

**Announcements:**

Details regarding computational assignments using the Pari-GP software package will be given out next week. Reading assignment: § 3.1, § 3.2 [NK]. Chapter 12 on Data Encryption Standard.

**Topics for today:**

1. Review of Group Theory
2. Cyclic groups
3. RSA cryptosystem
4.  $E(x)$  and  $D(x)$

## 1. Review of Group Theory

**Thm (Lagrange):** Suppose  $H$  is a subgroup of a finite group  $G$ . Then  $|H|$  divides  $|G|$

Let  $x \in G$  and consider the subgroup generated by  $x$ ,  $\langle x \rangle = \{x^i \mid i \in \mathbf{Z}\}$ .  $order(x) := |\langle x \rangle|$ , is the smallest integer such that  $x^{order(x)} = 1$ .

**Corollary:**  $order(x)$  divides  $|G|$ , for all  $x \in G$

$(\mathbf{Z}/N\mathbf{Z}, +)$  denotes the additive group modulo  $N$  and  $(\mathbf{Z}/N\mathbf{Z}^*, *)$  denotes the multiplicative group modulo  $N$

Let  $G = (\mathbf{Z}/N\mathbf{Z}^*, *)$ ,  $|G| = \phi(N) = \prod_{p^\alpha \parallel N} \phi(p^k)$  and  $\phi(p^k) = p^k - p^{k-1}$ . For  $N = p$  prime,  $\phi(p) = p - 1$

**Corollary (Fermat's little Thm):**  $x^{p-1} \equiv 1 \pmod{p}$ ,  $\forall x \in \mathbf{Z}$ ,  $p \nmid x$

**Corollary (Euler's Thm):**  $x^{\phi(N)} \equiv 1 \pmod{N}$ ,  $\forall x \in \mathbf{Z}$ ,  $\gcd(x, N) = 1$

## 2. Cyclic groups

**Defn:** A group  $G$  is *cyclic* if  $G = \langle x \rangle$  for some  $x \in G$ .

**Examples:** For prime  $p$ ,  $(\mathbf{Z}/p\mathbf{Z}^*, *)$  is cyclic but in general  $(\mathbf{Z}/N\mathbf{Z}^*, *)$  is not cyclic. We'll talk more about this in the context of finite fields.

**Lemma:** A cyclic group is abelian

**Proof:**  $x^{(i)} = x \odot_G x \dots i \text{ times} \dots \odot_G x$ . So

$$\begin{aligned} x^{(i)} \odot_G x^{(j)} &= (x \odot_G \dots i \text{ times} \dots x) \odot_G (x \odot_G \dots j \text{ times} \dots x) \\ &= x^{(i+j)} \\ &= x^{(j+i)} \\ &= x^{(j)} \odot_G x^{(i)} \end{aligned}$$

*Remark:*

1. Say  $G = \langle 2, 5 \rangle = \{2x + 5y \text{ mod } 100\}$ . It's not clear that  $G$  is cyclic. Infact  $\langle 2, 5 \rangle = \langle 1 \rangle$ . What about  $\langle 4, 6 \rangle$ ? Answer:  $\langle 2, 5 \rangle = \langle 6 \rangle$ . Euclid tells us that  $\langle x, y \rangle = \langle \gcd(x, y) \rangle$
2. When the context/group in question is clear we'll drop the  $\odot_G$  operator. So  $x^i := x^{(i)}$  and  $\cdot := \odot_G$ . We may also drop the dot (" $\cdot$ ") and just concatenate group elements (" $xy$ ").

**Thm:** Every subgroup of a cyclic group is cyclic.

**Proof:** (This is true in general but we'll tackle only the finite case.). Suppose  $G = \langle x \rangle$  is finite. Let  $H \subseteq G$  is a subgroup. Let  $H = \{x^{n_1}, x^{n_2}, \dots, x^{n_k}\}$ . Taking a cue from remark (1), let  $d = \gcd(n_1, n_2, \dots, n_k)$ . By the Euclidean algorithm  $d = a_1 n_1 + a_2 n_2 + \dots + a_k n_k$  with  $a_i \in \mathbf{Z}$ . As  $x^{n_1} \in G$ , we have  $(x^{n_1})^{a_1} \in G$ . Similarly  $(x^{n_i})^{a_i} \in G$  and so is their product.

$\prod_i (x^{n_i})^{a_i} = x^{\sum_i a_i n_i} = x^d$ . On the other hand  $d | n_i$  for all  $i \Rightarrow x^{n_i} \in \langle x^d \rangle \Rightarrow H \subseteq \langle x^d \rangle$ . Clearly  $\langle x^d \rangle \subseteq H$  and hence we have equality.  $\diamond$

**Corollary:** Let  $G$  be cyclic of order  $n$  then for all divisors  $d$  of  $n$ , there exists a unique subgroup of  $G$  of order  $d$ . In fact, say  $G = \langle x \rangle$ , then  $\forall d | n, \exists! \langle x^{\frac{n}{d}} \rangle$

**Proof:** Exercise. Use above Thm and Lagrange's.

Recall that in HW #1, we proved that if  $G = \mathbf{Z}/N\mathbf{Z}$ , then  $\forall d | n, S_d = \langle \frac{n}{d} \rangle$  is the unique subgroup of order  $d$ . The above corollary is the essence of the identity  $\sum_{d|n} \phi(d) = N$ .

### 3. RSA cryptosystem

The RSA is an asymmetric key scheme, very different from the symmetric schemes we've been discussing. The system works as follows:

- Private Computation/Initialization:
  1. Choose two big primes  $p$  and  $q$ , with  $p \neq q$  (of the same length). Let  $N := pq$ . So  $\phi(N) = (p-1)(q-1)$ .
  2. Choose  $e \in \mathbf{Z}/N\mathbf{Z}$  such that  $(e, \phi(N)) = 1$
  3. Compute  $d$  such that  $d = e^{-1} \bmod \phi(N)$  that is  $ed \equiv 1 \bmod \phi(N)$
- Encryption function: Public key =  $(N, e)$

$$\begin{array}{ccc} E : \mathbf{Z}/N\mathbf{Z}^* & \rightarrow & \mathbf{Z}/N\mathbf{Z}^* \\ x & \mapsto & x^e (= x^e \bmod N) \end{array}$$

- Decryption function: Private key =  $d$

$$\begin{array}{ccc} D : \mathbf{Z}/N\mathbf{Z}^* & \rightarrow & \mathbf{Z}/N\mathbf{Z}^* \\ x & \mapsto & x^d (= x^d \bmod N) \end{array}$$

In practice  $e$  is taken to be a small value like 3 so that encryption is relatively quick. Note that this doesn't imply that  $d$  is small. Perhaps it's a good idea to choose a random  $e$  and test that it is relatively prime to  $\phi(N)$ . Observe that using extended gcd algorithm to do the previous check gives  $d$  as a byproduct.

Say  $N \sim 2^{512}$  so  $e < 2^{512}$ . Doing naive exponentiation by iterative multiplication turns out to be expensive. Is there a better method? *Repeated Squaring Method*

- Suppose  $e$  is even then  $x^e = (x^{\frac{e}{2}})^2$ . So with one group operation (squaring) we are able to reduce the size of the problem by half ( $e \rightarrow \frac{e}{2}$ ).
- On the contrary say  $e$  is odd then  $x^e = (x^{\frac{e-1}{2}})^2 \cdot x$ . So with two group operations (squaring + multiplication) we are able to reduce the size of the problem by half ( $e \rightarrow \frac{e}{2}$ ).
- Repeating this procedure recursively gets the job done in  $\leq 2 \log n$  modular multiplications (as opposed to  $O(N)$  using the naive method.)

Assuming multiplication takes  $O(\log^2 N)$  bit ops, repeated squaring to compute  $x^e \bmod N$  will consume  $O(\log^3 N)$  time. Generating primes  $p$  and  $q$  is a time consuming process, at some point we'll look at prime # generation and primality testing. Notice that by taking  $e$  to be small like 3, for example exponentiation is brought down to  $O(\log^2 N)$ . In *theory* it's safer to choose a random  $e$ !

#### 4. $E(x)$ and $D(x)$

$$E(D(x)) = D(E(x)) = x \text{ for all } x \in \mathbf{Z}/N\mathbf{Z}^*$$

$$\begin{aligned} D(E(x)) &= D(x^e) \\ &= (x^e)^d \\ &= x^{ed} \\ &= x^{1+k\phi(N)} \text{ since } ed = 1 + k\phi(N) \\ &= x(x^{\phi(N)})^k \\ &= x \text{ by Euler's Thm} \end{aligned}$$

So  $E$  and  $D$  are well-defined maps and inverses of each other. Our message space  $\mathbf{Z}/N\mathbf{Z}^*$  is not just a set but a group.

$$\begin{aligned} E(xy) &= (xy)^e \\ &= x^e y^e \text{ since group is abelian} \\ &= E(x)E(y) \end{aligned}$$

So  $E$  as a map from  $\mathbf{Z}/N\mathbf{Z}^*$  to itself respects multiplication and also inverses  $E(x^{-1}) = (x^{-1})^e = (x^e)^{-1} = E(x)^{-1}$ .  $D(x)$  behaves similarly. So these maps  $E, D$  are bijective and respect the *group structure*. We'll try to develop an attack on RSA leveraging the multiplicative property of  $E(x)(= E(x)E(y))$  and  $D(x)(= D(x)D(y))$ .

Suppose somebody gave us a magic box  $A$  which decrypts 1% of all ciphertxts, that is  $A(x) = D(x)$ . Let  $x$  be an arbitrary (ciphertxt) input. Suppose we are lucky and  $x$  is in the 1% good cases where  $A$  can be used, we are done. Otherwise we'll randomize  $x$  so that the result falls into  $L$  the set of good cases and we can recover the plaintext.

Construct  $x.y$  by multiplying  $x$  by  $y$  where  $y = E(z)$ , and  $z$  is randomly picked from  $\mathbf{Z}/N\mathbf{Z}^*$ . So  $x.y$  is a random element in  $\mathbf{Z}/N\mathbf{Z}^*$ . We know that with 1% probability  $xy$  will fall into our lucky set  $L$ . So we randomize  $x$  with

different  $y$  until  $xy \in L$ . When this happens we see that:  $A(xy) = D(xy) = D(x).D(y) = D(x).D(E(z)) = D(x).z$  and we can recover  $D(x)$  as we know  $z$  and  $A(xy)$ .  $D(z) = A(xy)z^{-1}$ .

**Question:** Therefore if we can solve a *tiny* fraction of all instances, RSA is broken. What does this say about the hardness of solving RSA?