LECTURE 5: APPLICATIONS TO CRYPTOGRAPHY AND COMPUTATIONS

Modular arithmetics that we have discussed in the previous lectures is very useful in Cryptography and Computer Science. Here we discuss several of these applications.

1. Diffie–Hellman key exchange

Suppose two people, Alice and Bob, want to use an insecure communication channel to agree on a secret "shared key" that they can use to do further encryption for long messages. The Diffie–Hellman key exchange method, which is used in many of the web browsers, provides a way. It proceeds as follows:

(1) Alice and Bob agree on a big prime number p and a non-zero residue x modulo p. This is public information which is also available to an adversary. A secret key will be among non-zero residues modulo p.

(2) Alice chooses a large secret integer $a < p$, Bob chooses a large secret integer $b < p$. These are their "private keys".

(3) Alice computes her "public key"

$$A \equiv x^a \pmod{p}$$

and sends it to Bob using insecure communication. Likewise, Bob computes his public key

$$B \equiv x^b \pmod{p}$$

and sends it to Alice. An adversary could also get access to A and B.

(4) Finally, Alice computes $B^a \pmod{p}$, and Bob computes $A^b \pmod{p}$. We observe that it follows from basic properties of modular arithmetic that

$$B^a = (x^b)^a \equiv x^{ab} \equiv (x^a)^b \equiv A^b \pmod{p}.$$

This residue is the secret key that Alice and Bob can use for further communications.

This scheme is based on the assumption that given a residues x and y modulo p, it is difficult to find an exponent a such that

$$x^a \equiv y \pmod{p}.$$

This is called the *discrere logarithm problem.* At present time, there is no efficient (polynomial-time) algorithm for solving this problem, but there is an efficient algorithm using quantum computers.

2. Public-Key Cryptography: the RSA cryptosystem

Suppose that Alice wishes to securely send a message to Bob, avoiding Eve malevolently deciphering this message. We suppose that a message constitutes a number a in the range $1 \leqslant a < N$ for some large N (longer messages could be send in pieces). How do we achieve secure communication? We will provide a sketch of the RSA cryptosystem, invented by Rivest, Shamir and Adleman.

(1) Bob picks two large primes p and q in an essentially random manner, with $p \neq q$. He computes $N = pq$. Bob also chooses a natural number r coprime to $\phi(N)$ that is not too small. Notice that since Bob knows the prime factorisation of N, he is able to compute $\phi(N) = (p-1)(q-1)$ quickly, and hence obtain a suitable integer r (for instance, by trial and error) using the Euclidean Algorithm. Bob publishes integers N and r, but keeps the primes p and q secret. We note that since Bob knows $\phi(N)$ he can also find an integer s such that
$$sr \equiv 1 \pmod{\phi(N)}.$$
This integer can be found by solving the equation $xr + y\phi(N) = 1$ with a help of the Euclid algorithm.

(2) Now Alice would like to send to Bob a secret integer a with $1 \leqslant a < N$. She computes
$$b \equiv a^r \pmod N,$$
and send b over an insecure channel.

(3) Finally, Bob computes
$$b^s \pmod N.$$
It follows from the theorem below that this is exactly the secret number a.

Theorem 2.1. *For all integers a, one has $b^s \equiv a \pmod N$.*

Proof. We observe that since $sr \equiv 1 \pmod{\phi(N)}$,
$$sr = 1 + k\phi(N)$$
for some $k \in \mathbb{Z}$.

Suppose first that $(a, N) = 1$. Then it follows from Euler's Theorem that
$$a^{\phi(N)} \equiv 1 \pmod N.$$
Hence, since $sr \equiv 1 \pmod{\phi(N)}$, we obtain that
$$b^s = a^{sr} = a(a^{\phi(N)})^k \equiv a \pmod N$$

Since $N = pq$, it follows that when $(a, N) \neq 1$, then one has $(a, N) = p$, q or pq. In the latter case, we have $a = pq = N$, and then the conclusion is trivial. Suppose then that $(a, N) = p$, so that $p \mid a$ and $(a, q) = 1$. In this situation the former condition yields
$$b^s \equiv a^{sr} \equiv 0 \equiv a \pmod p,$$

and in view of Fermat's Little theorem ($a^{q-1} \equiv 1 \pmod{q}$), the latter yields

$$b^s = a^{sr} = a(a^{q-1})^{k(p-1)} \equiv a \pmod{q}.$$

Thus $b^s \equiv a \pmod{p}$ and $b^s \equiv a \pmod{q}$, whence $b^s \equiv a \pmod{pq}$. The situation in which $(a, N) = q$ may be analysed in a similar manner, and so this completes the proof. □

It remains to discuss the feasibility and security of this cryptosystem. The first observation to make is that all of the operations required to make use of the RSA cryptosystem are fast. The application of the Euclidean Algorithm, and the operation of taking powers modulo N, have running time $O(\log N)$ arithmetic operations. This is proportional to the number of digits in N. Second, we need to have available plenty of large prime numbers (p and q) in order to derive good public keys. Fortunately, there are relatively fast primality tests available. The security of the RSA cryptosystem depends on the difficulty of factoring large integers N and computing $\phi(N)$.

Aside: A probabilistic test is available with running time polynomial in $\log n$ that can discern, provably, that a number n is composite. For the numbers that survive this test, the Adleman-Pomerance-Rumely test can establish primality, or compositeness, provably in deterministic time $O((\log n)^{c \log\log\log n})$, which is close to polynomial in $\log n$. More recently, Agrawal, Kayal and Saxena have devised an algorithm that has running time polynomial in $\log n$.

The naive factorisation algorithm supplies a factorisation of a composite integer in running time $O(\sqrt{n})$ arithmetic operations. The fastest available factorisation algorithm for very large integers is the Number Field Sieve, with running time $\exp(c(\log n)^{1/3}(\log\log n)^{2/3})$ arithmetic operations to factor a large integer n, wherein c is a suitably large positive constant. This is much larger than polynomial in $\log n$. If a quantum computer can be built, then Shor's Quantum Algorithm would factor integers n in a time polynomial in $\log n$, and would constitute a threat to the RSA cryptosystem.

3. Searching for prime numbers

It crucial for many application to have an efficient way for generating large prime numbers and, in particular, testing that a given number is prime. One of the most basic test is based on the Fermat's Little Theorem. Recall that if the number n is prime, then for any $a = 1, \dots, n-1$, we have $a^{n-1} \equiv 1 \pmod{n}$. Hence, if we find $a = 1, \dots, n-1$ such that

$$a^{n-1} \not\equiv 1 \pmod{n}, \tag{3.1}$$

then a is composite. This motivates the following definition.

Definition 3.1. A number $a = 1, \dots, n-1$ is called a *Fermat witness* for n if (3.1) holds.

Example 3.2. Let $n = 2^{2^5} + 1$ Fermat thought that n is prime, but it is not. Although

$$2^{n-1} \equiv 1 \pmod{n},$$

one can check that

$$3^{n-1} \not\equiv 1 \pmod{n}.$$

Let $n = 2^{2^{14}} + 1$. This number has 4933 digits. While

$$2^{n-1} \equiv 1 \pmod{n},$$

one can check that

$$3^{n-1} \not\equiv 1 \pmod{n},$$

so that n is composite. Compositeness of n was first shown in 1961, but a nontrivial factor of n was found in 2010.

Theorem 3.3. *If n is composite, then there exists at least one Fermat witness.*

Proof. Indeed, if a be a proper divisor of n, then a divides both a^{n-1} and n, so that (3.1) is impossible. □

However, finding a Fermat witness could be difficult. We show that in most cases the proportion of Fermat witnesses for composite numbers exceeds 50%.

Theorem 3.4. *Suppose that $b^{n-1} \not\equiv 1 \pmod{n}$ for some b with $(b, n) = 1$. Then*

$$|\{a = 1, \dots, n-1 : \, a^{n-1} \not\equiv 1 \pmod{n}\}| > \frac{n-1}{2}.$$

Proof. We consider the sets

$$\begin{aligned} A &= \{a = 1, \dots, n-1 : \, a^{n-1} \equiv 1 \pmod{n}\}, \\ B &= \{a = 1, \dots, n-1 : \, (a, n) = 1, a^{n-1} \not\equiv 1 \pmod{n}\}, \\ C &= \{a = 1, \dots, n-1 : \, (a, n) > 1\}. \end{aligned}$$

These sets are disjoint, $A \cup B \cup C = \{1, \dots, n-1\}$, and $B \cup C$ are precisely the Fermat witnesses for n.

For $b \in B$ and $a \in A$,

$$(ab)^{n-1} \equiv a^{n-1} b^{n-1} \equiv b^{n-1} \not\equiv 1 \pmod{n},$$

and $(ab, n) = 1$. This shows $Ab \pmod{n} \subset B$. If $ab \equiv a'b \pmod{n}$, then $a = a'$, so that the size of $Ab \pmod{n}$ is equal to the size of A. Hence, we deduce that $|B| \geqslant |A|$. We obtain

$$n - 1 = |A| + |B| + |C| \geqslant |A| + |A| + 1 > 2|A|.$$

This shows that $|A| < (n-1)/2$ and $|B \cup C| > (n-1)/2$, as required.

□

Unfortunately, this theorem does not apply for some composite numbers. We say that n is a *Carmichael number* if n is composite and $a^{n-1} \equiv 1 \pmod{n}$ for all a such that $(a, n) = 1$. The first five Carmichael numbers are 561, 1105, 1729, 2465, 2821. Alford, Granville, and Pomerance proved that there are infinitely many Carmichael numbers.

A refined version of the Fermat test, which nowadays is used in many computational programmes, goes under the name — Miller-Rabin test. It is based on the following observation.

Theorem 3.5. *If $n > 2$ is a prime number and $n - 1 = 2^e k$, then for al $a = 1, \dots, n-1$, either $a^k \equiv 1 \pmod{n}$ or $a^{2^i k} \equiv -1 \pmod{n}$ for some $i = 0, \dots, e-1$.*

Proof. We observe that

$$\begin{aligned} a^{2^e k} - 1 &= (a^{2^{e-1} k})^2 - 1 \\ &= (a^{2^{e-1} k} - 1)(a^{2^{e-1} k} + 1) \\ &= \cdots \\ &= (a^k - 1)(a^k + 1)(a^{2k} + 1) \cdots (a^{2^{e-1} k} + 1). \end{aligned}$$

If n is a prime number, then $a^{n-1} - 1 \equiv 0 \pmod{n}$, and n must divide at least one of the factors in the above product. This implies the theorem. □

This result provides a useful way to test primality.

Definition 3.6. Let $n > 1$ be an odd integer and $n - 1 = 2^e k$ for odd k. A number $a = 1, \dots, n-1$ is called a *Miller-Rabin witness* for n if

$$a^k \not\equiv 1 \pmod{n} \quad \text{and} \quad a^{2^i k} \not\equiv -1 \pmod{n} \quad \text{for all } i = 1, \dots, n-1.$$

It was shown if n is odd and composite, then the proportion of Miller–Rabin witnesses is always at least 75%. Hence, it is easier to find a Miller–Rabin witness than a Fermat witness.

Aside: It was proven assuming The Generalised Riemann Hypothesis (a very difficult conjecture in Number Theory) that if n is odd and composite, then there exists a a Miller–Rabin witness of size at most $2(\log n)^2$. Hence, the Rabin–Miller test is expected to provide a polynomial-time algorithm for testing primality.