Introduction to mathematical cryptography

Péter Maga

Contents

Preface

These lecture notes are written to provide a text to my Introduction to Mathematical Cryptography course at Budapest Semesters in Mathematics. The main source is [1], even the structure is borrowed from there. Note also that in [1], both the material and the collection of examples are much more extended.

Chapter 1
Introduction

1.1 The principal goal of cryptography, Kerckhoff's principle

The principal goal of cryptography is to **allow two people to exchange confidential information, even if they can only communicate via a channel monitored by an adversary**.

Assume for example that Bob wants to send a message to Alice in such a way that Eve – who reads/listens/spies the communication of Alice and Bob – cannot understand the message (Alice, Bob and Eve are the usual participants of the cryptographic setup).

The scheme of the solution is the following. Bob sends through the communication *something else* than his original message. Eve can read only this *something else*. Alice knows how this *something else* should be understood to get the original message.

Example 1.1.1 (an ancient method)**.** We are in ancient times. Bob shaves the head of a slave. Then tattoos the message on the bald head. After hair has regrown, he sends the slave to Alice. Alice shaves the slave's head again and reads the message. (Here the assumption on the monitoring of the channel is that if Bob sent the slave with a letter to Alice, then Eve would steal the message from the slave and would read it.)

Example 1.1.2 (another ancient method – Caesar cipher)**.** Bob shifts the alphabet the following way: he replaces each letter in his message with the letter which follows three later in the alphabet, e.g.

```
We attack the castle tomorrow.
Zh dwwdfn wkh fdvwoh wrpruurz.
```

which gives rise to the ciphertext (it is usual to rewrite the cipher in five-letter blocks, and leaving punctuation):

```
ZHDWW DFNWK HFDVW OHWRP RUURZ.
```

He writes down only the message 'ZHDWW DFNWK HFDVW OHWRP RUURZ', and sends this to Alice. Now Alice knows that she has to count three characters back and gets the original message, at least, in the form 'WEATT ACKTH ECAST LETOM ORROW', but it is obvious where words start and end. However, even if Eve can read the ciphertext, it is just meaningless for her.

Example 1.1.3 (Caesar's cipher improved – simple substitution ciphers)**.** Set $\mathfrak{A} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \ldots, \mathrm{z}\}$ for the alphabet. Then assume Alice and Bob agrees on a bijection $\pi : \mathfrak{A} \to \mathfrak{A}$ (this is called a permutation of $\mathfrak{A}$). Now if Bob wants to send a message to Alice, simply applies π to each letter, and sends the resulting text to Alice. Again, this text is a gibberish for Eve, while Alice knows how to recover the original message. Caesar cipher in Example 1.1.2 is just a special case of this when π is a shift of the letters.

What happens if Eve knows *what* the basic principle behind the encryption is? In Example 1.1.1, she simply captures and shaves the slave, and the cryptosystem is broken. However, in Example 1.1.2, the information that the encoding procedure is *some* shift of the alphabet still leaves 26 possibilities for the size of the shift (and hence for the original message). Even worse, in Example 1.1.3, the number of possibilities is $26! = 26 \cdot 25 \cdot \ldots \cdot 1 > 10^{26}$.

Motivated by this phenomenon, Kerckhoff's principle says that the security of a cryptosystem should depend only on the secrecy of the *key* (this is the actual shift vector in Example 1.1.2 or the actual permutation in Example

1.1.3), not on the secrecy of the encryption *algorithm* (this is the fact that Alice and Bob use a shift in Example 1.1.2 or a permutation 1.1.3 of the alphabet to encode messages).

Already on this point, it is convenient to agree on the following ability of Eve (which is very natural to assume): she is able to recognize the message when she sees that. For example, in the Caesar cipher (Example 1.1.2), when she tries all the 26 possibilities for the translation vector, she will observe when she hits it. Although in principle it could happen that some other key gives another meaningful message, this is extremely unlikely (and its probability decreases very fast when the length of the message grows).

1.2 Cryptanalysis

In fact, Caesar's cipher can be easily broken: it is just 26 trials for the shift vector, and this can be done even by a human quite fast. Note that Eve has to try the first few characters in each possible shift, for an incorrect guess, most likely the first few letters will give rise to a gibberish.

This is not the case with the simple substitution cipher, the number of possible keys is quite large, $26! > 10^{26}$, which is too much even for a modern PC. However, even such a ciphertext can be revealed relatively easily. The point is that simple substitutions do not alter the characteristic of the underlying natural language, say, English in our case.

So Eve can argue as follows. In a normal English text (which is supposed the message to be), the most frequent characters are 'e', 't', 'a', 'o', 'n'. Even their frequency can be easily computed (you can find online tables telling them, or, by "bare" hand, you can open a long pdf file and count the occurrences of these letters). Also, you can consider the bigrams 'th', 'he', 'an', 're', 'er', and the trigrams 'the', 'and', 'ing'. Now given the ciphertext, if we count the often occurring letters, bigrams and trigrams, we may have reasonable guesses on the substitution: the point is that, for example, no matter which letter stands for 'e', it will stand for 'e' always, and since 'e' is the most often letter in English, this unknown character will be somewhat often in the cipher (and almost surely the most often, if the cipher is long enough). The same is true for other frequent letters, and also for bigrams and trigrams. As soon as some parts of the text are revealed, we can figure out further substitutions. After some trial and error, we can recover the text. Although this might seem complicated, it works surprisingly fast, read [1, Section 1.1.1] to see this is action.

As a historical note, we remark that cryptanalysis, letter frequency counts were known by Arab scholars in the 14th and the 15th century. The same time, in Italian states, more complicated cryptosystems were used than simple substitution ciphers, which suggests that cryptanalysis via frequency analysis was known there as well.

1.3 Symmetric ciphers: mathematical formulation

Definition 1.3.1 (symmetric cryptosystem)**.** By a symmetric cryptosystem, we mean a 5-tuple

$$(\mathscr{K},\mathscr{M},\mathscr{C},e,d),$$

where $\mathscr{K},\mathscr{M},\mathscr{C}$ are sets, and $e:\mathscr{K}\times\mathscr{M}\to\mathscr{C}$, $d:\mathscr{K}\times\mathscr{C}\to\mathscr{M}$ are functions satisfying that for any $k\in\mathscr{K}$ and $m\in\mathscr{M}$,

$$d(k,e(k,m))=m.$$

Here, $\mathscr{K}$ is the set of possible keys, $\mathscr{M}$ is the set of possible messages, $\mathscr{C}$ is the set of possible ciphers. The functions e and d are the encrypting and decrypting algorithms, respectively: for a given key $k\in\mathscr{K}$, e computes $c\in\mathscr{C}$ from $m\in\mathscr{M}$, while d computes $m\in\mathscr{M}$ from $c\in\mathscr{C}$. In practice, for a fixed $k\in\mathscr{K}$, $e(k,\cdot)$ and $d(k,\cdot)$ are often denoted by e_k and d_k, respectively.

In the realization, $\mathscr{K},\mathscr{M},\mathscr{C},e,d$ are known to everyone (including Eve), while the paricular $k\in\mathscr{K}$ in use is known only to Alice and Bob.

For example, in Example 1.1.2, the set of messages and ciphers are both the possible (finite) texts. The set of keys are the positive integers up to 26: $\mathscr{K}=\{1,2,3,\dots,26\}$. Now e_k, for an integer $k\in\mathscr{K}$, shifts each letter by k, while d_k shifts it back.

In Example 1.1.3, the set of messages and ciphers are the same, but this time $\mathscr{K}$ is the set of all permutations of the alphabet, a much bigger set than in Example 1.1.2.

We impose a few informal requirements on cryptosystems:

(1) knowing $k\in\mathscr{K}$ and $m\in\mathscr{M}$, $e_k(m)\in\mathscr{C}$ must be easy to compute;

(2) knowing $k \in \mathscr{K}$ and $c \in \mathscr{C}$, $d_k(c) \in \mathscr{M}$ must be easy to compute;

(3) given one or more encoded ciphertexts $c_1, \dots, c_n \in \mathscr{C}$ encrypted using the key $k \in \mathscr{K}$, without knowing k, it must be hard to compute any of the plaintexts $d_k(c_1), \dots, d_k(c_n)$.

These are just natural conditions: the first two say that if Alice and Bob agree on the key, their alogithms to encyrpt and decrypt texts are fast; the third one says that Eve – not knowing the particular key – cannot compute easily the original message.

A fourth condition is somewhat more complicated:

(4) for any fixed $k \in \mathscr{K}$, given any number of pairs $(m_1, c_1), \dots, (m_n, c_n)$ (i.e. for any $1 \leqslant j \leqslant n$, $m_j \in \mathscr{M}$, $c_j \in \mathscr{C}$, $e_k(m_j) = c_j$, $d_k(c_j) = m_j$), if $c \in \mathscr{C} \setminus \{c_1, \dots, c_n\}$, then without knowing k, it must be hard to compute $d_k(c)$. This is security against a *chosen plaintext attack*.

For example, the Caesar cipher and even its generalized version, the simple substitution cipher (recall Examples 1.1.2, 1.1.3) satisfy the conditions (1), (2), but one can easily see that they do not satisfy (4). Indeed, in the Caesar cipher, even a single given pair (m_1, c_1) determines k, while in the simple substitution cipher, the pairs $(\texttt{'a'}, k(\texttt{'a'})), \dots, (\texttt{'z'}, k(\texttt{'z'}))$ determine k. It is not that obvious what we have seen in Section 1.2: even (3) does not hold for them, meaning that they are not secure in the sense that Eve does not have to organize any ingenious attack on the cryptosystem (like the chosen plaintext attack).

1.4 Asymmetric ciphers: mathematical formulation

In Examples 1.1.2 and 1.1.3, the role of Alice and Bob are symmetric. There is the key set $\mathscr{K}$, and they know which element $k \in \mathscr{K}$ they use (this has to be kept in secret). Such setups are the symmetric ciphers.

However, the relation of Alice and Bob is asymmetric (Bob sends, Alice reads the message), which might be utilized.

Definition 1.4.1 (asymmetric cryptosystem)**.** By an asymmetric cryptosystem, we mean a 7-tuple

$$\mathscr{K}, \mathscr{K}_{\text{pub}}, \mathscr{K}_{\text{priv}}, \mathscr{M}, \mathscr{C}, e, d,$$

where $\mathscr{K}, \mathscr{K}_{\text{pub}}, \mathscr{K}_{\text{priv}}, \mathscr{M}, \mathscr{C}$ are sets satisfying

$$\mathscr{K} \subseteq \mathscr{K}_{\text{pub}} \times \mathscr{K}_{\text{priv}},$$

and $e : \mathscr{K}_{\text{pub}} \times \mathscr{M} \to \mathscr{C}$, $d : \mathscr{K}_{\text{priv}} \times \mathscr{C} \to \mathscr{M}$ are functions satisfying that for any $(k_{\text{pub}}, k_{\text{priv}}) \in \mathscr{K}$ and $m \in \mathscr{M}$,

$$d(k_{\text{priv}}, e(k_{\text{pub}}, m)) = m.$$

Here, $\mathscr{K}$ is the set of possible keys, $\mathscr{M}$ is the set of possible messages, $\mathscr{C}$ is the set of possible ciphers. The functions e and d are the encrypting and decrypting algorithms, respectively: for a given key $(k_{\text{pub}}, k_{\text{priv}}) \in \mathscr{K}$, e computes $c \in \mathscr{C}$ from $m \in \mathscr{M}$, while d computes $m \in \mathscr{M}$ from $c \in \mathscr{C}$. In practice, for a fixed $(k_{\text{pub}}, k_{\text{priv}}) \in \mathscr{K}$, $e(k_{\text{pub}}, \cdot)$ and $d(k_{\text{priv}}, \cdot)$ are often denoted by $e_{k_{\text{pub}}}$ and $d_{k_{\text{priv}}}$, respectively.

In the realization, $\mathscr{K}_{\text{pub}}, \mathscr{K}_{\text{priv}}, \mathscr{M}, \mathscr{C}, e, d$ are known to everyone (including Eve). It is only Alice who knows $\mathscr{K}$, and she picks a particular key $(k_{\text{pub}}, k_{\text{priv}}) \in \mathscr{K}$. Now she makes k_{pub} public, and keeps k_{priv} in secret. If Bob wants to send a message $m \in \mathscr{M}$ to Alice, then he applies $e_{k_{\text{pub}}}$ to m, and sends $e_{k_{\text{pub}}}(m)$ to Alice. Now Alice applies $d_{k_{\text{priv}}}$ to the incoming cipher, obtaining

$$d_{k_{\text{priv}}}(e_{k_{\text{pub}}}(m)) = m.$$

The requirements are modified the obvious way.

In this setup, there is nothing special about Bob, he has the same information as Eve has. Yet, if the system works well, Eve is unable to read Bob's messages to Alice. Although this seems more complicated and somewhat artificial, this scheme is commonly used. Think for example of your e-mail address: basically everyone knows it, none of your friends has a particular role, yet each of them can send you an e-mail which is hidden from anyone else[1].

[1] At least would be, in a world where webmail providers do not read clients' mails – in ours they do.

1.5 Mathematical background

Up to this point, messages and ciphers were both textual objects. This is natural, our communication is textual, so most messages are simply texts. However, the communication of our computers (which deliver the messages nowadays) are rather based on numbers, and – more importantly – fancy properties of numbers provide us with extremely ingenious cryptosystems. For this reason, given any text, we – more precisely: our softwares – transform it to numbers. Our computers use the ASCII encoding scheme to convert characters to bytes. A byte consists of eight bits, where a bit is a 0 or a 1 (it is the abbreviation of the binary digit). For example, 01000001 is a byte, and in ASCII, it stands for the character '`A`'. This encoding scheme is completely public, its purpose is not to hide information but on the contrary: it guarantees that if we send an e-mail to somebody, our computer transforms it to a number which can be transformed back to the same e-mail by his or her computer.

Of course, when Bob's e-mail is encoded it in ASCII, and is sent to Alice, when Eve captures it, she is also able to read it. For this reason, Bob, after making the ASCII code, makes the message to a cipher (both are numbers this time). Ideally, when Eve reads the cipher, and decodes its ASCII, she will see a gibberish. Alice, who knows what to do, transforms the cipher back to the ASCII encoded version of the original message, then decodes it.

Since from now on, messages and ciphers are all numbers (in fact, positive integers) and encryption procedures will rely on deep interrelations between them, we review now the necessary mathematical background (not only about positive integers, but also about certain abstract algebraic structures) for convenience.

1.5.1 The fundamental theorem of arithmetic

In this section, we state and prove the fundamental theorem of arithmetic: the fact that any nonzero integer can be written – essentially uniquely – as the product of prime (irreducible) numbers. The heart of the matter is in fact that primes and irreducibles coincide among rational integers.

Proposition 1.5.1 (euclidean division)**.** *Given integers a, b, $b \neq 0$. Then there exist integers c, d satisfying $a = bc + d$ and $|d| < |b|$.*

Proof. Let $a \geqslant 0$, $b > 0$, the remaining cases are similar. Induct on a. For $a = 1$, the statement is trivial ($c = 1, d = 0$ if $b = 1$ and $c = 0, d = 1$ if $b > 1$). Now assume that the statement holds for any $0 \leqslant a' < a$. If $a < b$, then $c = 0, d = a$. If $a \geqslant b$, then $a - b = bc' + d'$ with $|d'| < |b|$ by induction, so $a = b(c' + 1) + d'$. □

Proposition 1.5.2. *Assume $a, b \in \mathbf{Z}$. Then there exists an integer $\gcd(a,b)$ satisfying $\gcd(a,b) \mid a, b$ and also that whenever $d \mid a, b$, $d \mid \gcd(a,b)$.*

Proof. If $b = 0$, then $\gcd(a,b) = a$ does the job. Otherwise, consider the sequence $(a, b, d_1, \ldots, d_n, 0)$, where each d_i is defined via the euclidean division $d_{i-2} = c_{i-1}d_{i-1} + d_i$ (with $d_{-1} = a, d_0 = b, d_{n+1} = 0$). It is clear that such a sequence of euclidean divisions terminates, since the absolute value decreases in each step. Set $\gcd(a,b) = d_n$. It is clear that $d_n \mid d_{n+1}, d_n$, and then by induction, $d_n \mid d_i, d_{i-1}$ implies $d_n \mid d_{i-2}$. Also, if $d \mid d_{i-2}, d_{i-1}$ (which holds for $i = 1$), then $d \mid d_i$, yielding $d \mid d_n = \gcd(a,b)$. □

Observe that $\gcd(a,b)$ is well-defined only up to sign.

Definition 1.5.3 (greatest common divisor)**.** The greatest common divisor of $a, b \in \mathbf{Z}$ is the nonnegative number which satisfies the conditions imposed on $\gcd(a,b)$ in Proposition 1.5.2.

Definition 1.5.4 (euclidean algorithm)**.** The sequence of euclidean divisions in the proof of Proposition 1.5.2 is called the euclidean algorithm.

Proposition 1.5.5. *Assume $a, b \in \mathbf{Z}$. Then $\gcd(a,b) = au + bv$ for some $u, v \in \mathbf{Z}$.*

Proof. If $b = 0$, the statement is trivial. Otherwise, we can create the same sequence $(a, b, d_1, \ldots, d_n, 0)$ as in the proof of Proposition 1.5.2. Clearly $d_{-1} = a, d_0 = b$ are integer combinations of a and b. Also, if d_{i-2}, d_{i-1} are integer combinations, then so is d_i. □

Definition 1.5.6 (prime numbers)**.** A nonzero integer p is said to be prime, if $p \nmid 1$, and whenever $p \mid ab$, $p \mid a$ or $p \mid b$.

Definition 1.5.7 (irreducible numbers)**.** A nonzero integer p is said to be irreducible, if $p \nmid 1$, and whenever $p = ab$, $a \mid 1$ or $b \mid 1$.

Proposition 1.5.8. *An integer p is prime if and only if it is irreducible.*

Proof. Assume p is prime, and let $p = ab$. Then $a, b \neq 0$. If $a \nmid 1$ and $b \nmid 1$, then $1 < |a|, |b| < p$. Therefore $p \nmid a, b$, which is a contradiction.

Assume p is irreducible, and $p \mid ab$. If $p \mid a$, we are done. If $p \nmid a$, then $\gcd(a, p) = 1$, since p is irreducible. Then there exist integers u, v satisfying $au + pv = 1$. Multiplying by b, we obtain $abu + pbv = b$, the left-hand side is divisible by p, so is the right-hand side. □

Theorem 1.5.9 (fundamental theorem of arithmetic)**.** *Every nonzero integer can be written as a product of prime (irreducible) numbers. The decomposition is unique, apart from factors dividing* 1.

Proof. First we prove the existence by induction on $|n|$. For $|n| = 1$, it is trivial. Assume that the statement holds for any n' with $|n'| < |n|$. If n is irreducible, we are done. If not, we can write it as a product $n = ab$ with $|a|, |b| < |n|$. We are done by induction.

Now we prove the uniqueness. Assume n has two decompositions $p_1 \cdot \ldots \cdot p_k = q_1 \cdot \ldots \cdot q_l$. Here, p_1 divides the left-hand side, so it divides the right-hand side as well. Then, since it is a prime, it divides a factor of the right-hand side, say, q_1. Then $p_1 \mid q_1$, and also $q_1 \mid p_1$, since q_1 is irreducible. Dividing by them, we can complete the proof by induction. □

1.5.2 Groups, rings and fields

Definition 1.5.10 (group)**.** Assume G is a set, and there is a binary operation on it, e.g. a function $\cdot$ from $G \times G$ to G, which satisfies the following conditions (for a better notation, we write $x \cdot y$ in place of $\cdot(x, y)$):

- for any $x, y, z \in G$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ (associativity);
- there exists $e \in G$ satisfying that for any $x \in G$, $x \cdot e = e \cdot x = x$ (unit element);
- for any $x \in G$, there exists $y \in G$ satisfying $x \cdot y = y \cdot x = e$ (inverse).

Then we say that $(G, \cdot)$ is a group (or G is a group, if the operation is clear).

Notational convention: the inverse of an element x is often denoted by x^{-1}.
In most applications below, our groups will be abelian.

Definition 1.5.11 (abelian (commutative) group)**.** Assume $(G, \cdot)$ is a group such that for any $x, y \in G$,

$$x \cdot y = y \cdot x.$$

Then we say that G is an abelian (commutative) group.

Example 1.5.12. The structures $(\mathbf{Z}, +)$, $(\mathbf{Q}, +)$, $(\mathbf{R}, +)$, $(\mathbf{C}, +)$, $(\mathbf{Q}^\times, \cdot)$, $(\mathbf{R}^\times, \cdot)$, $(\mathbf{C}^\times, \cdot)$, $(\mathbf{Q}_+, \cdot)$, $(\mathbf{R}_+, \cdot)$ are all abelian groups.

Example 1.5.13. Given any set S, denote by $\mathrm{Perm}(S)$ the set of its permutations. Two permutations $\pi, \sigma \in \mathrm{Perm}(S)$ can be concatanated by the composition

$$\pi \circ \sigma \in \mathrm{Perm}(S): \qquad \forall x \in S: (\pi \circ \sigma)(x) = \pi(\sigma(x)).$$

Then $(\mathrm{Perm}(S), \circ)$ is a group: composing functions is associative, the identical permutation is the unit element, and the inverse permutation of each permutation is its inverse. Note that this group is not abelian if $\#S \geqslant 3$.

Proposition 1.5.14. *Assume G is a group. Then its unit element is unique. Also, the inverse of each element of G is unique.*

Proof. Assume e, f are unit elements. Then

$$\begin{aligned} e &= ef \text{ (since } f \text{ is a unit element)} \\ &= f \text{ (since } e \text{ is a unit element),} \end{aligned}$$

and the proof of the uniqueness of the unit element is complete.

Similarly, let $x \in G$, and assume y, z are both inverses of x. Then (denoting by e the unit element)

$$\begin{aligned} y &= ye \text{ (since } e \text{ is the unit element)} \\ &= y(xz) \text{ (since } z \text{ is the inverse of } x) \\ &= (yx)z \text{ (by associativity)} \\ &= ez \text{ (since } y \text{ is the inverse of } x) \\ &= z \text{ (since } e \text{ is the unit element),} \end{aligned}$$

and the proof of the uniqueness of the inverse is complete. □

Proposition 1.5.15. *Assume G is a group, and $a, b \in G$ are fixed elements. Then the equation $ax = b$ has a unique solution. The same holds for the equation $ya = b$.*

Proof. In the first equation, observe that $x = a^{-1}b$ is a solution: indeed, $a(a^{-1}b) = (aa^{-1})b = b$, where we used associativity and that aa^{-1} is the unit element. Also, if x is any solution, then multiplying both sides of the equation by a^{-1} on the left, we obtain $x = a^{-1}b$.

The proof is similar for the equation $ya = b$: $y = ba^{-1}$ is a solution, and it must be the only one, which can be seen by multiplying the both sides on the right by a^{-1}. □

Proposition 1.5.16. *Assume G is a finite group, and $x \in G$. Then for any $n \in \mathbf{N}$, $(x^n)^{-1} = (x^{-1})^n$.*

Proof. Assume the unit element is e. By associativity,

$$x^n(x^{-1})^n = \underbrace{x \cdot \ldots \cdot x}_{n \text{ times}} \cdot \underbrace{x^{-1} \cdot \ldots \cdot x^{-1}}_{n \text{ times}} = e,$$

and the proof is complete. □

Therefore, the conventions $x^0 = e$ (the unit element) and $x^{-n} = (x^n)^{-1}$ for $n \in \mathbf{N}$ are completely consistent, so the notation x^n makes sense for $n \in \mathbf{Z}$.

Proposition 1.5.17. *Assume G is a finite group with unit element e. Then for any element $x \in G$, there is a positive integer $n \in \mathbf{N}$ such that*

$$x^n = \underbrace{x \cdot \ldots \cdot x}_{n \text{ times}} = e.$$

Proof. Consider the elements $x = x^1, x^2, x^3, \ldots \in G$. Since G is finite, there exist positive integers $k < l$ such that $x^k = x^l$. Multiplying both sides by $(x^k)^{-1} = x^{-k}$, we obtain $e = x^{l-k}$, and here, $l - k \in \mathbf{N}$. □

Definition 1.5.18 (order of element)**.** Assume G is a finite group with unit element e. For any element $x \in G$, we define its order ($\mathbf{o}(x)$ in notation) the smallest positive integer n satisfying $x^n = e$.

Proposition 1.5.19. *Assume G is a finite group with unit element e, and $x \in G$. Then for any $k \in \mathbf{N}$, $x^k = e$ if and only if k is a multiple of $\mathbf{o}(x)$.*

Proof. First assume $\mathbf{o}(x) \mid k$, i.e. $k = l\mathbf{o}(x)$, where $l \in \mathbf{N}$. Then

$$x^k = (x^{\mathbf{o}(x)})^l = e^l = e.$$

Conversely, assume $\mathbf{o}(x) \nmid k$. Then by Proposition 1.5.1, for some $c \in \mathbf{N} \cup \{0\}$ and $0 < d < \mathbf{o}(x)$, $k = c\mathbf{o}(x) + d$. Then

$$x^d = x^{k-\mathbf{o}(x)} = x^k x^{-\mathbf{o}(x)} = ee^{-1} = e,$$

which contradicts the minimal choice of $\mathbf{o}(x)$. □

Definition 1.5.20 (subgroup)**.** Assume $(G, \cdot)$ is a group. If $H \subseteq G$, and $(H, \cdot)$ is a group, then we say that H is a subgroup of G. In notation: $H \leqslant G$.

Example 1.5.21. We have $(\mathbf{Z}, +) \leqslant (\mathbf{Q}, +) \leqslant (\mathbf{R}, +) \leqslant (\mathbf{C}, +)$. Also $(\mathbf{Q}_+, \cdot) \leqslant (\mathbf{Q}^\times, \cdot)$.

Proposition 1.5.22. *Assume G is a group, and H_i $(i \in I)$ are subgroups of G. Then $H = \cap_{i \in I} H_i$ is also a subgroup of G.*

Proof. Associativity just follows from the associativity of G. We have to check that H is closed under multiplication and taking inverses. First, if $x, y \in H$, then for each $i \in I$, $x, y \in H_i$, and then $xy \in H_i$, since H_i is a subgroup. Therefore, $xy \in H_i$ for each $i \in I$, implying $xy \in H$. The proof for inverses goes similarly: if $x \in H$, then $x \in H_i$ for each $i \in I$, and since H_i is a subgroup, $x^{-1} \in H_i$, giving $x \in H$. □

Definition 1.5.23 (generated subgroup)**.** Assume G is a group, and $A \subseteq G$. Then the subgroup generated by A is

$$\langle A \rangle = \bigcap_{A \subseteq H \leqslant G} H.$$

The intersection is nonempty, since $A \subseteq G$, and it is a subgroup, since it is an intersection of subgroups.

Proposition 1.5.24. *For a group G and $A \subseteq G$,*

$$\langle A \rangle = \{a_1 \cdot \ldots \cdot a_n : \textit{ for each } 1 \leqslant i \leqslant n,\ a_i \in A \textit{ or } a_i^{-1} \in A\}.$$

Proof. The right-hand side is clearly a subgroup: it is closed under multiplication and taking inverses, also it is containing A, so it is taken into account in the definition of $\langle A \rangle$. Also, any $H \leqslant G$ containing A contains the elements $a_1, \ldots, a_n$ and hence all the products on the right-hand side. □

Corollary 1.5.25. *For a group G and $x \in G$, $\langle x \rangle = \langle \{x\} \rangle = \{x^n : n \in \mathbf{Z}\}$.* □

Theorem 1.5.26 (Lagrange)**.** *Assume G is a finite group and H is a subgroup of G. Then $\#H \mid \#G$.*

Proof. Introduce the following relation on the pair of elements of G: $x \sim y$, if for some $h \in H$, $xh = y$. This is an equivalence relation: $1 \in H$ implies $x \sim x$; if $xh \sim y$ for some $h \in H$, then $yh^{-1} = x$ and $h^{-1} \in H$; if $x \sim y \sim z$, then for some $h_1, h_2 \in H$, $xh_1 = y$, $yh_2 = z$, then $x(h_1h_2) = z$ and $h_1h_2 \in H$, yielding $x \sim z$. Then G is partitioned into equivalence classes, and we claim that each equivalence class has the same number of elements as H (this clearly implies the statement). Take any equivalence class C, let x be a representative of it. Then take the function $f(h) = xh$ for $h \in H$, obviously $f(H) \subseteq C$. Also, for any $y \in C$, there is h satisfying $xh = y$, then $f(h) = y$, thus $f(H) = C$. Then f surjects H onto C, it suffices to see that it also injects H into C. Assume that for h, h', $f(h) = f(h')$. Then $xh = xh'$, so multiplying by x^{-1} on the left, $h = h'$. □

Corollary 1.5.27. *For a finite group G and $x \in G$, $\mathbf{o}(x) \mid \#G$.*

Proof. The elements of $\langle x \rangle$ are exactly the elements $x = x^1, x^2, x^3, \ldots, x^{\mathbf{o}(x)}$ (and these elements are distinct: if any two of them coincide, then their quotient is the unit element e, contradicting the minimal choice of the order of x). □

Definition 1.5.28 (ring)**.** Assume R is a set with two binary operations $+, \cdot$ such that

- $(R, +)$ is an abelian group;
- for any $x, y, z \in R$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ (associativity of $\cdot$);
- for any $x, y, z \in R$, $x \cdot (y + z) = x \cdot y + x \cdot z$ and $(x + y) \cdot z = x \cdot z + y \cdot z$ (distributivity).

Then we say that $(R, +, \cdot)$ is a ring.

Terminological and notational conventions: the unit element of $(R, +)$ is often called the zero element of the ring, and they will often be denoted by 0.

Our rings $(R, +\cdot)$ will exclusively be commutative, and will have unit elements for the operation $\cdot$ (not only for $+$).

Definition 1.5.29 (commutative ring with unit element)**.** We say that $(R, +, \cdot)$ is a commutative ring with unit element, if

- for any $x, y \in R$, $x \cdot y = y \cdot x$ (commutativity);
- there exists $1 \in R$ such that $1 \neq 0$, and for any $x \in R$, $x \cdot 1 = 1 \cdot x = x$ (unit element for $\cdot$).

Definition 1.5.30. Assume $(\mathbf{F}, +, \cdot)$ is a commutative ring with unit element 1, where $(\mathbf{F} \setminus \{0\}, \cdot)$ is an abelian group. Then we say that $(\mathbf{F}, +, \cdot)$ is a field.

Terminological and notational conventions: $(\mathbf{F}, +)$ is called the additive group, $(\mathbf{F}^\times, \cdot)$ is called the multiplicative group, where $\mathbf{F}^\times$ stands for $\mathbf{F} \setminus \{0\}$.

Example 1.5.31. The following structures are all commutative rings with unit element: $(\mathbf{Z}, +, \cdot)$, $(\mathbf{Q}, +, \cdot)$, $(\mathbf{R}, +, \cdot)$, $(\mathbf{C}, +, \cdot)$. The following structures are all fields: $(\mathbf{Q}, +, \cdot)$, $(\mathbf{R}, +, \cdot)$, $(\mathbf{C}, +, \cdot)$. Given any field $\mathbf{F}$ and $n \in \mathbf{N}$, the $n \times n$ matrices over $\mathbf{F}$ form the ring $\mathrm{Mat}(n, \mathbf{F})$, which has a unit element but is not commutative, if $n \geqslant 2$.

1.5.3 Residue classes

Definition 1.5.32 (congruence)**.** Given $m \in \mathbf{Z}$, we say that $a \equiv b \bmod m$ (in words: a is congruent to b modulo m) if $m \mid (a-b)$.

Proposition 1.5.33 (remainders)**.** *Given $m \in \mathbf{Z} \setminus \{0\}$ and $a \in \mathbf{Z}$, there exist $0 \leqslant b < |m|$ and $-|m|/2 < c \leqslant |m|/2$ satisfying $a \equiv b \equiv c \bmod m$.* □

Proposition 1.5.34. *Given $m \in \mathbf{Z}$. Being congruent modulo m is an equivalence relation, by which we mean that $a \equiv a \bmod m$ (reflexivity), $a \equiv b \bmod m$ implies $b \equiv a \bmod m$ (symmetry), $a \equiv b \bmod m$ and $b \equiv c \bmod m$ imply $a \equiv c \bmod m$ (transitivity).* □

Definition 1.5.35 (residue classes)**.** Let $m \in \mathbf{Z}$. Then the equivalence classes defined via modulo m congruency are said to be modulo m residue classes.

Proposition 1.5.36. *Let $m \in \mathbf{Z} \setminus \{0\}$. The number of residue classes modulo m is $|m|$.* □

Proposition 1.5.37. *Let $m \in \mathbf{Z} \setminus \{0\}$. The modulo m residue classes form a commutative ring, where addition, multiplication and taking additive inverse are the usual addition, multiplication and taking additive inverse, all reduced modulo m. The residue class of 1 is the multiplicative unit.*

Proof. Let a, b be arbitrary representatives of two residue classes. Then for any $k, l \in \mathbf{Z}$, we have

$$\begin{aligned}(a+km)+(b+lm) &= (a+b)+(k+l)m \equiv a+b \bmod m,\\ (a+km)\cdot(b+lm) &= ab+(kb+la)m+klm^2 \equiv ab \bmod m,\\ -(a+km) &= -a-km \equiv -a \bmod m,\end{aligned}$$

showing that the operations can be performed via any representatives. Then obviously the residue class of 1 is a multiplicative unit. □

Theorem 1.5.38 (Chinese remainder theorem)**.** *Assume $m, n \in \mathbf{N}$ satisfy $\gcd(m,n) = 1$. Then for any $a, b \in \mathbf{Z}$, there exists a unique residue class c modulo mn satisfying $c \equiv a \bmod m$, $c \equiv b \bmod n$.*

Proof. Define the function $f : x \bmod mn \mapsto (x \bmod m, x \bmod n)$. There are mn residue classes modulo mn, and the number of possible values of this function is also mn. It suffices to prove that f is a bijection, which holds if and only if it is a surjection.

To see this, take $u, v \in \mathbf{Z}$ satisfying $mu + nv = 1$. Then take the number $c = mub + nva$. Then

$$c \equiv nva \equiv mua + nva \equiv a \bmod m, \qquad c \equiv mub \equiv mub + nvb \equiv b \bmod n.$$

The proof is complete. □

Corollary 1.5.39. *Assume $m_1, \ldots, m_n$ satisfy $\gcd(m_i, m_j) = 1$ for all $1 \leqslant i < j \leqslant n$. Then for any $a_1, \ldots, a_n \in \mathbf{Z}$, there exists a unique residue class c modulo $m_1 \cdot \ldots \cdot m_n$ satisfying $c \equiv a_i \bmod m_i$ for each $1 \leqslant i \leqslant n$.* □

1.5.4 Multiplicative groups

Given $m \in \mathbf{N}$, we denote by $\mathbf{Z}_m$ the ring of residue classes modulo m. Assume $\gcd(a, m) = 1$.

Proposition 1.5.40. *The function $x \mapsto xa : \mathbf{Z}_m \to \mathbf{Z}_m$ is a bijection.*

Proof. Since $\mathbf{Z}_m$ is finite, it suffices to show that the function is surjective. Since $\gcd(a, m) = 1$, for some $u, v \in \mathbf{Z}$, $au + mv = 1$. Then if the residue class c modulo m is given, $x \equiv uc \bmod m$ is mapped to c via $x \mapsto xa \bmod m$, since $c = auc + mvc \equiv auc \bmod m$. □

Corollary 1.5.41. *The coprime residue classes form a group. (This is the unit group of $\mathbf{Z}_m$ and will be denoted by $\mathbf{Z}_m^\times$ from now on.)* □

Corollary 1.5.42. *If p is a prime, the residue classes modulo p form a field (denoted by $\mathbf{F}_p$ from now on).* □

The following function is of extreme importance in number theory.

Definition 1.5.43 (Euler's number of coprime residue classes function)**.** Define

$$\varphi(n) = \sum_{\substack{d \leqslant n \\ \gcd(d,n)=1}} 1.$$

With this notation, we see that the group $\mathbf{Z}_m^\times$ has $\varphi(m)$ elements.

Proposition 1.5.44. *Assume $n = p_1^{\alpha_1} \cdot \ldots \cdot p_r^{\alpha_r}$. Then*

$$\varphi(n) = n \prod_{j=1}^{r} \left(1 - \frac{1}{p_j}\right).$$

Proof. Assume n has prime factors $p_1, \ldots, p_r$. From the set $\{1, \ldots, n\}$, sift out the numbers that are divisble by some of $p_1, \ldots, p_r$. That is, by the inclusion-exclusion principle,

$$\varphi(n) = n + \sum_{j=1}^{r} (-1)^j \sum_{1 \leqslant i_1 < \ldots < i_j \leqslant r} \frac{n}{p_{i_1} \cdot \ldots \cdot p_{i_j}} = n \prod_{j=1}^{r} \left(1 - \frac{1}{p_j}\right).$$

The proof is complete. □

Now Corollary 1.5.27 has the following consequences.

Corollary 1.5.45. *Assume $m \in \mathbf{N}$ and a is coprime to m. Then the order of a (in the multiplicative group modulo m) divides $\varphi(m)$.* □

Corollary 1.5.46 (Euler-Fermat)**.** *Assume $m \in \mathbf{N}$. Then*

$$a^{\varphi(m)} \equiv 1 \bmod m$$

for any a coprime to m. □

Corollary 1.5.47 (Fermat)**.** *Assume p is a prime. Then*

$$a^p \equiv a \bmod p$$

for any $a \in \mathbf{Z}$. □

Finally, we describe the group structure of $\mathbf{Z}_m^\times$. By the Chinese remainder theorem (Corollary 1.5.39),

$$\mathbf{Z}_m^\times \cong \mathbf{Z}_{p_1^{\alpha_1}}^\times \times \ldots \times \mathbf{Z}_{p_r^{\alpha_r}}^\times,$$

if $m = p_1^{\alpha_1} \cdot \ldots \cdot p_r^{\alpha_r}$ is the canonical form. Therefore, it suffices to describe $\mathbf{Z}_{p^\alpha}^\times$ for primes p and positive integers α. This group has $p^\alpha - p^{\alpha-1} = p^{\alpha-1}(p-1)$ elements.

Proposition 1.5.48. *If p is an odd prime, then $\mathbf{Z}_{p^\alpha}^\times$ is cyclic for any $\alpha \in \mathbf{N}$. For $p = 2$, $\mathbf{Z}_{2^\alpha}^\times$ is cyclic for $\alpha = 1, 2$, and is not cyclic but generated by the elements -1 (of order 2) and 5 (of order $2^{\alpha-2}$) for $\alpha \geqslant 3$.*

Proof omitted.

1.5.5 Computability

In this section, we start to investigate the length of computations. Although a precise definition could be as well given using the notion of Turing machine, for our purpose an informal description will be completely sufficient.

Assume given an algorithm solving a problem. For example, let the problem be the addition of positive integers (given in base 10), and the algorithm is what we learned in the elementary school. Basically, the algorithm consists of two things: one is the addition table

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

and the other one is how to use this: the manipulation with the digits learned in the school. During this manipulation, we perform the addition of digits using the table (call them basic additions). If there are two numbers of n digits, then we have to use n basic additions (one at each digit). Also, there might be 1's taken on, at most n times, so the number of basic additions is at most $2n$. This means that given the basic addition table and the school algorithm, they can add up two numbers of n digits in $2n$ many steps, where by one step, we mean one reference to the basic addition table.

This is the general scheme: we can call *anything* a step, but that must be fixed once for all; also, we give an algorithm which can use the step as many times as it is needed. Then the input comes, and the algorithm computes the output. The running time (or time cost) of the algorithm (on the given input) is the number of steps used to compute the output.

Assume that the input is the positive integer n. We say that an algorithm computes the output in polynomial time, if the running time can be estimated from above by a polynomial of $\log n$. Why $\log n$? Simply because the length of the input is just $\log_{10} n$ (and in other bases, just a constant times $\log_{10} n$), and not n. Similarly, the algorithm is exponential, if it is exponential in $\log n$. If the input consists of more than one numbers, say, $n_1,\dots,n_k$, then a polynomial algorithm is an algorithm whose running time can be estimated by a polynomial of $\max(\log n_1,\dots,\log n_k)$.

From the school, we know that addition, subtraction, multiplication and euclidean division can be performed by polynomial algorithms.

Proposition 1.5.49. *With input a,b, the number $\gcd(a,b)$ can be computed in polynomial time.*

Proof. Recall the notation of the proof of Proposition 1.5.2: assume we have two numbers, $a \geqslant b > 0$, and then the sequence $(a,b,d_1,\dots,d_n,0)$, where each d_i is defined via the euclidean division $d_{i-2} = c_{i-1}d_{i-1} + d_i$ (with $d_{-1} = a, d_0 = b, d_{n+1} = 0$) reproduces $d_n = \gcd(a,b)$. If we take the remainder d_i in each step such that $|d_i| \leqslant |d_{i-1}|/2$ (which can be done), then it is clear that the number of d_i's is at most $1+\log_2 a$. Since each euclidean divisions can be performed in polynomial time, the time cost altogether is

$$\mathrm{Pol}(\log_2 a) + \mathrm{Pol}(\log_2 b) + \mathrm{Pol}(\log_2 d_1) + \ldots + \mathrm{Pol}(\log_2 d_{n-1}) \leqslant (1+\log_2 a)\,\mathrm{Pol}(\log_2 a) = \mathrm{Pol}(\log_2 a).$$

The proof is complete. □

In this proof, we spelled out that there are $\mathrm{Pol}(\log)$ many euclidean divisions, and each of them can be computed in $\mathrm{Pol}(\log)$ many steps, hence the algorithm runs in $\mathrm{Pol}(\log)$ time. From now on, for brevity, when we have proved that *something* can be computed in polynomial time, we will use that *something* as a *step* in more complex algorithms: since the product of polynomials is a polynomial again, this simplification does not affect whether an algorithm is polynomial or not. For example, computing the gcd can be handled as a single step in a complex algorithm. Note that this simplification refers only to the polynomial being of the running time: the degree of the polynomial of course varies.

Proposition 1.5.50. *With input $a,b,N \in \mathbf{N}$, the residue class $a^b \bmod N$ can be computed in polynomial time.*

Proof. First of all, write b as

$$b = \sum_{0 \leqslant j \leqslant 1+\log_2 b} \varepsilon_j 2^j,$$

where each ε_j is 0 or 1. (This can be done in polynomial time using only euclidean divisions: divide b by $2^0, 2^1, 2^2, 2^3, \dots, 2^{1+\log_2 b}$. Alternatively, you may think of the input as numbers already given in base 2.) Now

$$a^b = a^{\sum_{0 \leqslant j \leqslant 1+\log_2 b} \varepsilon_j 2^j} = \prod_{0 \leqslant j \leqslant 1+\log_2 b} a^{\varepsilon_j 2^j}.$$

Here, each factor with $\varepsilon_j = 0$ is just 1. When $\varepsilon_j = 1$, then

$$a^{\varepsilon_j 2^j} = a^{2^j} = \underbrace{(((a^2)^2)^{\cdots})^2}_{j \text{ many } 2\text{'s}}.$$

Now raise a to square (this is polynomial, since it is just a multiplication), then reduce it modulo N (a euclidean division). Then raise the result to square again, and reduce it modulo N. Doing this j many times, we get $a^{\varepsilon_j 2^j} \bmod N$ in polynomial time. Now we have to multiply together the factors $a^{\varepsilon_j 2^j}$ for all the j's, this is polynomial many multiplications and modulo N reductions again. □

1.5.6 Probability theory

Definition 1.5.51 (probability space)**.** By a (finite, discrete) probability space, we mean a pair $(\Omega, \Pr)$, where Ω is a finite set, and Pr is a function from 2^Ω to $\mathbf{R}_{\geqslant 0}$ satisfying

(1) $\Pr(\Omega) = 1$, $\Pr(\emptyset) = 0$;

(2) if $A, B \subseteq \Omega$ are disjoint (i.e. $A \cap B = \emptyset$), then $\Pr(A \cup B) = \Pr(A) + \Pr(B)$.

Example 1.5.52 (uniform distribution)**.** On any finite set Ω, we can define the uniform distribution by assigning the same probability $(\#\Omega)^{-1}$ to each one-element set, i.e. for any $A \subseteq \Omega$,

$$\Pr(A) = \frac{\#A}{\#\Omega}.$$

The subsets of Ω are the *events*, and the corresponding Pr values are their *probabilities*. Intuitionally we really think of some randomness and the probability of a given event is the chance that the given event occurs.

For any $A \subseteq \Omega$, we define A^c to be the complement of A, obviously

$$\Pr(A^c) = 1 - \Pr(A).$$

Definition 1.5.53 (conditional probability)**.** Given a probability space Ω, for any $A, B \subseteq \Omega$, if $\Pr(B) > 0$, we define the conditional probability

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)}.$$

The meaning of this notion is the following: the probability of A given that B occurs.

Proposition 1.5.54. *Assume $A, B \subseteq \Omega$ satisfy $\Pr(B), \Pr(B^c) > 0$. Then*

$$\Pr(A) = \Pr(A \mid B)\Pr(B) + \Pr(A \mid B^c)\Pr(B^c).$$

Proof. The statement follows by a simple calculation from the definitions. Indeed,

$$\Pr(A) = \Pr((A \cap B) \cup (A \cap B^c)) = \Pr(A \cap B) + \Pr(A \cap B^c) = \Pr(A \mid B)\Pr(B) + \Pr(A \mid B^c)\Pr(B^c),$$

and the proof is complete. □

Proposition 1.5.55 (Bayes's theorem)**.** *Assume $A, B \subseteq \Omega$ satisfy $\Pr(A), \Pr(B) > 0$. Then*

$$\Pr(A \mid B) = \frac{\Pr(B \mid A)\Pr(A)}{\Pr(B)}.$$

Proof. Again, by a simple calculation,

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)} = \frac{\Pr(B \cap A)}{\Pr(B)} = \frac{\Pr(B \mid A)\Pr(A)}{\Pr(B)},$$

and the proof is complete. □

Definition 1.5.56. We say that the events $A_1, \ldots, A_n$ are independent, if for any $S \subseteq \{1, \ldots, n\}$,

$$\Pr\left(\bigcap_{i \in S} A_i\right) = \prod_{i \in S} \Pr(A_i),$$

(we can freely exclude $S = \emptyset$; or use the conventions: the empty product is 1, and the empty intersection is Ω).

1.6 The XOR cipher and pseudorandom sequences

Assume, the message is a number $0 \leqslant m \leqslant 2^t - 1$, i.e. a binary number on t bits. Now Alice and Bob agree on a binary number k also on t bits. So in this case,

$$\mathscr{M} = \mathscr{C} = \mathscr{K} = \{0-1 \text{ sequences of length } t\}.$$

Define now the $\oplus$ operation as the bitwise addition, i.e. if $a = \sum_{j=0}^{t-1} a_j 2^j$, $b = \sum_{j=0}^{t-1} b_j 2^j$ (where a_j, b_j's are binary digits, 0 or 1 each), then let

$$a \oplus b = \sum_{j=0}^{t-1} c_j 2^j,$$

where $c_j = 0$ if $a_j = b_j$, and $c_j = 1$ if $a_j \neq b_j$.

Given m and k, let $e_k(m) = m \oplus k$. One can easily see that then $d_k = e_k$:

$$d_k(e_k(m)) = (m \oplus k) \oplus k = m \oplus (k \oplus k) = m \oplus \text{'}\underbrace{0 \ldots 0}_{t \text{ zeros}}\text{'} = m.$$

Since XOR addition can be computed fast, Bob can easily encrypt his message, and Alice can easily decrypt it. However, since Eve does not know k, she essentially has to check all possible k's between 0 and $2^t - 1$, which is hopeless, it t is large enough.

In some sense, this cryptosystem is as perfect as can be. However, there are a few problems. First, it is vulnerable against a chosen plaintext attack, since even a single pair (m, c) reveals $k = m \oplus c$.

Another problem is that if Alice sends to message with the same key, say $c_1 = e_k(m_1), c_2 = e_k(m_2)$, then Eve can XOR them to get

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2.$$

It is not obvious how Eve can use this information, yet she has managed to get rid of k, and this is something that Alice and Bob would like to avoid.

Third, the key must be as long as the message is, which requires very long key sequences. And since Alice and Bob would like to use different keys for each encryption, they have to generate many k's. Their security is the best possible, if they generate random k's, for example, tossing a $0-1$ coin t times gives a k, and they can repeat this as many times as they wish. But the setup is that they are at different places, their t-long tosses will almost certainly result differently. How can they solve this problem? Is it possible to securely and efficiently send long messages using only a single short key?

Assume there exists a function $R : \mathscr{K} \times \mathbf{N} \to \{0, 1\}$ satisfying the following properties:

(1) for any $k \in \mathscr{K}$, $j \in \mathbf{N}$, it is easy to compute $R(k, j)$;

(2) from any $j_1, \ldots, j_n$ and corresponding $R(k, j_1), \ldots, R(k, j_n)$, it is hard to figure out k;

(3) from any $j_1, \ldots, j_n$ and corresponding $R(k, j_1), \ldots, R(k, j_n)$, it is hard to guess the value of $R(k, j)$ with better than a 50% chance of success, if $j \notin \{j_1, \ldots, j_n\}$.

Now Alice and Bob agree on a key k (a number on 200 bits, say), and they agree that for the first message, they will use $R(k, 1), \ldots, R(k, 1000000)$, for the second one, they will use $R(k, 1000001), \ldots, R(k, 2000000)$, and so on.

Such functions R are called pseudorandom number generators, however, it is unknown at the moment whether pseudorandom number generators exist.

Chapter 2
Discrete logarithms and Diffie-Hellman

Let us turn our attention back to asymmetric ciphers, those having a public and a private part. Their significance in computer age is bigger than ever, since it may easily happen that Alice and Bob would like to communicate although they have never met before (so they cannot be sure that any piece of information exchanged by them is handled confidentially).

How can Alice build up her cryptosystem? Given the sets $\mathscr{M}$ and $\mathscr{C}$, she would like to publish a function $k_{\text{pub}} : \mathscr{M} \to \mathscr{C}$ in such a way that $k_{\text{pub}}^{-1} = k_{\text{priv}}$ can not be easily computed. For the first sight, this does not make too much sense: if it is difficult to invert k_{pub} for Eve (and Bob, and anyone else), then so is for Alice herself.

However, this is not the case. When Alice constructs k_{pub}, she uses some additional information, a trapdoor. This trapdoor plays an essential role in the construction of k_{pub}, and – ideally – has the following two properties: without the trapdoor, it is difficult, while using the trapdoor, it is easy to invert k_{pub} (and hence to get k_{priv}).

The second condition is the simpler one. To know that something is computationally easy, it suffices to have a good algorithm in hand, which solves the problem in little time. The critical point is the first condition. How to ensure that something is computationally difficult? Possibly there is no available method to invert a certain function but tomorrow there will be.

The cryptosystems presented below all depend on computational problems. Those which – according to the current state of art – are computationally difficult (i.e. at the moment, there are no fast alogrithms to solve them) are considered to be secure.

2.1 The discrete logarithm problem

One of such computational problems is the discrete logarithm problem (DLP from now on). It is basically the following: given a finite abelian group G, and an element g generating it, for any $a \in G$, compute the smallest positive $k \in \mathbf{N}$ such that

$$\underbrace{g \cdot \ldots \cdot g}_{k \text{ times}} = a.$$

This k is called the discrete logarithm of a (with base g), and we will denote it by $\log_g a$. The basic principle is the following: if we can perform the group operation fast, then from k, it is easy to get a (think of the repetitive squaring used in the proof of Proposition 1.5.50), while from a, it is not obvious to get k. This asymmetry gives us some chance to come up with a public key cryptosystem.

Following this principle, we will see below that using a finite abelian group, we can build up certain cryptosystems whose security rely a lot on the difficulty of the DLP. But is the DLP computationally difficult? We do not know it in general, but certainly it depends very much on the group.

For example, assume that the group is the additive group of the prime field $\mathbf{F}_p$. Then any $0 \neq g \in \mathbf{F}_p$ generates $\mathbf{F}_p$. Given a one may compute its logarithm as follows: if $\log_g a = k$, then

$$a \equiv \underbrace{g + \ldots + g}_{k \text{ times}} \equiv gk \bmod p.$$

Now we see that k is nothing else but the multiplicative inverse of g multiplied by a (all understood modulo p). And the multiplicative inverse of g can be computed in polynomial time: either we write $1 = gu + pv$ and then

$u \bmod g^{-1} \bmod p$ (and this can be done in polynomial time, since u is just produced by a euclidean algorithm, recall the proof of 1.5.5) or by Euler-Fermat (Corollary 1.5.46), $g^{p-2} \bmod p$ is the inverse of g modulo p (and this can also be computed in polynomial time, since this is just raising to a power with a modulus), recall Proposition 1.5.49 and Proposition 1.5.50. This altogether shows that the DLP in the additive group of prime fields is an easy problem, so certainly the methods below are useless with this group.

A much better choice is the multiplicative group $\mathbf{F}_p^\times$ of the same field. We already know that $\mathbf{F}_p^\times$ is cyclic, therefore, it is generated by some element. However, for our methods below, this is not really necessary, we can take an element g whose order modulo p is not too small, and can restrict to the subgroup generated by g. Then the DLP is the following: given g and a, compute the smallest possible k satisfying

$$g^k \equiv a \bmod p.$$

Current methods cannot solve this problem in polynomial time. This will be the starting point of the Diffie-Hellman key exchange and the ElGamal cryptosystem.

Later, we will learn how group structure can be attached to elliptic curves. It is also unknown but widely believed that the DLP is even more difficult in those groups. This gives us the possibility to build a cryptosystem on elliptic curves.

2.2 The Diffie-Hellman key exchange

In this section, we show how Alice and Bob can agree on a fixed number which is known only to them, even if they can only use a completely public channel for their whole communication. This is not as ambitious as sending complete encrypted messages, yet it is something: their common secret number can be for example used for a symmetric cipher.

First of all, they agree on a prime p and an element g whose order modulo p is sufficiently large. Now Alice takes a number a (and keeps it in secret), and Bob also takes a number b (and keeps it in secret). Then Alice computes $g^a \bmod p$, and sends it to Bob. Meanwhile, Bob computes $g^b \bmod p$, and sends it to Alice. Finally, Alice raises the incoming $g^b \bmod p$ to power a, while Bob raises the incoming $g^a \bmod p$ to power b. Both of them obtains the number $g^{ab} \bmod p$, so they have the common number (recall Proposition 1.5.50).

What happens to Eve? She just observes g^a and g^b (besides p, g). From this information, she should compute g^{ab} (this is called the Diffie-Hellman problem (DHP)). If she could solve effectively the DLP, she would immediately figure out the values a, b, and they lead to g^{ab}. Therefore, the DHP is not more difficult than the DLP. The converse in unknown (in pricinple, there can be a polynomial algorithm solving the DHP even if there is no algorithm solving the DLP in polynomial time).

2.3 The ElGamal cryptosystem

In this section, we describe a public key cryptosystem relying on the difficulty of the DLP for multiplicative groups of prime fields.

Again, Alice chooses a prime number p, and an element g of sufficiently large order modulo p. Also, she chooses a number a. Then she computes $A \equiv g^a \bmod p$, and publishes p, g, A (and keeps a in secret).

Then Bob chooses a number k (and keeps it in secret), and sends Alice the following two values:

$$c_1 \equiv g^k \bmod p, \qquad c_2 \equiv mA^k \bmod p,$$

where $1 \leqslant m \leqslant p-1$ is the message he wants to send.

Now Alice gets the pair (c_1, c_2), and computes $c_2(c_1^a)^{-1}$ modulo p. The result is

$$c_2(c_1^a)^{-1} \equiv mA^k((g^k)^a)^{-1} \equiv mg^{ak}g^{-ak} \equiv m \bmod p,$$

so the message is recovered.

This is called the ElGamal cryptosystem. It is clear what k_{pub} is (note that it uses an external parameter k chosen by Bob). Now k_{priv} is (with current methods) is difficult to compute from k_{pub}, however, using the trapdoor a, it can be done fast.

It is clear that solving the DLP breaks the ElGamal, since Eve can easily compute a from A, then she can compute the same way m from (c_1, c_2) as Alice does.

Now we prove that breaking the ElGamal in general is at least as difficult as solving the DHP.

Proposition 2.3.1. *Assume $p, g, A \equiv g^a \bmod p$ are fixed. If there is an algorithm, which computes m from (c_1, c_2) in polynomial time for any input (c_1, c_2), then there is an algorithm solving the DHP with inputs g^a, g^b in polynomial time.*

Proof. Assume we have to compute g^{ab} from g^a, g^b (all modulo p) in DHP. Set $c_1 \equiv g^b \bmod p$ and $c_2 = 1$. The computed m-value is g^{-ab} modulo p, its multiplicative inverse (computed in polynomial time) is g^{ab} modulo p. □

Chapter 3
Integer factorization and RSA

In Chapter 2, the invertibility of k_{pub} reached by chosing the computationally complicated DLP: it is easy to raise a certain power, but it is hard to difficult to figure out the power from the result. In this chapter, we will use the following: it is easy to compute the product of prime powers, but it is hard to figure out the prime factorization of a given number.

3.1 The RSA cryptosystem

Alice chooses two large prime numbers p,q, they will serve for her as the trapdoor. Now she computes the product $N = pq$, and chooses a further number e coprime to $\varphi(N)$. Then she publishes N,e. Alice also computes the multiplicative inverse d of e modulo $\varphi(N)$ (that is, $de = \varphi(N)u+1$). For her, this is easy to do: $\varphi(N) = (p-1)(q-1)$.

If Bob wants to send the message $1 \leqslant m \leqslant N$ to Alice, then he computes $c \equiv m^e \bmod N$ and sends it to Alice.

Then Alice simply raises c mod N to power d, obtaining

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{\varphi(N)u+1} \equiv m \cdot (m^{\varphi(N)})^u \equiv m \bmod N,$$

at least when $\gcd(m,N) = 1$. For a randomly chosen message $1 \leqslant m \leqslant N$

$$\Pr(\gcd(m,N) = 1) = \frac{\varphi(N)}{N} = \frac{N-p-q+1}{N} > 1 - \frac{1}{p} - \frac{1}{q},$$

which is very close to 1, if p,q are large.

Eve's problem is the following: she knows only N and e. If she could factorize N to pq, then it would immediately give $\varphi(N)$, leading to d fast. But there is no known method to factorize large numbers fast.

3.2 Primality testing

To implement RSA, one needs large prime numbers, but how can we get them? To answer this question, we first cite the prime number theorem:

$$\lim_{x\to\infty} \frac{\#\{p \leqslant x\colon\ p \text{ is prime}\}}{x/\log x} = 1.$$

Informally, we may say that up to x, there are approximately $x/\log x$ prime numbers. Reformulating this, we may say that if we pick a random large integer n, then we expect a prime in the set

$$\{n, n+1, \ldots, n+2\lceil \log n \rceil\}.$$

So we choose some n, and then we take the integers $n, n+1, \ldots$ until we bump into a prime. The number of steps we have to take is estimated from above by the gap of consecutive primes. The prime number theorem suggests that this is something like $\log n$ on average. However, it is known that the prime gap around X can be larger than any constant multiple of $\log X$ (it can be bigger than $100 \log X$, say). Conjecturally, the gap is always smaller than a constant multiple of $\log^2 X$ and the truth of this would suffice for a polynomial prime-generating algorithm.

But this is just one part of the problem. Okay, assume there *is* a prime between n and $n+100\log^2 n$. How will we recognize it? Given a large number, how fast can we decide if it is prime or not?

Of course, we can try any number up to $\sqrt{n}$ and if none of them divides n (except for 1 of course), then n is a prime. If n is large, this is awfully slow, so we need a better algorithm.

3.2.1 Fermat's little theorem and Carmichael numbers

Assume a number n is given, and we have to decide if n is a prime or not.

First assume n is a prime. Recall Fermat's little theorem (Corollary 1.5.47), which tells us then that

$$a^n \equiv a \bmod n.$$

So if we pick a number a such that

$$a^n \not\equiv a \bmod n,$$

then we can be sure that n is not a prime. This gives us a primality test: take some a, and if $a^n \not\equiv a \bmod n$, then n is composite.

Unfortunately, the converse is not true at all, for example,

$$2^{341} \equiv 2 \bmod 341,$$

but $341 = 11 \cdot 31$ is not a prime. Even worse, there exist composite integers n such that for any $a \in \mathbf{N}$,

$$a^n \equiv a \bmod n.$$

Such numbers are called Carmichael numbers, and although they are rare compared to primes, there are infinitely many of them (the smallest one is $561 = 3 \cdot 11 \cdot 17$).

So if our randomly found number n is a Carmichael number, then it is not a prime, yet we have no chance to prove its compositeness via Fermat's little theorem.

3.2.2 The Miller-Rabin test

The fundament of the Miller-Rabin test is the following observation.

Proposition 3.2.1. *Assume p is an odd prime, and write $p-1$ as $2^k q$, where q is an odd number. Then for any integer a coprime to p, one of the following two alternatives hold. Either $a^q \equiv 1 \bmod p$ or one of $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ is -1 modulo p.*

Proof. Fix an a as in the statement. If $a^q \equiv 1 \bmod p$, then we are done. If not then consider the number $a^{2^k q} \equiv a^{p-1}$, which is 1 modulo p by Euler-Fermat (Corollary 1.5.46). Since $\mathbf{F}_p$ is a field, the only residue classes x satisfying $x^2 \equiv 1 \bmod p$ are ± 1. Therefore since the square of $a^{2^{k-1}q}$ is 1 mod p, it must be either 1 or -1 modulo p. If it is -1, then we are done. If it is 1, we can go further and take $a^{2^{k-2}q}$: again, since its square is 1, it is ± 1 modulo p. We continue this, and since $a^q \not\equiv 1 \bmod p$, at some point, we must get -1 modulo p. □

Like Fermat's little theorem, this immediately gives us a prime test. Assume n is given. If it is even, then it is composite (unless it is 2). If it is odd, take $n-1$ and halve it as many times as possible, altogether getting $n-1 = 2^k q$ for some odd number q and positive integer k.

Choose a random $1 \leqslant a \leqslant n-1$. Check first if it is coprime to n (this can be done fast by Proposition 1.5.49). If $\gcd(a,n) > 1$, then n is composite. If $\gcd(a,n) = 1$, then compute $a^q, a^{2q}, \ldots, a^{2^k q}$ modulo n. They are at most $1 + \log_2 n$ many numbers, and they can be computed in polynomial time by Proposition 1.5.50. If a^q is not 1, and none of them is -1 modulo n, then we can be certain that n is composite.

But we have already experienced that things might go wrong: it can happen in principle that something is a fake-prime, by which we mean that it is composite yet either $a^q \equiv 1 \bmod n$ or one of $a^q, a^{2q}, \ldots, a^{2^k q}$ is $-1 \bmod n$ for all possible choices for a. Why this method is better is the fact that there are no fake-primes (i.e. there are no analogues for Carmichael numbers here).

Proposition 3.2.2. *If n is an odd composite number, then there exists a coprime residue classe a such that $a^q \not\equiv 1 \bmod n$ and $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q} \not\equiv -1 \bmod n$.*

Proof. First, when n is a power of a prime, say, p^α, then take a generator a of the cyclic group $\mathbf{Z}_n^\times$. Then, since

$$\gcd(\varphi(n), n-1) = \gcd(p^\alpha - p^{\alpha-1}, p^\alpha - 1) = (p-1)\gcd(p^{\alpha-1}, p^{\alpha-1} + \ldots + p + 1) = p - 1,$$

$a^{n-1} \not\equiv 1 \bmod n$. Then obviously none of $a^q, a^{2q}, \ldots, a^{2^{k-1}q}$ is ± 1 modulo n.

Now assume n is not a power of a prime, then write $n = n_1 n_2$, where $1 < n_1, n_2 < n$ and $\gcd(n_1, n_2) = 1$. By the Chinese remainder theorem, there exists a residue class a modulo n such that $a \equiv 1 \bmod n_1$ and $a \equiv -1 \bmod n_2$. This clearly implies $a \not\equiv \pm 1 \bmod n$, and then $a^q \not\equiv \pm 1 \bmod n$. On the other hand, $a^2 \equiv 1 \bmod n_1$ and $a^2 \equiv 1 \bmod n_2$, implying $a^2 \equiv 1 \bmod n$. Then $a^q \not\equiv \pm 1 \bmod n$, and $a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q} \equiv 1 \not\equiv -1 \bmod n$. □

In fact, it can be proved that at least 75% of the residue classes do this job. This gives rise to a random primality testing: take a convincing number of residue classes $a_1, \ldots, a_h \bmod n$, and run the Miller-Rabin test with them. If any of them shows n is composite, then we n is composite. If all of them allows n to be prime, then we can say n is a prime with a high confidence. How high? Let us estimate this using Bayes's theorem.

Choose a number at random which is approximately n. Then denote by X the event that the chosen number is composite, by X' that it is a prime, and by Y that it survives the Miller-Rabin test with h numbers. Then

$$\Pr(X \mid Y) = \frac{\Pr(Y \mid X)\Pr(X)}{\Pr(Y \mid X)\Pr(X) + \Pr(Y \mid X')\Pr(X')}.$$

Here, from the prime number theorem, we know that $\Pr(X') \approx \log n/n$ and $\Pr(X) \approx 1 - \log n/n$. Also, if the number is composite, it survives Miller-Rabin by probability at most 4^{-h}, so $\Pr(Y \mid X) \leqslant 4^{-h}$, while primes certainly survive Miller-Rabin, so $\Pr(Y \mid X') = 1$. Then

$$\Pr(X \mid Y) \lesssim \frac{4^{-h}(1 - \log n/n)}{4^{-h}(1 - \log n/n) + \log n/n}.$$

We see that this converges to 0 very fast: even if h is chosen to be $\lfloor \log_4 n \rfloor$ (which keeps the whole test polynomial), the probability of judging a composite number to be a prime tends to 0 (as n tends to infinity).

Also, it is known that if the Generalized Riemann Hypothesis is true, then for any composite number n there is an $a \leqslant 2\log^2 n$ which proves the compositeness of n in the Miller-Rabin test. Hence assuming the Generalized Riemann Hypothesis, the Miller-Rabin test is not only a probabilistic, but a deterministic test which decides the primality of a number in polynomial time.

3.2.3 The Agrawal-Kayal-Saxena polynomial test

However, even the classical Riemann Hypothesis is open, especially so are the generalized versions. In this section, we present the AKS primality test, which decides primality in polynomial time *unconditionally*. Note also that this has only theoretical significance: first, as we will see, the AKS test is very complicated, while Miller-Rabin is easy to implement; second, almost all mathematicians strongly believe that the Riemann Hypothesis (and all its reasonable generalizations) are true – all in all, for practical purposes, Miller-Rabin is totally fine.

Assume n is a large number, fixed once for all.

Proposition 3.2.3. *There exists a number $1 \leqslant r = O(\log^{O(1)} n)$ such that the order of n in $\mathbf{Z}_r^\times$ is greater than $\log_2^2 n$.*

Proof. Using the fact that for any $x \geqslant 2$,

$$\prod_{\substack{p \leqslant x \\ p \text{ prime}}} p < 4^x,$$

we can easily see that the number of prime divisiors of $n^i - 1$ is $O(\log^{O(1)} n)$ for any $1 \leqslant i \leqslant \log_2^2 n$. Then picking the first prime r which does not appear as a prime divisor of

$$\prod_{1 \leqslant i \leqslant \log_2^2 n} (n^i - 1),$$

$r = O(\log^{O(1)} n)$ and the order of n modulo r is bigger than $\log_2^2 n$. □

Now with this r in hand, for any $1 \leqslant a \leqslant n$, we can consider the polynomial $(X+a)^n$ in variable X. If n is prime, then

$$(X+a)^n \equiv X^n + a^n \equiv X^n + a \bmod n,$$

the second congruence holds because $a^n \equiv a \bmod n$ by Fermat's little theorem 1.5.47, the first congruence holds because in the binomial expression, all intermediate term have a factor n.

This implies, in particular, that if n is a prime, then

$$(X+a)^n \equiv X^n + a \bmod (n, X^r - 1). \tag{3.2.1}$$

The point is that although the polynomial $(X+a)^n$ takes a long time to compute, its modulo X^r-1 reduction can be computed in polynomial time. Indeed, write n in base 2, and apply repeated squarings modulo X^r-1 (like in the proof of Proposition 1.5.50).

Proposition 3.2.4. *If n is not a prime or a power of a prime, then there exists a number $1 \leqslant a = O(\log^{O(1)} n)$ such that (3.2.1) fails or $\gcd(a,n) > 1$.*

Proof. We prove by contradiction. Assume n is not a prime, nor a power of a prime, yet $\gcd(a,n) = 1$ and (3.2.1) are true for any $a = O(\log^{O(1)} n)$. Then n has a prime divisor $p > \log^C n$ with some number C (bigger than the $O(1)$ in the bound for a, in particular, we may assume $p > r$).

Now consider the field $\mathbf{F} = \mathbf{F}_p(X)$, where X is a primitive rth root of unity, and from now on, we compute in $\mathbf{F}$. The hypothesis (3.2.1) gives

$$(X+a)^n = X^n + a,$$

for any $1 \leqslant a \leqslant A$, where $A = O(\log^{O(1)} n)$ (and we choose A to be bigger than $2r$). Also, we have

$$(X+a)^p = X^p + a,$$

for $1 \leqslant a \leqslant A$, as above, since p is a prime. Then, for $1 \leqslant a \leqslant A$,

$$(X^p)^{n/p} + a = X^n + a = (X+a)^n = ((X+a)^p)^{n/p} = (X^p + a)^{n/p}.$$

Since X^p is a primitive rth root of unity again (and every primitive rth of unity is the pth power of a primitive rth root of unity), this implies

$$X^{n/p} + a = (X+a)^{n/p},$$

for any $1 \leqslant a \leqslant A$.

For any $1 \leqslant m \leqslant n$ coprime to r, we define the ring homomorphism $\phi_m : \mathbf{F} \to \mathbf{F}$ sending X to X^m, and we say that m is *nice*, if

$$\left(\widetilde{X} + a\right)^m = \phi_m\left(\widetilde{X} + a\right) = \widetilde{X}^m + a$$

holds for all $1 \leqslant a \leqslant A$ and for any $\widetilde{X} = X^t$, where t is coprime to r (i.e. $\widetilde{X}$ runs through all the primitive rth roots of unities).

Obviously, 1 is nice, and we have already seen that $n, p, n/p$ are all nice. Also, if m and m' are both nice, then so is their product:

$$\left(\widetilde{X} + a\right)^{mm'} = \left(\left(\widetilde{X} + a\right)^m\right)^{m'} = \left(\widetilde{X}^m + a\right)^{m'} = \widetilde{X}^{mm'} + a.$$

Let $G \subseteq \mathbf{F}^\times$ denote the multiplicative group generated by the elements $X + a$ with $1 \leqslant a \leqslant A$. Obviously, for any $z \in G$, $\phi_m(z) = z^m$.

Lemma 3.2.5. *Assume that there are exactly t nice residue classes modulo r of the form $p^i(n/p)^j$ for $i, j \geqslant 0$. Then $\#G \leqslant n^{\sqrt{t}}$.*

Proof of this lemma. For some $0 \leqslant i, j, i', j' \leqslant \sqrt{t}$ with $(i,j) \neq (i',j')$, we have

$$m = p^i(n/p)^j \equiv p^{i'}(n/p)^{j'} = m' \bmod r,$$

implying $\phi_m(z) = \phi_{m'}(z)$ for any $z \in G$. The numbers m, m' here are distinct positive integers not exceeding $n^{\sqrt{t}}$. Then for any $z \in G$, $z^m - z^{m'} = 0$, which means that each $z \in G$ is a root of the polynomial $Z^m - Z^{m'}$. Since a nonzero polynomial over a field cannot have more roots than its degree, $\#G \leqslant \max(m, m') \leqslant n^{\sqrt{t}}$. □

Lemma 3.2.6. *Assume that there are at least t nice residue classes modulo r. Then $\#G \geqslant 2^t$.*

Proof of this lemma. Make all the products with less than t factors from the elements $X+a$ (with $1 \leqslant a \leqslant A$, and allowing repetitions). Since $A > 2r \geqslant 2t$, there are at least 2^t many such products, and we claim that they are distinct.

By contradiction, let $P(X) = Q(X)$ for two such products, where $P(Z), Q(Z)$ are distinct polynomials of degree less than t. Since $P(X), Q(X) \in G$, for any nice m,

$$P(X^m) = \phi_m(P(X)) = \phi_m(Q(X)) = Q(X^m).$$

If $m_1, \ldots, m_t$ are nice numbers of distinct remainders modulo r, then this means that $X^{m_1}, \ldots, X^{m_t}$ are all distinct roots of the polynomial $P(Z) - Q(Z)$, which is a contradiction: the polynomial is nonzero and has degree less than t, while it has at least t roots. □

Since the order of n modulo r is at least $\log_2^2 n$, we see that the numbers $p^i(n/p)^j$ fall into at least $\log_2^2 n$ residue classes modulo n. Then $t > \log_2^2 n$ in the notation of the lemmas above. Then

$$2^t = 2^{t-\log_2^2 n} 2^{\log_2^2 n} = 2^{t-\log_2^2 n} 2^{\log_2^2 n} = 2^{t-\log_2^2 n} n^{\log_2 n},$$

and

$$n^{\sqrt{t}} = n^{\sqrt{t}-\log_2 n} n^{\log_2 n} = 2^{\sqrt{t}\log_2 n - \log_2^2 n} n^{\log_2 n},$$

so

$$2^t > n^{\sqrt{t}},$$

the lower bound beating the upper bound, which is obviously a contradiction. □

Now the AKS test is the following. Given n, find r (as in Proposition 3.2.3) in polynomial time. Then for the polynomially many $1 \leqslant a \leqslant A$ (as in Proposition 3.2.4), check in polynomial time if it is coprime to n and then if the identity (3.2.1) holds. If they always hold, then n is a prime power (otherwise, it is composite). Now it is easy to check in polynomial time if n is in fact a prime: all the possible square roots, cube roots etc. can be found by dyadic splits of $[1,n]$ (this is polynomially many steps), and the number of possible exponents is trivially at most $\log_2 n$.

3.3 Multiplayer RSA – the bad way

In this section, we make our first attempt to construct a secure way of communication between n participants – based on RSA. Assume that a Trusted Authority takes two large prime numbers p, q, and computes their product $N = pq$. Then also computes, for any $1 \leqslant i \leqslant n$, a pair (e_i, d_i) such that $e_i d_i \equiv 1 \bmod \varphi(N)$. Now each e_i is made public, but the Trusted Authority sends only each decrypting exponent d_i to the ith participant. Now we see that the participants can communicate as we have seen above.

Unfortunately, there are several problems with this setup. First of all, the Trusted Authority is able to read any message, but let us put this aside. An even bigger problem is that basically any participant can read any message, even that ones which have been sent to other than her/him.

Assume the jth paricipant wants to decrypt a message sent to the kth participant. First, (s)he computes a factorization uv of $e_j d_j - 1$ such that $\gcd(u, e_k) = 1$, and each prime factor of v divides e_k. This is easy to do: first (s)he computes $v_1 = \gcd(e_k, e_j d_j - 1)$, then $v_2 = \gcd(e_k, (e_j d_j - 1)/v_1)$, then $v_3 = \gcd(e_k, (e_j d_j - 1)/v_1 v_2)$ and so on, until the gcd becomes 1. Then $v = v_1 \cdot \ldots \cdot v_t$ and $u = (e_j d_j - 1)/v$, and it is easy to see that both conditions hold.

Now since $\gcd(\varphi(N), e_k) = 1$, $\gcd(\varphi(N), v) = 1$. Since $\varphi(N) \mid uv$, this implies $\varphi(N) \mid u$. Then (s)he can compute a d_k' such that $e_k d_k' \equiv 1 \bmod u$, which in particular implies $e_k d_k' \equiv 1 \bmod \varphi(N)$. Maybe this d_k' is not the same as d_k but serves as well as d_k for decryptions.

But even if the participants are honest, an external eavesdropper can make use of the fact that there are several exponents in the setup. It is quite likely that in a long communication, it happens that a message is sent to two different recipients, i.e. the eavesdropper can intercept a pair $c_1 \equiv m^{e_1}, c_2 \equiv m^{e_2} \bmod N$. With a little luck, the exponents are coprime, and then $u, v \in \mathbf{Z}$ satisfying $e_1 u + e_2 v = 1$ are computable. But then

$$c_1^u c_2^v \equiv m^{e_1 u} m^{e_2 v} \equiv m^{e_1 u + m_2 v} = m \bmod N.$$

So messages sent to many paricipants are simply readable.

Chapter 4
Probability and information theory

4.1 The Vigenère cipher and its cryptanalysis

For awhile, we return to alphabetic encryptions. In Chapter 1, we have already seen such ciphers: these replace each character with some other *fixed* character – these are called *monoalphatbetic* ciphers. Now we turn to the Vigenère cipher, an example of *polyalphabetic* ciphers, where each letter is replaced with some other one, but this cipher letter may vary.

First Alice and Bob agree on a keyword. Then Bob encrypts his message as follows: he determines the shift he uses on each letter of the message by using the letters of the keyword one by one. The keyword letter `a` stands for no shift, `b` stands for a shift by one, and so on.

Example 4.1.1. Assume the keyword is 'CHESS'. Encrypt the message 'Learning mathematics is fun'. Then

```
LEARN INGMA THEMA TICSI SFUN
CHESS CHESS CHESS CHESS CHES
---------------------------
NLEJF KUKES VOIES VPGKA UMYF,
```

so the ciphertext is `NLEJF KUKES VOIES VPGKA UMYF`.

The principle behind is the following. The letters of the keyword fix a variety of permutations of the alphabet, so breaking the cipher via counting common characters, bigrams, trigrams (which breaks simple substitution ciphers) is more difficult to perform in this case. Even if the length of the keyword has only five characters (four different, one is doubled), the common letter 'e' implies four common letters 'g', 'l', 'i', 'w' ('w' appearing with double frequency because of the double 's' in the keyword), and the common trigram 'the' in a long text implies four common trigrams 'voi', 'alw', 'xzw', 'lzg', 'ljl'. All in all, the characteristic of the language is totally mixed up.

For some time in history, the Vigenère cipher was considered to be unbreakable. Then in the 19th century, when basic statistical tools were developed, this view turned out to be very far from truth. In this section, we only describe the main methods to attack the cipher and suggest read [1, Section 4.2.2] to see how these approaches work in action.

The first goal is to determine the length of the keyword. We present two methods to do this.

Consider the repeated fragments (bigrams and trigrams) in the ciphertext, and list all the distances between the repetitions. Some of the repetitions may occur only by chance, but the length of the keyword is likely to divide many of the distances. This method depends on the fact that certain fragments are very frequent in natural languages (think of trigrams 'the' or 'ing' in English). In a not too short cipher, there is a considerable amount of occurrences of 'the', and if the key is not too long, some of them will be encrypted the same way. This method is known as the Kasiski test.

Another method is the index of coincidence test, which relies on individual letters rather than on two- or three-letter fragments. For each letter of the alphabet $i \in \mathfrak{A} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \ldots, \mathrm{z}\}$, let $F_i(\mathbf{s})$ be its frequency, i.e. the number of occurrences of i in a given string $\mathbf{s}$. With this notation, define, for any string $\mathbf{s}$, the index of coincidence

$$\mathrm{IndCo}(\mathbf{s}) = \frac{1}{\mathrm{length}(\mathbf{s})(\mathrm{length}(\mathbf{s})-1)} \sum_{i \in \mathfrak{A}} F_i(\mathbf{s})(F_i(\mathbf{s})-1).$$

If a very long string is completely random, then all $F_i(\mathbf{s})$ are essentially $\text{length}(\mathbf{s})/26$, so we expect

$$\text{IndCo}(\mathbf{s}) \approx \frac{1}{26} \approx 0.0385.$$

However, if you consider a long text $\mathbf{t}$ written in English, then you will find that

$$\text{IndCo}(\mathbf{t}) \approx 0.0685.$$

Although the difference of these two numbers seems small, it can be applied in practice. Assume there is a ciphertext $c_1, \ldots, c_n$. Now for some $k \in \mathbf{N}$, split the ciphertext into k pieces $\mathbf{s}_0, \ldots, \mathbf{s}_{k-1}$:

$$\mathbf{s}_i = c_i c_{i+k} c_{i+2k} \ldots c_{i+\lfloor n/k \rfloor k}.$$

For each $0 \leqslant i \leqslant k-1$, compute $\text{IndCo}(\mathbf{s}_i)$. If k is the length of the keyword, then characters in $\mathbf{s}_i$ are the encrypted versions of the plaintext using the same encryption, i.e. $\text{IndCo}(\mathbf{s}_i)$ is close to the index of coincidence 0.0685 of a natural English text (provided that $\mathbf{s}_i$ is not too short). On the other hand, if k is not the right guess for the keyword, then $\text{IndCo}(\mathbf{s}_i)$'s are expected to be close to the index of coincidence 0.0385 of a random text. All in all, if the cipher is long enough, and the key is short enough, this method is expected to tell us the right key length.

Having figured out the key length, we introduce another coefficient which can help to figure out the difference between the shifts applied to get the $\mathbf{s}_i$'s. This is the mutual index of coincidence, defined as

$$\text{MutIndCo}(\mathbf{s}, \mathbf{t}) = \frac{1}{\text{length}(\mathbf{s})\,\text{length}(\mathbf{t})} \sum_{i \in \mathfrak{A}} F_i(\mathbf{s}) F_i(\mathbf{t}),$$

for any strings $\mathbf{s}$ and $\mathbf{t}$. Again, if $\mathbf{s}$ and $\mathbf{t}$ are encrypted with the same shift, then their mutual index of coindidence is large, otherwise, it is small. For any $\mathbf{s}_i$ created above and any $\sigma \in \mathfrak{A}$, we define $\mathbf{s}_i + \sigma$, which is a further shift of each character of $\mathbf{s}_i$ by σ (as in the Vigenère cipher: `a` stands for no shift, `b` stands for a shift by one, and so on). For any $0 \leqslant i < j \leqslant k-1$, and any $\sigma \in \mathfrak{A}$, compute

$$\text{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma).$$

Denoting the keyword by $\beta_1 \ldots \beta_k$, and singling out the large values of $\text{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma)$, we find a linear system of equations of the form $\beta_i - \beta_j = \sigma$. This system can easily be overdetermined, when we have to leave out a few equations, but after some trial and error, we can determine $\beta_1 \ldots \beta_k$ up to a shift which is *uniform* at all the k places. This leaves 26 possibilities, and trying them all, we arrive at the original plaintext.

4.2 Collision and meet-in-the-middle attacks

Proposition 4.2.1. *Assume $A_1, \ldots, A_n$ are independent events such that*

$$\Pr(A_1), \ldots, \Pr(A_n) \geqslant p.$$

Then

$$\Pr((A_1 \cup \ldots \cup A_n)^c) \leqslant e^{-np}.$$

Proof. First, we claim, for any $x \in [0,1]$,

$$1 - x \leqslant e^{-x}.$$

Indeed, equality holds for $x = 0$, and the derivative (with respect to x) of the left-hand side is -1, while that of the right-hand side is not less than -1 for any $x \in [0,1]$. (Alternatively, writing $-x$ in place of x in the Taylor expansion $e^x = \sum_{j=0}^{\infty} x^j/j!$,

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \pm \ldots,$$

and for $0 \leqslant x \leqslant 1$, this is indeed at least $1-x$, since for any $j \geqslant 1$, $x^{2j}/(2j)! - x^{2j+1}/(2j+1)! \geqslant 0$.)

Then from the independeny of $A_1, \ldots, A_n$,

$$\Pr((A_1 \cup \ldots \cup A_n)^c) = \Pr(A_1^c \cap \ldots \cap A_n^c) = \prod_{i=1}^{n} \Pr(A_i^c) \leqslant (1-p)^n \leqslant (e^{-p})^n = e^{-np}.$$

The proof is complete. □

This statement enables us to attack certain problems with probabilistic algorithms (at the moment, this means that the algorithm is deterministic, and it solves the problem with high probability). We illustrate this phenomenon via the DLP.

Proposition 4.2.2. *Assume G is a finite abelian group on N elements, and g is a generator. Let $a \in G$ be given. Then the DLP $g^x = a$ can be solved in $O(\sqrt{N})$ steps with high probability ($O(C\sqrt{N})$ steps lead to probability $1 - e^{-C^2}$ for any $C > 0$), where a step is an exponentiation in the group (which can be computed in $O(\log N)$ group multiplications, recall the repetitive squaring from Proposition 1.5.50).*

Proof. For some fixed $1 \leqslant n \leqslant N$, pick the elements $y_1, \dots, y_n$ and $z_1, \dots, z_n$, and compute

$$g^{y_1}, \dots, g^{y_n}, ag^{z_1}, \dots, ag^{z_n}.$$

If there is a match, i.e.

$$g^{y_i} = ag^{z_j}$$

for some $1 \leqslant i, j \leqslant n$, then $a = g^{y_i - z_j}$. We claim that such a match occurs with high probability, if n is on the magnitude $\sqrt{N}$, the y_i's are chosen independently and the z_j's are chosen to be distinct.

As soon as $z_1, \dots, z_n$ are fixed, each y_i gives a match with probability n/N. Since the y_i's are independent, by Proposition 4.2.1,

$$\Pr(\text{there is no match}) \leqslant e^{-n^2/N}.$$

Here, if n is chosen to be $\lceil C\sqrt{N} \rceil$, the probability of not getting a match is at most e^{-C^2}. □

4.3 Perfect secrecy and entropy

Now we try to understand how much information a given ciphertext reveals about the plaintext. For a proper formalization, we introduce the random variables M, K, C: M is a random plaintext ($M : \Omega \to \mathscr{M}$), K is a random key ($K : \Omega \to \mathscr{K}$), and C is a random ciphertext ($C : \Omega \to \mathscr{C}$). We denote by f_M, f_K, f_C their density functions, i.e.

$$f_M(m) = \Pr(M = m), \qquad f_K(k) = \Pr(K = k), \qquad f_C(c) = \Pr(C = c).$$

Assume that each $f_M(m), f_K(k), f_C(c)$ is positive, i.e. drop the possible messages, keys and ciphers of zero probability. (Although theoretically, impossibility does not mean the same as zero probability.)

Also, we may introduce the conditional probabilities

$$f_{M|K}(m \mid k) = \Pr(M = m \mid K = k), \qquad f_{M|C}(m \mid c) = \Pr(M = m \mid C = c), \qquad f_{K|C}(k \mid c) = \Pr(K = k \mid C = c),$$
$$f_{K|M}(k \mid m) = \Pr(K = k \mid M = m), \qquad f_{C|M}(c \mid m) = \Pr(C = c \mid M = m), \qquad f_{C|K}(c \mid k) = \Pr(C = c \mid K = k).$$

Definition 4.3.1 (perfect secrecy)**.** We say that the cryptosystem has perfect secrecy, if for all $m \in \mathscr{M}$ and $c \in \mathscr{C}$,

$$f_{M|C}(m \mid c) = f_M(m).$$

This definition indeed grabs the notion that c does not reveal information on m: intercepting the ciphertext c, Eve has the same chance when guessing m as a priori.

Proposition 4.3.2. *If a cryptosystem has perfect secrecy, then $\#\mathscr{K} \geqslant \#\mathscr{M}$.*

Proof. Assume $\#\mathscr{K} < \#\mathscr{M}$. Take any $m \in \mathscr{M}$. Since for each $k \in \mathscr{K}$, $e_k(m)$ is a well-defined element of $\mathscr{C}$, we see that by assumption, there exists an element $c \in \mathscr{C}$ which does not appear as $e_k(m)$ when k runs through $\mathscr{K}$. This means that

$$f_{M|C}(m \mid c) = 0.$$

On the other hand, $f_M(m) > 0$, which excludes perfect secrecy. □

In efficient cryptosystems, however, a single key is used many times and for many plaintexts, therefore, perfect secrecy is impossible to reach. Alice and Bob still have the endeavor to build a cryptosystem which gives Eve the least possible information.

To measure information, Shannon introduced the notion of entropy in communication theory.

Definition 4.3.3 (entropy)**.** Given a random variable X taking the values $x_1,\dots,x_n$ with probabilities $p_1,\dots,p_n$, respectively, the entropy is defined as

$$H(X) = H(p_1,\dots,p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

For any $n \in \mathbf{N}$, set

$$U_n = \{(x_1,\dots,x_n) \in \mathbf{R}^n : x_1,\dots,x_n > 0 \text{ and } x_1+\dots+x_n = 1\},$$

and

$$U = \bigcup_{n=1}^{\infty} U_n.$$

With this notation, we may think of the entropy function H as a function from U to $\mathbf{R}$. It is also symmetric, by which we mean that restricted to any U_n, it is invariant under any permutation of the n coordinates.

Proposition 4.3.4 (characterization of the entropy)**.** *The entropy is defined up to a constant multiple by the following three axioms (and the fact that H is a symmetric function from U to $\mathbf{R}$):*

(H1) H is continuous on any U_n;

(H2) H is monotonically increasing on the uniform distributions, i.e. for any $m,n \in \mathbf{N}$, if $m > n$, then

$$H\left(\underbrace{\frac{1}{m},\dots,\frac{1}{m}}_{m \text{ many}}\right) > H\left(\underbrace{\frac{1}{n},\dots,\frac{1}{n}}_{n \text{ many}}\right);$$

(H3) H is decomposable, i.e. if $(p_0,p_1,\dots,p_n) \in U_{n+1}$, and $q_1,\dots,q_m \in U_m$, then

$$H(q_1p_0,\dots,q_mp_0,p_1,\dots,p_n) = H(p_0,p_1,\dots,p_n) + p_0H(q_1,\dots,q_m)$$

(obviously $(q_1p_0,\dots,q_mp_0,p_1,\dots,p_n) \in U_{n+m}$).

Proof. For any $n \in \mathbf{N}$, set

$$f(n) = H\left(\underbrace{\frac{1}{n},\dots,\frac{1}{n}}_{n \text{ many}}\right).$$

We claim that for any $m,n \in \mathbf{N}$, $f(mn) = f(m) + f(n)$. Indeed, decomposing all the $1/n$'s to $1/(mn)$'s in

$$H\left(\underbrace{\frac{1}{n},\dots,\frac{1}{n}}_{n \text{ many}}\right)$$

and applying (H3) n times, we obtain,

$$H\left(\underbrace{\frac{1}{mn},\dots,\frac{1}{mn}}_{mn \text{ many}}\right) = H\left(\underbrace{\frac{1}{n},\dots,\frac{1}{n}}_{n \text{ many}}\right) + H\left(\underbrace{\frac{1}{m},\dots,\frac{1}{m}}_{m \text{ many}}\right),$$

which gives $f(mn) = f(m) + f(n)$.

Writing here $m = n = 1$, we obtain $f(1) = H(1) = 0$ immediately. Then $0 < f(2) < f(3) < \dots$ by (H2). We claim that $f(n)/\log n$ is constant. To this aim, assume $p > 2$ is a prime number. For a large number $k \in \mathbf{N}$, sandwich p^k between two consecutive powers of 2:

$$2^{\lfloor \log_2 p^k \rfloor} < p^k < 2^{\lceil \log_2 p^k \rceil}.$$

Applying f and $f(mn) = f(m) + f(n)$, we obtain

$$\lfloor \log_2 p^k \rfloor f(2) < kf(p) < \lceil \log_2 p^k \rceil f(2).$$

Then

$$(\log_2 p^k - 1) f(2) < kf(p) < (\log_2 p^k + 1) f(2),$$

which gives

$$(k \log_2 p - 1) f(2) < kf(p) < (k \log_2 p + 1) f(2).$$

Dividing by $k \log p$,

$$\frac{f(2)}{\log 2} - \frac{f(2)}{k \log p} < \frac{f(p)}{\log p} < \frac{f(2)}{\log 2} + \frac{f(2)}{k \log p}.$$

Letting $k \to \infty$, this implies that $f(p)/\log p$ is the same positive number for each prime p. This clearly implies that $f(n)/\log n$ is a fixed positive constant.

From this, it is clear that for some $c > 0$,

$$H\left(\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}_{n \text{ many}}\right) = -c \sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n}$$

holds for all $n \in \mathbf{N}$. Via (H3), this implies, for $(p_1, \ldots, p_n) \in U_n \cap \mathbf{Q}^n$,

$$H(p_1, \ldots, p_n) = -c \sum_{i=1}^{n} p_i \log_2 p_i.$$

(Indeed, write each p_i as a_i/N, where N is a common denominator, $a_i \in \mathbf{N}$, and apply (H3) for each i.)

Now (H1) completes the proof. □

Definition 4.3.5 (equivocation (conditional entropy))**.** Assume there are two random variables X, Y with values $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$, respectively. Their equivocation (conditional entropy) is defined as

$$H(X \mid Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} \Pr(Y = y_j) \Pr(X = x_i \mid Y = y_j) \log_2 \Pr(X = x_i \mid Y = y_j).$$

Proposition 4.3.6. *We have*

$$H(X,Y) = H(Y) + H(X \mid Y), \tag{4.3.1}$$

where by X, Y, we mean the random variable $X \times Y$, which takes the value (x,y) with probability $\Pr(X = x, Y = y)$ (and analogously to more variables). Also, if X and Y are independent, then

$$H(X,Y) = H(X) + H(Y). \tag{4.3.2}$$

Proof. We prove (4.3.1) first. Assuming the values of X are $x_1, \ldots, x_m$ and those of Y are $y_1, \ldots, y_n$, we introduce the notation $p_{i,j} = \Pr(X = x_i, Y = y_j)$ for any $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$. Then for any $1 \leqslant j \leqslant n$,

$$\Pr(Y = y_j) = \sum_{u=1}^{m} p_{u,j},$$

and for any $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$,

$$\Pr(X = x_i \mid Y = y_j) = \frac{p_{i,j}}{\Pr(Y = y_j)} = \frac{p_{i,j}}{\sum_{u=1}^{m} p_{u,j}}$$

The left-hand side of (4.3.1) is, by definition,

$$H(X,Y) = -\sum_{i=1}^{m} \sum_{j=1}^{n} p_{i,j} \log_2 p_{i,j}.$$

On the right-hand side

$$H(X \mid Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p_{i,j}\log_2\left(\frac{p_{i,j}}{\sum_{u=1}^{m} p_{u,j}}\right).$$

Then

$$H(X,Y) - H(X \mid Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p_{i,j}\log_2\left(\sum_{u=1}^{m} p_{u,j}\right) = -\sum_{j=1}^{n}\Pr(Y=y_j)\log_2\Pr(Y=y_j) = H(Y),$$

and the proof of (4.3.1) is complete.

Now assume X and Y are indpendent random variables with values $x_1,\dots,x_m$, $y_1,\dots,y_n$, respectively. Let $\Pr(X=x_i)=p_i$, $\Pr(Y=y_j)=q_j$, by independency, $\Pr(X=x_i,Y=y_j)=p_iq_j$ then. Now

$$H(X \mid Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p_iq_j\log_2 p_i = -\sum_{i=1}^{m} p_i\log_2 p_i = H(X),$$

and then (4.3.2) follows from (4.3.1). □

Proposition 4.3.7. *We have*

$$H(X \mid Y) \leqslant H(X).$$

Proof. Assuming the values of X are $x_1,\dots,x_m$ and those of Y are $y_1,\dots,y_n$, we introduce the notation $p_{i,j} = \Pr(X=x_i,Y=y_j)$ for any $1\leqslant i\leqslant m$, $1\leqslant j\leqslant n$. Then for any $1\leqslant j\leqslant n$,

$$\Pr(Y=y_j) = \sum_{u=1}^{m} p_{u,j}.$$

Then

$$H(X \mid Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p_{i,j}\log_2\left(\frac{p_{i,j}}{\sum_{u=1}^{m} p_{u,j}}\right).$$

Computing the derivative of $t\log(t/T)+(1-t)\log((1-t)/T)$ with respect to t (for a fixed $0<T<1$ and $0<t<T$), we obtain

$$\frac{d}{dt}\left(t\log\frac{t}{T}+(1-t)\log\frac{1-t}{T}\right) = \log t - \log(1-t),$$

and we see that the maximum of $H(X \mid Y)$, for a fixed Y is attained at $p_{i,j}=p_{i',j}$ for any $1\leqslant i<i'\leqslant m$ and any $1\leqslant j\leqslant n$ by a simple analytic argument. Indeed, by compactness, there is a maximum, and if $p_{i,j}\neq p_{i',j}$, they can be replaced with their average, this increases the value (for interior $p_{i,j},p_{i',j}\neq 0$, this follows from the derivative; while on the boundary, the statement follows from continuity: if a boundary point admitted a bigger value than the midpoint then so would do a small perturbation of it). This implies that for a fixed Y, $H(X \mid Y)$ is maximal, when X is independent of Y. For such X's (4.3.1) and (4.3.2) give $H(X \mid Y)=H(X)$, and this completes the proof. □

In the cryptographic setup, we would like to measure the information that the cipher gives on the key, and define $H(K \mid C)$ to be the key equivocation. The higher its value is, the less information the cipher reveals about the key.

Proposition 4.3.8. *If M and K are independent, then*

$$H(K \mid C) = H(M)+H(K)-H(C).$$

Proof. First, by (4.3.1), we have

$$H(M,K,C) = H(C \mid M,K)+H(M,K).$$

Here, since C is determined by M and K, $H(C \mid M,K)=0$, and by (4.3.2),

$$H(M,K) = H(M)+H(K).$$

Similarly, since M is determined by K and C,

$$H(M,K,C) = H(M \mid K,C)+H(K,C) = H(K,C).$$

Then

$$H(K \mid C) = H(K,C)-H(C) = H(M,K,C)-H(C) = H(M,K)-H(C) = H(M)+H(K)-H(C).$$

The proof is complete. □

4.4 The redundancy of natural languages

If $\mathbf{L}$ is a language, denote by L the random variable where the values are the letters of the language, and the probabilities are the relative frequencies. Set L^2 for the bigrams, L^3 for the trigrams and so on. Then the entropy of the language is defined to be

$$H(\mathbf{L}) = \lim_{n\to\infty} \frac{H(L^n)}{n}.$$

Proposition 4.4.1. *For any* $\mathbf{L}$*,* $H(\mathbf{L})$ *exists.*

Proof. For any $m,n \in \mathbf{N}$, we have

$$\frac{H(L^{nm})}{nm} \leqslant \frac{H(\overbrace{L^n,\ldots,L^n}^{m \text{ many}})}{nm} = \frac{H(L^n)}{n}.$$

Now for a fixed n, and letting $m \to \infty$, introducing $m = nc_m + d_m$, where $0 \leqslant d_m < n$,

$$\frac{H(L^m)}{m} = \frac{H(L^{nc_m+d_m})}{m} \leqslant \frac{H(L^n)}{n} + o(1).$$

Clearly the sequence $(H(L^n)/n)_{n\in\mathbf{N}}$ is bounded (each element is between 0 and $H(L)$). Assume

$$S = \limsup_{n\to\infty} \frac{H(L^n)}{n}, \qquad I = \liminf_{n\to\infty} \frac{H(L^n)}{n}$$

are different. Fix n such that $H(L^n)/n < I + (S-I)/3$. Now take m's tending to infinity such that $H(L^m)/m > S - (S-I)/3$. This obviously gives a contradiction, when $o(1)$ in the above calculation gets smaller than $(S-I)/3$. □

Experimentally, it seems that

$$1 \leqslant H(\text{English}) \leqslant 1.5.$$

Note also that

$$\log_2 26 \approx 4.7.$$

This means that the English language (like any other natural language) admits a high redundancy: everything which is expressed, could be expressed on less than one third of the used space. This explains the phenomenenon that when you apply cryptanalysis to a simple substitution cipher (recall Section 1.2), you do not have to decrypt all the characters, the plaintext becomes readable earlier.

Chapter 5
Elliptic curves and cryptography

5.1 Elliptic curves and their abelian group structure

Given a field $\mathbf{F}$ of characteristic bigger than 3, and elements $A, B \in \mathbf{F}$ such that $4A^3 + 27B^2 \neq 0$, the elliptic curve defined by them is defined on the projective plane is

$$E = \{[X,Y,Z] \in \mathbf{P}^2(\mathbf{F}) : Y^2Z = X^3 + AXZ^2 + BZ^3\}.$$

On the subset $\{Z \neq 0\}$, this is the set (with affine coordinates $x = XZ^{-1}$, $y = YZ^{-1}$)

$$E_{Z\neq 0} = \{(x,y) \in \mathbf{F}^2 : y^2 = x^3 + Ax + B\},$$

and when $Z = 0$, then automatically $X = 0$, giving

$$E_{Z=0} = \{[0,1,0]\},$$

from now on, this point $[0,1,0]$ will be denoted by $\mathscr{O}$.

Given two different points $P = [X_P, Y_P, Z_P]$ and $Q = [X_Q, Y_Q, Z_Q]$ on E, they determine a line passing through both of them, namely, when $P \neq Q$, then

$$L_{P,Q} = \{\lambda_1[X_P, Y_P, Z_P] + \lambda_2[X_Q, Y_Q, Z_Q] : [\lambda_1, \lambda_2] \in \mathbf{P}^1(\mathbf{F})\},$$

while if $P = Q$, then it is the tangent line of E at P. To understand this tangent line, let us use the affine coordinates. On $E_{Z\neq 0}$, at a point $(x,y) \in E$ (assume for a moment that $y \neq 0$),

$$\frac{d}{dx}y^2 = \frac{d}{dx}(x^3 + Ax + B), \qquad \frac{2ydy}{dx} = 3x^2 + A, \qquad \frac{dy}{dx} = \frac{3x^2 + A}{2y}.$$

This gives that the tangent line at $(x_P, y_P) \in E_{Z\neq 0, Y\neq 0}$ is

$$\left\{(x,y) \in \mathbf{F}^2 : y = \frac{3x_P^2 + A}{2y_P}x - \frac{3x_P^2 + A}{2y_P}x_P + y_P\right\}.$$

Taking the projectivization, we obtain the line

$$\left\{[X,Y,Z] \in \mathbf{P}^2(\mathbf{F}) : 2y_PY = (3x_P^2 + A)X + ((-3x_P^2 - A)x_P + 2y_P^2)Z\right\},$$

and we see that this definition gives the tangent line even for $y_P = 0$: it contains the point $[X_P, Y_P, Z_P]$, and also any point with coordinates $[X_P, Y, Z_P]$, which is just the vertical line on the xy-plane (and this is the tangent line, when the slope is infinity, i.e. when we divide by $y_P = 0$ above).

At the point $\mathscr{O}$, we see that the tangent line is $\{Z = 0\}$: indeed, setting $Z = 0$, we obtain the equation $0 = X^3$, which has a triple root at $X = 0$, so $\mathscr{O}$ is a triple intersection point of E and $L_{\mathscr{O},\mathscr{O}}$.

All in all, given any two point $P, Q \in E$, we can attach a line $L_{P,Q}$ passing through them, and if $P = Q$, then further being tangent to E.

We claim this line intersects E in a further point of E. We go by cases.

If $P = Q = \mathscr{O}$, then $\mathscr{O}$ is the third intersection point (recall that $\mathscr{O}$ is a triple intersection of E and $L_{\mathscr{O},\mathscr{O}}$).

If $P = \mathscr{O}$, and $Q = [X_Q, Y_Q, Z_Q] \neq \mathscr{O}$, then we have two subcases. If $Y_Q \neq 0$, then $[X_Q, -Y_Q, Z_Q]$ is the third intersection point. If $Y_Q = 0$, then $L_{\mathscr{O},Q}$ is the tangent line at Q, so the third intersection point is Q itself.

If $P = [X_P, Y_P, Z_P] \neq \mathscr{O}, Q = [X_Q, Y_Q, Z_Q] \neq \mathscr{O}$, and $x_Q = x_P, y_Q = -y_P$ in the affine notation (including the case $Y_P = Y_Q = 0$, i.e. $P = Q$), then $L_{P,Q}$ is a vertical line on the xy-plane, which intersects E further in $\mathscr{O}$.

Now assume $P = [X_P, Y_P, Z_P] \neq \mathscr{O}, Q = [X_Q, Y_Q, Z_Q] \neq \mathscr{O}$, and their x-coordinates (on the xy-plane) are different. Then the line passing through them is $y = mx + c$ for some $m, c \in \mathbf{F}$. Then

$$(mx + c)^2 = x^3 + Ax + B$$

has two roots x_P, x_Q (counted with multiplicity, when $P = Q$). Then factoring them out, we obtain a third solution, giving rise to a third intersection point.

Then we may define an addition on E. Given two points $P, Q \in E$, take the third intersection point of $L_{P,Q}$ and E (call it R), and then the third intersection point of $L_{R,\mathscr{O}}$ and E. This will be the point $P + Q$.

Theorem 5.1.1. *Under this addition, the points of E form an abelian group with unit element $\mathscr{O}$.*

One can easily check that the third intersection point R is nothing else but $-P - Q$.

5.2 A sketch of the proof of Theorem 5.1.1

In this section, we are goint to sketch the proof of Theorem 5.1.1. Commutativity, unit element and invertibility are easy to see, the difficult (and deep) part is the associativity. To this aim, we assume that the underlying field $\mathbf{F}$ is algebraically closed: we are free to do this, since the points are always defined as third intersection points, and if two intersection points are in a certain subfield, then so is the third one.

5.2.1 The resultant and Bézout's theorem

For fixed $n \in \mathbf{N} \cup \{0\}$, denote by $\mathscr{P}_n \subset \mathbf{F}[Z]$ the vector space of polynomials of degree smaller than n (completed with the zero polynomial). Define the linear map

$$\rho(A, B) = PA + QB, \qquad \rho : \mathscr{P}_n \times \mathscr{P}_m \to \mathscr{P}_{m+n},$$

where $P(Z) = a_m Z^m + \ldots + a_1 Z + a_0$ and $Q(Z) = b_n Z^n + \ldots + b_1 Z + b_0$ are fixed polynomials (with $a_m, b_n \neq 0$). Taking the basis $(Z^{n-1}, \ldots, Z, 1, Z^{m-1}, \ldots, Z, 1)$ in $\mathscr{P}_n \times \mathscr{P}_m$ and the basis $(Z^{m+n-1}, \ldots, Z, 1)$ in $\mathscr{P}_{m+n}$, we see that the matrix of ρ is

$$S = \begin{pmatrix} a_m & a_{m-1} & \ldots & a_0 & 0 & \ldots & 0 \\ 0 & a_m & \ldots & a_1 & a_0 & \ldots & 0 \\ & & & \ldots & & & \\ 0 & \ldots & 0 & a_m & a_{m-1} & \ldots & a_0 \\ b_n & b_{n-1} & \ldots & b_0 & 0 & \ldots & 0 \\ 0 & b_n & \ldots & b_1 & b_0 & \ldots & 0 \\ & & & \ldots & & & \\ 0 & \ldots & 0 & b_n & b_{n-1} & \ldots & b_0 \end{pmatrix}.$$

Now ρ is singular if and only if $\det S = 0$. Singularity is also equivalent to that P and Q share a common root: if $\deg \gcd(P, Q) > 0$, then ρ is not surjective (since $\deg(\gcd(P, Q))$ divides all elements of $\rho(\mathscr{P}_n \times \mathscr{P}_m)$), while if $\deg \gcd(P, Q) = 0$, then ρ is surjective (since all elements of $\mathscr{P}_{m+n}$ is a polynomial combination of P and Q).

Therefore, $\det S$ is zero if and only if P and Q share a common root. This quantity $\det S$ is called the resultant of P and Q.

Proposition 5.2.1. *Let A be a unique factorization domain, and assume $f, g \in A[x]$ are nonzero polynomials. Then f and g have a common nonconstant factor if and only if the equation $uf + vg = 0$ has a nontrivial solution such that $\deg v < \deg f$ and $\deg u < \deg g$.*

Proof. First, if f and g share a nonconstant factor $h \in A[x]$. Then setting $(g/h)f + (-f/h)g = 0$, and obviously $\deg(f/h) < \deg f$, $\deg(-g/h) < \deg g$. For the converse, take an admissible pair (u, v), then each irreducible factor of f divides vg. Since $\deg v < \deg f$, there must be an irreducible factor of f which appears as a divisor of g. □

Theorem 5.2.2 (Bézout)**.** *Assume $f,g \in \mathbf{F}[x,y]$ are polynomials such that their degrees are m,n, respectively. If f,g do not share common factors, then they have at most mn common zeros.*

Proof. Assume the corresponding projectivizations are $F(X,Y,Z) = a_0Z^m + \ldots + a_{m-1}Z + a_m$, and $G(X,Y,Z) = b_0Z^n + \ldots + b_{n-1}Z + b_0$, where a_j, b_j are the degree j part of f,g, respectively. It is easy to see that no common factor of f,g implies the same for F,G. Then there are no nontrivial solutions of $uF + vG = 0$ such that $\deg v < \deg F$, $\deg u < \deg G$. This means that the resultant S made of the coefficients $a_0,\ldots,a_m,b_0,\ldots,b_n$ (note that these are now polynomials) is not constant zero. Then $\deg\det S \leqslant mn$, and by the fundamental theorem of algebra, we may factorize $\det S$ as $\prod(\xi x - \zeta y)$, and the number of factors is at most mn. Each such factor gives at most $\max(m,n)$ possibilities for Z, so there are finitely many common zeros $[X:Y:Z]$. Changing coordinates, we may choose a base point which is not a common zero, neither lies on a line connecting to common zeros (since $\mathbf{F}$ is infinite). With this base point as $[0:0:1]$, and repeating the above argument, each factor $(\xi x - \zeta y)$ gives at most one Z. □

5.2.2 The Cayley-Bacharach theorem

If $F(X,Y,Z)$ is a homogeneous polynomial of degree 3 over the base field $\mathbf{F}$, then its zero set $\gamma = \{[X:Y:Z] \in P\mathbf{F}^2\}$ is called a cubic curve.

Theorem 5.2.3 (Cayley-Bacharach)**.** *Assume γ_1 and γ_2 are two cubic curves sharing exactly nine intersection points $P_1,\ldots,P_9$. Then if γ is a further cubic curve passing thorugh the first eight intersection points $P_1,\ldots,P_8$, then γ is a linear combination of γ_1 and γ_2 (by this we mean that if γ_1,γ_2,γ are the vanishing sets of F_1,F_2,F, respectively, than for some $\lambda,\mu \in \mathbf{F}^2$, $\lambda F_1 + \mu F_2 = F$). In particular, γ passes through P_9.*

Proof. We prove by contradiction, assume F_1,F_2 are given as in the statement, and F is a degree 3 homogeneous polynomial vanishing at $P_1,\ldots,P_8$, which is not a linear combination of F_1 and F_2.

First we claim that no four of $P_1,\ldots,P_9$ lie on any line l (the vanishing set of the homogeneous degree 1 polynomial L), since by Bézout's theorem, this would mean that this L is a factor of F_1,F_2, leading to infinitely many common points of γ_1,γ_2. Similarly, no seven of $P_1,\ldots,P_9$ can lie on any conic q (the vanishing set of the homogeneous degree q polynomial Q), since by Bézout's theorem, this would mean that this Q is a factor of F_1,F_2, leading to infinitely many common points of γ_1,γ_2.

Then we claim that any five of $P_1,\ldots,P_9$ determine a unique conic passing through them. First, for the existence, observe that a conic has an equation

$$aX^2 + bXY + cZ^2 + dXZ + eYZ + fZ^2 = 0,$$

with not all of a,b,c,d,e,f being zero. Observe that fixing a point is nothing else but a linear condition on the coefficients a,b,c,d,e,f. Giving five such conditions still leaves a nonzero solution. Now assume there are two conics q_1,q_2 passing through the same five points. By Bézout's theorem, the corresponding polynomials Q_1,Q_2 must share a common factor L (with vanishing set l, a line). This line l can contain at most three of the points, so the line containing the remaining two points must be another common factor of Q_1,Q_2, leading to $q_1 = q_2$.

Case 1: three of the first eight points, say, P_1,P_2,P_3 are collinear. Let the line containing them be l. Let q be the conic containing $P_4,\ldots,P_8$. Now let X be a further point on l, and Y be a further point not on either l or q. Since F,F_1,F_2 are linearly independent, there is a nonzero linear combination of them $G = aF + bF_1 + cF_2$ which vanishes at both X and Y. Then the vanishing set of G passes through $P_1,\ldots,P_8,X$, so it must be the union of l and q (because it has 4 common points with l, and Bézout applies: $4 > 3\cdot 1$; so the vanishing set of G must be the union of l and a conic, and this conic must be q, because of 5 common points. However $Y \notin l \cup q$, which is a contradiction.

Case 2: Case 1 does not hold, and six of the first eight points, say, $P_1,\ldots,P_6$ are on a conic. Let the conic containing them be q. Let l be the line containing P_7,P_8. Now let X be a further point on q, and Y be a further point not on either q or l. Since F,F_1,F_2 are linearly independent, there is a nonzero linear combination of them $G = aF + bF_1 + cF_2$ which vanishes at both X and Y. Then the vanishing set of G passes through $P_1,\ldots,P_8,X$, so it must be the union of l and q (because it has 7 common points with q, and Bézout applies: $7 > 3\cdot 2$; so the vanishing set of G must be the union of q and a line, and this conic must be l, because of 2 common points). However $Y \notin q \cup l$, which is a contradiction.

Case 3: none of Case 1 and Case 2 holds. Let l be the line passing through P_1,P_2, and q the conic passing through $P_3,\ldots,P_7$. In this case then, by assumption, $P_8 \notin l \cup q$. Let X and Y be further points on l but not on q. Since F,F_1,F_2 are linearly independent, there is a nonzero linear combination of them $G = aF + bF_1 + cF_2$ which

vanishes at both X and Y. Then, as above, the vanishing set of G must be $l \cup q$. The contradiction follows from that on the one hand, G vanishes at P_8 (since it is a linear combination of F, F_1, F_2), but on the other hand, $P_8 \notin l \cup q$.

The proof is complete. □

5.2.3 Completion of the sketch

Assume P, Q, R are points on the elliptic curve, and assume the points $\mathcal{O}, P, Q, R, P+Q, Q+R, -(P+Q), -(Q+R), -((P+Q)+R)$ are all different. Now consider the following cubics c_1, c_2, c_3. Let $c_1 = E$ itself. Let

$$\begin{aligned} c_2 = &\text{line passing through } P \text{ and } Q \text{ and } -(P+Q) \\ &\cup \text{line passing through } P+Q \text{ and } R \text{ and } -((P+Q)+R) \\ &\cup \text{line passing through } \mathcal{O} \text{ and } Q+R \text{ and } -(Q+R). \end{aligned}$$

Finally, let

$$\begin{aligned} c_3 = &\text{line passing through } Q \text{ and } R \; -(Q+R) \\ &\cup \text{line passing through } P \text{ and } Q+R \\ &\cup \text{line passing through } \mathcal{O} \text{ and } P+Q \text{ and } -(P+Q). \end{aligned}$$

One can prove that c_1 and c_2 have exactly 9 common points (by the condition, it follows that E cannot contain a line, so by Bézout's theorem, the number of intersection points is at most 9), and c_3 passes through 8 of them. Then it passes through $-((P+Q)+R)$ as well, so it has to be equal to $-(P+(Q+R))$.

Of course, there can be many coincidences among the points. Having some topology in hand (e.g. over the complexes), this can be handled by continuity. As soon as we have the theorem in hand for the complex field, we might say that the intersection (or tangent) points are computed from algebraic formulae, and the coincidence giving associativity must be a formal coincidence, which then must hold in all fields.

5.3 The elliptic curve discrete logarithm problem

Letting $\mathbf{F}_p$ the prime field for some $p > 3$, we may consider an elliptic curve defined by the equation $y^2 = x^3 + Ax + B$ over it (with the discriminant condition $4A^3 + 27B^2 \neq 0$).

Let us estimate the number of points on E. There is one point at infinity, so restrict to the solutions of $y^2 = x^3 + Ax + B$ in $\mathbf{F}^2$. Now for any x, $x^3 + Ax + B$ is either square or not, and it is a square by probability $\approx 50\%$. This is because

$$\mathbf{F}^\times \ni x \mapsto x^2 \in \mathbf{F}^{\times 2}$$

is a two-folded cover of the quadratic residues (indeed, $x^2 = (-x)^2$, and for any $a \in F$, $x^2 - a$ has at most two solutions). Since $x^3 + Ax + B = 0$ has at most three solutions, we may say that essentially for half of the possible x's, there are two y's, so the number of points on E should be around $\#F$.

A deep theorem of Hasse tells us that this is the truth.

Theorem 5.3.1 (Hasse)**.** *We have*

$$|\#\mathbf{F} + 1 - \#E| \leqslant 2\sqrt{\#\mathbf{F}}.$$

Proof omitted.

As we have seen earlier, from any x, the value $x^3 + Ax + B \bmod p$ can be computed in polynomial time. To get a point on E, we should compute its squarer-root. Using random algorithms, this can be done fast with high probability, but if $p \equiv 3 \bmod 4$, then we also have a simple deterministic way. We will return to this question later.

Using this, we may find points of E by random methods: pick a random x, compute $x^3 + Ax + B$, compute its $(p+1)/4$th power, and check if its square is $x^3 + Ax + B$ or not. With high probability, after a few trials, we find a point $P \in E$. Having some points P and Q in hand, it is easy to compute $P + Q$ (the coefficients determining $L_{P,Q}$ come from simple algebraic manipulations, leading to $R = -P - Q$, then $L_{R,\mathcal{O}}$ and $P + Q$ come similarly).

Given a point $P \in E$ on an elliptic curve, we may consider

$$nP = \underbrace{P + \ldots + P}_{n \text{ many}} \in E.$$

Then nP can be computed in polynomial time with the same trick we applied in the proof of Proposition 1.5.50. Indeed, write

$$n = \sum_{0 \leqslant j \leqslant 1+\lceil \log_2 n \rceil} \varepsilon_j 2^j, \qquad \varepsilon_j \in \{0,1\}.$$

Then $P, 2P = P+P, 4P = 2P+2P, \ldots, 2^{1+\lceil \log_2 n \rceil}P = 2^{\lceil \log_2 n \rceil}P + 2^{\lceil \log_2 n \rceil}P$ can be computed in polynomial time, so is their weighted sum (with weights ε_j).

The elliptic curve discrete logarithm problem (ECDLP from now on), is the following: given $P, Q \in E$, provided that $Q = nP$ for some $n \in \mathbf{N}$, compute the smallest such n. Since even the DLP for multiplicative groups over prime fields seems to be a computationally difficult problem, and the group operation in an elliptic curve is much more complicated than that of multiplicative groups of prime fields, we expect that the ECDLP is also computationally difficult. So far, this seems to be the truth, there is no known algorithm which would solve the problem in fewer than $O(\sqrt{p})$ steps (and this can be achieved by a meet-in-the-middle attack, using the collision phenomenon).

5.4 Elliptic curve cryptography

5.4.1 The elliptic curve Diffie-Hellman

Alice and Bob agree on a prime p, an elliptic curve E, and a point $P \in E$. They make them public (they can agree on it publicly). Now Alice chooses a number n_A and Bob chooses a number n_B, and they keep them in secret. Then Alice sends $Q_A = n_A P$ to Bob, and Bob sends $Q_B = n_B P$ to Alice. Now Alice computes $n_A Q_B$, and Bob computes $n_B Q_A$, and this will be their secret key. Observe that

$$n_A Q_B = n_A n_B P = n_B n_A P = n_B Q_A,$$

so they have the same point $n_A n_B P \in E$ in hand. However, Eve known only $n_A P$ and $n_B P$, and from this, she should compute somehow $n_A n_B P$, i.e. she should solve the Diffie-Hellman problem over elliptic curves.

5.4.2 The elliptic curve ElGamal

It starts similarly, $p, E, P \in E$ are fixed and public. Now Alice chooses a number n_A and keeps it in secret. Now she computes $Q = n_A P$ and makes it public. If Bob wants to send a message $M \in E$ to Alice, he chooses an ephemeral key $k \in \mathbf{N}$ and computes the following two points:

$$C_1 = kP, \qquad C_2 = M + kQ.$$

Then Alice computes $C_2 - n_A C_1$, obtaining

$$C_2 - n_A C_1 = M + kQ - n_A kP = M + kQ - kQ = M.$$

For Eve, to break the cipher in general, should solve a problem not easier than the elliptic curve Diffie-Hellman problem (recall Proposition 2.3.1).

Chapter 6
Attacking the underlying problems

6.1 The discrete logarithm problem

6.1.1 A babystep-giantstep algorithm

Proposition 6.1.1. *Assume g generates the abelian group G of order N. Then the DLP $g^x = h$ can be solved in $O(\sqrt{N})\,\mathrm{Pol}(\log N)$ steps.*

Proof. For $n \geqslant \sqrt{N}+1$ Make the following two lists:

$$1, g, \ldots, g^n, \qquad h, hg^{-n}, \ldots, hg^{-n^2}.$$

We claim that the two lists have a common element. Ineed, the solution x satisfies $x = nq + r$ for some $0 \leqslant q, r \leqslant n$. Then $h = g^{qn+r}$, which implies $h^{-qn} = h^r$.

Therefore, the two lists indeed intersect nontrivially. Assume hence $g^i = hg^{-jn}$ for some $0 \leqslant i, j \leqslant n$. Then $x = i + jn$ is a solution to the DLP $g^x = h$. □

6.1.2 The Pohlig-Hellman algorithm

The following proposition reduces the DLP in a group to the DLP in its Sylow subgroups.

Proposition 6.1.2. *Assume G is a cyclic group of order N, where $N = \prod_{j=1}^{r} p_j^{\alpha_j}$. Assume we can solve the DLP in $O(S(p^\alpha))$ time for any element, whose order is p^α. Then for G, we can solve the DLP in $O(\sum_{j=1}^{r} S(p_j^{\alpha_j}))\,\mathrm{Pol}(\log N)$ time.*

Proof. For any $1 \leqslant j \leqslant r$, set $g_j = g^{N/p_j^{\alpha_j}}$, and $h_j = h^{N/p_j^{\alpha_j}}$. Now for any $1 \leqslant j \leqslant r$, solve the problem

$$g_j^{y_j} = h_j$$

in $O(S(p_j^{\alpha_j}))$ time.

Then in $O(\log N)$ time, using the Chinese remainder theorem (Corollary 1.5.39), we get

$$x \equiv y_j \bmod p_j^{\alpha_j} \qquad \text{for each } 1 \leqslant j \leqslant r.$$

Then, for any $1 \leqslant j \leqslant r$, for some $z_j \in \mathbf{Z}$, $x = y_j + p_j^{\alpha_j} z_j$, and then

$$(g^x)^{N/p_j^{\alpha_j}} = \left(g^{y_j + p_j^{\alpha_j} z_j}\right)^{N/p_j^{\alpha_j}} = h_j = h^{N/p_j^{\alpha_j}}.$$

This means that

$$\frac{N}{p_j^{\alpha_j}} x \equiv \frac{N}{p_j^{\alpha_j}} \log_g(h) \bmod N,$$

which implies

$$x \equiv \log_g(h) \bmod p_j^{\alpha_j}.$$

Since this holds for any $1 \leqslant j \leqslant r$, $x \equiv \log_g(h)$ modulo N. □

Further, the prime power groups can be reduced to prime groups.

Proposition 6.1.3. *Assume G is a cyclic group on p^e elements. Assume we can solve the DLP in $O(S(p))$ time for any element, whose order is p. Then in G, we can solve the DLP in $O(eS(p))\mathrm{Pol}(e\log p)$ time.*

Proof. Assume we have to solve $g^x = h$. We know that for some $0 \leqslant x_0, x_1, \ldots, x_{e-1} \leqslant p-1$

$$g^{x_0 + x_1 p + \ldots + x_{e-1} p^{e-1}} = h.$$

Then raising to power p^{e-1},

$$(g^{p-1})^{x_0} = h^{p-1},$$

and we obtain x_0 in $O(S(p))$ steps. Recursively, assume $x_0, x_1, \ldots, x_{j-1}$ are already determined. Then

$$g^{x_j p^j + \ldots + x_{e-1} p^{e-1}} = h g^{-x_0 - x_1 p - \ldots - x_{j-1} p^{j-1}},$$

and raising to power p^{e-1-j}, we obtain

$$(g^{p-1})^{x_j} = \left(h g^{-x_0 - x_1 p - \ldots - x_{j-1} p^{j-1}}\right)^{p-1},$$

and x_j is computed in $O(S(p))$ time. □

6.1.3 The index calculus method

In this section, we present the index calculus method, which solves the DLP in the multiplicative group of $\mathbf{F}_p$.

First of all, fix a parameter B, and solve the discrete logarithm problem for any prime $q \leqslant B$, i.e. compute $\log_g(q)$. How to do this? For some random numbers $1 \leqslant i \leqslant p-1$, compute

$$g_i \equiv g^i \bmod p.$$

If g_i has a prime factor bigger than B, then discard it, otherwise factorize it as

$$g_i = \prod_{\substack{q \leqslant B \\ q \text{ prime}}} q^{u_{q,i}}.$$

Then by Euler-Fermat (Corollary 1.5.46),

$$i \equiv \sum_{\substack{q \leqslant B \\ q \text{ prime}}} u_{q,i} \log_g(q) \bmod p-1.$$

Here, if the number of congruences is large enough, then we may be able to solve this congruence system for $\log_g(q)$ (be careful: $\mathbf{Z}_{p-1}$ is not a field).

Now in the DLP $g^x \equiv h \bmod p$, compute, for $k = 1, 2, \ldots$, the value $hg^{-k} \bmod p$ until we arrive at a value which has only prime factors not exceeding B. Then

$$hg^{-k} = \prod_{\substack{q \leqslant B \\ q \text{ prime}}} q^{e_q},$$

which immediately gives

$$\log_g(h) = k + \sum_{\substack{q \leqslant B \\ q \text{ prime}}} e_q \log_g(q).$$

Of course, the algorithm depends on the choice of the parameter B. For any $0 \leqslant \varepsilon \leqslant 1$, introduce the notation

$$L_\varepsilon(X) = e^{(\log X)^\varepsilon (\log\log X)^{1-\varepsilon}}.$$

It turns out that to find enough congruences, B has to run up to $L_{1/2}(p)^{1/\sqrt{2}}$ which altogether gives rise to a subexponential algorithm (which is still far worse than polynomial, but much better than exponential). With further improvements, the running time can be decreased to $L_{1/3}(p)$ (still subexponential).

6.2 Factorization algorithms

6.2.1 Pollard's $p-1$ method

When attacking RSA, there is a given large number $N = pq$, where we do not know the prime factors p and q.

Starting out from a small number a, say, $a = 2$, consider the following sequence

$$\gcd\left(a^{1!}-1,N\right), \qquad \gcd\left(a^{2!}-1,N\right), \qquad \gcd\left(a^{3!}-1,N\right), \qquad \dots$$

If $n!$ in the exponent is not too large, then these numbers can be computed in a reasonable time. Now assume that for some not too large n, $(p-1)|n!$. Then, by Euler-Fermat (Corollary 1.5.46),

$$a^{n!} = (a^{p-1})^{n!/(p-1)} \equiv 1^{n!/(p-1)} = 1 \bmod p.$$

If we are lucky enough that q does not divide $a^{n!}-1$, then $\gcd(a^{n!}-1,N)$ will be p, which gives the factorization of N.

How difficult it is to compute $\gcd(a^{n!}-1,N)$? First, $a^{n!}$ is uncomputably large (with current tools) even for $n = 200$. However, we need it only modulo N, and the modulo N powering can be done fast, think of Proposition 1.5.50. Even better, for an actual calculation, we may get use of the identity

$$a^{(n+1)!} = (a^{n!})^{n+1},$$

so when we are interested in $a^{(n+1)!}$ (modulo N), we only have to raise $a^{n!}$ (modulo N) to power $n+1$ (modulo N). Altogether, this means that even $n \approx \log N$ is an acceptable number for the number of trials in a polynomial algorithm.

This means that prime numbers p,q are insecure for the aim of RSA, if one of $p-1$ and $q-1$ has only small prime factors.

6.2.2 Lenstra's elliptic curve factorization

Given N to be factorized, consider an elliptic curve $y^2 = x^3 + Ax + B$ modulo N. For the first sight, this seems nonsense, since N is not a prime, so $\mathbf{Z}_N$ is not a field. Nevertheless, when computing the sum of two points, we just add, subtract, multiply and divide, and all these make sense, at least if the number we divide by is coprime to N.

For a point $P \in E$, its multiples

$$P, \qquad 2P, \qquad 3P, \qquad \dots$$

can be computed in most cases. What happens, when we cannot compute nP for some $n \in \mathbf{N}$? It means that during its computation $(n-1)P+P$, we draw the line of slope $(y_{(n-1)P}-y_P)(x_{(n-1)P}-x_P)^{-1}$, and this slope does not make sense, meaning that $x_{(n-1)P}-x_P$ is not coprime to N. Assuming that it is divisible by p, this means that $nP = \mathcal{O}$ when considered over $\mathbf{F}_p$.

This gives us an algorithm: take a point P on E, and compute

$$P, \qquad 2!P, \qquad 3!P, \qquad \dots.$$

When $n!$ gets divisible by the order of P modulo p, the resulting point is $\mathcal{O}$ over $\mathbf{F}_p$. If we cannot perform the calculation, it means that we tried to divide by a number (the difference of two x-coordinates) not coprime to N, that is, we have some $d > 1$ such that $d \mid N$. Computing $\gcd(d,N)$, we either find a proper divisor of N, or we find N itself. In the former case, we are done, in the latter one, we pick a new curve and a new point.

Again, an actual computation can be made faster by utilizing

$$(n+1)!P = (n+1)\cdot n!P.$$

There is a technical subtlety in the choice of the curve and the point on it. Earlier, we agreed that given a curve, a random trial gives a good x-coordinate by 50% chance, and then computing the square root is easy. However, this is not true now, since we do not know the factorization of N: if we compute x^3+Ax+B, we cannot tell if it a square. Instead, choose the point P first, and adjust the elliptic curve: pick a random A, then set B such that $y^2 = x^3+Ax+B$ holds for P.

Chapter 7
Additional topics

In this chapter, our main source is [2].

7.1 Interactive proofs

7.1.1 How to store the last move in chess?

Assume now Alice and Bob play chess via phone (they tell each other their moves using the chessboard coordinates). After some time, they decide to pend the game. In tournament chess, the player on move writes her/his move to a piece of paper, and gives it to the arbiter in a closed envelope. The next day the arbiter opens the envelope and makes the assigned move on the board, then the game continues. The goal is to deprive both players from clear advantages over each other[1].

Of course, Alice and Bob could go for a Trusted Authority and use the same scheme. Nevertheless, they can solve this problem on their own. Alice generates two large primes p,q such that the first few, say, ten digits of p (which is the smaller prime by their agreement) encrypts her move. Then she sends the product $N = pq$ to Bob. Bob is unable to read out Alice's move, but the next day, when they continue their game, Alice tells the factorization. Of course, Bob checks it, so that Alice cannot alter her move.

This way it is possible to create "electronic envelopes": Alice can put a certain piece of information in a deposit, which cannot be altered even by herself, but cannot be read by anyone else until she lets them.

7.1.2 A zero-knowledge proof of that a certain number is square modulo N

First assume p and q are large prime numbers, which Peggy (the prover) keeps in secret. She publishes their product $N = pq$, and also claims that a certain residue class y mod N is a square modulo N, i.e. there exists a residue class x mod N satisfying $x^2 \equiv y \bmod N$. Her goal is to prove this to Victor (the verifier) without telling him an actual square root. Of course, she could in principle tell the prime factors, but doing so would actually tell the square roots.

Proposition 7.1.1. *Assume p is an odd prime number, and* $\gcd(a,p) = 1$. *Then*

$$a^{\frac{p-1}{2}} \equiv \begin{cases} 1 \bmod p, \text{ if } a \text{ is a square modulo } p, \\ -1 \bmod p, \text{ if } a \text{ is not a square modulo } p. \end{cases}$$

Proof. Assume first that $a \equiv b^2 \bmod p$ for some resiude class b modulo p. By Euler-Fermat (Corollary 1.5.46), we have

$$a^{\frac{p-1}{2}} \equiv (b^2)^{\frac{p-1}{2}} \equiv b^{p-1} \equiv 1.$$

Assume then a is not a square modulo p. Again, by Euler-Fermat (Corollary 1.5.46), we certainly have

$$\left(a^{\frac{p-1}{2}}\right)^2 \equiv 1 \bmod p,$$

[1] In competitive chess, there are essentially no pending games any more, because of the fast development of engines in the last few decades.

and we see that $a^{(p-1)/2}$ is a root of the polynomial x^2-1 over $\mathbf{F}_p$, hence it must be ± 1. Therefore, it suffices to exclude $a^{(p-1)/2} \equiv 1 \bmod p$ in this case. Observe that $x \mapsto x^2$ is a two-folded cover from $\mathbf{F}_p^\times$ to the nonzero squares: indeed, if $x_1^2 \equiv x_2^2 \bmod p$, then

$$0 \equiv x_1^2 - x_2^2 \equiv (x_1 - x_2)(x_1 + x_2) \bmod p, \qquad x_1 \equiv \pm x_2 \bmod p.$$

Then the number of nonzero square residue classes is exactly $(p-1)/2$. Then consider the polynomial $x^{(p-1)/2}-1$ over $\mathbf{F}_p$. It has at most $(p-1)/2$ roots, since the number of roots cannot exceed the degree (over a field), and recall that we have already found that many roots: namely, the nonzero squares. This means that nonsquare residue classes are not roots, so $a^{(p-1)}/2$ cannot be 1 modulo p. □

Proposition 7.1.2. *If p is a prime such that $p \equiv 3 \bmod 4$, and $a \in \mathbf{N}$ is coprime to p such that it is a square modulo p, then $a^{(p+1)/4}$ is a square root of a.*

Proof. We have the following simple calculation (using Proposition 7.1.1):

$$\left(a^{\frac{p+1}{4}}\right)^2 \equiv a^{\frac{p+1}{2}} \equiv a \cdot a^{\frac{p-1}{2}} \equiv a \bmod p,$$

and the proof is complete. □

This already tells us how to compute the square root modulo a prime number congruent to 3 modulo 4. When a prime number is 1 modulo 4, we may use Cipolla's algorithm, which relies on the following proposition.

Proposition 7.1.3. *Assume p is an odd prime, and a is coprime to p such that it is a square modulo p. Choose $b \in \mathbf{F}_p^\times$ such that $b^2 - a$ is not a square in $\mathbf{F}_p$. Then*

$$\left(b + \sqrt{b^2-a}\right)^{\frac{p+1}{2}}$$

is a square root of a, where the element $b + \sqrt{b^2-a}$ is understood in the field $\mathbf{F}_{p^2}$.

Proof. Set $\omega = \sqrt{b^2-a}$. We will calculate in the field $\mathbf{F}_{p^2} = \mathbf{F}_p(\omega)$. Then the Frobenius automorphism $\sigma : x \mapsto x^p$ maps ω to $-\omega$: indeed, $\sigma(\omega)^2 = \sigma(\omega^2) = \sigma(b^2-a) = b^2-a$ by Corollary 1.5.47, therefore, $\sigma(\omega)^2 = \pm\omega$ (its square is fixed under σ); and the polynomial $x^p - x$ has at most p roots, so σ cannot fix anything outside $\mathbf{F}_p$. Then

$$\left((b+\omega)^{\frac{p+1}{2}}\right)^2 = (b+\omega)^{p+1} = (b+\omega)^p(b+\omega) = (b-\omega)(b+\omega) = b^2 - \omega^2 = b^2 - (b^2-a) = a,$$

and the proof is complete. □

So when $p \equiv 1 \bmod 4$, we may compute the square root as follows: we pick random b's, apply Proposition 7.1.1 to decide if b^2-a is a square or not modulo p, until we arrive at a nonsquare (it is likely that this happens fast: note that we win by probability $1/2$ in each trial). This nonsquare (or rather, its square root in the bigger field) gives us a square root of a modulo p.

All in all, if Victor learns p and q, he can easily compure a square root x of $y \equiv x^2 \bmod N$.

To solve this problem, Peggy generates another square, $s \equiv r^2 \bmod N$, and sends s to Victor. Now Victor chooses a random bit $\beta \in \{0,1\}$ and sends it to Peggy. If $\beta = 0$, Peggy sends r to Victor, who checks $s \equiv r^2 \bmod N$. If $\beta = 1$, Peggy sends xr to Victor, who checks $ys \equiv (xr)^2 \bmod N$. Observe that in none of the cases Peggy tells too much information on x to Victor: indeed, when $\beta = 0$, then r has nothing to do with y; while if $\beta = 1$, the original square root x is perturbed by a random square root. This means basically that Peggy does not inform Victor about x. However, Victor still learns something, at least with some probability: if one of y and s is not a square, then Peggy fails his check with probability $1/2$. This is not very much convincing, but Victor can tell Peggy in advance: "I will challenge you one hundred times. In each challenge, give me another square s, and I will ask you to send me the square root of either s or ys, chosen at random. If you fail any of the challenges, I will refuse to believe that your number y is a square. If you pass all the challenges, I will believe that your number y is indeed a square, since the probability $1 - 2^{-100}$ is more than enough for me."

7.1.3 Using our password

When we withdraw cash from an ATM, we use our password, and the computer system of the bank checks it. There is a weak point in this procedure: if a hacker gets access to the files of the bank, (s)he can learn our passord.

Using the complexity of factorization, we may find a solution to this weakness. Assume our password is a large prime p. We also generate another large prime number q, and tell the bank their product $N = pq$ (and, as usual, we keep p, q in secret). The computer system of the bank stores only N, and when withdrawing cash, we tell p to the ATM, and the system checks if p divides N, but does not store p.

Now even if the hacker manages to get access to the files of the bank, and learns our number N, (s)he is still unable to use our password p without factorizing N.

What happens if we cannot trust that the bank indeed does not store our prime number p? Then after the first withdrawal, p is available for the bank – and also for the skillful hacker.

A version of the RSA can solve this problem, namely, instead of telling the bank the prime divisor p, we only prove to the bank that we know the prime factorization of N. To this aim, we tell the bank not only N, but also a number e, and at cash withdrawals, the following protocol can be applied. The ATM generates a random number $1 \leqslant x \leqslant N$, and tells us $y \equiv x^e \bmod N$. We can easily tell what the original x was: letting d be the multiplicative inverse of e (which is easily computable for us), $y^d \equiv x \bmod N$ (recall Section 3.1). Now the ATM checks if we really recovered x, and if yes, it accepts that we know the factorization of N.

7.2 Identification

There are certain situations when it is important for Alice to be able to check whether the incoming message have been really encrypted Bob. In this section we present a few situtions where such issues are addressed.

7.2.1 An RSA-based digital signature

In the following setup, we turn the RSA upside down. Now Bob chooses two large primes p, q and exponents d, e such that $de \equiv 1 \bmod \varphi(N)$, where $N = pq$. Then publishes N and e and keeps the remaining numbers in secret. If his message is $1 \leqslant m \leqslant N$, a number coprime to N, he computes $m^d \bmod N$, and sends it to Alice. Then Alice raises the incoming message to power e, obtaining

$$(m^d)^e \equiv m \bmod N,$$

recall Section 3.1. For a random number, its eth power is most likely a gibberish, so if Alice gets a meaningful message, she learns that it must have been sent by Bob indeed. This way Bob's m^d is not only a message, but it also serves as a personal identifier of him, an example of a digital signature.

7.2.2 Multiplayer RSA – the good way

In this section, we assume that there are n participants, each of them using a public key cryptosystem (let us index them by $1 \leqslant i \leqslant n$). To this aim, each of them chooses two large prime numbers p_i, q_i, and exponents d_i, e_i such that $d_i e_i \equiv \bmod \varphi(N_i)$, where $N_i = p_i q_i$. Now each of them publishes N_i, e_i.

Since the moduli are different, this time they cannot use residue classes, so the possible messages are positive integers. Now assume the ith participant wants to send the message $m \in \mathbf{N}$ to the jth participant. First of all, (s)he writes m in base N_i:

$$m = \sum_{k=0}^{t} m(i,k) N_i^k.$$

Now (s)he applies her/his own decryption function d_i to each digit $m(i,k)$. They altogether give rise to a certain number, say,

$$D_i(m) = \sum_{k=0}^{t} (m(i,k)^{d_i} \bmod N_i) N_i^k.$$

Now (s)he rewrites this number $D_i(m)$ in base N_j:

$$D_i(m) = \sum_{k=0}^{t} D_i(m)(j,k) N_j^k,$$

and applies the addressee's encryption function e_j to each digit $D_i(m)(j,k)$, giving, say

$$E_j(D_i(m)) = \sum_{k=0}^{t} ((D_i(m)(j,k))^{e_j} \bmod N_j) N_j^k.$$

Then the (s)he sends the number $E_j(D_i(m))$ to the addressee.

Now the recipient applies her/his own decryption function d_j to each digit, this gives back $D_i(m)(j,k)$, as usual:

$$((D_i(m)(j,k))^{e_j})^{d_j} \equiv D_i(m)(j,k) \bmod N_j,$$

at least when $\gcd(D_i(m)(j,k), N_j) = 1$, but this holds with high probability (at least when t is much smaller than all the p_i, q_i's, which we assume from now on). Now these $D_i(m)(j,k)$'s give rise to $D_i(m)$:

$$D_i(m) = \sum_{k=0}^{t} D_i(m)(j,k) N_j^k.$$

Finally, the recipient rewrites $D_i(m)$ in base N_i,

$$D_i(m) = \sum_{k=0}^{t} (m(i,k)^{d_i} \bmod N_i) N_i^k.$$

then applies sender's encryption function to each digit, giving back, as usual:

$$(m(i,k)^{d_i})^{e_i} \equiv m(i,k) \bmod N_i,$$

at least when $\gcd(m(i,k), N_i) = 1$, but this holds with high probability. Then

$$m = \sum_{k=0}^{t} m(i,k) N_i^k,$$

so the reader obtains the original message. But this is not the only outcome of the method. Simultaneously, the jth participant can be almost certain that the sender was the ith participant, so the method provides digital signatures as a by-product.

7.3 Secret sharing

The basic problem of secret sharing is the following: given some information, share certain pieces of it among n people such that when they get together, they can read out the information, but no subgroup of $n-1$ people can do so.

A simple solution is the following. Let the secret be a residue class S modulo m. Choose then $n-1$ random values $D_1, \ldots, D_{n-1}$ modulo m, and set

$$D_n \equiv S - D_1 - \ldots - D_{n-1} \bmod m.$$

Now the ith participant receives the value D_i for $1 \leqslant i \leqslant n$. It is clear that together they can reveal S, since

$$S \equiv D_1 + \ldots + D_n \bmod m.$$

However, if $n-1$ of them gets together, and share their knowledge, they still cannot have a good guess on S, which can be any residue class modulo m.

A more general problem is the following: there are n participants and the goal is to give each of them some piece of the given secret S such that any t of them can reveal the whole S, but no $t-1$ of them can do so. The following idea is due to Shamir.

Let the secret S be a number in any fixed field $\mathbf{F}$. Set $a_0 = S$, and choose random numbers $a_1, \ldots, a_{t-1}$. Construct the polynomial

$$f(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1} \in \mathbf{F}[x].$$

We see that $f(0) = S$. Take random nonzero elements $x_1, \ldots, x_n \in \mathbf{F}^\times$ and let the ith participant receive the value $y_i = f(x_i)$.

Proposition 7.3.1. *Let* $\mathbf{F}$ *be a field. Assume that* $a_1,\dots,a_m,b_1,\dots,b_m \in \mathbf{F}$ *such that the* a_i*'s are distinct. Then there exists a unique polynomial* $f \in \mathbf{F}[x]$ *such that it is either the zero polynomial or its degree is at most* $m-1$ *which satisfies* $f(a_i) = b_i$ *for any* $1 \leqslant i \leqslant m$.

Proof of uniqueness. Assume $f(x)$ and $g(x)$ are two such polynomials. Then consider the polynomial $h(x) = f(x) - g(x)$. Then either $h(x) = 0$ or $\deg h(x) \leqslant m-1$. Note on the other hand that $a_1,\dots,a_m$ are all roots of $h(x)$, and the number of roots cannot exceed the degree of a polynomial over a field. Then the possibility $\deg h(x) \leqslant m-1$ is excluded, so $h(x) = 0$, and then $f(x) = g(x)$. □

Proof of existence via Lagrange's interpolation. Consider the polynomial

$$f(x) = \sum_{i=1}^{m} b_i \prod_{\substack{1 \leqslant j \leqslant m \\ j \neq i}} \frac{x - a_j}{a_i - a_j}.$$

It is clear that $f(x) = 0$ or $\deg f(x) \leqslant m-1$. Also, for a single monomial,

$$\prod_{\substack{1 \leqslant j \leqslant m \\ j \neq i}} \frac{x - a_j}{a_i - a_j} \equiv \begin{cases} 1, \text{ if } x = a_i, \\ 0, \text{ if } x = a_{i'} \text{ for some } 1 \leqslant i' \leqslant m \text{ with } i' \neq i. \end{cases}$$

From this, the statement is obvious. □

Proof of existence via Newton's interpolation. We induct on $m \in \mathbf{N}$. The statement for $m = 1$ is obvious, since the polynomial can be chosen to be the constant b_1. Now assume that $g(x)$ is an appropriate polynomial for the first $m-1$ pairs in the input: $g(x) = 0$ or $\deg g(x) \leqslant m-2$, and $g(a_i) = b_i$ for $1 \leqslant i \leqslant m-1$. Then set

$$f(x) = g(x) + (b_m - g(a_m)) \prod_{j=1}^{m-1} \frac{x - a_j}{a_m - a_j}.$$

Clearly $f(x)$ satisfies the degree conditions, $f(a_i) = g(a_i) = b_i$ for $1 \leqslant i \leqslant m-1$, while at a_m, we have

$$f(a_m) = g(a_m) + (b_m - g(a_m)) \prod_{j=1}^{m-1} \frac{a_m - a_j}{a_m - a_j} = g(a_m) + b_m - g(a_m) = b_m,$$

and the proof is complete. □

Although both proofs provide an algorithm, Newton's one is faster in practice (although it is not as explicit as Lagrange's one).

With this tool in hand, it is clear that t people can figure out the shared secret: they simply interpolate the value $f(0) = S$. However, given only $t-1$ pieces of the secret, the polynomial can still take any value of $\mathbf{F}$ at 0.

Bibliography

[1] J. Hoffstein, J. Pipher, and J. H. Silverman. *An Introduction to Mathematical Cryptography*. Undergraduate Texts in Mathematics. Springer, 2008.

[2] L. Lovász. *Algoritmusok bonyolultsága*. University text.