Authentication Protocols

BİL 448/548
Internet Security Protocols

Ali Aydın Selçuk

Entity Authentication

- Authentication of people, processes, etc.
- Non-cryptographic
  - Address-based (E-mail, IP, etc.)
  - Passwords
  - Biometrics
- Cryptographic
  - Symmetric key
  - Public key

Authentication Tokens

- What you know (password schemes)
- What you have (keys, smart cards, etc.)
- What you are (fingerprints, retinal scans, etc.)

Password Problems

- Eavesdropping
- Stealing password files
- On-line password guessing
- Off-line guessing attacks
  - Dictionary attacks
  - Exhaustive search
- Careless users writing down passwords

On-line Password Guessing

Careless choices (first names, initials, etc.); poor initial passwords

Defenses: After wrong guesses,

- Lock the account
  - Not desirable, can be used for DoS
- Slow down
- Alert users about unsuccessful login attempts
- Don't allow short or guessable passwords

Off-line Password Guessing

- Stealing & using password files
- Passwords should not be stored in clear. Typically, they're hashed and stored.
- Attacks:
  - Exhaustive search
  - Dictionary attacks
- Defenses:
  - Don't allow short/guessable passwords
  - Don't make password files readable
  - *Salting:* Mix a random number to each hash
    - Store <*username*, *rand*, *H*(*pwd*,*rand*)> and use *rand* to hash the input password at each login.
    - Why? How does this help to slow down the attack?

Eavesdropping

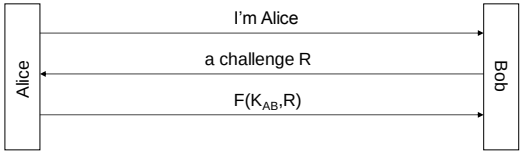
- Watching the screen
- Watching the keyboard
- Login Trojan horses. Solutions:
  - Different appearance
  - Interrupt command for login
- Keyboard sniffers. Solutions:
  - Good system administration
- Network sniffers. Solutions:
  - Cryptographic protection
  - One-time passwords

Cryptographic Authentication

- Password authentication subject to eavesdropping
- Alternative: Cryptographic challenge-response
  - Symmetric key
  - Public key

Symmetric Key Challenge-Response

An example protocol:

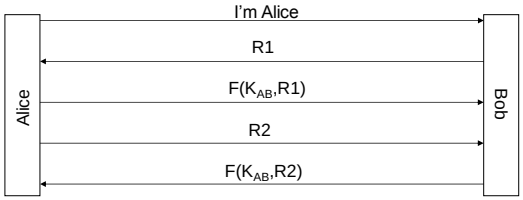


- F is either:
  - block cipher (how?)
  - hash function (how?)
- What about a stream cipher?! (As in WEP)

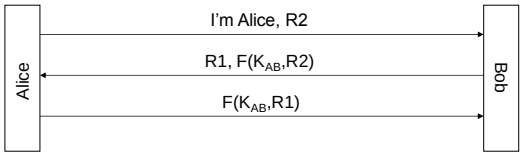
Mutual Authentication

Both Alice and Bob authenticate each other

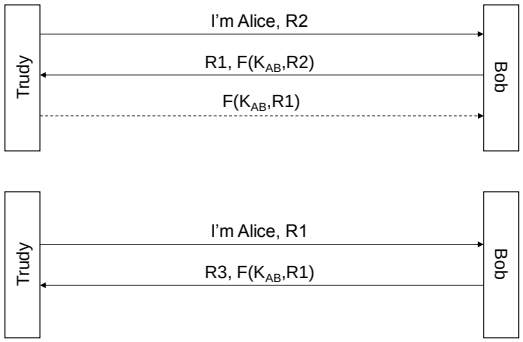
An example protocol:



Some saving:



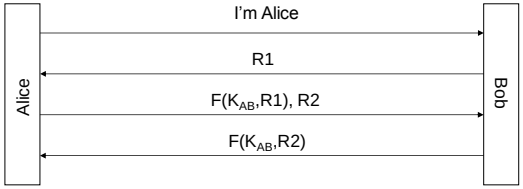
Reflection attack:



- Solutions:
  - Different keys for Alice and Bob
  - Formatted challenges, different for Alice and Bob
- Principle: Initiator should be the first to prove its identity

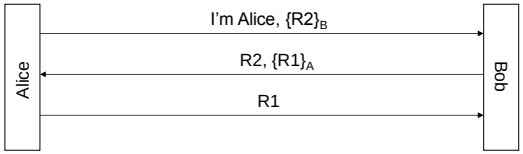
Another weakness: Trudy can do dictionary attack against K_{AB} acting as Alice, without eavesdropping.

Solution against both problems:



(Dictionary attack still possible if Trudy can impersonate Bob.)

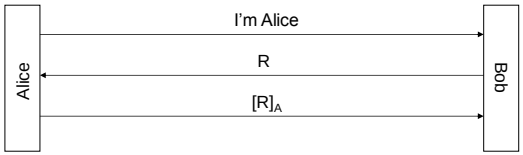
Mutual authentication with PKC:



- Problem: How can the public/private keys be remembered by ordinary users?
- They can be stored in an electronic token (USB), or can be retrieved from a server with password-based authentication & encryption.

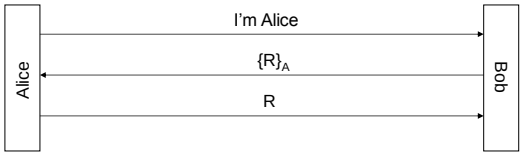
Public Key Challenge-Response

By signature:



Public Key Challenge-Response

By decryption:



Public Key C-R Pitfalls

- Problem: Bob (or Trudy) can get Alice to sign/decrypt any text he chooses.
- Solutions:
  - Never use the same key for different purposes (e.g., for login and signature)
  - Have formatted challenges

Nonces

- *Nonce*: Something created for one particular occasion
- Nonce types:
  - Random numbers
  - Timestamps
  - Sequence numbers
- Random nonces: if unpredictability is needed
- Timestamps: require syn. clocks
- Obtaining random nonces from timestamps: Encrypt/hash the timestamp with a secret key.
- Seq.no.: Fine if predictability is not a problem