- The opponent O knows $u = a^X \bmod p$, $v = a^Y \bmod p$, a, p.
  If O is able to calculate discrete log's, the system is broken, i.e. Breaking the DH problem is no harder than calculating discrete log's.

Def 7.6 | Diffie-Hellman-Problem (DHP)

Given p, a a PE mod p, $a^X \bmod p$, $a^Y \bmod p$. Calculating $a^{XY} \bmod p$ is the Diffie-Hellman problem.

An efficient alg. to solve the DHP would break the DH scheme.

Open question: Does an efficient alg for solving DHP lead to an efficient alg. for discrete log's?

7.2 Shamir's no-key protocol

Prop 7.7 Let p be prime, $a, b \in \mathbb{Z}_{p-1}^*$. Then

$$\forall m \in \mathbb{Z}_p \quad m^{ab a^{-1} b^{-1}} \equiv m \pmod p$$

Proof: $a^{-1}, b^{-1} \in \mathbb{Z}_{p-1}^*$ exist by def. satisfying

$$a \cdot a^{-1} \equiv 1 \pmod{p-1} \qquad b \cdot b^{-1} \equiv 1 \pmod{p-1}, \text{ i.e.}$$

$$a \cdot a^{-1} = s(p-1)+1 \qquad b \cdot b^{-1} = t(p-1)+1 \qquad \text{for some } s, t \in \mathbb{Z}$$

Hence, for all $m \in \mathbb{Z}_p$

$$m^{ab a^{-1} b^{-1}} = m^{(s(p-1)+1)(t(p-1)+1)}$$

$$= m \cdot \underbrace{m^{(p-1)}}_{\equiv 1 \bmod p,\ \text{Fermat}}{}^{(st(p-1)+s+t)} \equiv m \pmod p$$

A sends a key m to B as follows

- Initial setup: a prime p is chosen and published
- Protocol actions:

A and B choose secret random numbers $a, b \in \mathbb{Z}^*_{p-1}$ and calculate $a^{-1}, b^{-1} \bmod (p-1)$, respectively

$A \to B$: $c_1 = m^a \bmod p$ (A locks, sends to B)

$B \to A$: $c_2 = (c_1)^b \bmod p$ (B locks, sends to A)

$A \to B$: $c_3 = (c_2)^{a^{-1}} \bmod p$ (A unlocks, returns to B)

B deciphers $m = (c_3)^{b^{-1}} \bmod p$ (B unlocks, reads m)

$$(c_3)^{b^{-1}} = m^{a b a^{-1} b^{-1}} \equiv m \pmod p$$

<u>Observe</u>: no authentication provided, protection from passive adversaries only

8. Public key Encryption

Asymmetric cryptosystem which does not need to exchange secret keys.

<u>Idea</u>: [by Diffie Hellman (76), earlier but not published paper by James Ellis (70) paper released by British government 97)]

- All user share the same e, d (en- decryption function)
- Each user has a pair of keys (K, L) such that

$$d(\underbrace{e(M, K)}_{c}, L) = M \qquad \forall M \in \mathcal{M}$$

K is public, L is private key

- Requirements

(i) $c = e(M, K)$ "easy" given M, K, solving for M "infeasible" given c and K

(ii) $M = d(c, L)$ "easy" given c, L

Hence, $f_K(M) = e(M, K)$ is a one-way function with "trapdoor" L

- Further requirements

(i) (K, L) easy to generate

(ii) There are sufficiently many pairs (K, L), exhaustive search impossible.

8.1 The RSA cryptosystem (Rivest, Shamir, Adleman, 1978)

(previously invented by Cocks (73), not published, released '97)

RSA – System

(i) Choose $p \neq q$ (large prime numbers), compute $n = p \cdot q$

(ii) Choose $d \in \mathbb{Z}^*_{\varphi(n)}$, i.e. $\gcd(d, \varphi(n)) = 1$

Compute $e = d^{-1} \mod \varphi(n)$

(iii) Public key (e, n), private key d

(iv) Message $m \in \{1, \dots, n-1\}$

Encryption : $c = m^e \mod n$

Decryption : $b = c^d \mod n$

Questions : 1) $b = m$? 2) Security 3) Implementation

Prop. 8.1 $p \neq q$ prime, $x, y \in \mathbb{N}$

$x \equiv y \pmod p \wedge x \equiv y \pmod q \Leftrightarrow x \equiv y \pmod{p \cdot q}$

Proof: $p \mid x - y$, $q \mid x - y \Leftrightarrow p \cdot q \mid x - y$ (since p, q are relatively prime)

Prop 8.2 Let $p \neq q$ prime, $n = p \cdot q$, $d, d^{-1} \in \mathbb{Z}^*_{\varphi(n)}$

$0 \leq m < n$, $c = m^{d^{-1}} \mod n$. Then $m = c^d \mod n$

$\Rightarrow$ Decryption in the RSA system works

Proof : $d^{-1} d \equiv 1 \pmod{\varphi(n)} \Rightarrow \exists t : d^{-1} d = t \underbrace{(p-1)(q-1)}_{\varphi(n) = \varphi(p) \cdot \varphi(q)} + 1$

(i) $\gcd(m, p) = 1$

$$\big(\underbrace{m^{d^{-1}}}_{c}\big)^d \equiv m^{t(p-1)(q-1)+1} \equiv m \cdot \underbrace{(m^{p-1})}_{\equiv 1 \pmod p,\ \text{Fermat}}{}^{t(q-1)} \equiv m \pmod p$$

(ii) $\gcd(m,p) = p$ $\quad p \mid m$, i.e., $m \equiv 0 \pmod p$

$\Rightarrow (m^{d-1})^d \equiv 0 \equiv m \pmod p$

Analogously $(m^{d-1})^d \equiv m \pmod q$

Using Prop 8.1: $(m^{d-1})^d \equiv m \pmod{n = p \cdot q}$

<u>Security of RSA</u>

Chosen plaintext attacks is most relevant, since anybody can encrypt an arbitrary number of any messages using the public key.

Hence, known: d^{-1}, n, arbitrary many pairs (m, c)

a) Factoring of ~~…~~ n. Use p, q to compute

$d = (d^{-1})^{-1} \bmod \varphi(n) = (p-1)(q-1)$ the private key.

But, Factoring is infeasible.

b) Computing square roots modulo n allows factoring.

<u>Prop 8.3</u> Let $n = p \cdot q$, $p \neq q$ prime, x a nontrivial solution of $x^2 \equiv 1 \pmod n$, i.e., $x \not\equiv \pm 1 \pmod n$. Then

$\gcd(x+1, n) \in \{p, q\}$

<u>Proof: Exercise</u>

<u>Hence</u>: Computing square roots is no easier than factoring.

c) Computing $\varphi(n)$ without factoring n.

Any efficient alg for computing $\varphi(n)$ yields an efficient alg. for factoring.

<u>Hence</u>, computing $\varphi(n)$ is no easier than factoring.

<u>Proof</u>: Let $n = p \cdot q$, p, q prime (unknown)

$\varphi(n) = (p-1)(q-1)$ is known

$\varphi(n) = (p-1)(q-1) = \underbrace{pq}_{n} - p - q + 1$

$\Leftrightarrow$ $p + q = n - \varphi(n) + 1$ (1)

$(p-q)^2 - (p+q)^2 = -4\underbrace{pq}_{n} \Leftrightarrow (p-q)^2 = (p+q)^2 - 4n$ (2)

$\Rightarrow q = \frac{1}{2}((p+q) - (p-q))$ (3)

(1) yields $(p+q)$, from (2) obtain $(p-q)$, q follows by (3)

d) Computing $(d^{-1})^{-1}$ (without knowing $\varphi(n)$)

<u>Prop 8.4</u> Let $n = p \cdot q$, p, q prime. Any efficient alg for computing $b^{-1} \bmod \varphi(n)$ leads to an efficient probabilistic alg for factoring n with error probability $< 1/2$

<u>Proof</u>: Stinson p. 139-141

Repeat the above alg until a factorization is found.

<u>Hence</u>, computing $b^{-1} \bmod \varphi(n)$ is no easier than factoring

<u>Remarks</u>

a) If d is known, n can be efficiently factored
If the private key d is detected, it is not sufficient to compute some new d, d^{-1}, also change p, q.

b) Never let somebody observe your decryption process

c) Conjecture of RSA (78): An efficient alg for breaking the RSA system yields an efficient factoring alg. (still open question)