Applications to cryptography

In this lecture we apply some of the theory we have developed to Cryptography, which is the subject of sending secret messages. Our main application is to describe the RSA cryptosystem, a widely used method for secure data transmission.

We begin with a simple example of a *private key cryptosystem*, which most people are familiar with in some form. Let us write each letter from A to Z as a number in $\mathbb{Z}_{26}$. For convenience we also write each number with 2 digits. We have the following correspondence:

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Now, suppose we have a message we would like to send in secret: ALGEBRA. We first write this in numbers as 00110604011700, by replacing A with 00, and so on. Now we encrypt the message using our private encryption key: the function $f : \mathbb{Z}_{26} \to \mathbb{Z}_{26}$ given by $f(k) = k + 10 \pmod{26}$, say. Thus 00 gets sent to $f(0) = 0 + 10 \equiv 10 \pmod{26}$, 11 is sent to $f(11) = 11 + 10 \equiv 21 \pmod{26}$, and so on. Our message 00110604011700 transforms to

$$10211614110110$$

We may write this in letters as KVQOLBK. We send the message in this form to a friend. Once our friend receives the message, to decode it they only need apply f^{-1}, i.e. the function on $\mathbb{Z}_{26}$ which is $k \pmod{26} \mapsto k - 10 \pmod{26}$.

The above cryptosystem, which is ancient and used for example by Caesar, is called a *private key* cryptosystem because the key (in this case the function f or its inverse) must be kept private for the system to work effectively. This scheme also has the property that anyone who has the key, i.e. knowledge of the function f, can easily encode and also decode messages. This is because the function f^{-1} is easy to find if f is known, and vice versa. In particular, anyone who can encode messages can also decode messages.

We now describe a more sophisticated cryptosystem which is used in the modern world and relies on some of the algebra we have developed thus far. It is called *RSA*, named after Rivest, Shamir, and Adleman, who described the method in 1978. The algorithm was actually discovered earlier in 1973 by Clifford Cocks working for British intelligence, but this information was only declassified in 1997.

To begin, choose two large distinct prime numbers at random, say p and q. These numbers are meant to be kept secret from the public. Let $n = pq$ and $N = (p-1)(q-1)$. Choose an integer e which satisfies $1 < e < N$ and $\gcd(e, N) = 1$. We know that there exists an integer d that satisfies $ed \equiv 1 \pmod{N}$, and we compute this. The scheme is as follows:

Public information (encryption key): the number $n = pq$ and integer e.

Private information (decryption key): the prime numbers p, q and the integer d.

Note that although $n = pq$ is known to the public, the factorization of n into the prime numbers p and q is private information. The robustness of RSA is based on the principle that factoring numbers is hard!

Now we describe the simplest version of the algorithm. Suppose someone from the public, "Bob", wants to send an encrypted message to "Alice", who holds the private information of the decryption key. Bob knows only the information n, e while Alice knows p, q and d. Assume Bob's message is in the format of a number $m \in \mathbb{Z}_n^\times$. The algorithm is as follows:

1. Bob encodes his message $m \in \mathbb{Z}_n^\times$ by computing $c = m^e \pmod{n}$.
2. Bob sends his encrypted message $c \in \mathbb{Z}_n^\times$ to Alice.
3. Alice then uses d to compute $c^d \pmod{n}$, which recovers $m \in \mathbb{Z}_n^\times$.

Why does this work? First, we should check that $c^d \equiv m \pmod{n}$, i.e. that Alice's decryption actually recovers the original message. We need the following:

▶ **Let a, b be relatively prime. Then $\phi(ab) = \phi(a)\phi(b)$.**

To see this, recall that $\phi(a) = |\mathbb{Z}_a^\times|$. We define a map from $\mathbb{Z}_{ab}^\times$ to the Cartesian product $\mathbb{Z}_a^\times \times \mathbb{Z}_b^\times$ be sending $k \pmod{ab}$ to the pair $(k \pmod{a}, k \pmod{b})$. This is well-defined: if k is relatively prime to ab it is then certainly relatively prime to each of a and b. It is a 1-1 map: if $k, k' \in \mathbb{Z}_{ab}^\times$ are sent to the same pairs, then $k \equiv k' \pmod{a}$ and $k \equiv k' \pmod{b}$, and so $k - k'$ is divisible by both a and b. Since $\gcd(a, b) = 1$, it follows that $k - k'$ is divisible by ab, and so $k \equiv k' \pmod{ab}$. It is onto: let r, s be such that $ar \equiv 1 \pmod{b}$ and $bs \equiv 1 \pmod{a}$. Then for any pair $(k_1 \pmod{a}, k_2 \pmod{b}) \in \mathbb{Z}_a^\times \times \mathbb{Z}_b^\times$ consider

$$k = k_1 bs + k_2 ar$$

This gives an element of $\mathbb{Z}_{ab}^\times$ and it reduces mod a to k_1, and reduces mod b to k_2. Thus the map is onto. We have thus produced a 1-1 onto map from $\mathbb{Z}_{ab}^\times$ to $\mathbb{Z}_a^\times \times \mathbb{Z}_b^\times$. In particular these sets have the same size. Thus $\phi(ab) = |\mathbb{Z}_{ab}^\times| = |\mathbb{Z}_a^\times \times \mathbb{Z}_b^\times| = \phi(a)\phi(b)$.

Now return to the RSA algorithm. Note that $N = (p-1)(q-1) = \phi(p)\phi(q) = \phi(pq) = \phi(n)$. Recall that Alice has Bob's encrypted message $c \in \mathbb{Z}_n^\times$ and she computed $c^d \in \mathbb{Z}_n^\times$. Since $ed \equiv 1 \pmod{N}$ we have $ed - 1 = kN$ for some $k \in \mathbb{Z}$. Then

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{1+kN} \equiv m \cdot (m^N)^k \equiv m \cdot (m^{\phi(n)})^k \equiv m \cdot 1 \equiv m \pmod{n}$$

which is of course Bob's original message $m \in \mathbb{Z}_n^\times$. Note that in this computation we used the fact that $m^{\phi(n)} \equiv 1 \pmod{n}$, which is Euler's Theorem!

Example: Suppose $p = 43$ and $q = 97$. Then $n = pq = (43)(97) = 4171$. Note here $N = \phi(n) = (43-1)(97-1) = 4032$. Alice chooses the encryption number $e = 13$, which is relatively prime to $N = 4032$. Alice releases the encryption key information

$$n = 4171, \qquad e = 13$$

to the public. However, only she knows $p = 43$ and $q = 97$. Alice needs to compute her private decryption key: this is a number d such that $ed \equiv 1 \pmod{4171}$. She computes $d = 1861$ using the Euclidean algorithm.

Next, Bob wants to send the message $m = 3161$ to Alice. He wants to compute $m^e \equiv 3161^{13} \pmod{4171}$. To do this he first writes $13 = 2^3 + 2^2 + 1$ and then

$$c \equiv m^e \equiv 3161^{13} = 3161^{(2^3+2^2+1)} \equiv ((3161^2)^2)^2 \cdot (3161^2)^2 \cdot 3161 \pmod{4171}$$

Then $3161^2 = 9991921 \equiv 2376 \pmod{4171}$. Next, $(3161^2)^2 \equiv 2376^2 = 5645376 \equiv 2013 \pmod{4171}$, and $((3161^2)^2)^2 \equiv 2013^2 = 4052169 \equiv 2128 \pmod{4171}$. He finally computes

$$c \equiv m^e = (2128)(2013)(3161) \equiv 2582 \pmod{4171}$$

Thus $c = 2582$ is the encrypted message, and Bob sends this to Alice. She can then decipher the message by using $d = 1861$ to compute $2582^d \equiv m \pmod{4171}$. To do this she can write 1861 in binary as $1861 = 2^{10} + 2^9 + 2^8 + 2^6 + 2^2 + 1$ and proceed as above:

$$2582^{2^1} \equiv 1466, \quad 2582^{2^2} \equiv 1091, \quad 2582^{2^3} \equiv 1456, \quad 2582^{2^4} \equiv 133, \quad 2582^{2^5} \equiv 1005,$$

$$2582^{2^6} \equiv 643, \quad 2582^{2^7} \equiv 520, \quad 2582^{2^8} \equiv 4356, \quad 2582^{2^9} \equiv 2363, \quad 2582^{2^{10}} \equiv 2971.$$

Thus $c^d \equiv 2582^{1861} \equiv (2971)(2363)(3456)(643)(1091)(2582) \equiv 3161 \pmod{4171}$, which is exactly Bob's original message m.

How can someone intercept the message and decipher it? They would have to know $d \pmod{N}$. You can compute $d \pmod{N}$ from the knowledge of e and N using the Euclidean algorithm, and of course $N = (p-1)(q-1)$. So knowing the prime numbers p and q will allow for anyone to decipher encrypted messages. However, the public only knows e and $n = pq$. And it is very difficult to factor large numbers! So if p and q (and hence n) are very large, it will be very difficult for people to decipher intercepted messages.