Professor John R. Black
University of Colorado at Boulder
Fall 2004

CSCI 6268 Notes on Groups and RSA

These are some rough notes for the lectures we did recently concerning groups and related notions in number theory and RSA cryptography.

Definition 1 *A group is a nonempty set G along with a binary operation $\circ$. We must have that G is closed under the operation $\circ$, and we must have the following three properties:*

- *Associative Property (for all $a, b, c \in G$, $(a \circ b) \circ c = a \circ (b \circ c)$).*
- *Existence of an identity element (there exists some $e \in G$ such that for all $a \in G$, $a \circ e = e \circ a = a$).*
- *Inverses for all elements (for all $a \in G$ there exists some $b \in G$ such that $ab = ba = e$).*

If a group G is also commutative we call G an Abelian group.

Examples of groups are $\mathbb{Z}$ under addition, $\mathbb{Q}$ under addition, $\mathbb{R}$ under addition. But $\mathbb{Z}$ under multiplication is NOT a group (inverses exist only for 1 and -1). If you take $\mathbb{Q} - \{0\}$, you get a group under multiplication (check this).

Finite Groups. The groups we just mentioned are finite. But there can be finite groups too. The simplest example is the group $G = \{0\}$ under addition (check this). But that's not very interesting.

It turns out that $\mathbb{Z}$ is a group under addition mod n for any $n > 1$. We write this group as $\mathbb{Z}_n$ and its elements are $\{0, 1, \cdots, n-1\}$. The inverse of any element $a \in \mathbb{Z}_n$ is simply $n - a$, and the identity element is 0, of course.

For multiplication in $\mathbb{Z}$ we have to be careful. If we allow 0 to be in our group we won't have an inverse for it, so we must disallow it. If we take the set $\{1, 2, \cdots, p-1\}$ where p is prime and the operation is multiplication mod p, we get a group. (We didn't prove this, but it's a fact.) We write this group as $\mathbb{Z}_p^*$ where the asterisk reminds us that the operation in the group is multiplication.

What happens if we try doing multiplication modulo some composite? Say we try doing multiplication mod n for some $n > 1$. As we saw in class, we have to be careful once more: any number between 1 and $n-1$ that is not relatively prime to n will not have an inverse. Take for example the element 2 and try to find its inverse mod 4. It doesn't have one! So when we speak of the group $\mathbb{Z}_n^*$ we mean the set $\{1 \leq i \leq n-1 : (i, n) = 1\}$. Then everything will have an inverse. For example, $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ and the operation is (of course) multiplication mod 15. You can find all the inverses in this set as a good exercise.

In general, how do you go about finding inverses in $\mathbb{Z}_n^*$? Well, for any element $a \in \mathbb{Z}_n^*$, you need to find some element b such that $ab = 1$. You know that $(a, n) = 1$ by definition of $\mathbb{Z}_n^*$. So using Euclid's extended algorithm, we find two integers u and v such that $au + nv = 1$ and $u \in [1..n-1]$ with $(u, n) = 1$. We know we will ALWAYS succeed in doing this, though we didn't prove it. Now just set $b = u$ and you're done. That is the inverse of a. And remember, sometimes a number is its own inverse; for example 14 is the inverse of 14 in the group $\mathbb{Z}_{15}^*$.

How many elements are there in $\mathbb{Z}_n^*$? The answer is (of course) the number of elements in the set $\{1 \leq i \leq n-1 : (i,n) = 1\}$. So what we're really asking is how many numbers are there between 1 and $n-1$ which are relatively prime to n. This is precisely the definition of $\phi(n)$. (This is called the "Euler phi function.") A few handy facts about this function:

- If p is prime, then $\phi(p) = p - 1$.
- If p and q are distinct primes then $\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$
- If p is prime, then $\phi(p^s) = p^s - p^{s-1}$. (Notice that this rule implies the first rule above: just set $s = 1$. We didn't present this rule in class, because we're not going to need it for RSA.)

Lagrange's Theorem. We had a really famous result in lecture called Lagrange's Theorem. It says something like this: let G be a finite group of n elements and with operation $\circ$. Then for any $a \in G$, $a^n = a \circ a \circ \cdots \circ a$ (n appearances of a) is equal to e. (Where "e" is the identity element of the group. If we were in $\mathbb{Z}$ under addition then e would be 0, for example.)

We did some examples of Lagrange's Theorem with various groups including $\mathbb{Z}_{15}^*$. You can do them yourself for additional practice if you like.

Public-Key Cryptography. Suppose Alice and Bob want to send information privately between each other, but there is some evil adversary trying to learn the content of the communications. If Alice and Bob share a piece of secret information, called the "key" then this is not too hard. But without this, it's quite unclear how they can hope to succeed.

The solution is called Public-Key Cryptography. And the most well-known and widely-used method is called RSA (after its inventors Rivest, Shamir, and Adleman). Here's how RSA works: Alice chooses two primes, p and q and sets $n = pq$. Alice also selects some number e which is relatively prime to $\phi(n) = (p-1)(q-1)$. She then computes d as the inverse of e in the group $\mathbb{Z}_{\phi(n)}^*$. That should be confusing to you; why in this weird group? It will become clear later.

She then publishes to the world the value (e, n). She wants EVERYONE to know this value. But she guards the value d with her life. No one gets to know this number but Alice.

Now when Bob wishes to send some message (encoded as a number) $M \in \mathbb{Z}_n^*$, he computes $C = M^e \bmod n$ and sends this to Alice. When Alice receives C she computes $C^d \bmod n$ and gets M, which is Bob's original message.

There are a couple of things to clear up here. First, how do we know that $(M^e)^d \bmod n$ is going to always be M again? And how do we know that some adversary cannot compute M from just having the public info (e, n) along with C? And how do we do all this efficiently in software? (Those modular exponentiations look computationally difficult!) We now address each of these issues.

First, why is $(M^e)^d$ equal to M in the group $\mathbb{Z}_n^*$? This uses some of the rules we have been working on this week. Let's go through it. We know that e and d are inverses in a *different* group: $\mathbb{Z}_{\phi(n)}^*$. That means that in $\mathbb{Z}$ we know $ed = 1 + k\phi(n)$ for some k. So what happens when we take $M^{ed} = M^{1+k\phi(n)}$ in the group $\mathbb{Z}_n^*$?

$$\begin{aligned} M^{ed} &= M^{1+k\phi(n)} \\ &= M \cdot M^{k\phi(n)} \\ &= M \cdot (M^{\phi(n)})^k \\ &= M \cdot 1^k \\ &= M. \end{aligned}$$

Can you see why we said that $M^{\phi(n)} = 1$ above? It's Lagrange's theorem! Remember, $\phi(n)$ is exactly the size of $\mathbb{Z}_n^*$, so we must get 1 when we raise ANY group element to that power.

So the method works. Now, why is it secure? Well, that would take us a long time to answer, but the short answer is that "it is BELIEVED to be as hard to crack RSA as it is to factor n." And factoring is believed to be hard when p and q are large primes.

Finally the last question: how do we do these computations quickly on a computer. Everything is easy except for one part: raising a big number to a big power seems like it would take a long time. The advantage we have here is that we are doing all arithmetic modulo n, so we can always keep our intermediate results between 1 and $n-1$ if we just keep applying the modulus (we may go as high as n^2 just before we apply the modulus, but no higher than this). Computing $M^e \bmod n$ is called the "modular exponentiation problem" and we need a way to solve it quickly on a computer. We didn't do this in lecture, but in case you're interested, here it is.

The trick to compute this stuff fast is called "repeated squaring." I went over it in lecture, but I'll give you an example quickly to remind you how it works: suppose we have to compute $513^{134} \bmod 2911$. First we convert the exponent into binary: 134 is $10000110_2 = 2^7 + 2^2 + 2^1$. Then we begin squaring 513 as needed. Each time we square, we reduce modulo 2911 because we want to keep the numbers as small as possible for the sake of efficiency.

$$\begin{aligned}
513^1 \bmod 2911 &= 513 \\
513^2 \bmod 2911 &= 1179 \\
513^4 \bmod 2911 &= 1179^2 \bmod 2911 = 1494 \\
513^8 \bmod 2911 &= 1494^2 \bmod 2911 = 2210 \\
513^{16} \bmod 2911 &= 2210^2 \bmod 2911 = 2353 \\
513^{32} \bmod 2911 &= 2353^2 \bmod 2911 = 2798 \\
513^{64} \bmod 2911 &= 2798^2 \bmod 2911 = 1125 \\
513^{128} \bmod 2911 &= 1125^2 \bmod 2911 = 2251.
\end{aligned}$$

Now we're set up to compute $513^{134} \bmod 2911$ using the numbers above:

$$\begin{aligned}
513^{134} \bmod 2911 &= 513^{2^7+2^2+2^1} \bmod 2911 \\
&= 513^{128+4+2} \bmod 2911 \\
&= 513^{128} \cdot 513^4 \cdot 513^2 \bmod 2911 \\
&= 2251 \cdot 1494 \cdot 1179 \bmod 2911 \\
&= 1622.
\end{aligned}$$

This method is vastly more efficient than trying to compute $513^{134} \bmod 2911$ by direct means. But we could do it:

```
% bc
n = 513^134
n
1431287653425434770823600543915338715814185788826663241367705381446 6
7846610641378565038805170710708514413583444723736510121548120749642 9
5909724616466787653990716187248952976480271156361156673026083808090 9
0927743748175405556994655899830632624637970227371667996429149088419 4
8215685319619732351842415020581867025911607878213784007291917408658 3
```

```
981864660548090303679489

n % 2911
1622
```

But if you tried something like $171241571292^{1275881999911} \bmod 99128447125111$ even `bc` would fail. You would HAVE to use modular exponentiation! (In case you're interested, the answer is 57233387830710.)