Recap: p - prime

$\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ with operations $+_p$: addition mod p

$\times_p$: mult. mod p

Fermat's Little Theorem (FLT):

$\forall\, a \in \underbrace{\mathbb{Z}_p \setminus \{0\}}$, $\quad exp_p(a, \underbrace{p-1}) = 1$

$\mathbb{Z}_p^* \equiv$ set of all elements in $\mathbb{Z}_p$ co-prime to p $\qquad$ $\uparrow$ $|\mathbb{Z}_p^*|$

Last lecture, we generalized this for general natural numbers.

[Euler's Thm] $\quad \forall n \geq 2, \quad \forall\, a \in \mathbb{Z}_n^*, \quad exp_n(a, |\mathbb{Z}_n^*|) = 1$

Proof of FLT used the following properties of $(\mathbb{Z}_p^*, \times_p)$

(1) $1 \in \mathbb{Z}_p^*$ $\qquad$ (2) $a, b \in \mathbb{Z}_p^* \Rightarrow a \times_p b \in \mathbb{Z}_p^*$

(3) $\times_p$ is assoc. & comm.

(4) $\forall\, a \in \mathbb{Z}_p^*, \; \exists\, b \in \mathbb{Z}_p^*$ s.t. $a \times_p b = 1$.

The above properties also hold for $\mathbb{Z}_n^*$, for any n:

(1) and (3) are immediate.

(2): follows from defn. of gcd

(4): Bézout's identity / Extd. Euclid's Algorithm.

LECTURE 12 : PUBLIC KEY ENC. (PKE)

Setup $\rightarrow$ pk, sk

Encrypt(pk, m) $\rightarrow$ ct

Decrypt(sk, ct) $\rightarrow$ m.

<u>Correctness</u>: For all (pk, sk) sampled by Setup,
for all messages m,
Decrypt(sk, Encrypt(pk, m)) = m.

<u>Security</u>: For any adversary that only has pk (but not sk), Encrypt(pk, m) hides the message m.

The first PKE scheme was given by Ron Rivest, Adi Shamir and Leonard Adleman, and is the most widely used PKE scheme currently (called RSA-PKE after the last names of Rivest, Shamir, Adleman).

Before seeing the RSA PKE scheme, let us see a template scheme that satisfies correctness, but may not be secure. It is based on the following observation:

<u>Observation</u>: Suppose $a \in \mathbb{Z}_N^*$, d and e are numbers s.t. $d \cdot e \bmod (|\mathbb{Z}_N^*|) = 1$

Then $a = \exp_N(a, d \cdot e)$
$= \exp_N(\exp_N(a, e), d)$

<u>Proof</u>: Using Euler's Theorem,

$$\exp_N(a, d \cdot e) = \exp_N(a, k \cdot |\mathbb{Z}_N^*| + 1)$$
$$= \exp_N(a, k \cdot |\mathbb{Z}_N^*|) \times_N \exp_N(a, 1)$$
$$= 1 \times_N a$$

∎

Using this observation, we build a PKE scheme as follows:

pk = (modulus N, exponent e)

sk = d s.t. $e \cdot d \bmod |\mathbb{Z}_N^*| = 1$

message is an element in $\mathbb{Z}_N^*$

Encrypt $(pk = (p, e), a \in \mathbb{Z}_N^*)$

ct = $\exp_N(a, e)$.

Even though modulus N is very large, ct can be computed efficiently using repeated squaring.

Decrypt $(sk = d, \; ct)$:

Output $\exp_N(ct, d)$.

Correctness follows from the observation above, what about security? Recall, adversary sees $pk = (p, e)$ and ciphertext. It must compute m.

In order to run Extd. Euclid's Algorithm, adversary needs to know $|\mathbb{Z}_N^*|$. Given N, can the adversary always compute $|\mathbb{Z}_N^*|$?

Consider $\mathbb{Z}_N = \{0, 1, 2, \ldots, N-1\}$. Let $N = p \cdot q$

p, q are primes.

$$|\mathbb{Z}_N^*| = pq - p - q + 1 = (p-1)(q-1)$$

$\varphi(N)$: Euler's Totient Function

<u>Observation</u>: If N is product of two primes,

Computing $\varphi(N) \Leftrightarrow$ Computing prime factors of N

believed to be hard computational problem

Proof : If $\varphi(N)$ can be computed efficiently, then we have $p \cdot q = N$, $p+q = \underbrace{N - \varphi(N)+1}_{Z}$

$\Rightarrow \frac{N}{p} + p = Z$

Solve for p. ∎

Rivest, Shamir and Adleman conjectured that the template PKE scheme, when instantiated with N = product of two large primes, is secure. This is called the RSA problem (or RSA assumption). It is very well studied, and so far, the best known algorithms run in super polynomial time.

RSA Problem :

1. Sample unif. rand. $\underbrace{\text{large}}_{\in [1, 2^{1000}]}$ primes p, q.
   $N = p \cdot q$.

2. Sample random $m \leftarrow \mathbb{Z}_N^*$

3. Sample random exponent $\underbrace{e}_{\in [1, 2^{1000}]}$ s.t. $\gcd(e, \varphi(N)) = 1$.

4. Given $N, e, \exp_N(m, e)$, compute m.

So far, the largest modulus for which RSA problem has been solved is an 800-bit modulus. It is believed that solving RSA problem for 2000-bit moduli is computationally infeasible, even with modern supercomputers.

RSA-PKE : Summary and some details

Setup :

$\in [2^{1000}]$

1. Sample large primes p, q.

How to sample primes :

1. sample 10^6 unif. rand. numbers in $[2^{1000}]$
2. Check if any of these are prime. If so, output the first one.

The Prime Number Theorem states that there are roughly $2^{1000}/1000$ primes in the range $[2^{1000}]$

Therefore, a random number in $[2^{1000}]$ will be prime w.p. $\sim 1/1000$. Hence, sampling around 10^6 random numbers ensures that one of them will be prime with high probability.

Checking primality: the naive algorithm takes time $2^{1000/2}$, which is very expensive. However, there are efficient algorithms that can check if n is prime, in time $\log^2(n)$.

2. Set $N = p \cdot q$ and $\varphi(N) = (p-1)(q-1)$.

3. Sample random prime $e \leftarrow [2^{1000}]$
s.t. $\gcd(e, \varphi(N)) = 1$

e doesnt need to be prime, we only need that $\gcd(e, \varphi(N)) = 1$.

Again, sample uniformly random number e and check if e is prime and $\gcd(e, \varphi(N)) = 1$. If so, output e, else repeat.

This process terminates in polylog(N) steps, since $\varphi(N)$ can have at most $\log(N)$ prime factors, and there are $N/\log N$ prime numbers in this range.

4. Compute d s.t. $e \cdot d \bmod \varphi(N) = 1$.

2 ways of doing this:

1. Using Extd. Euclid's algorithm:

Since $\gcd(e, \varphi(N)) = 1$, $\exists\ s, t \in \mathbb{Z}$ s.t.

$$s \cdot e + t \cdot \varphi(N) = 1$$

$\Rightarrow$ $d = s \bmod \varphi(N)$ satisfies $e \cdot d \bmod \varphi(N) = 1$.

This approach takes polylog(N) time, since Extd. Euclid's algorithm takes $O(\log m + \log n)$

time, where m & n are the inputs.

2. Set $d = e^{\varphi(\varphi(N)) - 1} \bmod \varphi(N)$.

First, note that $e^{\varphi(\varphi(N)) - 1} \bmod \varphi(N)$

$$= \left(e \bmod \varphi(N)\right)^{\varphi(\varphi(N)) - 1} \bmod \varphi(N)$$

$e' = e \bmod \varphi(N)$ is an element in $\mathbb{Z}^*_{\varphi(N)}$

(since $\gcd(e, \varphi(N)) = 1$).

As a result, using Euler's theorem,

$$(e')^{\varphi(\varphi(N))} \bmod \varphi(N) = 1.$$

Hence $e' \cdot (e')^{\varphi(\varphi(N)) - 1} \bmod \varphi(N) = 1$, and

therefore $d = e^{\varphi(\varphi(N)) - 1}$ satisfies $e \cdot d \bmod \varphi(N) = 1$.

However, is it efficient to compute d in this manner? This would require computing $\varphi(\varphi(N))$, and this may not be efficiently feasible, given just p and q.

5. Set $pk = (N, e)$, $sk = d$.

$\mathrm{Enc}\left(pk = (N, e),\ m \in \mathbb{Z}^*_N\right)$:

Output $\underbrace{\exp_N(m, e)}$.

← can be computed in $\mathrm{polylog}(N)$ time

We wanted to encrypt messages (which can be arbitrary bit strings). One way to do this: partition the message into small chunks, encode each chunk separately by encoding it as an element in $\mathbb{Z}_N^*$.

Dec (sk, ct) :

Output $\underbrace{\exp_N(ct, sk)}$.

can be computed in poly log (N) time.

End of PKE