18.200 Lecture notes May 6, 2024

Cryptography, RSA and Secret Sharing

Lecturer: Ankur Moitra

1 Crytography and Cryptology

Cryptography is the art and science of keeping secrets. It can be traced back to ancient times, for example with the Caesar cipher that works as follows: If we have a message such as

HAIL CAESAR

that we would like to send, we decide on a secret key k which is an integer between 0 and 25. Then we modularly shift the letters in the message by k. For example, if we choose $k = 3$, we would get the garbled message:

KDLO FHDVDU

Now how do we decode a message? If we know the secret key, we can just modularly shift the letters by k in the opposite direction. For example, try for yourself decoding the following message:

HW WX EUXWH?

From this simple example, we can already see the basic elements of a cryptographic scheme:

(1) An encryption is a function that maps a message to ciphertext.

(2) A decryption is a function that maps ciphertext to a message

(3) A secret key is a parameter used to specify the particular encryption/decryption functions from among a family of possibilites.

We can also learn an important lesson from this simple cryptographic scheme: They might not necessarily be as secure as they seem. For example, while the message above looks garbled, we could find the true message from the ciphertext with a simple attack. We could try all possible secret keys in the hopes that only the correct one will result in a readable message.

Another popular and simple cryptographic scheme, which will turn out to have a different issue, is the following: A substitution cipher fixes a permutation from letter to letters. For example, we could have the function $\phi(A) = R$, $\phi(B) = C$, $\phi(C) = D$, $\phi(D) = Q$, and so on. Now the message

BAD CAB

becomes

CRQ DRC

and we could similarly decrypt using the inverse of ϕ. Now there are tons of possible secret key! In fact there are $26! \approx 4\dot{1}0^{26}$. But there are still simple attacks that allow an eavesdropper to figure out the substitution cipher. One approach is to make deductions based on how often certain letters appear in the ciphertext. In particular, it is reasonable to assume that the most common letter in a large enough collection of cyphertexts is what the letter E gets mapped to, and so on. For centuries, cryptography was more art than science. But by now we have cryptographic schemes that appear to be secure (at least against classical computers) and are widely used everyday!

2 Public Key Cryptosystems

The traditional way of creating secret codes (used in various degrees of sophistication for centuries) is that both the sender and the receiver share a secret, called a key. The sender scrambles the message in a complicated way that depends on the key. The receiver then uses the key to unscramble the message. If these codes are constructed properly (something which is surprisingly hard to do), it seems virtually impossible for somebody without the key to decode the message, even if they have many examples of pairs of messages and their encodings to work from in trying to deduce the key.

The drawback of this method is ensuring that every pair of people who need to communicate secretly have a shared secret key. Distributing and managing these keys is surprisingly difficult even when these keys are used for espionage. This would be extremely difficult for secret communication over the internet, when you may wish to send your credit card securely to a store you have never before heard of. Luckily, there is another way to proceed.

Diffie and Hellman in 1976 came up with a scheme for handling such communications, called a *public key cryptosystem*. It is based on the assumption that there is a wide class of functions that are relatively easy to compute but extraordinarily difficult to invert unless you possess a secret.

According to this scheme, each communicator or recipient, say Bob or B, publishes in a well defined place (a kind of telephone directory) a description of his function, f_B from this class; this is Bob's *public key*. Bob knows also the inverse of f_B, this is his *private key*. The assumption is that this inverse is extremely difficult to compute if one does not know some private information.

Suppose now that someone, say Alice or A, would like to send a message m to B. She looks up Bob's public key and sends $m' = f_B(m)$. Since Bob knows his own private key, he can recover $m = f_B^{-1}(m')$. The problem here is that Bob has no guarantee that Alice sent the message. Maybe someone else claiming to be Alice sent it. So, instead, suppose that Alice sends the message $m' = f_B(f_A^{-1}(m))$ to Bob. Notice that Alice needs to know Bob's public key (which she can find in the directory) and also her own private key, which is known only to her. Having received this message, Bob can look up Alice's public key and recover m by computing $m = f_A(f_B^{-1}(m'))$; again, for this purpose, knowledge of Bob's private key and Alice's public key is sufficient. Anyone else would have to solve the said-to-be-extraordinarily-difficult task of inverting the action of one or another of these functions on some message in order to read the message or alter it in any way at all. This is the basic setup for a public-key cryptosystem. One can also use it for digital signatures. If Alice wants to show to anyone that she wrote message m, she can publish or send $m' = f_A^{-1}(m)$, and anyone can test it came from Alice by computing $f_A(m')$.

We will discuss one of the classes of problems that have been suggested and deployed for communications of this public key type. It was actually developed here at M.I.T., a number of years ago and is known as the RSA public key cryptosystem. It was invented by three MIT people,

Ron Rivest, Adi Shamir and Len Adleman in 1977. Their scheme is based on the fact that it is easy to multiply two large numbers together, but it appears to be hard to factor a large number. The record so far for factoring has been to factor a 768-bit number (i.e., 232 digits) given in an RSA challenge, and this took the equivalent of 20,000 years of computing on a single-core machine... The task of factoring a 1024-bit number appears to be 1,000 harder with the current algorithms.

3 The RSA code

For this code, choose two very large prime numbers (say with several hundred digits), p and q, and form the product $N = pq$. Choose a number $z < N$ such that z is relatively prime to $(p-1)(q-1) = N - p - q + 1$. Knowing p and q (or $(p-1)(q-1)$) we can find the multiplicative inverse y to z modulo $(p-1)(q-1)$ by the extended Euclidean algorithm. The pair (N, z) constitutes the public key, and (N, y) constitutes the private key.

If we want to encode a message, we first view it as a number in base N. Every digit is a number between 0 and $N-1$ and we will encode each digit $0 \leq m < N$ separately. The sender computes $s = m^z \bmod N$ and transmits s. Upon receiving s, the receiver, who knows the private key, computes $s^y \bmod N$. The claim is that this is precisely m, i.e. $m = s^y \bmod N$.

If one could factor N into $N = pq$ then one can easily compute y from z (by the extended Euclid algorithm) and therefore break this public-key cryptosystem. However, as we said previously, factoring large numbers appears to be very challenging.

Why does this scheme work? Let $x = s^y$; we want to show that $m = x \bmod N$. We have that

$$x = s^y = m^{yz} \pmod{N},$$

and since z and y are multiplicative inverses modulo $(p-1)(q-1)$, we get that

$$x = m^{1+k(p-1)(q-1)} = mm^{k(p-1)(q-1)} \pmod{N}.$$

We want to prove that this is equal to $m \bmod N$. We will first show that $x \equiv m \pmod{p}$ and $x \equiv m \pmod{q}$ and then deduce from the Chinese remainder theorem that $x \equiv m \pmod{pq}$. To show that $x \equiv m \pmod{p}$, we need to consider two cases. First, if m is a multiple of p, then x is also a multiple of p, and so $x \equiv m \equiv 0 \pmod{p}$. Otherwise, if m is not a multiple of p then it must be relatively prime to p (since p is prime): $\gcd(m, p) = 1$. Thus we can apply Fermat's Little Theorem, which tells us that $m^{p-1} \equiv 1 \pmod{p}$, and thus

$$m^{k(p-1)(q-1)} \equiv 1^{k(q-1)} \equiv 1 \pmod{p}.$$

Multiplying by m, we indeed obtain $x \equiv m \pmod{p}$.

We can apply the same argument to q, to obtain that $x \equiv m \pmod{q}$ also. Thus the Chinese remainder theorem tells us that $x \equiv m \pmod{N}$, and we have proved the correctness of the RSA scheme.

In order to use RSA, we need to show how to generate large primes, and also how to efficiently compute $m^z \bmod N$ (or $s^y \bmod N$) when z is very large.

4 Raising a Number to a High Power

In this section, we show how to raise a number to a high power modulo N efficiently, say $m^z \bmod N$. The technique is known as repeated squaring and the idea is very simple.

If z is a power of 2, say 2^k, we can compute $m^z \bmod N$ by starting with m, and repeatedly (k times) squaring it and taking it modulo N. This requires only k multiplications. For example, to compute $3^{32} \bmod 83$, we first compute $3^2 \equiv 9 \pmod{83}$ then $3^4 \equiv 9^2 \equiv 81 \pmod{83}$, then $3^8 \equiv 81^2 \equiv (-2)^2 \equiv 4 \pmod{83}$ then $3^{16} \equiv 4^2 \equiv 16 \pmod{83}$, and finally $3^{32} \equiv 16^2 \equiv 256 \equiv 7 \pmod{83}$. Even though 3^{32} is more than 10^{15}, the fact that we did every operation modulo 83 meant that we never had to deal with large numbers.

In general, when z is not necessarily a power of 2, we write z's binary representation. The binary representation can be obtained by repeatedly subtracting 0 or 1 to make z even and divide by 2; the sequence of 0's and 1's we had to subtract correspond to the binary representation from least to most significant bits. Suppose that z requires d bits, and let z_k denote the k leading bits of z. Observe that $z_k = 2z_{k-1}$ or $z_k = 2z_{k-1} + 1$ depending on the kth bit of z. We compute $m^{z_k} \bmod N$ for $k = 1, \cdots, d$. For $k = 1$, this is simply $m^1 \bmod N$. Once we have computed $a_{k-1} = m^{z_{k-1}} \bmod N$, it is easy to compute a_k. Square a_{k-1}, multiply it by m if the kth leading bit is a 1, and do these operations modulo N. We then get a_k and we can repeat.

Example. Suppose $z = 201$. Then z can be represented by 11001001 in binary. These bits (in reverse order) are obtained from: $z_8 = 201 = \mathbf{1} + 2 \cdot 100$, $z_7 = 100 = \mathbf{0} + 2 \cdot 50$, $z_6 = 50 = \mathbf{0} + 2 \cdot 25$, $z_5 = 25 = \mathbf{1} + 2 \cdot 12$, $z_4 = 12 = \mathbf{0} + 2 \cdot 6$, $z_3 = 6 = \mathbf{0} + 2 \cdot 3$, $z_2 = 3 = \mathbf{1} + 2 \cdot 1$ and $z_1 = \mathbf{1}$.. We then compute $m^{z_k} \bmod N$ for $k = 1$ to 8 in 8 steps, in which (i) we square and multiply by m in steps 1, 2, 5, 8 and (ii) we only square in steps 3, 4, 6, and 7.

5 Digital Signatures

Suppose now we not only want to send a message securely but also want to ensure that a message really did come from Alice. In the physical world, we might use a signature to verify the identity of the author. Can we design a digital analogue of a signature? It turns out that we can use RSA to do exactly that.

Recall that in the RSA scheme, Alice finds two very large prime numbers p and q and forms the composite $N = pq$. She then finds a number $z < N$ such that z is relatively prime to $(p-1)(q-1) = N - p - q + 1$ and its multiplicative inverse y to z modulo $(p-1)(q-1)$ by the extended Euclidean algorithm. Finally she publishes the pair (N, z), which is called the public key, and keeps the pair (N, y) to herself, which is called the private key.

The main insight is that the role of the private and public key can be swapped. Suppose Alice's public key is (N_A, z_A) and her private key is (N_A, y_A). Similarly let Bob's public key be (N_B, z_B) and his private key be (N_B, y_B). Now we can construct a digital signature as follows: If Alice wants to send an authenticated message m to Bob, she applies the decryption function $s = m^{y_A} \bmod N_A$. Now Alice can encrypt the message as usual as $c = m^{z_B} \bmod N_B$ and transmits (c, s). Upon receiving this transmission, Bob first computes the decrypted message $m = c^{y_B} \bmod N_B$. He then checks the signature by using Alice's public key to compute $s^{z_A} \bmod N_A$. This should be equal to m. The main point is that forging the signature without knowing Alice's private key is difficult. In fact this scheme not only ensures that the message did come from Alice, but that it has not

been tampered with along the way. Modern digital signature schemes often work with a hash of the message, to keep signatures short.

6 Secret Sharing Schemes

In this section, we will build on the notes on modular arithmetic and elementary group theory, and fields and study polynomial interpolation which we will apply to construct secret sharing schemes. The goal of a secret sharing scheme is to divide a secret among n parties so that any k of them can reconstruct the secret but any fewer gives absolutely no information about the secret.

Recall $(\mathbb{F}, +, *)$ is a *field* if

1. $(\mathbb{F}, +)$ is an Abelian group with identity element denoted 0.
2. $(\mathbb{F} - \{0\}, *)$ is an Abelian group with identity element denoted $1 \neq 0$.
3. *Distributive property.* For all $a, b, c \in \mathbb{F}$, $a * (b + c) = (a * b) + (a * c)$.

Moreover $(\mathbb{Z}_m, \oplus, \otimes)$ is a field if and only if m is a prime. The rationals, real or complex numbers (with usual addition and multiplication) are also fields. Also recall the fundamental theorem of algebra:

Theorem 1. *A polynomial of degree $d \geq 1$ with coefficients in a field $\mathbb{F}$ can have at most d roots in $\mathbb{F}$.*

With this in hand, we can now introduce Lagrange interpolation:

Theorem 2. *Given $d + 1$ pairs $(x_0, y_0), (x_1, y_1), \cdots, (x_d, y_d)$ where each x_i and y_j is in a field $\mathbb{F}$ and the points $x_0, x_1, \cdots, x_d$ are distinct, there is a unique degree d polynomial $P(x)$ that passes through these points.*

In particular by passing through the points, we mean that $P(x_i) = y_i$ for all $i = 0, 1, \cdots d$. In fact you can write down the interpolating polynomial explicitly as

$$P(x) = \sum_{i=0}^{d} y_i P_i(x)$$

where

$$P_i(x) = \prod_{j=0, j\neq i}^{d} \frac{x - x_j}{x_i - x_j}$$

Proof. For any x_i we have $P_{i'}(x)$ is 1 if $i = i'$ and 0 otherwise. Thus $P(x_i) = y_i$ for all i. Now to prove uniqueness, suppose for the sake of contradiction that there is another polynomial $Q(x)$ also of degree at most d that passes through these same points. But then $P(x) - Q(x)$ is a non-zero polynomial of degree at most d but has $d + 1$ zeros, at $x_0, x_1, \cdots, x_d$ which contradicts the fundamental theorem of algebra. □

Now we can introduce Adi Shamir's famous secret sharing scheme, which works as follows:

1. Let p be a prime with $p > n$. Choose random coefficients $a_1, a_2, \cdots, a_{k-1}$ where each $a_i \in \mathbb{Z}_p$

2. Let $a_0 = s$ where s is the secret. Set $P(x) = a_0 + a_1 x + a_2 x^2 \cdots a_{k-1} x^{k-1}$.

3. Send participant i the pair $(i, P(i))$.

Lemma 1. *Any k participants can reconstruct the secret s.*

Proof. Suppose the k participants are $i_1, i_2, \cdots, i_k$. Then they can find an interpolating polynomial of degree at most $k-1$ at the points

$$(i_1, P(i_1)), (i_2, P(i_2)), \cdots, (i_k, P(i_k))$$

All the i's are distinct and thus the interpolating polynomial exists and is unique. Then they can compute $P(0)$ to recover the secret s. □

Defining precisely what we mean by security is outside the scope of this course, but we will show informally that $k-1$ participants cannot meaningfully reconstruct the secret. Let's imagine that $i_1, i_2, \cdots, i_{k-1}$ are the $k-1$ participants and let i_k be any other non-participant. Then we claim that any interpolating polynomial $Q(x)$ of degree at most $k-1$ that passes through

$$(i_1, P(i_1)), (i_2, P(i_2)), \cdots, (i_{k-1}, P(i_{k-1})), (i_k, y_k)$$

for $y_k \neq P(i_k)$ must have the property that $Q(0) \neq s$. The reason is if it did then $P(x) - Q(x)$ would be a non-zero but would have k zeros which again contradicts the fundamental theorem of algebra.

18.200 Principles of Discrete Applied Mathematics

Spring 2024