

# CS 4803

## Computer and Network Security

Alexandra (Sasha) Boldyreva  
Authenticated key exchange

1

### Diffie-Hellman key exchange

- Secure against passive eavesdropping...
- ...but insecure against a man-in-the-middle attack

2

### Adding key exchange

- Not sufficient to simply “add on” key establishment before/after authentication
- Need “authenticated key exchange”

3

### Overview

- Protocol design is subtle
  - Small changes can make a protocol insecure!
  - Historically, designed in an “ad-hoc” way, by checking protocol for known weaknesses
  - Great example of where provable security helps!

4

## Example

- “Reverse” challenge-response
  - I.e., send a ciphertext and have user decrypt it
  - Mutual authentication (if decrypts “validly”)??
- Weaknesses?
  - Uses encryption for authentication

5

## Example

- User sends time,  $MAC_K(\text{time})$ 
  - What if she had used encryption, or a hash?
  - What about just sending  $MAC_K(\text{time})$ ?
- Considerations?
  - Requires (loosely) synchronized clocks
  - Must guard against replay...
  - What if user has same key on multiple servers?
  - Clock reset attacks
  - No mutual authentication

6

## Adding mutual authentication

- Double challenge-response (symmetric key) in 4 rounds
- Variant in which user sends nonce first?
  - Insecure (reflection attack)...
  - Also vulnerable to off-line password guessing without eavesdropping
  - To improve security, make protocol asymmetric
  - Security principle: let initiator prove its identity first

7

## Using timestamps?

- User sends time,  $MAC_K(\text{time})$ , server responds with  $MAC_K(\text{time}+1)$
- Vulnerabilities?

8

## Establishing a session key

- Double challenge-response; compute session key as  $F_K(R+2)$ 
  - Secure against passive attacks if  $F$  is a pseudorandom permutation...
  - Active attacks? And how to fix it...

9

## Public-key based...

- Include  $E_{pk}(\text{session-key})$  in protocol?
- Encrypt session-key and sign the result?
  - Potentially vulnerable to replay attacks
- User sends  $E(R_1)$ ; server sends  $E(R_2)$ ; session key is  $R_1+R_2$ 
  - Reasonable...

10

## One-way authentication

- If only the server has a known public key (e.g., SSL)
  - Server sends  $R$
  - Client sends  $E_{pk}(R, \text{password}, \text{session-key})$
- Insecure in general!!!
  - But secure if encryption scheme is chosen appropriately
- Can extend to give mutual authentication

11

## Authenticated Diffie-Hellman

- Add signatures/MACs and nonces to Diffie-Hellman protocol
- Variation: HMQV (improved MQV)

12

## Using session keys

- Generally, want to provide both secrecy and integrity for subsequent conversation
  - Use encrypt-then-MAC
  - Use sequence numbers to prevent replay attacks
  - Periodically refresh the session key

13

## Mediated authentication

- E.g., using KDC
- Simple protocol:
  - Alice requests to talk to Bob
  - KDC generates  $K_{AB}$  and sends it to Alice and Bob, encrypted with their respective keys
  - Note: no authentication here, but impostor can't determine  $K_{AB}$

14

## Improvement...

- Have KDC send to Alice the encryption of  $K_{AB}$  under Bob's key
  - Reduces communication load on KDC
  - Resilient to message delays in network

15

## Needham-Schroeder

- $A \rightarrow KDC: N_1, \text{Alice}, \text{Bob}$
- $KDC \rightarrow A: K_A(N_1, \text{Bob}, K_{AB}, \text{ticket}), \text{ where ticket} = K_B(K_{AB}, \text{Alice})$
- $A \rightarrow B: \text{ticket}, K_{AB}(N_2)$
- $B \rightarrow A: K_{AB}(N_2-1, N_3)$
- $A \rightarrow B: K_{AB}(N_3-1)$

16

## Analysis?

- $N_1$  assures Alice that she is talking to KDC
  - Prevents key-replay, helps prevent attack when Bob's key is compromised and then changed
- Important: authenticate "Bob" in message 2, and "Alice" in ticket
- Uses encryption to authenticate... ☹
  - Leads to actual flaw if, e.g., ECB mode is used!
- Vulnerable if Alice's key is compromised
  - Bob's ticket is always valid
  - Use timestamps, or request (encrypted) nonce from Bob at the very beginning of the protocol

17

## Otway-Rees

- $A \rightarrow B$ :  $N_C, K_A(N_A, N_C, \text{Alice}, \text{Bob})$
- $B \rightarrow \text{KDC}$ :  $K_A(\dots), K_B(N_B, N_C, \text{Alice}, \text{Bob})$ 
  - KDC checks that  $N_C$  is the same...
- $\text{KDC} \rightarrow B$ :  $N_C, K_A(N_A, K_{AB}), K_B(N_B, K_{AB})$
- $B \rightarrow A$ :  $K_A(\dots)$
- $A \rightarrow B$ :  $K_{AB}(\text{timestamp})$ 
  - Note: KDC already authenticated Bob

18

## Analysis?

- $N_C$  should be unpredictable, not just a nonce
  - Otherwise, can impersonate B to KDC
    - Send first message: (next  $N_C$ ), "garbage"
    - B forwards to KDC along with encryption of the next  $N_C$
    - Next time A initiates a conversation, replay previous message from B
- Still uses encryption for authentication... ☹
  - Serious attack if ECB is used
    - Replace  $K_{AB}$  with  $N_C$

19

## Kerberos

- (May discuss in more detail later)
- $A \rightarrow \text{KDC}$ :  $N_1, \text{Alice}, \text{Bob}$
- $\text{KDC} \rightarrow A$ :  $K_A(N_1, \text{Bob}, K_{AB}, \text{ticket})$ , where  $\text{ticket} = K_B(K_{AB}, \text{Alice}, \text{expiration time})$
- $A \rightarrow B$ :  $\text{ticket}, K_{AB}(\text{time})$
- $B \rightarrow A$ :  $K_{AB}(\text{time}+1)$

20