CS 3510 — Algorithms, Fall 2022

Lecture 6: RSA

January 27, 2022

Lecturer: Frederic Faulkner *Scribe: Aditya Diwakar*

Homework 2 (1B) is due on Saturday and Homework 3 (1C) will also come out on Saturday. Homework 1 grades will be out tonight. Lecture started with a recap (please see Lecture 5 notes).

Modular Review

Remember that the multiplicative inverse of $x \pmod N$ is a if $ax \equiv 1 \pmod N$ (also means $a \equiv_N x^{-1}$) which means we can do interesting things such as solving for unknowns:

$$\begin{aligned} x \cdot y &\equiv_N 5 \\ x^{-1}(xy) &\equiv_N 5(x^{-1}) \\ y &\equiv 5x^{-1} \end{aligned}$$

For today's lecture, we also need an additional theorem:

Theorem. *If p is prime and $gcd(a, p) = 1$ then $q^{p-1} \equiv 1 \pmod p$*

This is known as Fermat's primality test theorem.

Proof. Consider the set $\{1, 2, \dots, p-1\}$ then let a be an element within this set and create a new set defined by $\{a, 2a, 3a, (p-1)a\}$. The claim is that these two sets are the same (remember: in sets, order does not matter).

Why? The only way it could fail is if $x \not\equiv y \pmod p$ but $ax \equiv ay \pmod p$ then the set would shrink (in size). Since p is prime, then $a \pmod p$ has an inverse because $gcd(a, p) = 1$ (from the assumption), then let

$$\begin{aligned} ax &\equiv ay \pmod p \\ a^{-1}ax &\equiv a^{-1}y \pmod p \\ x &\equiv y \pmod p \end{aligned}$$

As an example of this, let us take the set $\{1, 2, 3, 4, 5, 6\}$ which is of size $7-1, p = 7$ and take a number $a = 4$, then under modulo 7, the set is:

$$\{4, 1, 5, 2, 6, 3\}$$

Each number was found by performing $mod(4 \times x)$ for each x in the original set. These sets are equivelant. This example is not a proof but an explanation of how the set don't shrink when they are of the form from above.

Hence, we can say that the product of the elements of $\{1, 2, \ldots, p-1\} \equiv_p \{a, 2a, \ldots, (p-1)a\}$. The left hand side has a product of $(p-1)!$ while the right hand side is $a^{p-1}(p-1)!$:

$$\begin{aligned} (p-1)! &\equiv_p a^{p-1}(p-1)! \\ (p-1)!((p-1)!)^{-1} &\equiv_p a^{p-1}(p-1)!((p-1)!)^{-1} \\ 1 &\equiv_p a^{p-1} \end{aligned}$$

which is the claim of the theorem. □

Theorem. *If for any a, $a^{N-1} \equiv_N 1 \implies N$ is not prime.*

Hence, if we wanted to know if p is prime. We can pick some a at random and determine if $a^{p-1} \pmod p \equiv 1$. If not, then this number p is definitely not prime. If it is, we can't make any definite conclusions, not yet at least. To make this answer more conclusive, we define a few more theorems/definitions:

Definition. a is a witness for N if $a^{N-1} \not\equiv 1 \pmod N$

where witnesses prove that N is compositive (not prime).

Theorem. *If N has 1 witness, at least half of the numbers $< N$ are witnesses.*

Proof. Suppose a is a witness for N ($a^{N-1} \not\equiv 1 \pmod N$) then we can divide all the numbers less than N into two buckets for witnesses and non-witnesses, and then multiply every non-witness by a giving you a distinct witness.

$$(a \cdot b)^{N-1} = a^{N-1} \underbrace{b^{N-1}}_{\text{not a witness}} \equiv a^{N-1} \not\equiv 1$$

The claim here being made is that if b and c are distinct non-witnesses and there exists a single witness a, then ab, ac are both witnesses.

As a result, we can multiply the entire set of nonwitnesses by a single witness to get more witnesses with the same cardinality as the original set of nonwitnesses.

$$|\{\text{non witnesses}\}| \times a = |\{\text{new set of witnesses}\}|$$

The assumption here is that we have some witness a.

Theorem. *At least half of the numbers less than N (under modulo N) are witnesses if there exists at least one witness.*

Equipped with this knowledge, we can write an algorithm to check primality:

```
Function IsPrime($N$):
    for $i \leftarrow 1$ to $100$ do
        if $a^{N-1} \not\equiv 1$ then
            return False
    return True
```

The runtime of this algorithm is $O(n^3)$ as computing a^{N-1} takes $O(n^3)$ time and the loop is iterating $O(1)$ times. But, should we believe this algorithm?

If it returns false, you should believe it because the existence of that value a means N is not prime. But, can we run `True` by mistake? Yes, but are the chances?

We want to know the probability that N is composite but the algorithm retures `True`. From the argument above, if it is composite, then there must exist a witness for N (maybe not in $1 \to 100$).

However, the existence of any witness implies that more than half the numbers less than N (under modulo) are witnesses. Therefore, the question is equivelant to asking what is the probability that none of the 100 numbers are in that half?

$$P(\text{picked 100 non-witnesses}) \leq \frac{1}{2^{100}}$$

However, there exist a class of composite numbers that have no witnesses which would fail our algorithm (blindspot). Carmichael numbers, such as 561, fail this type of primality test, but is beyond the scope of the course.

Theorem. *A random n bit number has $1/n$ chance of being prime.*

Hence, we can make an algorithm for generating random primes given by:

```
Function RandomPrime($N$):
    while $True$ do
        $x \leftarrow$ RandomBitNumber($N$)
        if IsPrime($x$) then
            return $x$
```

With a probability of $1/n$ for a random n bit number to be prime, we expect to iterate n times before finding a prime. The `IsPrime` method takes $O(n^3)$ time giving a final runtime of $n \cdot O(n^3) = O(n^4)$.

RSA Encryption

Alice wants to send a message to Bob over an insecure channel. Eve is a spy and wants to eavesdrop and get the message that Alice is sending Bob. How do we prevent this?

Before using the insecure channel, you *could* create a secret code with them, but it has a prerequisite of meeting them. That sounds terrible. With RSA, we don't need to do any of this but still create a secure communication.

How does RSA work?

1. Bob generates two large primes: P and Q
2. He calculates $N = PQ$ (product of primes)
3. He also picks e such that $gcd(e, (p-1)(q-1)) = 1$
4. Finally, Bob also gets d such that $ed \equiv 1 \pmod{(p-1)(q-1)}$
5. Bob publishes two numbers: N and e (public key) and does not publish while the private keys are P, Q, d.

How does Alice use this? If she has a message m, then she can encrypt the message as $m' \equiv m^e \pmod{N}$ and sends m' to Bob. Bob can decrypt this because he (and only he) has the private key:

$$(m')^d \pmod{N} = m$$

Why is this secure? Because computing d requires $(p-1)(q-1)$ which requires factoring $N = PQ$, but computers cannot factor large numbers quickly.

To prove this works: we need to prove the following:

$$\begin{aligned} \left(m^e \pmod{N}\right)^d \pmod{N} &= m \\ (m^e)^d \pmod{N} &= m \\ m^{ed} \pmod{N} &= m \end{aligned}$$

By definition of modulo, we can write this as $N \mid m^{ed} - m$ simplifying our question to does N divide $m^{ed} - m$ evenly? Well, since $ed \equiv 1 \pmod{(p-1)(q-1)}$, we can write ed as $1 + k(p-1)(q-1)$ giving $N \mid m^{1+k(p-1)(q-1)} - m$:

$$\begin{aligned} m^{1+k(p-1)(q-1)} &\equiv m \pmod{N} \\ m \cdot m^{k(p-1)(q-1)} &\equiv m \pmod{N} \\ m \cdot (m^{p-1})^{k(q-1)} &\equiv m \pmod{pq} \end{aligned}$$

From here, we can simplify it to two parts based on $\pmod{p}$ and $\pmod{q}$. Without loss of generality (WLOG),

$$\begin{aligned} m \cdot (m^{p-1})^{k(q-1)} &\equiv m \pmod{p} \\ m \cdot (1)^{k(q-1)} &\equiv m \pmod{p} \\ m &\equiv m \pmod{p} \end{aligned}$$

From $m \equiv m \pmod{p}$ and $m \equiv m \pmod{q}$, we can prove the original claim that $m^{ed} \equiv m \pmod{pq}$ proving the decryption of RSA. □

RSA Example

How do we actually go about getting these values? We have a few unknowns to find when doing RSA: $P, Q, N, (p-1)(q-1), e, d$. First, we need to pick primes P and Q.

For sake of example, let us pick $P = 7$ and $Q = 17$ making $N = PQ = (7)(17) = 119$. Further, $(p-1)(q-1) = (7-1)(17-1) = 6(16) = 96$.

The rule for e is that $\gcd((p-1)(q-1), e) = 1$ meaning e has to be relatively prime to 96. We can pick 5 since $\gcd(96, 5) = 1$. Since 96 and 5 are relatively prime, we can run the Extended Euclidean algorithm to get numbers a, b such that $1 = 5a + 96b$ and pick a.

After running, we find that $a = 77$ and $b = 4$ as $77(5) = 4(96) + 1$ meaning d (decryption key) is 77.

Hence, the private key is: ($d = 77$) while the public key is: ($N = 119, e = 5$).