6.857 Computer and Network Security **Fall Term, 1997**

Lecture 7 : September 25,1997

Lecturer: Ron Rivest *Scribe: Vijak Sethaput*

1 Review

- Z_p^* is cyclic since there exist generator g s.t. $\{g, g^2, ..., g^{p-1}\} = Z_p^*$
- **Definition 1** *Order(a): least x s.t. $a^x \equiv 1 \pmod{p}$*
- $(\forall a \in Z_p^*)$ $order(a)$ divides p-1
- a is a square iff $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$
- if $a = g^x$ for some generator g, $order(a) = \frac{p-1}{gcd(x,p-1)}$
- $a \equiv b \pmod{p}$, where $a = g^x, b = g^y \iff x \equiv y \pmod{p-1}$

2 Density of Primes

2.1 The Theorem

Theorem 1 *Prime Number Theorem*

if $\pi(x) =$ The number of primes between 1 and x, then $\pi(x) \approx \frac{x}{ln(x)}$

This approximation gets better for large values of x. Informally, the theorem states that if one picks a number less than x at random, the probablility that it is prime approaches $\frac{1}{ln(x)}$. Therefore, large prime number are relatively common, so large prime number can be found easily by random search (e.g. just picking an odd number at random and a test for primality will terminate in a reasonable about of time)

2.2 Variation

For the reason that will be clear later in the discussion of security algorithms, one may want $p-1$ to have a large prime factor. This can be done by finding large prime q and then check $p=(iq+1)$ where $i=2,4,6,..$ for primility. It is not necessary that one needs to find large prime q everytime.

3 Generator

Theorem 2 *g is a generator (mod p) iff* $g^{p-1} \equiv 1 \pmod{p}$

- Problem: factor $p-1$ requires knowing all prime q that divide $p-1$
- Solution: One should generate primes using $p = i \cdot q + 1$ e.g. $48 \cdot q + 1$
- generator are common

4 Discrete Log Problem

Definition 2 *Discrete Log Problem*

x is the discrete log (base g, modulo p) of y. If given prime p, generator g:$\{g^0, g^1, g^2, ...g^{(p-2)}\} = Z_P^*$ *,and* g^x *(mod p) then the adversary need to find x s.t.* $g^x = y \pmod{p}$

This problem is the foundation for many crypto system. Its security based on computational difficulties of the problem. Discrete log algorithms may be somewhat harder if you choose p to be large enough. However, if $p-1$ has only small factors, then this problem is easy to solve.

The next sections outline mathematical problems based on cryptography. These crypto systems are always prone to a mathematical breakthrough. There is no proof that they are hard, so an algorithm may exist.

5 Public Key Cryptography

The idea for public key systems originated with Merkle at UC Berkley when he was trying to develop an algorithm for communicating over secure lines without prior exchange for key as we have seen so far. Merkle came up with a puzzle scheme. This requires people who are communicating to solve just one puzzle but the adversary has to solve many more puzzles. Berkley wasn't interested in his work, so he went to talked to Diffie and Hellman at Standford about the idea. Diffie Hellman then came up with the key exchange protocol which is based on the discrete Log Problem.

6 Key Exchange Protocol

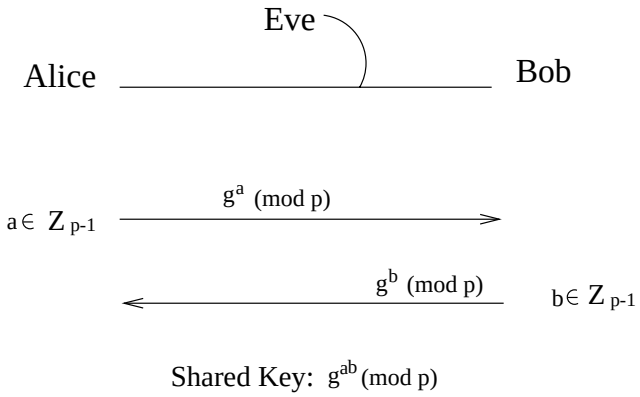

Figure 1: Diffie-Hellman Key Exchange Protocol.

As shown in the figure, Eve faces the Diffie-Hellman problem which can be state as follows:

Definition 3 *Diffie-Hellman Problem*

Given g, p, $g^a \pmod p$, $g^b \pmod p$ *The adversary has to compute* $g^{ab} \pmod p$

The nice property about this protocol is that it can bootstap the parties from no secret key to have shared secret key using global public value. The protocol is secured base on the fact that Discrete Log Problem is hard. However, if Eve start to be active, there will need to be more steps added.

7 El Gamel

7.1 Setting up:

- Alice choose private key a s.t. $0 <= a < p-1$ ($a \in Z_{p-1}$)
- Alice can then compute her public key, $g^a \pmod p$ which she can then publish somewhere

7.2 Privacy

To encrypt messages:

- Bob pick a random k s.t. $0 <= k < p-1$ ($k \in Z_{p-1}$) which corresponding to b in Diffie-Hellman Key exchange protocol.
- Now both parties can computer their share key g^{ka}
- Bob send $(c_1, c_2) = (g^k, M * g^{ka})$

To decrypt:

- Alice can now compute thier share key, $(g^k)^a$
- The message M is then $M = \frac{c_2}{(g^k)^a} = \frac{c_2}{(c_1)^a}$

7.3 Authentication

Signature:

The idea for signature is that only Alice can produce the signature but others just using Alice public key, g^a, can verify that $k^{-1} \pmod{(p-1)}$ exist.

- Alice generate a random k, $0 <= k < p-1$ and $\gcd(k, p-1) = 1$
- Alice then compute $r = g^k \pmod p$
- Signature $\sigma = (r, s)$ where $b = (M - ra)/k \pmod{(p-1)}$

Verify:

- $y = g^a$
- $y^r \cdot r^s \stackrel{?}{=} g^M \pmod{p}$
- $g^{ar} \cdot g^{ks} \stackrel{?}{=} g^M \pmod{p}$
- $g^{ar+ks} \stackrel{?}{=} g^M \pmod{p}$
- iff $M \stackrel{\surd}{=} ar + ks \pmod{(p-1)}$

Note that in this scheme the value of k need to be protected. Since, if adversary knows k then he/she can solve for a thus find out the secret.