

# Key Establishment and Authentication Protocols

EECE 412

# where we are

Protection					Assurance				
Authorization		Accountability		Availability		Requirements Assurance	Design Assurance	Development Assurance	Operational Assurance
Access Control	Data Protection	Audit		Service Continuity	Disaster Recovery				
		Non-Repudiation							
Authentication									
Cryptography									

“The security of a cryptosystem must not depend on keeping secret the crypto-algorithm. The security depends only on keeping secret the key”

Auguste Kerckhoff von Nieuwenhof

Dutch linguist

1883

# session key with mutual authentication using symmetric key



Alice

“I’m Alice”,  $R_A$

---

$R_B$ ,  $E(\text{“Bob”}, R_A, K_{AB})$

---

$E(\text{“Alice”}, R_B, K_S, K_{AB})$

---



Bob

# FPS session key with mutual authentication using symmetric key



Alice

“I’m Alice”,  $R_A$

$R_B, E(\text{“Bob”}, R_A, g^b \text{ mod } p, K_{AB})$

$E(\text{“Alice”}, R_B, g^a \text{ mod } p, K_{AB})$



Bob

# Outline

1. Diffie-Hellman key exchange (Stamp 4.4, Anderson 5.7.2.1, 5.7.2.2)
2. mutual authentication in networks (Stamp 9.1-9.3.3, Anderson Chapter 3)
3. perfect forward secrecy (Stamp 9.3.4, 9.3.5, 9.6, 9.7)

# learning objectives for this module

You should be able to

- analyze key establishment and authentication protocols and identify their vulnerabilities
- improve or design new key establishment and authentication protocols

# Notation

- $X \rightarrow Y : \{ Z \parallel W \}_{k_{X,Y}} == E(Z, W, k_{X,Y})$
- X sends Y the message produced by concatenating Z and W enciphered by key  $k_{X,Y}$ , which is shared by users X and Y
- $A \rightarrow T : \{ Z \}_{k_A} \parallel \{ W \}_{k_{A,T}}$
- A sends T a message consisting of the concatenation of Z enciphered using  $k_A$ , A's key, and W enciphered using  $k_{A,T}$ , the key shared by A and T
- $r_1, r_2$  nonces (“nonrepeating” random numbers)



# Diffie-Hellman Key Exchange

# important trivia

- Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)
- A “key exchange” algorithm
  - Used to establish a shared symmetric key
- Not for encrypting or signing
- Security rests on difficulty of **discrete log** problem:  
given  $g$ ,  $p$ , and  $g^k \bmod p$  find  $k$

# how it works

- Let  $p$  be prime, let  $g$  be a **generator**
  - For any  $x \in \{1,2,\dots,p-1\}$  there is  $n$  s.t.  $x = g^n \pmod p$
- 1. Alice selects secret value  $a$
- 2. Bob selects secret value  $b$
- 3. Alice sends  $g^a \pmod p$  to Bob
- 4. Bob sends  $g^b \pmod p$  to Alice
- 5. Both compute shared secret  $g^{ab} \pmod p$ 
  - Shared secret can be used as symmetric key

# why it's hard to attack

- Suppose that Bob and Alice use  $g^{ab} \bmod p$  as a symmetric key
- Trudy can see  $g^a \bmod p$  and  $g^b \bmod p$
- Note  $g^a g^b \bmod p = g^{a+b} \bmod p \neq g^{ab} \bmod p$
- If Trudy can find  $a$  or  $b$ , system is broken
- If Trudy can solve **discrete log** problem, then she can find  $a$  or  $b$

# the protocol

- **Public:**  $g$  and  $p$
- **Secret:** Alice's exponent  $a$ , Bob's exponent  $b$



Alice,  $a$

$$g^a \bmod p$$

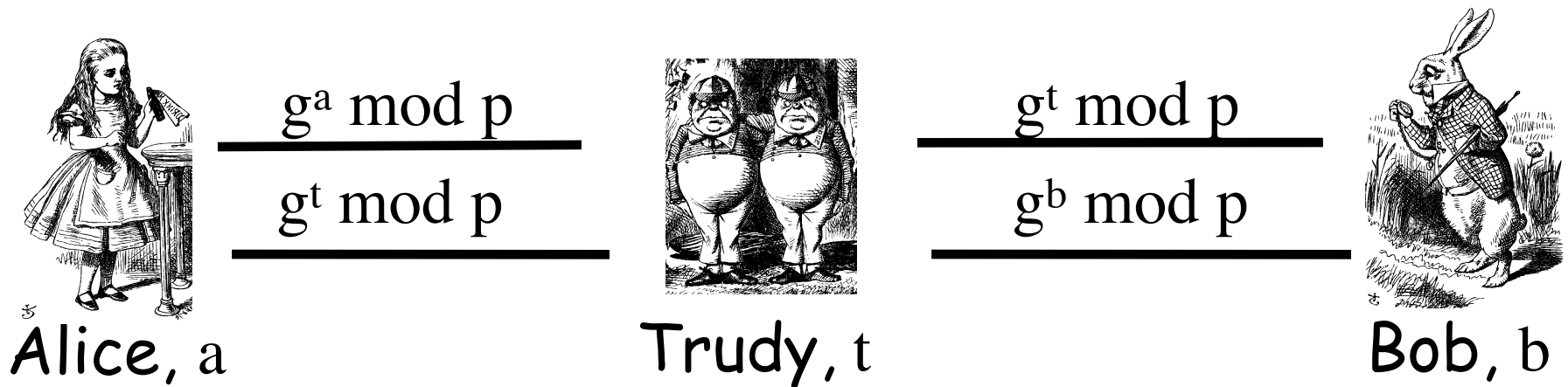
$$g^b \bmod p$$



Bob,  $b$

- Alice computes  $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- Bob computes  $(g^a)^b = g^{ab} \bmod p$
- Could use  $K = g^{ab} \bmod p$  as symmetric key

# Man-in-the-Middle Attack



- Trudy shares secret  $g^{at} \bmod p$  with Alice
- Trudy shares secret  $g^{bt} \bmod p$  with Bob
- Alice and Bob don't know Trudy exists!

# how to prevent MiM attack?

- Encrypt DH exchange with symmetric key
- Encrypt DH exchange with public key
- Sign DH values with private key
- Other?

You **MUST** be aware of MiM attack on Diffie-Hellman

# Authentication Protocols



# basics

- Alice must prove her identity to Bob
  - Alice and Bob can be humans or computers
- May also require Bob to prove he's Bob (mutual authentication)
- May also need to establish a session key
- May have other requirements, such as
  - Use only public keys
  - Use only symmetric keys
  - Use only a hash function
  - Anonymity, plausible deniability, etc., etc.

# why authentication can be hard?

- relatively simple on a stand-alone computer
  - “Secure path” is the primary issue
  - main concern is an attack on authentication software
- much more complex over a network
  - attacker can passively observe messages
  - attacker can replay messages
  - active attacks may be possible (insert, delete, change messages)

# simple authentication



Alice

“I’m Alice”

Prove it

My password is “frank”



Bob

- Simple and may be OK for standalone system
- But insecure for networked system
  - Subject to a replay attack (next 2 slides)
  - Bob must know Alice’s password

# authentication attack



Alice

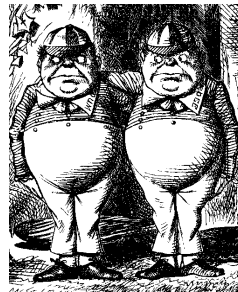
“I’m Alice”

Prove it

My password is “frank”

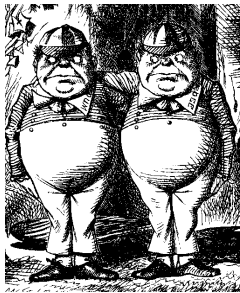


Bob



Trudy

# authentication Attack



Trudy

“I’m Alice”

---

Prove it

---

My password is “frank”

---



Bob

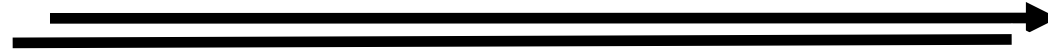
- This is a **replay** attack
- How can we prevent a replay?

# Simple Authentication



Alice

I'm Alice, My password is "frank"



Bob

- More efficient...
- But same problem as previous version

# Better Authentication



Alice

“I’m Alice”

Prove it

$h(\text{Alice's password})$



Bob

- Better since it hides Alice’s password
  - From both Bob and attackers
- But still subject to replay

# challenge-response

- To prevent replay, challenge-response used
- Suppose Bob wants to authenticate Alice
  - Challenge sent from Bob to Alice
  - Only Alice can provide the correct response
  - Challenge chosen so that replay is not possible
- How to accomplish this?
  - Password is something only Alice should know...
  - For freshness, a “number used once” or **nonce**



# simple challenge-response



Alice

“I’m Alice”

Nonce

$h(\text{Alice's password, Nonce})$



Bob

- Nonce is the **challenge**
- The hash is the **response**
- Nonce prevents replay, insures freshness
- Password is something Alice knows
- Note that Bob must know Alice's password

# general challenge-response



Alice

“I’m Alice”

---

Nonce

---

Something that could only be

---

from Alice (and Bob can verify)



Bob

- What can we use to achieve this?
- Hashed pwd works, crypto might be better

# symmetric key notation

- Encrypt plaintext  $P$  with key  $K$

$$C = E(P,K)$$

- Decrypt ciphertext  $C$  with key  $K$

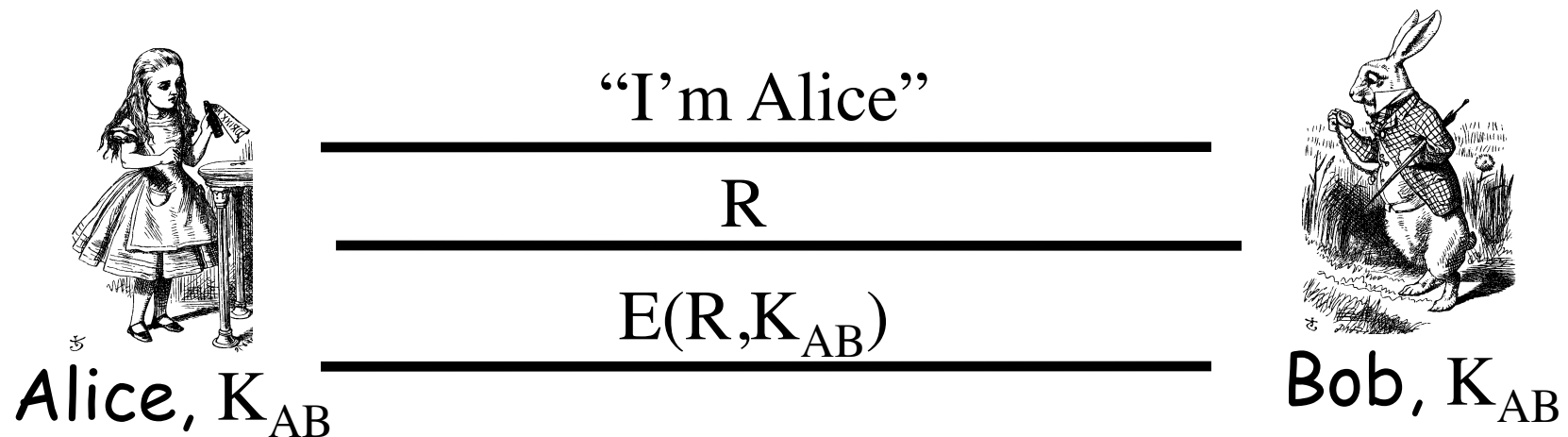
$$P = D(C,K)$$

- Here, we are concerned with attacks on **protocols**, not directly on the crypto
- We assume that crypto algorithm is secure

# authentication with symmetric key

- Alice and Bob share symmetric key  $K_{AB}$
- key  $K_{AB}$  known only to Alice and Bob
- authenticate by proving knowledge of shared symmetric key
- how to accomplish this?
  - must not reveal key
  - must not allow replay attack

# authentication with symmetric key



- Secure method for Bob to authenticate Alice
- Alice does not authenticate Bob
- Can we achieve mutual authentication?

# mutual authentication?



Alice

“I’m Alice”, R

$E(R, K_{AB})$

$E(R, K_{AB})$



Bob

- What’s wrong with this picture?
- “Alice” could be Trudy (or anybody else)!

# Mutual Authentication

- Since we have a secure one-way authentication protocol...
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has to work...

# Mutual Authentication



Alice

“I’m Alice”,  $R_A$

$R_B, E(R_A, K_{AB})$

$E(R_B, K_{AB})$

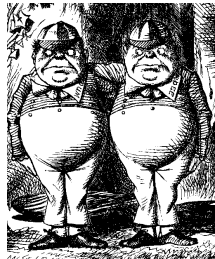


Bob

- This provides mutual authentication
- Is it secure?



# attack on mutual authentication



Trudy

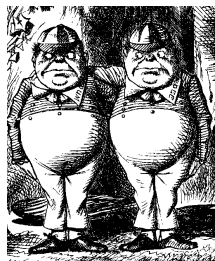
1. "I'm Alice",  $R_A$

2.  $R_B, E(R_A, K_{AB})$

5.  $E(R_B, K_{AB})$



Bob



Trudy

3. "I'm Alice",  $R_B$

4.  $R_C, E(R_B, K_{AB})$



Bob

# Notes on Mutual Authentication

- Our one-way authentication protocol **not** secure for mutual authentication
- Protocols are subtle!
- The “obvious” thing may not be secure
- Also, if assumptions or environment changes, protocol may not work
  - This is a common source of security failure
  - For example, Internet protocols

# mutual authentication with symmetric key



Alice

“I’m Alice”,  $R_A$

$R_B$ ,  $E(\text{“Bob”}, R_A, K_{AB})$

$E(\text{“Alice”}, R_B, K_{AB})$



Bob

- Do these “insignificant” changes help?
- Yes!

# session key with mutual authentication using symmetric key



Alice

“I’m Alice”,  $R_A$

---

$R_B$ ,  $E(\text{“Bob”}, R_A, K_{AB})$

---

$E(\text{“Alice”}, R_B, K_S, K_{AB})$

---



Bob

# Perfect Forward Secrecy

# Perfect Forward Secrecy

- The concern...
  - Alice encrypts message with shared key  $K_{AB}$  and sends ciphertext to Bob
  - Trudy records ciphertext and later attacks Alice's (or Bob's) computer to find  $K_{AB}$
  - Then Trudy decrypts recorded messages

**Perfect forward secrecy (PFS):** Trudy cannot later decrypt recorded ciphertext

- Even if Trudy gets key  $K_{AB}$  or other secret(s)
- Is PFS possible?

# Perfect Forward Secrecy

- For perfect forward secrecy, Alice and Bob cannot use  $K_{AB}$  to encrypt
- Instead they must use a **session key**  $K_S$  and forget it after it's used
- Problem: How can Alice and Bob agree on session key  $K_S$  and insure PFS?

# naïve session key protocol



Alice,  $K_{AB}$

$E(K_S, K_{AB})$

$E(\text{messages}, K_S)$



Bob,  $K_{AB}$

- Trudy could also record  $E(K_S, K_{AB})$
- If Trudy gets  $K_{AB}$ , she gets  $K_S$



# perfect forward secrecy

- Can use **Diffie-Hellman** for PFS
- Recall Diffie-Hellman: public  $g$  and  $p$



Alice,  $a$

$$g^a \bmod p$$

$$g^b \bmod p$$



Bob,  $b$

- But Diffie-Hellman is subject to MiM
- How to get PFS and prevent MiM?

# PFS session key via DH



Alice,  $a$

$$E(g^a \bmod p, K_{AB})$$

$$E(g^b \bmod p, K_{AB})$$



Bob,  $b$

- Session key  $K_S = g^{ab} \bmod p$
- Alice forgets  $a$ , Bob forgets  $b$

## Ephemeral Diffie-Hellman

- Not even Alice and Bob can later recover  $K_S$
- Other ways to do PFS?

# mutual authentication with symmetric key



Alice

“I’m Alice”,  $R_A$

$R_B$ ,  $E(\text{“Bob”}, R_A, K_{AB})$

$E(\text{“Alice”}, R_B, K_{AB})$



Bob

- Do these “insignificant” changes help?
- Yes!

# FPS session key with mutual authentication using symmetric key



Alice

“I’m Alice”,  $R_A$

$R_B, E(\text{“Bob”}, R_A, g^b \text{ mod } p, K_{AB})$

$E(\text{“Alice”}, R_B, g^a \text{ mod } p, K_{AB})$



Bob

# Outline

1. Diffie-Hellman key exchange (Stamp 4.4, Anderson 5.7.2.1)
2. mutual authentication in networks (Stamp 9.1-9.3.3)
3. perfect forward secrecy (Stamp 9.3.4, 9.3.5)

# learning objectives for this module

You should be able to

- analyze key establishment and authentication protocols and identify their vulnerabilities
- improve or design new key establishment and authentication protocols