CS 70 Discrete Mathematics and Probability Theory
Fall 2013 Vazirani Note 4

This note is partly based on Section 1.4 of "Algorithms," by S. Dasgupta, C. Papadimitriou and U. Vazirani, McGraw-Hill, 2007.
An online draft of the book is available at http://www.cs.berkeley.edu/ vazirani/algorithms.html

Public Key Cryptography

Bijections

This note introduces the fundamental concept of a function, as well as a famous function called RSA that forms the basis of public-key cryptography. A function is a mapping from a set of inputs A to a set of outputs B: for input $x \in A$, $f(x)$ must be in the set B. To denote such a function, we write $f : A \to B$.

Consider the following examples of functions, where both functions map $\{0,\ldots,m-1\}$ to itself:

$$f(x) = x+1 \bmod m$$

$$g(x) = 2x \bmod m$$

A bijection is a function for which every $b \in B$ has a unique *pre-image* $a \in A$ such that $f(a) = b$. Note that this consists of two conditions:

1. f is *onto*: every $b \in B$ has a pre-image $a \in A$.
2. f is *one-to-one*: for all $a, a' \in A$, if $f(a) = f(a')$ then $a = a'$.

Looking back at our examples, we can see that f is a bijection; the unique pre-image of y is $y-1$. However, g is only a bijection if m is odd. Otherwise, it is neither one-to-one nor onto. The following lemma can be used to prove that a function is a bijection:

Lemma: For a finite set A, $f : A \to A$ is a bijection if there is an *inverse* function $g : A \to A$ such that $\forall x \in A$ $g(f(x)) = x$.

Proof: If $f(x) = f(x')$, then $x = g(f(x)) = g(f(x')) = x'$. Therefore, f is one-to-one. Since f is one-to-one, there must be $|A|$ elements in the range of f. This implies that f is also onto. $\square$

RSA

One of the most useful bijections is the RSA function, named after its inventors Ronald Rivest, Adi Shamir and Leonard Adleman:

$$E(x) \equiv x^e \bmod N$$

where $N = pq$ (p and q are two large primes), $E : \{0,\ldots,N-1\} \to \{0,\ldots,N-1\}$ and e is relatively prime to $(p-1)(q-1)$. The inverse of the RSA function is:

$$D(x) \equiv x^d \bmod N$$

where d is the inverse of e mod $(p-1)(q-1)$.

Consider the following setting. Alice and Bob wish to communicate confidentially over some (insecure) link. Eve, an eavesdropper who is listening in, would like to discover what they are saying. Let's assume that Alice wants to transmit a message x (a number between 1 and $N-1$) to Bob. She will apply her *encryption function* E to x and send the encrypted message $E(x)$ (also called the *cyphertext*) over the link; Bob, upon receipt of $E(x)$, will then apply his *decryption function* D to it. Since D is the inverse of E, Bob will obtain the original message x. However, how can we be sure that Eve cannot also obtain x?

In order to encrypt the message, Alice only needs Bob's *public key* (N,e). In order to decrypt the message, Bob needs his *private key* d. The pair (N,e) can be thought of as a public lock - anyone can place a message in a box and lock it, but only Bob has the key d to open the lock. The idea is that since Eve does not have access to d, she will not be able to gain information about Alice's message.

We will now prove that $D(E(x)) = x$ (and therefore $E(x)$ is a bijection). We will require a beautiful theorem from number theory known as *Fermat's Little Theorem*, which is the following:

Theorem 4.1: **[Fermat's Little Theorem]** For any prime p and any $a \in \{1,2,\ldots,p-1\}$, we have $a^{p-1} = 1 \bmod p$.

Let S be the nonzero integers modulo p; that is, $S = \{1,2,\ldots,p-1\}$. Define a function $f : S \to S$ such that $f(x) \equiv ax \bmod p$. Here's the crucial observation: f is simply a bijection from S to S; it permutes the elements of S. For instance, here's a picture of the case $a = 3, p = 7$:

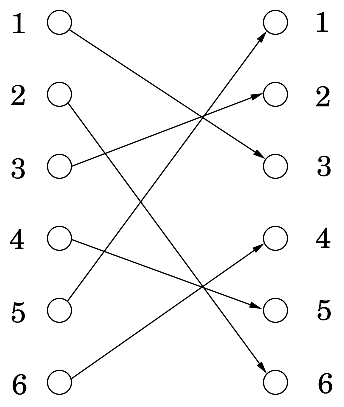

Figure 1: Multiplication by (3 mod 7)

With this intuition, we can now prove Fermat's Little Theorem:

Proof: Our first claim is that $f(x)$ is a bijection. We will then show that this claim implies the theorem.

To show that f is a bijection, we simply need to argue that the numbers $a \cdot i \bmod p$ are distinct. This is because if $a \cdot i \equiv a \cdot j \pmod p$, then dividing both sides by a gives $i \equiv j \pmod p$. They are nonzero because $a \cdot i \equiv 0$ similarly implies $i \equiv 0$. (And we *can* divide by a, because by assumption it is nonzero and therefore relatively prime to p.)

Now we can prove the theorem. Since f is a bijection, we know that the image of f is S. Now if we take the product of all elements in S, it is equal to the product of all elements in the image of f:

$$(p-1)! \equiv a^{p-1} \cdot (p-1)! \pmod p.$$

Dividing by $(p-1)!$ (which we can do because it is relatively prime to p, since p is assumed prime) then gives the theorem.

$\square$

Let us return to proving that $D(E(x)) = x$:

Theorem 4.2: Under the above definitions of the encryption and decryption functions E and D, we have $D(E(x)) = x \bmod N$ for every possible message $x \in \{0,1,\ldots,N-1\}$.

The proof of this theorem relies on Fermat's Little Theorem:

Proof of Theorem 6.2: To prove the statement, we have to show that

$$(x^e)^d = x \bmod N \qquad \text{for every } x \in \{0,1,\ldots,N-1\}. \tag{1}$$

Let's consider the exponent, which is ed. By definition of d, we know that $ed = 1 \bmod (p-1)(q-1)$; hence we can write $ed = 1 + k(p-1)(q-1)$ for some integer k, and therefore

$$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x(x^{k(p-1)(q-1)} - 1). \tag{2}$$

Looking back at equation (1), our goal is to show that this last expression in equation (2) is equal to 0 mod N for every x.

Now we claim that the expression $x(x^{k(p-1)(q-1)} - 1)$ in (2) is divisible by p. To see this, we consider two cases:

Case 1: *x is not a multiple of p.* In this case, since $x \neq 0 \bmod p$, we can use Fermat's Little Theorem to deduce that $x^{p-1} = 1 \bmod p$. Then $(x^{p-1})^{k(q-1)} \equiv 1^{k(q-1)} \bmod p$ and hence $x^{k(p-1)(q-1)} - 1 = 0 \bmod p$, as required.

Case 2: *x is a multiple of p.* In this case the expression in (2), which has x as a factor, is clearly divisible by p.

By an entirely symmetrical argument, $x(x^{k(p-1)(q-1)} - 1)$ is also divisible by q. Therefore, it is divisible by both p and q, and since p and q are primes it must be divisible by their product, $pq = N$. But this implies that the expression is equal to 0 mod N, which is exactly what we wanted to prove. $\square$

So we have seen that the RSA protocol is *correct*, in the sense that Alice can encrypt messages in such a way that Bob can reliably decrypt them again. But how do we know that it is *secure*, i.e., that Eve cannot get any useful information by observing the encrypted messages? The security of RSA hinges upon the following simple assumption:

Given N, e and $y = x^e \bmod N$, there is no efficient algorithm for determining x.

This assumption is quite plausible. How might Eve try to guess x? She could experiment with all possible values of x, each time checking whether $x^e = y \bmod N$; but she would have to try on the order of N values of x, which is completely unrealistic if N is a number with (say) 512 bits. This is because $N \approx 2^{512}$ is larger than estimates for the age of the Universe in femtoseconds! Alternatively, she could try to factor N to retrieve p and q, and then figure out d by computing the inverse of $e \bmod (p-1)(q-1)$; but this approach requires Eve to be able to *factor* N into its prime factors, a problem which is believed to be impossible to solve efficiently for large values of N. She could try to compute the quantity $(p-1)(q-1)$ without factoring N; but it is possible to show that computing $(p-1)(q-1)$ is equivalent to factoring N. We should point out that the security of RSA has not been formally proved: it rests on the assumptions that breaking RSA is essentially tantamount to factoring N, and that factoring is hard.

We close this note with a brief discussion of implementation issues for RSA. Since we have argued that breaking RSA is impossible because *factoring* would take a very long time, we should check that the computations that Alice and Bob themselves have to perform are much simpler, and can be done efficiently.

There are really only two non-trivial things that Alice and Bob have to do:

1. Bob has to find prime numbers p and q, each having many (say, 512) bits.
2. Both Alice and Bob have to compute exponentials mod N. (Alice has to compute $x^e \bmod N$, and Bob has to compute $y^d \bmod N$.)

Both of these tasks can be carried out efficiently. The first requires the implementation of an efficient test for primality as well as a rich source of primes. You will learn how to tackle each of these tasks in the algorithms course $CS170$. The second requires an efficient algorithm for modular exponentiation, which is not very difficult, but will also be discussed in detail in $CS170$.

To summarize, then, in the RSA protocol Bob need only perform simple calculations such as multiplication, exponentiation and primality testing to implement his digital lock. Similarly, Alice and Bob need only perform simple calculations to lock and unlock the the message respectively—operations that any pocket computing device could handle. By contrast, to unlock the message without the key, Eve would have to perform operations like factoring large numbers, which (at least according to widely accepted belief) requires more computational power than all the world's most sophisticated computers combined! This compelling guarantee of security without the need for private keys explains why the RSA cryptosystem is such a revolutionary development in cryptography.