

---

# KEY AGREEMENT PROTOCOLS

---

**CIS 400/628 — Spring 2005**  
**Introduction to Cryptography**

**This is based on Chapter 13 of Trappe and Washington**

# DIFFIE-HELLMAN KEY EXCHANGE

Alice & Bob want to exchange a ton of data using the nice & fast AES cryptosystem. — **But first they have to agree on a key.**

## Diffie-Hellman

### Setup

$p$ , a large prime **(Public)** and  $\alpha$ , a prim. elem. of  $\mathbb{Z}_p^*$  **(Public)**

### Alice

Chooses  $x \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$  **(Private)** and sends  $\alpha^x \pmod{p}$  to Bob.

### Bob

Chooses  $y \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$  **(Private)** and sends  $\alpha^y \pmod{p}$  to Alice.

### Alice

Computes  $k = (\alpha^y)^x = \alpha^{x \cdot y} \pmod{p}$ .

### Bob

Computes  $k = (\alpha^x)^y = \alpha^{x \cdot y} \pmod{p}$ .

### Eve

Knows  $\alpha^x \pmod{p}$  and  $\alpha^y \pmod{p}$ . Wants  $\alpha^{x \cdot y} \pmod{p}$ .

# THE MAN-IN-THE-MIDDLE ATTACK

Alice  $\longleftrightarrow$  Eve  $\longleftrightarrow$  Bob

## Eve

Chooses  $z \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$ .

Intercepts  $\alpha^x$  and  $\alpha^y$ .

Sends  $\alpha^z$  to Alice and Bob.

Eve computes  $k_{AE} = (\alpha^x)^z$  and  $k_{BE} = (\alpha^y)^z$ .

## Alice

believes she has exchanged a key with Bob.

## Bob

believes he has exchanged a key with Alice.

## Eve

reads everything & sends whatever she wants,

spoofing Alice & Bob.

**We need to fix this!!**

# STATION TO STATION (STS) PROTOCOL

Use signatures & a trusted authority (Trent)  
to defend against man-in-the-middle.

## Setup

Each user  $U$  has

$\text{sig}_U$  - a signature algorithm

$\text{ver}_U$  - a verification algorithm (established by Trent)

$p$ , a prime

$\alpha$ , a prim. elem. of  $\mathbb{Z}_p^*$

## Alice

Chooses  $x \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$  and computes  $\alpha^x \pmod{p}$ .

## Bob

Chooses  $y \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$  and computes  $\alpha^y \pmod{p}$ .

More...

## STATION TO STATION, CONTINUED

**Alice**

Sends  $\alpha^x$  to Bob.

**Bob**

Computes  $k = (\alpha^x)^y$ .

$E_k(\cdot)$  &  $D_K(\cdot)$  - say AES

Sends  $\alpha^y$  and  $E_K(\text{sig}_B(\alpha^y, \alpha^x))$  to Alice.

**Alice**

Computes  $K = (\alpha^y)^x$ .

Decrypts  $E_K(\text{sig}_B(\alpha^y, \alpha^x))$  and obtains  $\text{sig}_B(\alpha^y, \alpha^x)$ .

Asks Trent to verify that  $\text{ver}_B$  is Bob's verification alg.

Uses  $\text{ver}_B$  to verify Bob's signature.

Sends  $E_K(\text{sig}_A(\alpha^x, \alpha^y))$  to Bob.

**Bob**

Decrypts  $E_K(\text{sig}_A(\alpha^x, \alpha^y))$  & obtains  $\text{sig}_A(\alpha^x, \alpha^y)$ .

Asks Trent to verify that  $\text{ver}_A$  is A's verification alg.

Uses  $\text{ver}_A$  to verify Alice's sig.

**What is Eve to do?**

# KEY PRE-DISTRIBUTION

## Key Distribution

A TA (Trent) and  $n$  users

+ a secure channel between TA and each User

TA sends  $K$  to  $n$  users securely.

## Key Agreement

Two users + a public network

The users interact to agree on a key  $K$ .

## Key Pre-Distribution

TA and  $n$  users + a public network

+ a secure channel between TA and each User

For each pair of users  $U, V$  ( $U \neq V$ )

The TA constructs a key  $K_{UV}$  ( $= K_{VU}$ )

and sends it to  $U$  and  $V$  securely.

$\binom{n}{2}$  messages

each user stores  $n - 1$  keys

too many!

too many!

# BLOM'S DISTRIBUTION SCHEME

►  $p$ , prime with  $p > n = \#$  of users

► Keys chosen from  $\mathbb{Z}_p$

## SETUP

### TA

Chooses  $p$  as above. (public)

For each user  $U$ , chooses  $r_U \in \mathbb{Z}_p$ . (public) ( $U \neq V \implies r_U \neq r_V$ )

Chooses  $a, b, c \stackrel{\text{ran}}{\in} \mathbb{Z}_p$  (private)

For each user  $U$ , the TA computes:

$$a_U = a + b \cdot r_U \bmod p \quad (\text{private})$$

$$b_U = b + c \cdot r_U \bmod p \quad (\text{private})$$

and sends them securely to  $U$ .

### Each user $U$

Constructs  $g_U(x) = a_U + b_U \cdot x$ .

---

## When Alice & Bob want to communicate

Alice computes  $K_{AB} = g_A(r_B)$  and Bob computes  $K_{BA} = g_B(r_A)$ .

**CLAIM:**  $K_{AB} = K_{BA}$ .

proof on board

## BREAKING BLOM'S SCHEME: I

**Eve** wants to determine  $a$ ,  $b$ , and  $c$ . She knows:

$$a_E = a + b \cdot r_E$$

$$b_E = b + c \cdot r_E$$

Two equations, three unknowns, no dice

**Eve** also wants to determine  $K_{AB}$ . She knows:

$$K_{AB} = a + b \cdot (r_A + r_B) + c \cdot (r_A \cdot r_B)$$

$$a_E = a + b \cdot r_E$$

$$b_E = b + c \cdot r_E$$

Three equations, four unknowns:  $a$ ,  $b$ ,  $c$ , and  $K_{AB}$ .

**Fact:** For every possible value of  $K_{AB}$ , there is a solution for  $a$ ,  $b$ , and  $c$ .

**But what if Eve has a friend?**

## BREAKING BLOM'S SCHEME: II

Together Eve and Oscar know:

$$\left. \begin{aligned} a_E &\equiv a + b \cdot r_E \\ a_E &\equiv b + c \cdot r_E \\ a_O &\equiv a + b \cdot r_O \\ a_E &\equiv b + c \cdot r_O \end{aligned} \right\} \pmod{p}$$

Four equations, three unknowns:  $a$ ,  $b$ , and  $c$ .

So, Eve and Oscar together can break the scheme.

- ▶ The scheme can be generalized to be secure against coalitions of  $k$  users —  $k$  a parameter.
- ▶ E.g., There is a version that is secure against coalitions of 15 users, but fails against a 16 user coalition.

# TRANSPORT PROTOCOLS

**Alice**

Chooses  $k$  and sends it to **securely** to Bob. **OR**

**Trent** (The TA) acts as a key server:

Alice wants to talk to Bob.

She tells Trent & Trent issues a key to Alice and Bob for the session.

---

**Shamir's Three Pass Protocol** (Here Trent = Alice.)

**Alice** Publishes a prime  $p$  (with a hard disc. log problem)

**Alice** Chooses  $a \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$ .  $a^{-1} \equiv a^{-1} \pmod{p-1}$

**Bob** Chooses  $b \stackrel{\text{ran}}{\in} \mathbb{Z}_{p-1}^*$ .  $b^{-1} \equiv b^{-1} \pmod{p-1}$

**Alice** Sends  $K_1 = K^a \pmod{p}$  to Bob.

**Bob** Sends  $K_2 = K_1^b \pmod{p} = K^{a \cdot b} \pmod{p}$  to Alice.

**Alice** Sends  $K_3 = K_2^{a^{-1}} \pmod{p} = K^b \pmod{p}$  to Bob.

**Bob** Computes  $K = K_3^{b^{-1}} \pmod{p}$ . **Man-in-the-middle problems!**

# KERBEROS, I

---

Clients: **users, processes**

Servers: **gateways**

---

## The Dramatis Personæ

**Cliff** - a client

**Serge** - a server

**Trent** - a T.A. (**authentication server**)

**Grant** - a ticket granting server

## Before

Cliff and Serge share no secret data

## After

Serge will have verified Cliff's ID

A session key (for Cliff and Serge) will have been established.

## Background

The following is all **symmetric key** cryptography!

# KERBEROS, II

See drawing on board

## 1: Cliff → Trent

Requests ticket to ticket-granting server.

Cliff supplies his name and Grant's name.

## 2: Trent → Cliff

Checks out Cliff and if O.K.

Generates  $K_{CG}$

Sends Cliff  $T =_{\text{def}} e_{K_C}(K_{CG})$

$K_C = \text{Cliff's secret key}$

Constructs  $TGT =_{\text{def}} \text{Grant's ID} || e_{K_G}(\text{Cliff's ID, timestamp}_1, K_{CG})$

Sends Cliff  $TGT$ .

$K_G = \text{Grants's secret key}$

## 3: Cliff → Grant

Decrypts  $T$  to obtain  $K_{CG}$ .

Constructs  $Auth_{CG} =_{\text{def}} e_{K_{CG}}(\text{Cliff's ID, timestamp}_2)$ .

Sends  $TGT$  and  $Auth_{CG}$  to Grant.

# KERBEROS, III

See drawing on board

## 4: Grant → Cliff

Grant decrypts  $TGT$  and obtains: Cliff's ID,  $K_{CG}$ , and timestamp<sub>1</sub>.

Decrypts  $Auth_{CG}$  and obtains: Cliff's ID and timestamp<sub>2</sub>.

Checks that the two versions of Cliff's ID match.

Checks that the two timestamps are suff. close.

If OK, Grant generates  $K_{CS}$  = the Cliff-Serge session key.

Generates  $ServTicket \stackrel{\text{def}}{=} e_{K_S}(\text{Cliff's ID, timestamp}_3, \text{Exp-Time, } K_{CS})$ .

Sends  $ServTicket$  and  $e_{K_{CG}}(K_{CS})$  to Cliff.

$\text{Exp-Time}$  = how long  $K_{CS}$  is good for

$K_S$  = Serge's secret key

## KERBEROS, IV

### 5: Cliff → Serge

Cliff decrypts  $e_{K_{CG}}(K_{CS})$  and obtains  $K_{CS}$ .

Cliff constructs  $Auth_{CS} \stackrel{\text{def}}{=} e_{K_{CS}}$

$(\text{Cliff's ID, timestamp}_4)$ .

Cliff sends  $Auth_{CS}$  and **ServTicket** to Serge.

### Serge:

Decrypts **ServTicket** to obtain:

Cliff's ID,  $\text{timestamp}_3$ , Exp-Time, and  $K_{CS}$

Using  $K_{CS}$  decrypts  $Auth_{CS}$  to obtain:

Cliff's ID,  $\text{timestamp}_4$

Checks that the two versions of Cliff's ID match.

Checks that  $\text{timestamp}_4 \leq \text{timestamp}_3 + \text{Exp-Time}$ .

If OK, Cliff and Serge can chat using  $K_{CS}$ .

# PUBLIC KEY INFRASTRUCTURES (PKIS)

## Public Key Infrastructure

A set of protocols for publishing and certifying keys

## Certificate

Some information signed by its publisher,  
a **certification authority**.

- ▶ identity certification — id + email address + public keys
- ▶ credential certification — access rights

See §14.4 of T&W for more detail.

(This is a possible final paper topic.)

**NEXT**

**INFORMATION THEORY**