CLASS 18, GIVEN ON 11/01/2010, FOR MATH 25

1. An old cryptosystem: the Caesar cipher

We will take a brief detour to learn about a simple and elegant, but important, application of the theory we have been learning (and in particular, the Fermat-Euler theorem) to the real world. A problem which is as old as civilization itself is the question of how to transmit information from one party to another in such a way so that third parties ('adversaries') are unable to read that information, even if they intercept it mid-transit. Historically, this was most important to governments, especially during wartime, but in the present this problem is relevant to everybody, since lots of important information is transmitted over the Internet nowadays.

The practice of transmitting information in a way to prevent unwanted parties from reading that information even if they intercept it is called cryptography. An ancient method for encoding information in the Latin alphabet was used by Julius Caesar, when he was a general subjugating 'barbarian' civilizations for the then Roman Republic. When he wanted to transmit orders or information to his generals, he would take whatever text he wanted to transmit – known in cryptography as the *plaintext*, and then shift every letter by some constant amount. For instance, perhaps he took the plaintext and replaced each letter with the one which came three places later in the alphabet. In the case of X, Y, and Z, he would shift those to A, B, and C, respectively. A message like "HELLO" would then become "KHOOR". The result of encrypting a plaintext is known as the *ciphertext*. (Obviously he never sent the word "hello" since he spoke Latin, not English, but the idea is the same.) We will disregard the question of what happens if we want to transmit numbers in this way; the above scheme can easily be extended to account for numbers as well.

Mathematically, we can think of Caesar's method, nowadays known as the *Caesar cipher*, as an example of mod 26 arithmetic. If we think of 'A' as synonymous with the congruence class 0 mod 26, 'B' with 1 mod 26, ..., 'Z' with 25 mod 26, then the above example corresponds to the operation $x \mod 26 \mapsto x + 3 \mod 26$. Of course, there is nothing special with the map $x \mapsto x + 3$; we could have used any $x \mapsto x + a$, where probably we would want $a \not\equiv 0 \mod 26$ – otherwise, this map doesn't change the message at all, and would we worthless as an encryption method.

It is probably true that this encryption method was secure back in Caesar's time. More likely than not, his enemies were illiterate, if not in their own languages, then certainly in Latin, and even if some people had passing literacy in Latin, it seems somewhat unlikely that they knew enough to break the Caesar cipher. However, if you think about it, the Caesar cipher is very easy to break. Suppose we did not even know what the value of a in the map $x \mapsto x + a$ was, but we knew that there was some value of a which was being used in this way to encode a message. Then all we have to do is try out all 26 different values of a mod 26, and find the one result which actually makes sense as a message. For instance, "KHOOR" with $a = 3$ yields "HELLO", but with $a = 4$ yields "GDKKN", which makes no sense. It is basically impossible for any message of appreciable length to make sense for two different values of a.

So perhaps you decide that the Caesar cipher is weak and decide to modify it. One possible way is to use a map $x \mapsto ax + b$ instead of $x \mapsto x + a$. Of course, you need to make sure that a, b are selected in such a way to ensure that there is a unique way to decrypt any encrypted message. For instance, if $a = 26$, then $x \mapsto 26x + b \equiv b \mod 26$ would

not be effective, because every letter would change to the same letter, and there would be no way to recover the original message from an encrypted message. However, suppose we got around this problem. More generally, maybe we just decide to use some bijective map $f : \mathbb{Z}/26\mathbb{Z} \to \mathbb{Z}/26\mathbb{Z}$ to encrypt a message. It seems much harder to break this using brute force, because there are 26! different such maps; way too much for anyone to try by hand, although perhaps it might be breakable using brute force on a computer.

However, brute force is far from the best way to break such an encryption scheme. Suppose we intercept a message which we know is encoded using a map $f : \mathbb{Z}/26\mathbb{Z} \to \mathbb{Z}/26\mathbb{Z}$ to permute the letters of the alphabet. It is a statistical fact that in any English message of appreciable length (say, of paragraph length), that regardless of the actual subject matter of the message, certain letters appear most frequently. In particular, the letters 'e', 't' are almost always the most common letters in the English language. With this in mind, one can systematically try to break an encrypted message using this type of *frequency analysis*.

So the Caesar cipher seems quite unsecure against any adversary who knows some basic principles of cryptography. Modifications of the Caesar cipher, some of which are quite hard to break, even with frequency analysis, were developed all the way through the 19th century. However, all of the systems shared a common feature. Notice that in the Caesar cipher, knowing how to encrypt a message is equivalent to knowing how to decrypt a message, and conversely knowing how to decrypt a message is equivalent to knowing how to encrypt a message. This means that the means by which a message must be encrypted must be distributed to all appropriate parties somehow before the encryption method is actually used, and the encryption method must be kept secret from all potential adversaries.

For instance, if Caesar's generals wanted to decode their messages, they needed to know the actual value amount of shift to apply to each letter to encrypt/decrypt a message. This has to be agreed upon somehow; perhaps they were given instructions in Rome when they were together. But perhaps a general is captured during battle; there is the possibility that his captors will determine how to decrypt messages by searching his papers or simply forcing the knowledge out of him.

So in summary, the Caesar cipher is a simple method of encrypting text which does not require a lot of effort to encrypt or decrypt, but has the drawbacks that it can be easily broken via frequency analysis, and also the fact that knowledge of encryption is equivalent to knowledge of decryption.

2. A modern cryptosystem: RSA

Cryptography become more and more important as time went by; in World War II cryptography played a critical role in the advantage the Allies had over the Axis as the war went on. For instance, in the Battle of Midway, one of the most important naval battles in the Pacific Theater, the Americans had a decisive advantage over the Japanese when American codebreakers were able to decrypt Japanese communications and obtain information on the location and strength of Japanese forces before the battle began. It was already clear that mathematicians, and people who we would now describe as computer scientists, as well as engineers, had much to contribute to the practice of war.

We will describe a cryptosystem which was developed in the late 1970s by Rivest, Shamir, and Adleman, now known as RSA. It has the elegant feature of being easily implemented and understood by anybody with some knowledge of number theory, and of course has the practical feature that as of modern times, no one knows of any efficient way to break the RSA cryptosystem. We will not actually describe the precise way in which RSA is implemented; however, we will describe the basic mathematical idea underlying RSA. Improvements in its security have been made over the past few decades, but the basic idea remains intact.

In contrast to the example we used with the Caesar cipher, we will only transmit numbers using RSA. In practice, this is no drawback, because we can always agree in some predetermined way on how to convert letters and other characters to numbers. (Perhaps we use ASCII, or some other custom conversion.) It does not matter if this method is public knowledge or not.

So suppose there are two people, Alice and Bob, and Bob wants to send a message, which is the number x, securely to Alice. Here is how the RSA system works. Alice begins by choosing two large, distinct prime numbers, p and q. She keeps them secret, and does not tell anybody what they are. She calculates $N = pq$. She then calculates $\phi(N)$. We know that $\phi(N) = (p-1)(q-1)$, so for Alice it is easy for her to compute what $\phi(N)$ is. Alice then chooses some integer e satisfying $1 < e < \phi(N)$ and $\gcd(e, \phi(N)) = 1$. The second condition is not automatically satisfied for arbitrary choices of e, but it is fast to calculate $\gcd(e, \phi(N))$, using Euclid's algorithm, and in practice Alice will not have to make too many different choices of e to find one which is coprime to $\phi(N)$. This number e is sometimes called the *encryption exponent*.

Alice then publicly releases the ordered pair (N, e), which is called Alice's *public key*. (Remember, Alice keeps $\phi(N), p, q$ private.) Anybody can see this; certainly Bob can see this, but any potential adversaries can see this as well. If Bob wants to send the message x, which we now assume satisfies $0 \leq x < N$ (if not, Bob can break up his message into smaller pieces, each of which is represented by a number less than N, and transmit them in sequence), he calculates $x^e \mod N$. Notice that this is possible for Bob to do, because Alice has made N and e public knowledge, and is fast to do, since $x^e \mod N$ can be calculated quickly, even if N and e are large. Bob then sends $x^e \mod N$ to Alice. This is the encrypted message.

How does Alice decrypt this message? She begins by calculating a positive number d such that $de \equiv 1 \mod \phi(N)$. We sometimes call this d the *decryption exponent*. This is possible because $\gcd(e, \phi(N)) = 1$, and is fast, because she can use the extended version of Euclid's algorithm to solve this linear congruence. To decrypt $x^e \mod N$, the encrypted message she received from Bob, she raises this message to the dth power mod N. Because $de \equiv 1 \mod \phi(N)$, we may write $de = 1 + k\phi(N)$ for some positive integer k. We then have

$$(x^e)^d = x^{de} = x^{1+k\phi(N)} \equiv x(x^{\phi(N)})^k \mod N.$$

If $\gcd(x, N) = 1$, then the Fermat-Euler theorem tells us that $x^{\phi(N)} \equiv 1 \mod N$, so the above expression is $\equiv x \mod N$, which is Bob's message. As a matter of fact, one can check (HW assignment) that this is true even if $\gcd(x, N) \neq 1$, so that Bob does not need to worry about checking if $\gcd(x, N) = 1$.

Example. Alice choose $p = 2543, q = 3833$. (One can check these are both prime.) Then $N = pq = 9747319$, and $\phi(N) = 9740944$. Clearly $\gcd(5, \phi(N)) = 1$, so Alice chooses $e = 5$. She publishes $(9747319, 5)$ as her public key. She then privately calculates that $d = 1948189$ satisfies $de \equiv 1 \mod \phi(N)$.

Bob decides he wants to send the message $x = 2010$ to Alice. Using a computer or calculator, he finds that $x^5 \equiv 9220872 \mod N$, so he sends 9220872 to Alice. Alice receives this message, and then using her own computer or calculator determines that $9220872^d = 9220872^{1948189} \equiv 2010 \mod 9747319$, thereby recovering Bob's original message.

This describes how Bob can send encrypted messages to Alice, and then how Alice can decrypt the messages she receives. But for this to truly be a worthwhile cryptosystem, it

should be very difficult for adversaries to decrypt encrypted messages, given public knowledge of the cryptosystem they are using and Alice's public key.

For instance, let's pretend that the adversary Eve has intercepted $x' \mod N$ in transit (in the example above, say she found 9220872). Eve knows that Bob's message is the number x which satisfies $x^e \equiv x' \mod N$. How might she go about actually determining what x is?

One approach is to try to figure out what Alice knows. After all, since $N = pq$, if Eve is able to factor N, then she will have all the information she needs to determine d and then decrypt any messages sent to Alice. So at the very least, if Eve can factor N, she will be able to decrypt messages sent to Alice. Perhaps it is possible, using some very clever technique, to recover x from $x^e \mod N$ knowing only x and N; this is known as the RSA problem. Factorization is at least as hard as RSA (since factorization solves RSA), but it is currently unknown whether RSA is actually easier than factorization. Right now, any practical approach to breaking RSA requires factoring N.

In the example above, it is easy to factor N with a computer, or even by hand (in a few hours), since N is not particularly large. But this is only because p, q were chosen to be small primes. In practice, p, q will be hundreds of digits long. The reason why RSA is an effective cryptosystem in modern times is due to the fact that it is possible to find primes which are hundreds, if not thousands of digits long, using modern primality testing methods, but it is very time-consuming to factor numbers thousands of digits long if they have no small factors. As computing power grows, larger numbers can be factored, but then we can just start using larger values of p, q, which will be possible because increased computing power will allow us to generate larger primes. In this way, so long as no breakthrough in the problem of factoring integers or the RSA problem occurs, it is likely that RSA will remain secure even as computers get more powerful.