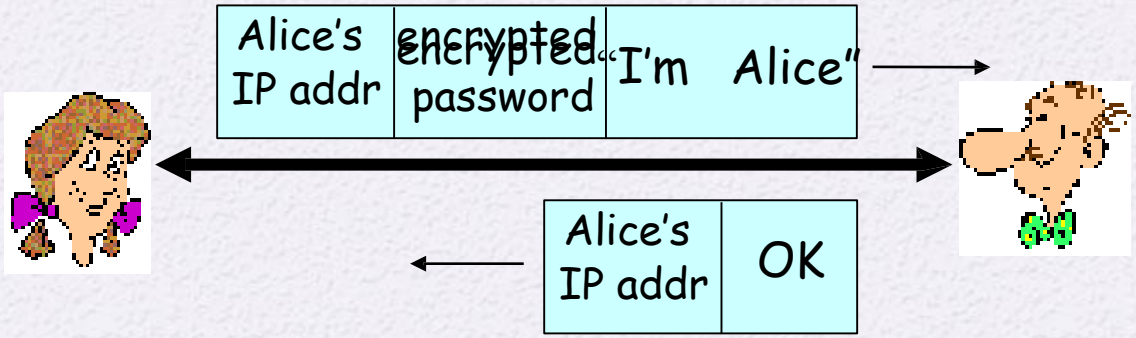


playback attack:
Trudy
records Alice's packet
and later
plays it back to Bob

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her encrypted password to "prove" it.

secret

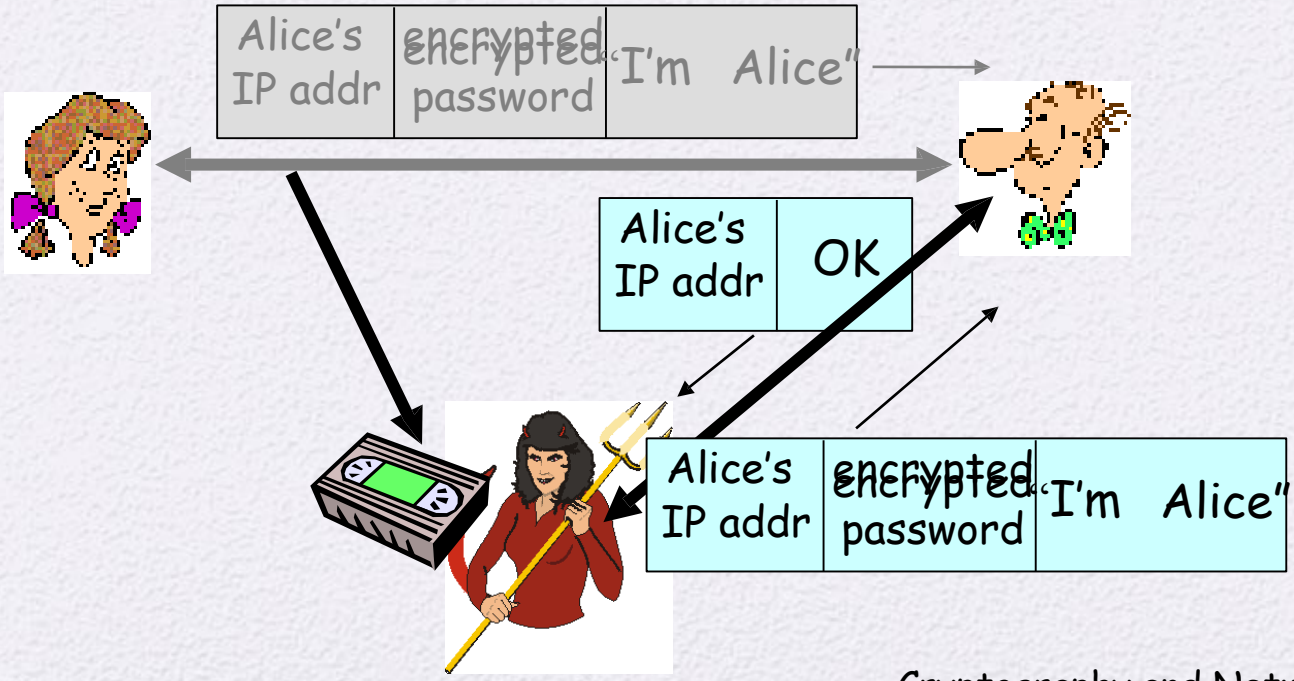


Failure scenario??

Authentication: another try

Protocol ap3.1: Alice says "I am Alice" and sends her encrypted password to "prove" it.

secret



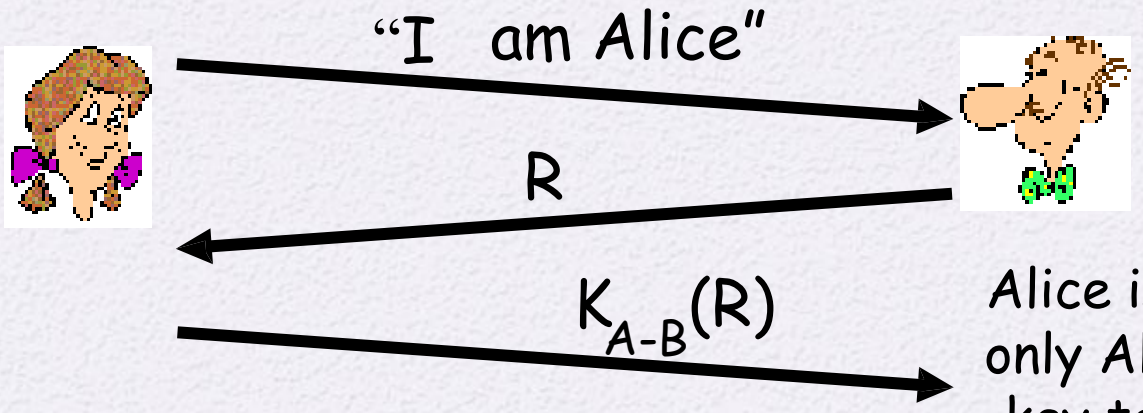
record and playback still works!

Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only once –in-a-lifetime

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



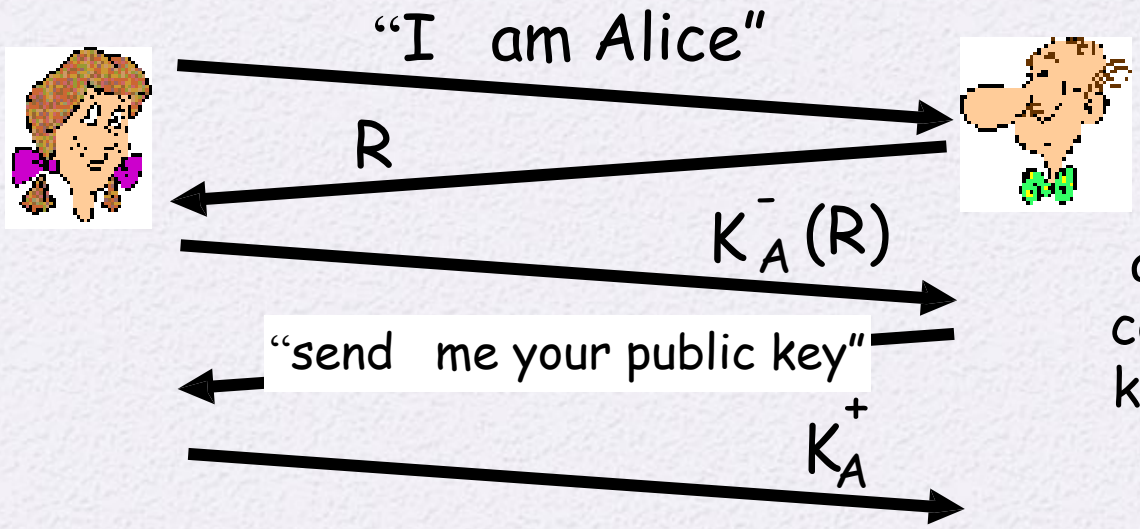
drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

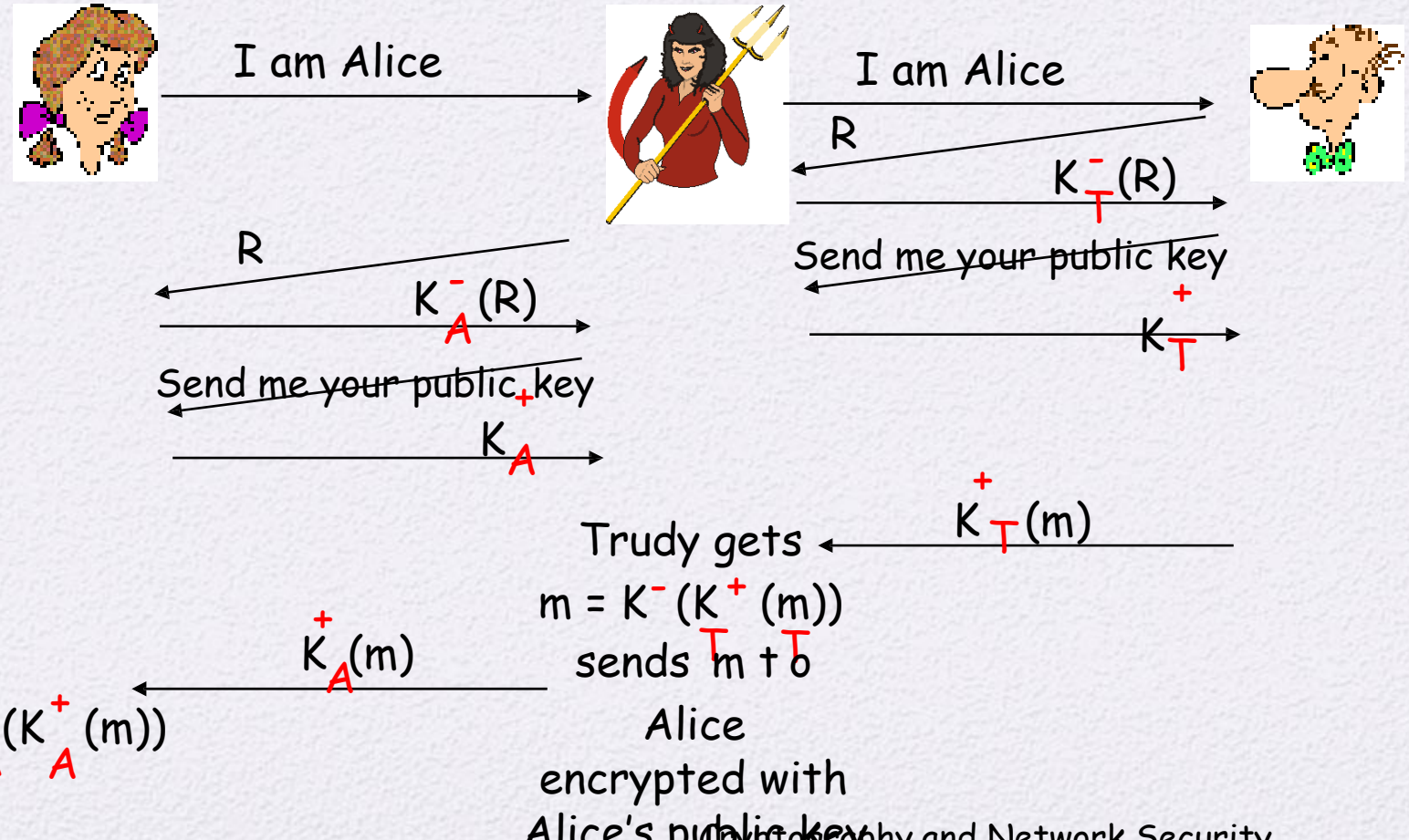
ap5.0: use nonce, public key cryptography



Bob computes $K_A^+(K_A^-(R)) = R$ and knows only Alice could have the private key, that encrypted R such that $K_A^+(K_A^-(R)) = R$

ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

Difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation)
- problem is that Trudy receives all messages as well!

Solution for this?--- public key certificate

Password Authentication

- Oldest(?) way to authenticate an entity.
- Each user has a password.
  - Host keeps a list of (user id, password).
- When a user needs to login, he sends the host his password.
  - Host checks password before granting access.

Problems with Password Authentication

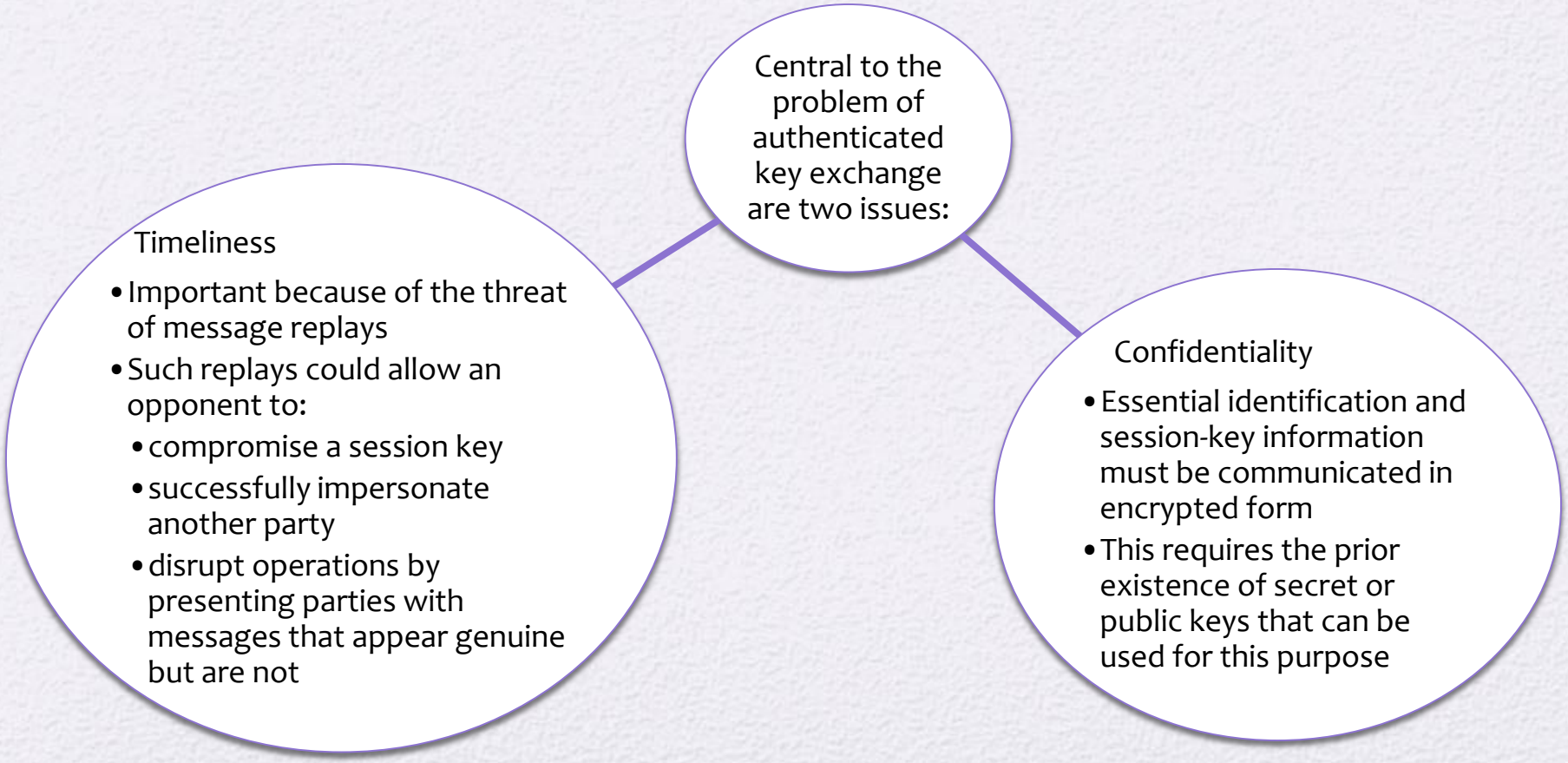
- The host's list of (user id, password) may be revealed to adversary.
  - This list becomes an attractive target of attack.
- The password may be eavesdropped in transmission.

No Password Storage in Clear

- We can address the first problem using one-way hash function.
  - Host stores H(password) instead of password.
  - Verifying password is still easy for host.
  - Adversary can't figure out password even if he sees H(password).

Mutual Authentication

- Protocols which enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys



- To prevent masquerade and to prevent compromise of session keys, essential identification and session-key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose.

Timeliness

- The second issue, timeliness, is important because of the threat of message replays. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party.

Replay Attacks

1. The simplest replay attack is one in which the opponent simply copies a message and replays it later

2. An opponent can replay a timestamped message within the valid time window

3. An opponent can replay a timestamped message within the valid time window, but in addition, the opponent suppresses the original message; thus, the repetition cannot be detected

4. Another attack involves a backward replay without modification and is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content

Approaches to Coping With Replay Attacks

- Attach a sequence number to each message used in an authentication exchange
  - A new message is accepted only if its sequence number is in the proper order
  - Difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with
  - Generally not used for authentication and key exchange because of overhead
- Timestamps
  - Requires that clocks among the various participants be synchronized
  - Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time
- Challenge/response
  - Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value

- Lamport Hash chain
- Therefore, any timestamp-based procedure must allow for a window of time sufficiently large to accommodate network delays yet sufficiently small to minimize the opportunity for attack

- The challenge-response approach is unsuitable for a connectionless type of application, because it requires the overhead of a handshake before any connectionless transmission, effectively negating the chief characteristic of a connectionless transaction.

One-Way Authentication

One application for which encryption is growing in popularity is electronic mail (e-mail)

- Header of the e-mail message must be in the clear so that the message can be handled by the store-and-forward e-mail protocol, such as SMTP or X.400
- The e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key

A second requirement is that of authentication

- The recipient wants some assurance that the message is from the alleged sender

Remote User-Authentication Using Symmetric Encryption

A two-level hierarchy of symmetric keys can be used to provide confidentiality for communication in a distributed environment

- Strategy involves the use of a trusted key distribution center (KDC)
- Each party shares a secret key, known as a master key, with the KDC
- KDC is responsible for generating keys to be used for a short time over a connection between two parties and for distributing those keys using the master keys to protect the distribution

- The use of a trusted key distribution center (KDC).
- Each party in the network shares a secret key, known as a master key, with the KDC. The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.

Needham and Schroeder

1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. A → B: $E(K_b, [K_s \| ID_A])$
4. B → A: $E(K_s, N_2)$
5. A → B: $E(K_s, f(N_2))$

Secret keys K_a and K_b are shared between A and the KDC and B and the KDC, respectively. The purpose of the protocol is to distribute securely a session key K_s to A and B. A securely acquires a new session key in step 2. The message in step 3 can be decrypted, and hence understood, only by B. Step 4 reflects B's knowledge of K_s, and step 5 assures B of A's knowledge of K_s and assures B that this is

Modified protocol

1. $A \rightarrow KDC: \quad ID_A \| ID_B$
2. $KDC \rightarrow A: \quad E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. $A \rightarrow B: \quad E(K_b, [K_s \| ID_A \| T])$
4. $B \rightarrow A: \quad E(K_s, N_1)$
5. $A \rightarrow B: \quad E(K_s, f(N_1))$

Improvement

1. $A \rightarrow B$: $ID_A \| N_a$
2. $B \rightarrow KDC$: $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. $KDC \rightarrow A$: $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

1. A initiates the authentication exchange by generating a nonce, N_a, and sending that plus its identifier to B in plaintext. This nonce will be returned to A in an encrypted message that includes the session key, assuring A of its timeliness.

1. A → B: $ID_A \| N_a$

B alerts the KDC that a session key is needed. Its message to the KDC includes its identifier and a nonce, N_b. This nonce will be returned to B in an encrypted message that includes the session key, assuring B of its timeliness.

2. B → KDC: $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$

The KDC passes on to A B's nonce and a block encrypted with the secret key that B shares with the KDC. The block serves as a "ticket" that can be used by A for subsequent authentications, as will be seen. The KDC also sends to A a block encrypted with the secret key shared by A and the KDC. This block verifies that B has received A's initial message (ID_B) and that this is a timely message and not a replay (N_a), and it provides A with a session key (K_s) and the time limit on its use (T_b).

3. KDC $\rightarrow$ A: $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$

4. A transmits the ticket to B, together with the B's nonce, the latter encrypted with the session key. The ticket provides B with the secret key that is used to decrypt $E(K_s, N_b)$ to recover the nonce. The fact that B's nonce is encrypted with the session key authenticates that the message came from A and is not a replay.

4. $A \rightarrow B: \quad E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

1. A → B: $E(K_b, [ID_A \| K_s \| T_b]) \| N'_a$
2. B → A: $N'_b \| E(K_s, N'_a)$
3. A → B: $E(K_s, N'_b)$

Suppress-Replay Attacks

- The Denning protocol requires reliance on clocks that are synchronized throughout the network
- A risk involved is based on the fact that the distributed clocks can become unsynchronized as a result of sabotage on or faults in the clocks or the synchronization mechanism
- The problem occurs when a sender's clock is ahead of the intended recipient's clock
  - An opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site
  - Such attacks are referred to as *suppress-replay attacks*