Fall 2024

Introduction: Algebra in Computer Science

In this course we will study the basics of abstract algebra, a branch of mathematics that contributed most to cryptology. This knowledge will allow you to get a deeper understanding of the existing methods and possibly design your own versions. We will also encounter and construct proofs of various mathematical statements, developing skills that can help you check and justify your algorithms.

1 Cryptography

Very generally, cryptology is the science of secure communication: developing and using various methods to keep messages secret. Thus, if P is plain text (a message to be transferred), and C the ciphertext (encoded message), then the *encryption function* $E_{k_E}(P) = C$, that depends on the *encryption key* k_E, converts plain text into ciphertext, and the *decryption function* $D_{k_D}(C) = P$, that depends on the *decription key* k_D performs the reverse operation. Thus we have

$$D_{k_D}(E_{k_E}(P)) = P, \qquad E_{k_E}(D_{k_D}(C)) = C.$$

The decription key is a number that needs to be kept secret: we assume the adversary knows the algorithm but not the key.

The relation between the encryption and decryption function is of major interest in cryptography. Indeed, we want the encryption function to be simple enough to encode efficiently large amount of information; at the same time we want the decryption function to be hard enough to guess, so that the third party could not easily decrypt the message.

Mathematics provides a source of this kind of asymmetric operations. For example, it is easy to find a product of two given primes $N = p_1 p_2$, even if these primes are very large. On the other hand, given a very large positive integer N that is a product of exactly two different primes, it is time-consuming to compute its (unique) prime factorization.

Below we recall two well known crypto protocols. The contents of the course provides mathematical foundations for these protocols.

2 Diffie-Hellman public key exchange

Suppose that Alice and Bob want to find (generate) a common secret key, that they can use later to send each other encoded messages.

1. They publicly agree on two positive integers, g and a large n.
2. Alice secretly chooses a large positive integer a, and Bob secretly chooses a large positive integer b.
3. Alice computes $g^a \equiv x \pmod n$, and Bob computes $g^b \equiv y \pmod n$.
4. They publicly exchange the obtained numbers x and y.
5. Alice computes $y^a \equiv (g^b)^a \equiv K \pmod n$ and Bob computes $x^b \equiv (g^a)^b \equiv K \pmod n$. The number K is the common secret key.

The security of the exchange is based on the asymmetry between the two operations: given g, n, a, it is relatively easy to compute x such that $g^a \equiv x \pmod n$, but given g, n, x, it is much harder to find $a \in \mathbb{N}$ such that $g^a \equiv x \pmod n$. The second operation is known as the *discrete logarithm problem*. Modular arithmetic plays central role in this construction. Moreover, a clever choice of the initial data g, n can significantly improve the security.

3 The RSA protocol

The purpose of the protocol is to allow anyone to send secret messages to the initiator A.

1. A chooses secretly two distinct large primes p and q, and computes $m = pq$ and $\phi(m) = (p-1)(q-1)$, where $\phi(m)$ is the Euler totient function of m: the number of positive integers below m that have no common divisors with m. Further, A chooses a number $1 \leq e \leq m-1$ such that e and $\phi(m)$ are relatively prime, and uses the Euclid's algorithm, that we will see in the course, to compute d such that $ed \equiv 1 \pmod{\phi(m)}$. This means that there exists an integer k such that $ed = k\phi(m) + 1$.

2. A publishes the *encryption key* (m, e) and keeps the *decription key* (m, d) secret.

3. Suppose that B wants to send a secret message x to A, where x is an integer such that $0 \leq x \leq m-1$ (this is not an important restriction as m is very large). Then B uses the published key to compute $x^e \equiv y \pmod{m}$ and sends y publicly to A.

4. Now A can decode the message by computing $y^d \equiv z \pmod{m}$. Note that $z \equiv x^{ed} \equiv x^{k\phi(m)+1} \equiv x \pmod{m}$ (the last congruence will follow from the results in the course).

The security of the protocol is based on the computational difficulty of factoring a large number $m = pq$ as a product of two primes. We will see that this last problem is equivalent to finding the Euler's function $\phi(m)$.

The RSA protocol can be generalized to obtain a certified signature, meaning that the receiver of the message will be sure that it was sent by the person claimed to be the sender.

Understanding the properties of Euler's totient function is one of the goals of the course, and we will see how this question involves more generally the study of the theory of *groups* and *rings*.

These are just a few direct applications of algebra in computer science. You will see more mathematics involved in your further CS courses. The purpose of this course is to provide you with a solid foundation to proceed in this direction.

References

[DH] W. Diffie, M.E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976), no.6, 644-654.

[RSA] R.L. Rivest, A. Shamir, L.M. Aldeman, *A method for obtaining digital signatures and public key signatures*, Communications of the ACM **21** (1978), 120-126.