CrypTool Version 1.3
Tutorial

November 22, 2001

In this CrypTool tutorial you will find predominantly mathematically oriented information for using encryption procedures. The main chapters have been written by various authors and are therefore independent from one another. At the end of most chapters you will wind literature and URLs.

You will receive information about the principles of symmetrical and asymmetrical encryption. A large section of the tutorial is dedicated to the fascinating topic of prime numbers. XXX You will also obtain an overview of the mathematical ideas behind modern cryptography.XXX

A further chapter is devoted to digital signatures, which are an essential component of e-business applications. The last chapter — elliptic curves — fits in well with this. The mathematics of elliptic curves forms the basis for rapid cryptographic algorithms for digital signatures; these algorithms are extremely well suited for implementation on smartcards.

Whereas the program teaches you how to use cryptography in practice, the tutorial provides those interested in the subject with a deeper understanding of the mathematical algorithms used.

The authors Bernhard Esslinger, Bartol Filipovic, Henrik Koy and Roger Oyono would like to take this opportunity to thank their colleagues in the company and at the universities of Frankfurt, Gieen und Karlsruhe. They particularly thank Dr. Peer Wichmann from the Karlsruhe computer science research centre (Forschungszentrum Informatik, FZI) for his down-to-earth support.

As at CrypTool, the quality of the tutorial is enhanced through your suggestions and ideas for improvement. We look forward to your feedback.

You will find the current version of CrypTool under
http://www.CrypTool.de, http://www.CrypTool.com or http://www.CrypTool.org.

The contact persons for this free tool are listed in the readme file for CrypTool.

Contents

Introduction

This tutorial is delivered together with CrypTool.

CrypTool is a program with an extremely comprehensive online help enabling you to use and analyse cryptographic procedures.

For the conventional encryption procedures, automatic analyses are available that enable you to determine the key with knowledge of the encrypted document and, possibly, further information (non-encrypted document or language of the document).

CrypTool was developed during the End-User Awareness program in order to increase employee awareness of IT security and provide them with a deeper understanding of the term security.

A further aim was to enable users to understand the cryptographic procedures implemented in the Deutsche Bank. XXX In this way, using CrypTool as a reliable reference implementation of the various encryption procedures, you can test the encryption implemented in other programs. XXX

1 Encryption procedures

1.1 Encryption

The purpose of encryption is to change data in such a way that only an authorised recipient is able to reconstruct the plaintext. This has the advantage that you can transmit encrypted data openly and nevertheless need not fear a perpetrator reading the data without authorisation. Authorised recipients possess a piece of secret information — called the key — which allows them to decrypt the data while it remains hidden from everyone else.

One encryption procedure has been proved to be secure — the *One-Time-Pad* . However, this procedure has several practical disadvantages (the key used must be selected randomly and must be just as long as the message to be protected), which means that it is hardly used except in closed environments such as for the hot wire between Moscow and Washington.

For all other procedures there is a (theoretical) possibility of breaking them. If the procedures are good, however, the time taken to break them is so long that it is practically impossible to do and these procedures can therefore be considered (practically) secure.

We basically distinguish between symmetric and asymmetric encryption procedures.

1.1.1 Symmetric encryption

For *symmetric* encryption the sender and recipient must possess a common (secret) key which they have exchanged before actually starting to communicate. The sender uses this key to encrypt the message and the recipient uses it to decrypt it.

The advantages of symmetric algorithms are the high speed with which data can be encrypted and decrypted. One disadvantage is the need for key management. In order to communicate with one another confidentially, sender and recipient must have exchanged a key using a secure channel before actually starting to communicate. Spontaneous communication between individuals who have never met therefore seems virtually impossible. If everyone wants to communicate with everyone else spontaneously at any time in a network of n subscribers, each subscriber must have previously exchanged a key with each of the other $n-1$ subscribers. A total of $n(n-1)/2$ keys must therefore be exchanged.

The most well-known symmetric encryption procedure is the DES– algorithm. The DES-algorithm has been developed by IBM in collaboration with the National Security Agency (NSA), and was published as a standard in 1975. Despite the fact that the procedure is relatively old, no effective attack on it has yet been detected. The most effective way of attacking consists of testing all possible keys until the right one is found (*brute–force– attack*). Due to the relatively short key length of effectively 56 bits (64 bits, which however include 8 parity bits), numerous messages encrypted using DES have in the past been broken. Therefore, the procedure can now only be considered to be conditionally secure. Symmetric alternatives to the DES procedure include the IDEA or Triple DES algorithms.

Up-to-the-minute procedures are the symmetric AES procedures. The associated Rijndael procedure was declared winner the AES award on 2 October 2000 and thus succeeds the DES

procedure.

1.1.2 Asymmetrc encryption

In the case of *asymmetric* encryption each subscriber has a personal pair of keys consisting of a *secret* key and a *public* key . The public key, as its name implies, is made public, e.g. in a key directory on the Internet.

If Alice wants to communicate with Bob, then she finds Bob's public key in the directory and uses it to encrypt her message to him. She then sends this ciphertext to Bob, who is then able to decrypt it again using his secret key. As only Bob knows his secret key, only he can decrypt messages addressed to him. Even Alice who sends the message cannot restore plaintext from the (encrypted) message she has sent. Of course, you must first ensure that the public key cannot be used to derive the private key.

Such a procedure can be demonstrated using a series of thief-proof letter boxes. If I have composed a message, I then look for the letter box of the recipient and post the letter through it. After that, I can no longer read or change the message myself, because only the legitimate recipient possesses the key for the letter box.

The advantage of asymmetric procedures is the easy key management. Let's look again at a network with n subscribers. In order to ensure that each subscriber can establish an encrypted connection to each other subscriber, each subscriber must possess a pair of keys. We therefore need $2n$ keys or n pairs of keys. Furthermore, no secure channel is needed before messages are transmitted, because all the information required in order to communicate confidentially can be transmitted openly. In this case, you simply have to pay attention to the accuracy (integrity and authenticity) of the public key. Disadvantage: Pure asymmetric procedures take a lot longer to perform than symmetric ones.

The most well-known asymmetric procedure is the RSA – algorithm, named after its developers Ronald Rivest, Adi Shamir and Leonard Adleman. The RSA- algorithm was published in 1978. The concept of asymmetric encryption was first introduced by Whitfield Diffie and Martin Hellman in 1976. Today, the ElGamal procedures also play a decisive role, particularly the Schnorr variant in the DSA (Digital Signature Algorithm).

1.1.3 Hybrid procedures

In order to benefit from the advantages of symmetric and asymmetric techniques together, hybrid procedures are usually used (for encryption) in practice.

In this case the data is encrypted using symmetric procedures: the key is a session key generated by the sender randomly that is only used for this message. This session key is then encrypted using the asymmetric procedure and transmitted to the recipient together with the message. Recipients can determine the session key using their secret keys and then use the session key to encrypt the message. In this way, we can benefit from the easy key management of asymmetric procedures and encrypt large quantities of data quickly and efficiently using symmetric procedures.

2 Prime numbers

(Bernhard Esslinger, May 1999, Update Nov. 2000)

What are prime numbers? Prime numbers are whole, positive numbers greater than or equal to 2 that can only be divided by 1 and themselves. All other natural numbers greater than or equal to 2 can be formed by multiplying prime numbers.

The *natural* numbers thus comprise $1, 2, 3, 4$, ...

- the number 1 (the unit value)
- the primes and
- the composite numbers.

Prime numbers are particularly important for 3 reasons:

- In number theory, they are considered to be the basic components of natural numbers, upon which numerous brilliant mathematical ideas are based.
- They are of extreme practical importance in modern cryptography (public-key cryptography). The most common public-key procedure, invented at the end of the 1970's, is RSA encryption. Only using (large) prime numbers for particular parameters can you guarantee that an algorithm is secure, both for the RSA procedure and for even more modern procedures (digital signature, elliptic curves).
- The search for the largest known prime numbers does not have any practical usage known to date, but requires the best computers, is an excellent benchmark (possibility for determining the performance of computers) and leads to new calculation methods on many computers (see also: http://www.mersenne.org/prime.htm).

Many people have been fascinated by prime numbers over the past two millenniums. Ambition to make new discoveries about prime numbers has often resulted in brilliant ideas and conclusions. The following section provides an easily comprehensible introduction to the basics of prime numbers. We will also explain what is known about the distribution (density, number of prime numbers in particular intervals) of prime numbers and how prime number tests work.

2.1 Prime numbers in mathematics

Every whole number has a factor. The number 1 only has one factor, itself, whereas the number 12 has the six factors $1, 2, 3, 4, 6, 12$. Many numbers can only be divided by themselves and by 1. With respect to multiplication, these are the atoms in the area of numbers. Such numbers are called prime numbers.

In mathematics, a slightly different (but equivalent) definition is used.

Definition 1. *A whole number $p \in \mathbf{N}$ is called prime if $p > 1$ and p only possesses the trivial factors ± 1 and $\pm p$.*

By definition, the number 1 is not a prime number. In the following sections, p will always denote a prime number.

The sequence of prime numbers starts with

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...$$

The first 100 numbers include precisely 25 prime numbers. After this, the percentage of primes constantly decreases. Prime numbers can be factorised in a unique *trivial* way:

$$5 = 1 \cdot 5, \quad 17 = 1 \cdot 17, \quad 1013 = 1 \cdot 1.013, \quad 1.296.409 = 1 \cdot 1.296.409.$$

All numbers that have 2 or more factors are called *composite*. These include

$$4 = 2 \cdot 2, \quad 6 = 2 \cdot 3$$

as well as numbers that *look like primes*, but are in fact composite:

$$91 = 7 \cdot 13, \quad 161 = 7 \cdot 23, \quad 767 = 13 \cdot 59$$

Theorem 1. *Each whole number m greater than 1 possesses a lowest factor greater than 1. This is a prime number p. Unless m is a prime number itself, then: p is less than or equal to the square root of m.*

All whole numbers greater than 1 can be expressed as a product of prime numbers — in a unique way. This is the claim of the 1st fundamental theorem of number theory (= fundamental theorem of arithmetic = fundamental building block of all positive integers).

Theorem 2. *Each element n of the natural numbers greater than 1 can be written as the product $n = p_1 \cdot p_2 \ldots p_m$ of prime numbers. If two such factorisations*

$$n = p_1 \cdot p_2 \cdots p_m = p'_1 \cdot p'_2 \cdots p'_{m'}$$

are given, then they can be reordered such that $m = m'$ and for all i: $p_i = p'_i$.

In other words: each natural number other than 1 can be written as a product of prime numbers in precisely one way, if we ignore the order of the factors. The factors are therefore unique (the *expression as a product of factors* is unique)! For example,

$$60 = 2 \cdot 2 \cdot 3 \cdot 5 = 2^2 \cdot 3^1 \cdot 5^1$$

And this — other than changing the order of the factors — is the only way in which the number 60 can be factorised. If you allow numbers other than primes as factors, there are several ways of factorising integers and the uniqueness is lost:

$$60 = 1 \cdot 60 = 2 \cdot 30 = 4 \cdot 15 = 5 \cdot 12 = 6 \cdot 10 = 2 \cdot 3 \cdot 10 = 2 \cdot 5 \cdot 6 = 3 \cdot 4 \cdot 5 = ...$$

The following section is aimed more at those familiar with mathematical logic: The 1st fundamental theorem only appears to be obvious . We can construct numerous other sets of numbers (i.e. other than positive whole numbers greater than 1), for which numbers in the set cannot be expressed uniquely as a product of the prime numbers of the set: In the set $M = \{1, 5, 10, 15, 20, ...\}$ there is no equivalent to the fundamental theorem under multiplication. The first five prime numbers of this sequence are $5, 10, 15, 20, 30$ (note: 10 is prime, because 5 is not a factor of 10 in this set — the result is not an element of the given basic set M). Because the following applies in M:

$$100 = 5 \cdot 20 = 10 \cdot 10$$

and $5, 10, 20$ are all prime numbers in this set, the expression as a product of prime factors is not unique here.

2.2 How many prime numbers are there?

For the natural numbers, the primes can be compared to elements in chemistry or the elementary particles in physics (see [Blum1999, S. 22]).

Although there are only 92 natural chemical elements, the number of prime numbers is unlimited. Even the Greek, Euclid knew this in the third century before Christ. His proof that there is an infinite number of primes is still considered to be a brilliant mathematical consideration and conclusion today (proof by contradiction). He assumed that there is only a finite number of primes and therefore a largest prime number. Based on this assumption, he drew logical conclusions until he obtained an obvious contradiction. This meant that something must be wrong. As there were no mistakes in the chain of conclusions, it could only be the assumption that was wrong. Therefore, there must be an infinite number of primes!

Euclid's proof by contradiction argues as follows:

Assumption: There is a *finite* number of primes.

Conclusion: Then these can be listed $p_1 < p_2 < p_3 < \cdots < p_n$, where n is the (finite) number of prime numbers. p_n is therefore the largest prime. Euclid now looks at the number $a = p_1 \cdot p_2 \cdots p_n + 1$. This number cannot be a prime number because it is not included in our list of primes. It must therefore be divisible by a prime, i.e. there is a natural number i between 1 and n, such that p_i divides the number a. Of course, p_i also divides the product $a - 1 = p_1 \cdot p_2 \cdots p_n$, because p_i is a factor of $a - 1$. Since p_i divides the numbers a and $a - 1$, it also divides the difference of these numbers. Thus: p_i divides $a - (a - 1) = 1$. p_i must therefore divide 1, which is impossible.

Contradiction: Our assumption was false.

Thus there is an *infinite* number of primes. (Cross-reference: overview under 2.6.1 of the number of prime numbers in various intervals).

2.3 The search for extremely large primes

The largest prime numbers known today have several hundred thousand digits, which is too big for us to imagine. The number of elementary particles in the universe is only estimated to be a 80-digit number. (See: overview under 2.6.3 of various orders of magnitude / dimensions.)

Almost all known huge prime numbers are special candidates, called *Mersenne numbers* of the form $2^p - 1$, where p is a prime. Marin Mersenne (1588-1648) was a French priest and mathematician. Not all Mersenne numbers are prime:

$$\begin{aligned} 2^2 - 1 = 3 \quad & \Rightarrow \text{prime} \\ 2^3 - 1 = 7 \quad & \Rightarrow \text{prime} \\ 2^5 - 1 = 31 \quad & \Rightarrow \text{prime} \\ \vdots \quad & \\ 2^{11} - 1 = 2.047 = 23 \cdot 89 \quad & \Rightarrow \text{NOT prime!} \end{aligned}$$

Even Mersenne knew that not all Mersenne numbers are prime (see exponent $p = 11$). However, he is to be thanked for the interesting conclusion that a number of the form $2^n - 1$ is not a prime number if n is a composite number:

Theorem 3 (Mersenne). *If $2^n - 1$ is a prime number, then n is also a prime number.*

Proof
The theorem of Mersenne can be proved by contradiction. We therefore assume that there exists a composite natural number n (with real factorisation) $n = n_1 n_2$, with the property that $2^n - 1$ is a prime number.

From

$$\begin{aligned} (x^r - 1)((x^r)^{s-1} + (x^r)^{s-2} + \cdots + x^r + 1) &= ((x^r)^s + (x^r)^{s-1} + (x^r)^{s-2} + \cdots + x^r) \\ &\quad -((x^r)^{s-1} + (x^r)^{s-2} + \cdots + x^r + 1) \\ &= (x^r)^s - 1 = x^{rs} - 1, \end{aligned}$$

we conclude

$$2^{n_1 n_2} - 1 = (2^{n_1} - 1)((2^{n_1})^{n_2 - 1} + (2^{n_1})^{n_2 - 2} + \cdots + 2^{n_1} + 1).$$

Because $2^n - 1$ is a prime number, one of the above two factors on the right-hand side must be equal to 1. This is the case if and only if $n_1 = 1$ or $n_2 = 1$. But this contradicts our assumption. Therefore the assumption is false. This means that there exists no composite number n, such that $2^n - 1$ is a prime. □

Unfortunately this theorem only applies in one direction (the inverse does not apply, no equivalence): see the above example $2^{11} - 1$, where 11 is prime.

Mersenne claimed that $2^{67} - 1$ is a prime number. There is also a mathematical history behind this claim: it first took over 200 years before Edouard Lucas (1842-1891) proved that this number

is composite. However, he argued indirectly and did not name any of the factors. Then Frank Nelson showed in 1903 which factors make up this composite number:

$$2^{67} - 1 = 147,573,952,588,676,412,927 = 193.707.721 \cdot 761,838,257,287$$

He admitted to having worked 20 years on the factorisation (expression as a product of factors) of this 21-digit decimal number!

Due to the fact that the exponents of the Mersenne numbers do not use all natural numbers, but only the primes, the *experimental space* is limited considerably. The currently known Mersenne prime numbers have the exponents

$$2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1.279, 2.203, 2.281, 3.217, 4.253, 4.423,$$
$$9.689, 9.941, 11.213, 19.937, 21.701, 23.207, 44.497, 86.243, 110.503, 132.049, 216.091,$$
$$756.839, 859.433, 1.257.787, 1.398.269, 2.976.221, 3.021.377, 6.972.593.$$

Thus 38 Mersenne prime numbers are currently known. The 19th number with the exponent $4,253$ was the first with at least $1,000$ digits in decimal system (the mathematician Samual Yates coined the expression *titanic* prime for this; it was discovered by Hurwitz in 1961); the 27th number with the exponent $44,497$ was the first with at least $10,000$ digits in the decimal system (Yates coined the expression *gigantic* prime for this. These names are now long outdated). The 37th number was found in January 1998 and has 909,526 digits in the decimal system, which corresponds to 33 pages in the newspaper!

These numbers can be found at the following URLs:

http://reality.sgi.com/chongo/prime/prime_press.html

> (The supercomputer manufacturer SGI Cray Research not only employed brilliant mathematicians but also used the prime number tests as benchmarks for their machines.)

http://www.utm.edu/

> (At the university of Tennessee you will find extensive research results about prime numbers.)

The largest prime number currently known (November 2000) is the Mersenne number

$$2^{6,972,593} - 1.$$

It was discovered in June 1999 and has $2,098,960$ digits in the decimal system (that corresponds to around 77 pages in the newspaper). Like the three Mersenne numbers listed above, this number was also discovered in the GIMPS project (Great Internet Mersenne Prime Search). Since 1995, dozens of experts and thousands of amateurs have been working on this project to find such numbers using distributed computer programs.

This search is also spurred on by a competition started by the non-profit organisation EFF (Electronic Frontier Foundation) using the means of an unknown donator. The participants are

rewarded with a total of 500,000 USD if they find the longest prime number. In promoting this project, the unknown donator is not looking for the quickest computer, but rather wants to draw people's attention to the possibilities offered by *cooperative networking*

http://www.eff.org/coopawards/prime-release1.html,

Edouard Lucas (1842-1891) held the record for the longest prime number for over 70 years by proving that $2^{127} - 1$ is prime. No new record is likely to last that long.

2.3.1 Prime number tests

In order to implement secure encryption procedures we need extremely large prime numbers (in the region of $2^{2,048}$, i.e. numbers with 600 digits in the decimal system!).

Up to now we have always looked for the prime factors in order to decide whether a number is prime. However, if even the smallest prime factor is enormous, the search takes too long. Factorising numbers using systematic computational division or using the sieve of Eratosthenes (see below) is only feasible using current computers for numbers with up to around 20 digits in the decimal system.

However, if we know something about the *construction* of the number in question, there are extremely highly developed procedures that are much quicker. In the 17th century, Fermat wrote to Mersenne that he presumed that all numbers of the form

$$F(n) = 2^{2^n} + 1$$

are prime for all whole numbers n greater than or equal to 0 (see below).

As early as in the 19th century, it was discovered that the 29-digit number

$$F(7) = 2^{2^7} + 1$$

is not prime. However, it was not until 1970 that Morrison/Billhart managed to factorise it.

$$\begin{aligned} F(7) &= 34,279,974,696,877,740,253,374,607,431,768,211,457 \\ &= 59,649,589,127,497,217 \cdot 574,689,200,685,129,054,721 \end{aligned}$$

Many rapid prime number tests are based on the (small) Fermat theorem put forward by Fermat in 1640 .

Theorem 4 (small Fermat). *Let p be a prime number and a be any whole number, then for all a*

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

This could also be formulated as follows:
Let p be a prime number and a be any whole number that is not a multiple of p (also $a \not\equiv 0 \bmod p$), then $a^{p-1} \equiv 1 \bmod p$.

If you are not used to calculating with remainders (modulo), please simply accept the theorem. What is important here is that this sentence implies that if this equation is not met for any whole

number a, then p is not a prime! The tests (e.g. for the first formulation) can easily be performed using the *test basis* $a = 2$.

This gives us a criterion for non-prime numbers, i.e. a negative test, but no proof that a number a is prime. Unfortunately Fermat's theorem does not apply — otherwise we would have a simple proof of the prime number property (or to put it in other words, we would have a simple prime number criterion).

Comment: Numbers that have the property

$$2^n \equiv 2 \bmod n$$

but are not prime are called *pseudoprime numbers*. The first pseudoprime number (i.e. not a prime) is

$$341 = 11 \cdot 31.$$

There are numbers that pass the Fermat test with all bases and yet are not prime: these numbers are called *Carmichael numbers*. The first of these is

$$561 = 3 \cdot 11 \cdot 17.$$

A stronger test is provided by Miller/Rabin: it is only passed by so-called *strong pseudoprime numbers*. Again, there are strong pseudoprime numbers that are not primes, but this is much less often the case than for (simple) pseudoprime numbers. The smallest strong pseudoprime number base 2 is

$$15.841 = 7 \cdot 31 \cdot 73.$$

If you test all 4 bases, $2, 3, 5$ and 7, you will find only one strong pseudoprime number up to $25 \cdot 10^9$, i.e. a number that passes the test and yet is not a prime number.

More extensive mathematics behind the Rabin test delivers the probability that the number examined is prime (such probabilities are currently around 10^{-60}).

Detailed descriptions of tests for finding our whether a number is prime can be found on Web sites such as:

http://www.utm.edu/research/primes/mersenne.shtml and

http://www.utm.edu/research/primes/prove/index.html.

2.4 The search for a formula for prime numbers

There are currently no useful, open (i.e. not recursive) formulae known that only deliver prime numbers (recursive means that in order to calculate the function the same function is used with a smaller variable). Mathematicians would be happy if they could find a formula that leaves gaps (i.e. does not deliver all prime numbers) but does not deliver any composite (non-prime) numbers.

Ideally, we would like, for the number n, to immediately obtain the n-th prime number, i.e. for $f(8) = 19$ or for $f(52) = 239$.

Ideas for this can be found at

http://www.prothsearch.net/index.html.

Cross-reference: the table under 2.6.2 contains the precise values for the nth prime numbers for selected n.

The following are some of the most common ideas for prime number formulae:

1. Mersenne numbers $f(n) = 2^n - 1$:
   As shown above, this formula seems to deliver relatively large prime numbers but - as for $n = 11$ $[f(n) = 2,047]$ - it is repeatedly the case that the result is not prime.

2. $F(k, n) = k \cdot 2^n \pm 1$ for n prime and k small primes:
   For this generalisation of the Mersenne numbers there are (for small k) also extremely quick prime number tests (see [Knuth1981]). This can be performed in practice using software such as the PROTHS software from Yves Gallot

   (http://www.utm.edu/research/primes/programs/gallot/proths.html).

3. *Fermatzahlen* $F(n) = 2^{2^n} + 1$:
   As mentioned above, Fermat wrote to Mersenne regarding this assumption. Surprisingly he would have been able obtain a positive result using the negative prime number test for $n = 5$ based on his small theorem.

$$
\begin{array}{lll}
F(0) = 2^{2^0} + 1 = 2^1 + 1 & = 3 & \mapsto \text{ prime} \\
F(1) = 2^{2^1} + 1 = 2^2 + 1 & = 5 & \mapsto \text{ prime} \\
F(2) = 2^{2^2} + 1 = 2^4 + 1 & = 17 & \mapsto \text{ prime} \\
F(3) = 2^{2^3} + 1 = 2^8 + 1 & = 257 & \mapsto \text{ prime} \\
F(4) = 2^{2^4} + 1 = 2^{16} + 1 & = 65.537 & \mapsto \text{ prime} \\
F(5) = 2^{2^5} + 1 = 2^{32} + 1 & = 4,294,967,297 = 641 \cdot 6,700,417 & \mapsto \text{ NOT prime!}
\end{array}
$$

4. Carmichael numbers: see above

5. Pseudoprime numbers: see above

6. Strong pseudoprime numbers: see above

7. Idea based on Euclid's proof (infinite many prime numbers) $p_1 \cdot p_2 \cdots p_n + 1$:

$$
\begin{array}{lll}
2 \cdot 3 + 1 & = 7 & \mapsto \text{ prime} \\
2 \cdot 3 \cdot 5 + 1 & = 31 & \mapsto \text{ prime} \\
2 \cdot 3 \cdot 5 \cdot 7 + 1 & = 211 & \mapsto \text{ prime} \\
2 \cdot 3 \cdots 11 + 1 & = 2311 & \mapsto \text{ prime} \\
2 \cdot 3 \cdots 13 + 1 & = 59 \cdot 509 & \mapsto \text{ NOT prime!} \\
2 \cdot 3 \cdots 17 + 1 & = 19 \cdot 97 \cdot 277 & \mapsto \text{ NOT prime!}
\end{array}
$$

8. As above but except +1: $p_1 \cdot p_2 \cdots p_n - 1$

$$\begin{array}{lll} 2 \cdot 3 - 1 & = 5 & \mapsto \text{ prime} \\ 2 \cdot 3 \cdot 5 - 1 & = 29 & \mapsto \text{ prime} \\ 2 \cdot 3 \cdots 7 - 1 & = 11 \cdot 19 & \mapsto \text{ NOT prime!} \\ 2 \cdot 3 \cdots 11 - 1 & = 2309 & \mapsto \text{ prime} \\ 2 \cdot 3 \cdots 13 - 1 & = 30029 & \mapsto \text{ prime} \\ 2 \cdot 3 \cdots 17 - 1 & = 61 \cdot 8369 & \mapsto \text{ NOT prime!} \end{array}$$

9. *Euclidean numbers* $e_n = e_0 \cdot e_1 \cdots e_{n-1} + 1$ with n greater than or equal to 1 and $e_0 := 1$. e_{n-1} is not the $(n-1)$th prime number, but the number previously found here. Unfortunately this formula is not open but recursive. The sequence starts with

$$\begin{array}{lll} e_1 = 1 + 1 & = 2 & \mapsto \text{ prime} \\ e_2 = e_1 + 1 & = 3 & \mapsto \text{ prime} \\ e_3 = e_1 \cdot e_2 + 1 & = 7 & \mapsto \text{ prime} \\ e_4 = e_1 \cdot e_2 \cdot e_3 + 1 & = 43 & \mapsto \text{ prime} \\ e_5 = e_1 \cdot e_2 \cdots e_4 + 1 & = 13 \cdot 139 & \mapsto \text{ NOT prime!} \\ e_6 = e_1 \cdot e_2 \cdots e_5 + 1 & = 3.263.443 & \mapsto \text{ prime} \\ e_7 = e_1 \cdot e_2 \cdots e_6 + 1 & = 547 \cdot 607 \cdot 1.033 \cdot 31.051 & \mapsto \text{ NOT prime!} \\ e_8 = e_1 \cdot e_2 \cdots e_7 + 1 & = 29.881 \cdot 67.003 \cdot 9.119.521 \cdot 6.212.157.481 & \mapsto \text{ NOT prime!} \end{array}$$

$e_9, ..., e_{17}$ are also composite, which means that this formula is not particularly useful. Comment: However, what is special about all these numbers is that any pair of them does not have a common factor other than 1. They are therefore *relatively prime* .

10. $f(n) = n^2 + n + 41$:
This sequence starts off very *promisingly*, but is far from being a proof.

$$\begin{array}{ll} f(0) = 41 & \mapsto \text{ prime} \\ f(1) = 43 & \mapsto \text{ prime} \\ f(2) = 47 & \mapsto \text{ prim} \\ f(3) = 53 & \mapsto \text{ prime} \\ f(4) = 61 & \mapsto \text{ prime} \\ f(5) = 71 & \mapsto \text{ prime} \\ f(6) = 83 & \mapsto \text{ prime} \\ f(7) = 97 & \mapsto \text{ prime} \\ \vdots & \\ f(39) = 1.601 & \mapsto \text{ prime} \\ f(40) = 11 \cdot 151 & \mapsto \text{ NOT prime!} \\ f(41) = 41 \cdot 43 & \mapsto \text{ NOT prime!} \end{array}$$

The first 40 values are prime numbers (which have the obvious regularity that their difference starts with 2 and increases by 2 each time), but the 41th and 42th values are not prime numbers. It is easy to see that $f(41)$ hl cannot be a prime number: $f(41) = 41^2 + 41 + 41 = 41(41 + 1 + 1) = 41 \cdot 43$.

11. $f(n) = n^2 - 79 \cdot n + 1.601$:
This function delivers prime numbers for all values from $n = 0$ to $n = 79$. Unfortunately $f(80) = 1,681 = 11 \cdot 151$ is not a prime number. To this date, no function has been found that delivers more prime numbers in a row. On the other hand, each prime occurs twice (first in the decreasing then in the increasing sequence), which means that the algorithm delivers a total of 40 difference prime values (the same ones as the function from point 10).

$f(0) = 1.601$	$\mapsto$	prime	$f(28) = 173$	$\mapsto$	prime
$f(1) = 1.523$	$\mapsto$	prime	$f(29) = 151$	$\mapsto$	prime
$f(2) = 1.447$	$\mapsto$	prime	$f(30) = 131$	$\mapsto$	prime
$f(3) = 1.373$	$\mapsto$	prime	$f(31) = 113$	$\mapsto$	prime
$f(4) = 1.301$	$\mapsto$	prime	$f(32) = 97$	$\mapsto$	prime
$f(5) = 1.231$	$\mapsto$	prime	$f(33) = 83$	$\mapsto$	prime
$f(6) = 1.163$	$\mapsto$	prime	$f(34) = 71$	$\mapsto$	prime
$f(7) = 1.097$	$\mapsto$	prime	$f(35) = 61$	$\mapsto$	prime
$f(8) = 1.033$	$\mapsto$	prime	$f(36) = 53$	$\mapsto$	prime
$f(9) = 971$	$\mapsto$	prime	$f(37) = 47$	$\mapsto$	prime
$f(10) = 911$	$\mapsto$	prime	$f(38) = 43$	$\mapsto$	prime
$f(11) = 853$	$\mapsto$	prime	$f(39) = 41$	$\mapsto$	prime
$f(12) = 797$	$\mapsto$	prime			
$f(13) = 743$	$\mapsto$	prime	$f(40) = 41$	$\mapsto$	prime
$f(14) = 691$	$\mapsto$	prime	$f(41) = 43$	$\mapsto$	prime
$f(15) = 641$	$\mapsto$	prime	$f(42) = 47$	$\mapsto$	prime
$f(16) = 593$	$\mapsto$	prime	$f(43) = 53$	$\mapsto$	prime
$f(17) = 547$	$\mapsto$	prime	$\cdots$		
$f(18) = 503$	$\mapsto$	prime	$f(77) = 1.447$	$\mapsto$	prime
$f(19) = 461$	$\mapsto$	prime	$f(78) = 1.523$	$\mapsto$	prime
$f(20) = 421$	$\mapsto$	prime	$f(79) = 1.601$	$\mapsto$	prime
$f(21) = 383$	$\mapsto$	prime	$f(80) = 11 \cdot 151$	$\mapsto$	NOT prime!
$f(22) = 347$	$\mapsto$	prime	$f(81) = 41 \cdot 43$	$\mapsto$	NOT prime!
$f(21) = 383$	$\mapsto$	prime	$f(82) = 1.847$	$\mapsto$	prime
$f(22) = 347$	$\mapsto$	prime	$f(83) = 1.933$	$\mapsto$	prime
$f(23) = 313$	$\mapsto$	prime	$f(84) = 43 \cdot 47$	$\mapsto$	NOT prime!
$f(24) = 281$	$\mapsto$	prime			
$f(25) = 251$	$\mapsto$	prime			
$f(26) = 223$	$\mapsto$	prime			
$f(27) = 197$	$\mapsto$	prime			

12. Polynomial functions $f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0$ (a_i in $\mathbb{Z}$, $n \geq 1$):

    There exists no such polynomial that for all x in $\mathbb{Z}$ only delivers prime values. For a proof of this, please refer to [Padberg1996, p. 83 f.], where you will also find further details about prime number formulae.

13. Catalan, after whom the so- called *Catalan numbers* $A(n) = (1/(n+1)) * (2n)!/(n!)^2$ are

named, conjectured that C_4 is a prime:

$$\begin{aligned} C_0 &= 2, \\ C_1 &= 2^{C_0} - 1, \\ C_2 &= 2^{C_1} - 1, \\ C_3 &= 2^{C_2} - 1, \\ C_4 &= 2^{C_3} - 1, \cdots \end{aligned}$$

(see http://www.utm.edu/research/primes/mersenne.shtml under Conjectures and Unsolved Problems).

This sequence is also defined recursively and increases extremely quickly. Does it only consist of primes?

$$\begin{array}{lll} C(0) = 2 & & \mapsto \text{ prime} \\ C(1) = 2^2 - 1 & = 3 & \mapsto \text{ prime} \\ C(2) = 2^3 - 1 & = 7 & \mapsto \text{ prime} \\ C(3) = 2^7 - 1 & = 127 & \mapsto \text{ prime} \\ C(4) = 2^{127} - 1 & = 170,141,183,460,469,231,731,687,303,715,884,105,727 & \mapsto \text{ prime} \end{array}$$

It is not (yet) known whether C_5 and higher elements are prime, but this is not very likely. In any case, it has not been proved that this formula delivers only primes.

2.5 Density and distribution of the primes

As Euclid discovered, there is an infinite number of primes. However, some infinite sets are *denser* than others. Within the set of natural numbers, there is an infinite number of even, uneven and square numbers.

The following proves that there are more even numbers than square ones:

- the size of the nth element:
  The nth element of the even numbers is $2n$; the nth element of the square numbers is n^2. Because for all $n > 2$: $2n < n^2$, the nth even number occurs much earlier than the nth square number. Thus the even numbers are distributed more densely and we can say that there are more even numbers than square ones.
- the number of values that are less than or equal to a certain *maximum value* x in $\mathbb{R}$ is:
  There are $[x/2]$ such even numbers and $[\sqrt{x}]$ square numbers. Because for large x the value $x/2$ is much greater than the square root of 2, we can again say that there are more even numbers.

Theorem 5. *For large n: The value of the nth prime $P(n)$ is asymptotic to $n \cdot ln(n)$, i.e. the limit of the relation $P(n)/(n \cdot \ln n)$ is equal to 1 if n tends to infinity.*

The definition is similar for the number of prime numbers $PI(x)$ that do not exceed the maximum value x:

Theorem 6. *$PI(x)$ is asymptotic to $x/ln(x)$.*

This is the **prime number theorem**. It was put forward by Legendre (1752-1833) and Gau (1777-1855) but not proved until over 100 years later.

(Cross-reference: overview under 2.6.1 of the number of prime numbers in various intervals).

For large n, $P(n)$ lies between $2n$ and n^2. This means that there are fewer primes numbers than even natural numbers but more prime numbers than square numbers.

These formulae, which only apply when n tends to infinity, can be replaced by more precise formulae. For $x \geq 67$:

$$ln(x) - 1,5 < x/PI(x) < ln(x) - 0,5$$

As we know that $PI(x) = x/\ln x$ only for very large x (x tending towards infinity), we can create the following overview:

x	$ln(x)$	$x/ln(x)$	$PI(x)(counted)$	$PI(x)/(x/ln(x))$
10^3	6.908	144	168	1.160
10^6	13.816	72,386	78,498	1.085
10^9	20.723	48,254,942	50,847,534	1.054

For a binary number (number in the binary system) of the length 250 bits (2^{250} is approximately $= 1.809251 * 10^{75}$):

$$PI(250) = 2^{250}/(250 \cdot \ln 2) \text{ is approximately } = 2^{250}/173.28677 = 1.045810 \cdot 10^{73}.$$

We can therefore expect that the set of numbers with a bit length of less than 250 contains approximately 10^{73} primes (a reassuring result?!).

We can also express this as follows: Let us consider a *random* natural number n. Then the probability that this number is prime is around $1/\ln(n)$. For example, let us take numbers in the region of 10^{16}. Then we must consider $16 \cdot \ln 10 = 36,8$ numbers (on average) until we find a prime. A precise investigation shows: There are 10 prime numbers between $10^{16} - 370$ and $10^{16} - 1$.

Under the heading *How Many Primes Are There* under

http://www.utm.edu/research/primes/howmany.shtml.

you will find numerous other details.

Using the following Web site:

http://www.math.Princeton.EDU/ãrbooker/nthprime.html

you can easily determine $PI(x)$.

The **distribution** of primes displays several irregularities for which no system has yet been found. (On the one hand, many occur closely together, like 2 and 3, 11 and 13, 809 and 811, on the other hand large gaps containing no primes also occur. For example, no primes lie between 113 and 127, 293 and 307, 317 and 331, 523 and 541, 773 and 787, 839 and 853 as well as between 887 and 907) (For details, see:

http://www.utm.edu/research/primes/notes/gaps.html).

This is precisely part of what motivates mathematicians to discover its secrets.

Sieve of Eratosthenes An easy way of calculating all $PI(x)$ primes less than or equal to x is to use the sieve of Eratosthenes. In the 3rd century before Christ, he found an extremely easy, automatic way of finding this out. To begin with, you write down all numbers from 2 to x, circle 2, then cross out all multiples of 2. Next, you circle the lowest number that hasn't been circled or crossed out (3) and again cross out all multiples of this number, etc. You only need to continue until you reach the largest number whose square is less than or equal to x.

Apart from 2, prime numbers are never even. Apart from 2 and 5, prime numbers never end in 2, 5 or 0. So you only need to consider numbers ending in 1, 3, 7, 9 anyway (there are infinite primes ending in these numbers; see [Tietze1973, vol. 1, p. 137]).

You can now find a large number of finished programs on the Internet - often complete with source code - allowing you to experiment with large numbers yourself. You also have access to large databases that contain either a large number of primes or the factorisation of numerous composite numbers. The **Cunningham project** for example, determines the factors of all composite numbers that are formed as follows:

$$f(n) = b^n \pm 1 \quad \text{for } b = 2, 3, 5, 6, 7, 10, 11, 12$$

(b is not equal to multiples of bases already used, such as $4, 8, 9$).

Details of this can be found under:

http://www.cerias.purdue.edu/homes/ssw/cun

2.6 Notes

Proven statements / theorems about primes

- For each number n in $\mathbf{N}$ there are n consecutive natural numbers that are not primes. A proof of this can be found in [Padberg1996, p. 79].
- The Hungarian mathematician Paul Erds (1913-1996) proved: Between each random number not equal to 1 and its double, there is at least one prime. (He was not the first to prove this theorem, but proved it in a simpler manner than those before him).
- There is a real number a such that the function $f : \mathbf{N} \to \mathbb{Z}$ where $n \mapsto a^{3^n}$ only delivers primes for all n (see [Padberg1996, p. 82]). Unfortunately, problems arise when we try to determine a (see below).

Proven statements / conjectures about primes

- The German mathematician Christian Goldbach (1690-1764) conjectured: Every even natural number greater than 2 can be represented as the sum of two prime numbers. Computers

have verified the Goldbach conjecture for all even numbers up to 10^8 but no general proof has yet been found.

- The German mathematician Bernhard Riemann (1826-1866) put forward a formula for the distribution of primes that would further improve the estimate. However, this has neither been proved nor contradicted (?) so far.

Open questions

Twin primes are prime numbers whose difference is 2. Examples include 5 and 7 or 101 and 103. Triplet primes, however, only occur once: 3, 5, 7. For all other sets of three consecutive uneven numbers, one of them is always divisible by 3 and thus not a prime.

- The number of twin primes is an open question: infinite or limited number? The largest twin primes known today are $1,693,965 \cdot 2^{66,443} \pm 1$.
- Does a formula exist for calculating the number of twin primes per interval?
- The above proof of the function $f : N \to Z$ with $n \mapsto a^{3^n}$ only guarantees the existence of such a number a. How can we determine this number a and will it have a value, making the function also of practical interest?
- Does a polynomial time algorithm exist for representing a number as its prime factors (see [Klee1997, p. 167])? This question can be divided into the two following questions:
- Does a polynomial time algorithm exist that decides whether a number is prime?
- Does a polynomial time algorithm exist that for a composite number n calculates a non-trivial (i.e. other than 1 and n) factor of n?

Further interesting topics regarding prime numbers This chapter doesn't consider the following number theory topics such as divisibility rules, modulus calculation, modular inverses, modular powers and roots, Chinese remainder theorem, Euler PHI function, perfect numbers.

2.6.1 Number of prime numbers in various intervals

Ten-sized intervals		Hundred-sized intervals		Thousand-sized intervals	
Interval	Number	Interval	Number	Interval	Number
1-10	4	1-100	25	1-1000	168
11-20	4	101-200	21	1001-2000	135
21-30	2	201-300	16	2001-3000	127
31-40	2	301-400	16	3001-4000	120
41-50	3	401-500	17	4001-5000	119
51-60	2	501-600	14	5001-6000	114
61-70	2	601-700	16	6001-7000	117
71-80	3	701-800	14	7001-8000	107
81-90	2	801-900	15	8001-9000	110
91-100	1	901-1000	14	9001-10000	112

Further intervals:

Intervall	Number	Average number per 1000
1 - 10,000	1229	122.900
1 - 100,000	9592	95.920
1 - 1,000,000	78,498	78.498
1 - 10,000,000	664,579	66.458
1 - 100,000,000	5,761,455	57.615
1 - 1,000,000,000	50,847,534	50.848
1 - 10,000,000,000	455,052,512	45.505

2.6.2 Indexing prime numbers (nth prime number)

Index	Precise value	Rounded value	Comment
1	2	2	
2	3	3	
3	5	5	
4	7	7	
5	11	11	
6	13	13	
7	17	17	
8	19	19	
9	23	23	
10	29	29	
100	541	541	
1000	7917	7917	
664,559	9,999,991	9.99999E+06	All prime numbers up to 1E+07 were known at the beginning of the 20th century.
1E+06	15,485,863	1.54859E+07	
6E+06	104,395,301	1.04395E+08	This prime was discovered in 1959
1E+07	179,424,673	1.79425E+08	
1E+09	22,801,763,489	2.28018E+10	
1E+12	29,996,224,275,833	2.99962E+13	

Comment: With gaps, extremely large prime numbers were discovered at an early stage.

Quell-URLs:

http://www.math.Pr inceton.EDU/ãrbooker/nthprime.html

Output of the nth prime number

See http://www.utm.edu/research/primes/notes/by_year.html

2.6.3 Orders of magnitude / dimensions in reality

In the description of cryptographic protocols and algorithms, numbers occur that are so large or so small that they are inaccessible to our intuitive understanding. It may therefore be useful to provide comparative numbers from the real world that surrounds us so that we can develop a feeling for the security of cryptographic algorithms. Some of the numbers listed below originate from [Schwenk1996] and [Schneier1996, p.18].

Probability that you will be hijacked on your next flight	$5.5 \cdot 10^{-6}$
Probability of 6 correct numbers in the lottery	$7.1 \cdot 10^{-8}$
Annual probability of being hit by lightning	10^{-7}
Risk of being hit by a meterorite	$1.6 \cdot 10^{-12}$

Time until the next ice age (in years)	14000	(2^{14})
Time until the sun dies away (in years)	10^9	(2^{30})
Age of the Earth (in years)	10^9	(2^{30})
Age of the universe (in years)	10^{10}	(2^{34})
Number of the Earth's atoms	10^{51}	(2^{170})
Number of the sun's atoms	10^{57}	(2^{190})
Number of atoms in the universe (without dark material)	10^{77}	(2^{265})
Volume of the universe (in cm^3)	10^{84}	(2^{280})

2.6.4 Special values in the binary and decimal systems

Dual system	Decimal system
2^{10}	1024
2^{40}	$1.09951 \cdot 10^{12}$
2^{56}	$7.20576 \cdot 10^{16}$
2^{64}	$1.84467 \cdot 10^{19}$
2^{80}	$1.20893 \cdot 10^{24}$
2^{90}	$1.23794 \cdot 10^{27}$
2^{112}	$5.19230 \cdot 10^{33}$
2^{128}	$3.40282 \cdot 10^{38}$
2^{150}	$1.42725 \cdot 10^{45}$
2^{160}	$1.46150 \cdot 10^{48}$
2^{250}	$1.80925 \cdot 10^{75}$
2^{256}	$1.15792 \cdot 10^{77}$
2^{320}	$2.13599 \cdot 10^{96}$
2^{512}	$1.34078 \cdot 10^{154}$
2^{768}	$1.55252 \cdot 10^{231}$
2^{1024}	$1.79769 \cdot 10^{308}$
2^{2048}	$3.23170 \cdot 10^{616}$

Calculation using GMP, for example: http://www.gnu.ai.mit.edu

References

[Bartholome1996] A. Bartholom, J. Rung, H. Kern,
Zahlentheorie fr Einsteiger, Vieweg 1995, 2. Auflage 1996.

[Blum1999] W. Blum,
Die Grammatik der Logik, dtv, 1999.

[Graham1989] R.E. Graham, D.E. Knuth, O. Patashnik,
Concrete Mathematics, Addison-Wesley, 1989.

[Klee1997] V. Klee, S. Wagon,
Ungelste Probleme in der Zahlentheorie und der Geometrie der Ebene, Birkhuser Verlag, 1997.

[Knuth1981] Donald E. Knuth,
The Art of Computer Programming, vol 2: Seminumerical Algorithms, Addison- Wesley, 1969; second edition, 1981.

[Lorenz1993] F. Lorenz,
Algebraische Zahlentheorie, BI Wissenschaftsverlag, 1993.

[Padberg1996] F. Padberg,
Elementare Zahlentheorie, Spektrum Akademischer Verlag, 1996, 2. Auflage.

[Pieper1983] H. Pieper,
Zahlen aus Primzahlen, Verlag Harri Deutsch, 1974, 3. Auflage 1983.

[Schneier1996] Bruce Schneier,
Applied Cryptography, Wiley and Sons, 2nd edition, 1996.

[Schwenk1996] J. Schwenk
Conditional Access. In: taschenbuch der telekom praxis 1996, Hrgb. B. Seiler, Verlag Schiele und Schn, Berlin.

[Tietze1973] H. Tietze,
Gelste und ungelste mathematische Probleme, Verlag C.H. Beck, 1959, 6. Auflage 1973.

URLs

1. http://www.utm.edu/
2. http://prothsearch.net/index.html
3. http://www.mersenne.org/prime.htm
4. http://reality.sgi.com/chongo/prime/prime_press.html
5. http://www.eff.org/coop-awards/prime-release1.html
6. http://www.informatik.tu-darmstadt.de/TI/LiDIA/
7. http://www.math.Princeton.EDU/ãrbooker/nthprime.html
8. http://www.utm.edu/research/primes/notes/by_year.html
9. http://www.cerias.purdue.edu/homes/ssw/cun

3 Introduction to Elementary Number Theory with Examples

(Bernhard Esslinger, July 2001, besslinger@web.de)

> *Eric Berne*[1]*:*
> Mathematical game theory postulates players who respond rationally. Transactional game theory, on the other hand, deals with games that are not rational, perhaps even **irrational and thereby closer to reality**.

3.1 Mathematics and cryptography

A large proportion of modern, asymmetric cryptography is based on mathematical knowledge – on the properties (laws") of whole numbers, which are investigated in elementary number theory. Here, the word elementary means that questions raised in number theory are essentially rooted in the set of natural and whole numbers.

Further mathematical disciplines currently used in cryptography include (see [Bauer1995, p. 2], [Bauer2000, p. 3]) :

- Group theory
- Combination theory
- Complexity theory
- Ergodic theory
- Information theory.

Number theory or arithmetic (the emphasis here is more on the aspect of performing calculations with numbers) was established by Carl Friedrich Gauss as a special mathematical discipline. Its elementary features include the greatest common divisor [2] (gcd), congruence (remainder classes), factorisation, the Euler-Fermat theorem and primitive roots. However, the most important aspect is prime numbers and their multiplicative operation.

For a long time, number theory was considered to be the epitome of pure research, the ideal example of research in the ivory tower. It delved into the mysterious laws of the realm of numbers", giving rise to philosophical considerations as to whether it described elements that exist everywhere in nature or whether it artificially constructed elements (numbers, operators and properties).

We now know that patterns from number theory can be found everywhere in nature. For example, the ratio of laevorotary and dextrorotary spirals in a sunflower is equal to two consecutive

[1] Eric Berne, Games People Play", rororo, (c) 1964, page 235.
[2] This article deals with the gcd (greatest common divisor) in Appendix A.

Fibonacci numbers[3], for example 21 : 34.

Also, at the latest when number theory was applied in modern cryptography, it became clear that a discipline that had been regarded as purely theoretical for centuries actually had a practical use. Today, experts in this field are in great demand on the job market.

Applications in (computer) security now use cryptography because this mathematical discipline is simply better and easier to prove than all other creative substitution procedures that have been developed over the course of time and better than all sophisticated physical methods such as those used to print bank notes [Beutelspacher1996, p. 4].

This article explains the basics of elementary number theory in a way that you can easily understand. It provides numerous examples and very rarely goes into any proofs (these can be found in mathematical textbooks).

For this purpose we will use both theory and examples to explain how to perform calculations in finite sets and describe how these techniques are applied in cryptography. Particular attention will be paid to the traditional Diffie-Hellman (DH) and RSA public key procedures.

> *Carl Friedrich Gauss (30.4.1777 - 23.2.1855):*
> Mathematics is the queen of sciences and number theory is the queen of mathematics.

3.2 Introduction to number theory

Number theory arose from interest in positive whole numbers $1, 2, 3, 4, ...$, also referred to as the set of natural numbers *natural numbers* $\mathbb{N}$. These are the first mathematical constructs used by human civilisation.

In ancient times, no distinction was made between number theory and numerology, which attributed a mystical significance to specific numbers. In the same way as astronomy and chemistry gradually detached themselves from astrology and alchemy during the Renaissance (from the 14th century), number theory also separated itself from numerology.

Number theory has always been a source of fascination – for both amateurs and professional mathematicians. In contrast to other areas of mathematics, many of the problems and theorems in number theory can be understood by non-experts. On the other hand, mathematicians often take a long time to find solutions to the problems or prove the theorems. It is therefore easy to pose good questions but quite another matter to find the answer. One example of this is what is known as Fermats Last (or large) Theorem [4].

[3] The sequence of Fibonacci numbers $(a_i)_{i\in\mathbb{N}}$ is defined by the recursive rule $a_1 := a_2 := 1$ and for all numbers $n = 1, 2, 3, ...$ we define $a_{n+2} := a_{n+1} + a_n$. This historical sequence can be found in many interesting forms in nature (for example, see [Graham1994, p. 290 ff] or the website of Ron Knott, which is devoted to Fibonacci numbers). A lot is known about the Fibonacci sequence and it is used today as an important tool in mathematics.

[4] One of the things you learn in mathematics at school is Pythagoras theorem, which states the following for a right-angle triangle: $a^2 + b^2 = c^2$, where a and b are the lengths of the sides containing the right angle and c is the length of the hypotenuse. Fermat famously proposed that $a^n + b^n \neq c^n$ for whole-number exponents $n > 2$.

Up until the mid 20th century, number theory was considered to be the purest area of mathematics, an area that had no practical use in the real world. This changed with the development of computers and digital communication, as number theory was able to provide several unexpected solutions to real-life tasks. At the same time, advances in information technology allowed specialists in number theory to make huge progress in factorising large numbers, finding new prime numbers, testing (old) conjectures and solving numerical problems that were previously impossible to solve. Modern number theory Number theory!modern is made up of areas such as:

- Elementary number theorie
- Algebraic number theorie
- Analytic number theorie
- Geometric number theorie
- Combinatorial number theorie
- Numeric number theorie
- Probability theorie.

All of the different areas are concerned with questions regarding whole numbers (both positive and negative whole numbers plus zero). However, they each have different methods of dealing with them.

This article only deals with the area of elementary number theory.

3.2.1 Convention

Unless stated otherwise:

- The letter $a, b, c, d, e, k, n, m, p, q$ are used to present whole numbers.
- The letter i and j represent natural numbers.
- The letter p always represents a prime number.
- Te sets $\mathbb{N} = \{1, 2, 3, ...\}$ and $\mathbb{Z} = \{..., -3, -2, -1, 0, 1, 2, 3, ...\}$ are the *natral* and *whole* numbers respectively.

Unfortunately, the letter in which Fermat made the claim did not have enough space for him to prove it. The theorem was not proved until over 300 years later [Wiles1994, p. 433-551].

Joanne K. Rowling[5]*:*
This isnt magic its logic a puzzle. A lot of the greatest wizards havent got an ounce of logic.

3.3 Prime numbers and the first fundamental theorem of elementary number theory

Many of the problems in elementary number theory are concerned with prime numbers.

Every whole number has divisors or factors. The number 1 has just one itself, whereas the number 12 has the six factors 1, 2, 3, 4, 6 and 12[6]. Many numbers are only divisible by themselves and by 1. When it comes to multiplication, these can be regarded as the atoms in the realm of numbers.

Definition 1. Prime numbers *are natural numbers greater than 1 that can only be divided by 1 and themselves.*

By definition, 1 is not a prime number.

If we write down the prime numbers in ascending order (prime number sequence), then we get:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...$$

The first 100 numbers include precisely 25 prime numbers. After this, the percentage of primes decreases, but never reaches zero.

We come across whole numbers that are prime fairly often. In the last decade only, three years were prime: 1993, 1997 and 1999. If they were rare, cryptography would not be able to work with them to the extent it does.

Prime numbers can be factorised in a unique (*trivial*) way:

$$\begin{aligned} 5 &= 1 * 5 \\ 17 &= 1 * 17 \\ 1013 &= 1 * 1013 \\ 1296409 &= 1 * 1296409. \end{aligned}$$

Definition 2. *Natural numbers greater than* 1 *that are not prime are called* **composite numbers**. *These have at least two factors other than* 1.

[5] Joanne K. Rowling, Harry Potter and the Philosophers Stone , Carlsen, (c) 1997, chapter Through the trapdoor , p. 310.

[6] Due to the fact that 12 has so many factors, this number – and multiples of this number – is often found in everyday life: the 12-hour scale on clocks, the 60 minutes in an hour, the 360-degree scale for measuring angles, etc. If we divide these scales into segments, the segments often turn out to be whole numbers. These are easier to use in mental arithmetic than fractions.

Examples of the decomposition of such numbers into prime factors:

$$\begin{aligned} 4 &= 2*2 \\ 6 &= 2*3 \\ 91 &= 7*13 \\ 161 &= 7*23 \\ 767 &= 13*59 \\ 1029 &= 3*7^3 \\ 5324 &= 22*11^3. \end{aligned}$$

Theorem 1. *Each composite number a has a lowest factor greater than* 1*. This factor is a prime number p and is less than or equal to the square root of a.*

All whole numbers greater than 1 can be expressed as a product of prime numbers — in a unique way.

This is the claim of the 1st *fundamental theorem of number theory* (= fundamental theorem of arithmetic = fundamental building block of all positive integers). This was formulated precisely for the first time by Carl Friedrich Gauss in his Disquisitiones Arithmeticae (1801).

Theorem 2. Gauss 1801 *Every even natural number greater than* 1 *can be written as the product of prime numbers. Given two such decompositions* $a = p_1 * p_2 * ... * p_n = q_1 * q_2 * ... * q_m$*, these can be resorted such that* $n = m$ *and, for all* $i, p_i = q_i$*.*

In other words: Each natural number other than 1 can be written as a product of prime numbers in precisely one way, if we ignore the order of the factors. The factors are therefore unique (the expression as a product of factors" is unique)!

For example, $60 = 2 * 2 * 3 * 5 = 2^2 * 3 * 5$. And this — other than changing the order of the factors — is the only way in which the number 60 can be factorised.

If you allow numbers other than primes as factors, there are several ways of factorising integers and the *uniqueness* is lost:

$$60 = 1 * 60 = 2 * 30 = 4 * 15 = 5 * 12 = 6 * 10 = 2 * 3 * 10 = 2 * 5 * 6 = 3 * 4 * 5 = ...$$

The 1st fundamental theorem only appears to be obvious. We can construct numerous other sets of numbers [7] for which numbers in the set *cannot* be expressed uniquely as a product of the prime numbers of the set.

In order to make a mathematical statement, therefore, it is important to state not only the operation for which it is defined but also the basic set on which the operation is defined.

For more details on prime numbers (e.g. how Fermats Little Theorem can be used to test extremely large numbers to determine whether they are prime), please refer to the article on prime numbers in this script.

[7] These sets are formed especially from the set of natural numbers and do not contain the number 1. An example of this can be found in this tutorial on page 10.

3.4 Divisibility, modulus and remainder classes

If whole numbers are added, subtracted or multiplied, the result is always another whole number.

The division of two whole numbers does not always result in a whole number. For example, if we divide 158 by 10 the result is the decimal number 15.8, which is not a whole number!

If, however, we divide 158 by 2 the result 79 is a whole number. In number theory we express this by saying that 158 is *divisible* by 2 but not by 10. In general, we say:

Definition 3. *A whole number n is* **divisible** *by a whole number d if the quotient n/d is a whole number c such that $n = c * d$.*

n is called a *multiple* of d, whereas d is called a *divisor* or factor of n.

The mathematical notation for this is $d|n$ (read d divides n). The notation $d \not| n$ means that d does not divide the number n.

In our example therefore: $10 \not| 158$ but $2|158$.

3.4.1 The modulo operation – working with congruence

When we investigate divisibility, it is only the remainder of the division that is important. When dividing a number n by m, we often use the following notation:

$$\frac{n}{m} = c + \frac{r}{m},$$

where c is a whole number and r is a number with the values $0, 1, ..., m-1$. This notation is called division with remainder, whereby c is called the whole-number quotient and r is the remainder of the division.

Example:

$$\frac{19}{7} = 2 + \frac{5}{7} \quad (c = 2, r = 5)$$

What do the numbers $5, 12, 19, 26, ...$ have in common for division by 7? The remainder is always $r = 5$. For division by 7, only the following remainders are possible:

$$r = 0, 1, 2, .., 6.$$

The numbers that result in the same remainder r when divided by 7 are combined to form the remainder class r modulo 7. Two numbers a and b belonging to the same remainder class modulo 7 are said to be congruent modulo 7. Or in general:

Definition 4. *The* **remainder class r modulo m** *is the set of all whole numbers a that have the same remainder r when divided by m.*

Examples:

Remainder class 0 modulo 4 $= \{x | x = 4*n;\ n \in \mathbb{N}\} = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$

Remainder class 3 modulo 4 $= \{x | x = 4*n+3;\ n \in \mathbb{N}\} = \{\ldots, -13, -9, -5, -1, 3, 7, 11, 15, \ldots\}$

As only the remainders $0, 1, 2, ..., m-1$ are possible for division modulo m, modular arithmetic works with finite sets. For each modulo m there are precisely m remainder classes.

Definition 5. *Two numbers $a, b \in \mathbb{N}$ are said to be congruent modulo $m \in \mathbb{N}$ if and only if they have the same remainder when divided by m.*

We write: $a \equiv b \pmod{m}$ (read a is congruent b modulo m), which means that a and b belong to the same remainder class. The modulo is therefore the divisor. This notation was introduced by Gauss. Although the divisor is usually positive, a and b can also be any whole numbers.

Examples:

$19 \equiv 12 \pmod{7}$, because the remainders are equal: $19/7 = 2$ remainder 5 and $12/7 = 1$ remainder 5.

$23103 \equiv 0 \pmod{453}$, because $23103/453 = 51$ remainder 0 and $0/453 = 0$ remainder 0.

Theorem 3. *$a \equiv b$ (mod m) if and only if, the difference $(a-b)$ is divisible by m, i.e. if $q \in \mathbb{Z}$ exists with $(a-b) = q*m$.*

These two statements are therefore equivalent.

Therefore: If m divides the difference, there exists a whole number q such that: $a = b + q*m$. As an alternative to the congruence notation, we can also use the divisibility notation: $m|(a-b)$.

Example of equivalent statements:
$35 \equiv 11 \pmod{3} \Longleftrightarrow 35-11 \equiv 0 \pmod{3}$, where $35-11 = 24$ is divisible by 3 without remainder while $35 : 3$ and $11 : 3$ leave the remainder 2.

Comment:
The above equivalence does not apply to the sum $(a+b)$!

Example:
$11 \equiv 2 \pmod{3}$, therefore $11 - 2 \equiv 9 \equiv 0 \pmod{3}$; but $11 + 2 = 13$ is not divisible by 3. The statement in Theorem 3 does not even apply to sums in one direction. It is correct for sums only if the remainder is 0 and only in the following direction: if a divisor divides both summands with no remainder, it also divides the sum with no remainder.

We can apply the above equivalence in Theorem 3 if we need a quick and easy method of determining whether large numbers are divisible by a certain number.

Example:
Is $69,993$ divisible by 7?
The number can be written in the form of a difference in which it is clear that each operand is divisible by 7: $69,993 = 70,0007$. Therefore, the difference is also divisible by 7.

Although these considerations and definitions may seem to be rather theoretical, we are so familiar with them in everyday life that we no longer think about the formal procedure. For example,

the 24 hours on a clock are represented by the numbers $1, 2, ..., 12$. We obtain the hours after 12 noon as the remainder of a division by 12 and know immediately that 2 oclock in the afternoon is the same as 14.00.

This modular arithmetic (based on division remainders) forms the basis of asymmetric encryption procedures. Cryptographic calculations are therefore not based on real numbers, as the calculations you performed at school, but rather on character strings with a limited length, in other words on positive whole numbers that cannot exceed a certain value. This is one of the reasons why we choose a large number m and calculate modulo m". That is, we ignore whole-number multiples of m and, rather than working with a number, we only work with the remainder when this number is divided by m. The result is that all results are in the range 0 to $m - 1$.

3.5 Calculations with finite sets

3.5.1 Laws of modular calculations

From algebra theorems it follows that essential parts of the conventional calculation rules are kept when we proceed to modular calculations over a basic set $\mathbb{Z}$. For example, addition remains commutative. The same goes for multiplication modulo m. The result of a division [8] is not a fraction but rather a whole number between 0 and $m - 1$.

The known laws apply:

1. **Associative law:**
   $((a + b) + c) \pmod m \equiv (a + (b + c)) \pmod m$
   $((a * b) * c) \pmod m \equiv (a * (b * c)) \pmod m$

2. **Commutative law:**
   $(a + b) \pmod m \equiv (b + a) \pmod m$
   $(a * b) \pmod m \equiv (b * a) \pmod m$

The associative law and commutative law apply to both addition and multiplication.

3. **Distributive law:**
   $(a * (b + c)) \pmod m \equiv (a * b + a * c) \pmod m$

4. **Reducibility:**
   $(a + b) \pmod m \equiv (a \pmod m + b \pmod m) \pmod m$
   $(a * b) \pmod m \equiv (a \pmod m * b \pmod m) \pmod m$

The order in which order the modulo operation is performed is irrelevant.

5. **Existence of an identity:**
   $(a + 0) \pmod m \equiv 0 + a \pmod m \equiv a \pmod m$
   $(a * 1) \pmod m \equiv 1 * a \pmod m \equiv a \pmod m$

[8] When dividing modulo m we cannot use every number because some numbers have the same property as zero. See footnote 11.

6. **Existence of an inverse element:**
   For all whole numbers a and m there exists a whole number $-a$ such that:
   $a + (-a) \pmod{m} \equiv 0 \pmod{m}$ (additive inverse)
   For each a ($a \neq 0 \pmod{p}$) where p is prime there exists a whole number a^{-1}, such that:
   $(a * a^{-1}) \pmod{p} \equiv 1 \pmod{p}$ (multiplicative inverse).

7. **Closedness**[9]:
   $a, b \in G \Longrightarrow (a + b) \in G$
   $a, b \in G \Longrightarrow (a * b) \in G$

3.6 Examples of modular calculations

As we have already seen:

For two natural numbers a and m, a mod m denotes the remainder obtained when we divide a by m. This means that $a \pmod{m}$ is always a number between 0 and $m - 1$.

For example, $1 \equiv 6 \equiv 41 \equiv 1 \pmod{5}$ because the remainder is always 1. Another example is: $2000 \equiv 0 \pmod{4}$ because 4 divides 2000 with no remainder.

Modular arithmetic only contains a limited quantity of non-negative numbers. The number of these is specified by a modulus m. If the modulo is $m = 5$, then only the 5 numbers in the set $\{0, 1, 2, 3, 4\}$ are used.

A calculation result larger than 4 is then transformed modulo 5. In other words, it is the remainder when the result is divided by 5. For example, $2 * 4 \equiv 8 \equiv 3 \pmod{5}$ because 3 is the remainder when we divide 8 by 5.

3.6.1 Addition und multiplication

The following shows the addition table [10] (mod 5) and the multiplication tables[11] for mod 5 and mod 6.

Example of an addition table

The result when we add 3 and 4 (mod 5) is calculated as follows: Calculate $3 + 4 = 7$ and keep subtracting 5 from the result until the result is less than the modulo: $7 - 5 = 2$. Therefore: $3 + 4 \equiv 2 \pmod{5}$.

[9] The property of closedness is always defined in relation to an operation in a set. See Appendix B

[10] Comment on subtraction modulo 5:
$2 - 4 \equiv -2 \equiv 3 \bmod 5$.
It is therefore not true modulo 5 that $-2 = 2$ (see also Appendix C).

[11] Comment on division modulo 6: Due to the special role of zero as the identity for addition, division by zero is not permitted:// XXX XXX

Addition table modulo 5:	+	0	1	2	3	4
	0	0	1	2	3	4
	1	1	2	3	4	0
	2	2	3	4	0	1
	3	3	4	0	1	2
	4	4	0	1	2	3

Example of an multiplication table:

The result of the multiplication $4 * 4 \pmod 5$ is calculated as follows: $4 * 4 = 16$ and subtract 5 until the result is less than the modulus.

$$16 - 5 = 11;\ 11 - 5 = 6;\ 6 - 5 = 1.$$

The table directly shows that $4 * 4 \equiv 1 \pmod 5$ because $16 : 5 = 3$ remainder 1. Multiplication is defined on the set $\mathbb{Z}$ excluding 0.

Multiplication table modulo 5:	*	1	2	3	4
	1	1	2	3	4
	2	2	**4**	**1**	3
	3	3	**1**	**4**	2
	4	4	3	2	1

3.6.2 Additive und multiplicative inverses

You can use the tables to read the inverses for each number in relation to addition and multiplication.

The inverse of a number is the number that gives the result 0 when the two numbers are added and 1 when they are multiplied. Thus, the inverse of 4 for addition mod 5 is 1 and the inverse of 4 for multiplication mod 5 is 4 itself, because

$$\begin{aligned} 4 + 1 &= 5 \equiv 0 \pmod 5; \\ 4 * 4 &= 16 \equiv 1 \pmod 5. \end{aligned}$$

The inverse of 1 for multiplication mod 5 is 1, while the inverse modulo 5 of 2 is 3 and, since multiplication is commutative, the inverse of 3 is again 2.

If we take a random number and add or multiply another number (here 4) and then add[12] or multiply the corresponding inverse (1 or 4) to the interim result (1 or 3), then the end result is the same as the initial value.

[12] In general $x + y + (-y) \equiv x \pmod m$ ($(-y)$ = additive inverse of $y \pmod m$)

Example:

$$2+4 \equiv 6 \equiv 1 \pmod 5; \quad 1+1 \equiv 2 \equiv 2 \pmod 5,$$
$$2*4 \equiv 8 \equiv 3 \pmod 5; \quad 3*4 \equiv 12 \equiv 2 \pmod 5.$$

In the set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$, all numbers have a **unique** inverse modulo 5 both for addition and for multiplication (except for 0).

In the case of modular addition, this is true for every modulo (not just for 5). This is not the case, however, for modular multiplication. Here a natural number a from the set $\{1, ..., m-1\}$ has one inverse if and only if it and the modulo m are coprime[13], in other words if a and m have no common prime factors.

Since $m = 5$ is prime, the numbers 1 to 4 are relatively prime to 5 and **each** of these numbers has a multiplicative inverse in mod 5.

A counterexample shows the multiplication table for mod 6 (since the modulus $m = 6$ is not prime, not all elements from $\mathbb{Z}_6 \setminus \{0\}$ are relatively prime to 6):

Multiplication table modulo 6:

*	1	2	3	4	5
1	1	2	3	4	5
2	2	**4**	**0**	**2**	4
3	3	**0**	**3**	**0**	3
4	4	**2**	**0**	**4**	2
5	5	4	3	2	1

In addition to 0, the numbers 2, 3 and 4 also have no unique inverse (we can also say they have **no** inverse, because the elementary property of an inverse is uniqueness).

The numbers 2, 3 and 4 have the factor 2 or 3 in common with the modulus 6. Only the numbers 1 and 5, which are relatively prime to 6, have multiplicative inverses, namely themselves.

The number of numbers that are relatively prime to the modulus m is the same as the number of numbers that have a multiplicative inverse (see the Euler function below).

For the two moduli 5 and 6 used in the multiplication tables, this means: the modulus 5 is a prime number itself. In mod 5, therefore, there are exactly $J(5) = 5 - 1 = 4$ numbers that are relatively prime to the modulus, that is all numbers from 1 to 4.

Since 6 is not a prime number, we write it as a product of its factors: $6 = 2 * 3$. In mod 6, therefore, there are exactly $J(6) = (2-1)*(3-1) = 1*2 = 2$ numbers that have a multiplicative inverse, that is 1 and 5.

Although it may seem difficult to calculate the table of multiplicative inverses for large moduli (this only applies to the areas of the table shaded dark grey), we can use Fermats Little Theorem

[13] Other names for coprime are relatively prime or pairwise prime. Two whole numbers a and b are coprime if and only if $\gcd(a, b) = 1$. If p is prime and a is a random whole number that is not a multiple of p, then p and a are coprime.

to create a simple algorithm for this [Pfleger1997, p. 80]. Quicker algorithms are described, for instance, in [Knuth1998] [14].

The following three examples[15] illustrate the properties of multiplicative inverses.
In the multiplication table mod 17, the following was calculated for $i = 1, 2, ..., 18$:

$(5 * i)/17 = a$ remainder r und $5 * i \equiv 1 \pmod{17}$

$(6 * i)/17 = a$ remainder r und $6 * i \equiv 1 \pmod{17}$

We need to find the i for which the product remainder has the value 1.

Multiplication table modulo 17

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5*i	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
remainder	5	10	15	3	8	13	**1**	6	11	16	4	9	14	2	7	12	0	5
6*i	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108
remainder	6	12	**1**	7	13	2	8	14	3	9	15	4	10	16	5	11	0	6

Between $i = 1, ..., m$, all values between $0, ..., m-1$ occur for the remainders, because both 5 and 6 are also relatively prime to the modulus $m = 17$.

The multiplicative inverse of 5 **(mod** 17**) is** 7**, while the inverse of** 6 **(mod** 17**) is** 3**.**

Multiplication table modulo 13

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5*i	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
remainder	5	10	2	7	12	4	9	**1**	6	11	3	8	0	5	10	2	7	12
6*i	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108
remainder	6	12	5	11	4	10	3	9	2	8	**1**	7	0	6	12	5	11	4

Between $i = 1, ..., m$, all values between $0, ..., m-1$ occur for the remainders, because both 5 and 6 are also relatively prime to the modulus $m = 13$.

The multiplicative inverse of 5 **(mod** 13**) is** 8**, while the inverse of** 6 **(mod** 13**) is** 11**.**

[14] Using Euclids extended theorem (extended gcd), we can calculate the multiplicative inverse and determine whether numbers have an inverse (see appendix A). Alternatively, we can also use the primitive roots.

[15] Mathematica can be used to calculated the tables for these examples as follows:

```
m = 17; iWidth = 18; iFactor1 = 5; iFactor2 = 6;
Print[ ''i '', Table[ i, {i, 1, iWidth } ] ];
Print[ iFactor1, ''*i '', Table[ iFactor1*i, {i, 1, iWidth } ] ];
Print[ ''remainder '', Table[ Mod[iFactor1*i, m], {i, 1, iWidth } ] ];
Print[ iFactor2, ''*i '', Table[ iFactor2*i, {i, 1, iWidth } ] ];
Print[ ''remainder '', Table[ Mod[iFactor2*i, m], {i, 1, iWidth } ] ];
```

Multiplication table modulo 12

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5*i	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
remainder	5	10	3	8	**1**	6	11	4	9	2	7	0	5	10	3	8	1	6
6*i	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108
remainder	6	0	6	0	6	0	6	0	6	0	6	0	6	0	6	0	6	0

We have calculated $5 * i \pmod{12}$ and $6 * i \pmod{12}$. Between $i = 1, ..., m$, not all values between $0, ..., m-1$ occur and 6 does not have an inverse mod 12, because 6 and the modulus $m = 12$ are not coprime.

The multiplicative inverse of 5 **(mod** 12**) is** 5**.**

3.6.3 Raising to the power

In modular arithmetic, raising to the power is defined as repeated multiplication – as usual except that multiplication is now slightly different. We can even apply the usual rules, such as:

$$\begin{aligned} a^{b+c} &= a^b * a^c \\ (a^b)^c &= a^{b*c} = a^{c*b} = (a^c)^b. \end{aligned}$$

Modular powers work in the same way as modular addition and modular multiplication:

$$3^2 \equiv 9 \equiv 4 \pmod{5}.$$

Even consecutive powers work in the same way:

Example 1:

$$(4^3)^2 \equiv 64^2 \equiv 4096 \equiv 1 \pmod{5}.$$

(1) We can speed up[16] the calculation by reducing the **interim results** modulo 5 but we need to take care because not everything then works in the same way as in standard arithmetic.

$$\begin{aligned} (4^3)^2 &\equiv (4^3 \pmod{5})^2 \pmod{5} \\ &\equiv (64 \pmod{5})^2 \pmod{5} \\ &\equiv 4^2 \pmod{5} \\ &\equiv 16 \equiv 1 \pmod{5}. \end{aligned}$$

(2) In standard arithmetic, consecutive powers can be reduced to a single power by multiplying the exponents:

$$(4^3)^2 = 4^{3*2} = 4^6 = 4096.$$

[16] The time required to calculate the multiplication of two numbers normally depends on the length of the numbers. We can observe this if we use the school method to calculate, for instance, $474 * 228$. The time required increases quadratically, because we need to multiply $3 * 3$ numbers. The numbers become considerably smaller if we reduce the interim result.

This is not quite as simple in modular arithmetic because this would give:

$$(4^3)^2 \equiv 4^{3*2 \pmod 5} \equiv 4^{6 \pmod 5} \equiv 4^1 \equiv 4 \pmod 5.$$

But as we saw above, the correct result is 1!!

(3) Therefore, the rule is slightly different for consecutive powers in modular arithmetic: we do not multiply the exponents in (mod m) but rather in (mod $J(m)$).

Using $J(5) = 4$ gives:

$$(4^3)^2 \equiv 4^{3*2 \pmod{J(5)}} \equiv 4^{6 \bmod 4} \equiv 4^2 \equiv 16 \equiv 1 \pmod 5.$$

This delivers the correct result.

Theorem 4. $(a^b)^c \equiv a^{b*c \pmod{J(m)}} \pmod m$.

Example 2:

$$3^{28} \equiv 3^{4*7} \equiv 3^{4*7 \pmod{10}} \equiv 3^8 \equiv 6561 \equiv 5 \pmod{11}.$$

3.6.4 Roots and logarithms

The inverses of the powers are also defined. The roots and logarithms are again whole numbers. Yet in contrast to the usual situation, they are not only difficult to calculate but, in the case of large numbers, cannot be calculated at all within a reasonable space of time.

Let us take the equation $a \equiv b^c \pmod m$.

a) **Taking the logarithm (determining c):** If we know a and b of the three numbers a, b and c that meet this equation, then every known method of finding c is approximately just as time-consuming as trying out all m possible values for c one after the other. For a typical m of the order of magnitude of 10^{180} for 600-digit binary numbers, this is a hopeless undertaking. More precisely, for suitably large numbers m, the time required according to current knowledge is proportional to $\exp\left(C * (\log m[\log\log m]^2)^{1/3}\right)$ with a constant $C > 1$.

b) **Calculating the root (determining b):** The situation is similar if b is the unknown variable and we know the values of a and c:
If we know the Euler function of m, $J(m)$, then we can easily[17] calculate d with $c * d \equiv 1 \pmod{J(m)}$ and use Theorem 4 to obtain:

$$a^d \equiv (b^c)^d \equiv b^{c*d} \equiv b^{c*d \ (mod\ J(m))} \equiv b^1 \equiv b \pmod m$$

the *c-th root* b of a.

If $J(m)$ cannot be determined[18], it is difficult to calculate the c-th root. This forms the basis for the security assumption used by the RSA encryption system (see Sub-section 4.3.1: the RSA procedure).

[17] See Appendix A: the greatest common divisor (gcd) of whole numbers.

[18] According to the first fundamental theorem of number theory and Theorem 9, we determine $J(m)$ by reducing m to prime factors.

The time required for inverting addition and multiplication, on the other hand, is simply proportional to $\log m$ or $(\log m)^2$. Powers (for a number x calculate x^a with a fixed) and exponents (for a number x calculate a^x with a fixed) are therefore typical one-way functions (XXX See Section Script and Subsection artikelXXX).

3.7 Groups and modular arithmetic

Mathematical *groups* play a decisive role in number theory and cryptography. We only talk of groups if, for a defined set and a defined relation (an operation such as addition or multiplication), the following properties are fulfilled:

- The set is closed
- A neutral element exists
- An inverse element exists
- The associative law applies.

The abbreviated mathematical notation is $(G, +)$ or $(G, *)$.

$\mathbb{Z}_n$: $\mathbb{Z}_n$ comprises all numbers from 0 to $n-1$: $\mathbb{Z}_n = \{0, 1, 2, ..., n-2, n-1\}$. $\mathbb{Z}_n$ is sometimes also called the remainder set R modulo n.

For example, 32-bit computers (standard PCs) only directly work with whole numbers in a finite set, that is the value range $0, 1, 2, ..., 2^{32} - 1$.

This value range is equivalent to the set $\mathbb{Z}_{2^{32}}$.

3.7.1 Addition in a group

If we define the operation mod+ on such a set where

$$a \bmod + \ b := (a + b) \pmod{n},$$

then the set $\mathbb{Z}_n$ together with the relation mod+ is a group because the following properties of a group are valid for all elements in $\mathbb{Z}_n$:

- $a \bmod + \ b$ is an element of $\mathbb{Z}_n$ (the set is closed),
- $(a \bmod + \ b) \bmod + \ c \equiv a \bmod + \ (b \bmod + \ c)$ (mod+ is associative),
- the neutral element is 0.
- each element $a \in \mathbb{Z}_n$ has an inverse for this operation, namely $n - a$
  (because $a \bmod + \ (n - a) \equiv a + (n - a) \pmod{n} \equiv n \equiv 0 \pmod{n}$).

Since the operation is commutative, i.e. $(a \bmod + \ b) = (b \bmod + \ a)$, this structure is even a commutative group.

3.7.2 Multiplication in a group

If we define the operation mod* on the set $\mathbb{Z}_n$ where

$$a \bmod * \ b := (a * b) \pmod{n},$$

then $\mathbb{Z}_n$ together with this operation is **usually not a group** because not all properties are fulfilled for each n.

Examples:

a) In $\mathbb{Z}_{15}$, for example, the element 5 does not have an inverse. That is to say, there is no a with
$5 * a \equiv 1 \pmod{15}$. Each modulo product with 5 on this set gives $5, 10$ or 0.

b) In $\mathbb{Z}_{55} \setminus \{0\}$, for example, the elements 5 and 11 do not have multiplicative inverses. That is to say, there is no $a \in \mathbb{Z}_{55}$ such that $5 * a1$ (*mod* 55) and no a such that $11 * a \equiv 1$ (*mod* 55).

This is because 5 and 11 are not relatively prime to 55. Each modulo product with 5 on this set gives $5, 10, 15, \ldots, 50$ or 0. Each modulo product with 11 on this set gives $11, 22, 33, 44$ or 0.

On the other hand, there are subsets of $\mathbb{Z}_n$ that form a group with the operation mod*. If we choose all elements in $\mathbb{Z}_n$ that are relatively prime to n, then this set forms a group with the operation mod*. We call this set $\mathbb{Z}_n^*$.

Definition 6. $\mathbb{Z}_n^*$:

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | \text{gcd}(a, n) = 1\}.$$

$\mathbb{Z}_n^*$ is sometimes also called the reduced remainder set R' modulo n.

Example: For $m = 10$ the following applies: $J(m) = 4$ and

full remainder set $R = \mathbb{Z}_n = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

reduced remainder set $R' = \mathbb{Z}_n^* = \{1, 3, 7, 9\}$.

Comment: R' or $\mathbb{Z}_n^*$ is always a genuine subset of R or $\mathbb{Z}_n$ because 0 is always an element of R but never an element of R'. Since 1 and $m - 1$ are always relatively prime to m, they are always elements of both sets.

If we select a random element in $\mathbb{Z}_n^*$ and multiply it by every other element in $\mathbb{Z}_n^*$, then the products [19] are all in $\mathbb{Z}_n^*$, and the results are also a unique permutation of the elements in $\mathbb{Z}_n^*$. Since 1 is always an element of $\mathbb{Z}_n^*$, there is a unique partner " in this set such that the product is 1. In other words:

Theorem 5. *Each element in $\mathbb{Z}_n^*$ has a multiplicative inverse.*

[19] This is due to the fact that $\mathbb{Z}_n^*$ is closed with respect to the multiplication and due to the gcd property: $a, b \in \mathbb{Z}_n^* \Rightarrow [\text{ggT}(a, n) = 1, \text{ggT}(b, n) = 1] \Rightarrow [\text{ggT}(a * b, n) = 1] \Rightarrow (a * b) \pmod{n} \in \mathbb{Z}_n^*$.

Example for $a = 3$ modulo 10 with $\mathbb{Z}_n^* = \{1, 3, 7, 9\}$:

$$\begin{aligned} 3 &\equiv 3 * 1 \pmod{10} \\ 9 &\equiv 3 * 3 \pmod{10} \\ 1 &\equiv 3 * 7 \pmod{10} \\ 7 &\equiv 3 * 9 \pmod{10} \end{aligned}$$

The unique inverse is an essential condition for cryptography (see Section 3.10: Proof of the RSA procedure with Euler-Fermat).

3.8 Euler function, Fermat's Little Theorem and Euler–Fermat

Given n, the number of numbers from the set $\{1, ..., n-1\}$ that are relatively prime to n is equal to the value of the Euler function $J(n)$.

Definition 7. [20] *The Euler function*[21] $J(n)$ *specifies the number of elements in* $\mathbb{Z}_n^*$.

$J(n)$ also specifies how many whole numbers have multiplicative inverses in mod n. $J(n)$ can be calculated if we know the prime factors of n.

Theorem 6. *For a prime number, the following is true:* $J(p) = p - 1$

Theorem 7. *If* m *is the product of two distincts primes, then:*

$$J(p * q) = (p - 1) * (q - 1) \quad \text{or} \quad J(p * q) = J(p) * J(q)$$

This case is important for the RSA procedure.

Theorem 8. *If* $m = p_1 * p_2 * ... * p_k$ *where* p_1 *to* p_k *are distinct prime numbers (i.e. no factor occurs more than once), then the following is true (as a generalisation of Theorem 7):*

$$J(m) = (p_1 - 1) * (p_2 - 1) * ... * (p_k - 1).$$

Theorem 9. *In general, the following is true for every prime number* p *and every* n *in* $\mathbb{N}$*:*

1. $J(p^n) = p^{n-1} * (p - 1)$
2. *If* $n = p_1^{e_1} * p_2^{e_2} * ... * p_k^{e_k}$, *where* p_1 *to* p_k *are distinct prime numbers, then:*

$$J(n) = [(p_1^{e_1 - 1}) * (p_1 - 1)] * ... * [(p_k^{e_k - 1}) * (p_k - 1)] = n * ([(p_1 - 1)/p_1] * ... * [(p_k - 1)/p_k])$$

Example:
$n = 2.701.125 = 3^2 * 5^3 * 7^4 \Longrightarrow J(n) = [3^1 * 2] * [5^2 * 4] * [7^3 * 6] = 1.234.800$

In order to prove the RSA procedure, we need Fermat's theorem and its generalisation (Euler-Fermat theorem).

[20] Leonhard Euler, Swiss mathematician, 15.4.1707 – 18.9.1783
[21] Is often described as the phi function $\Phi(n)$.

Theorem 10. Fermat's Little Theorem [22] *Let p be a prime number and a be a random whole number, then:*

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

An alternative formulation of Fermat's Little Theorem is as follows: Let p be a prime number and a be a random whole number that is relatively prime to p, then:

$$a^{p-1} \equiv 1 \pmod{p}.$$

Theorem 11. Euler-Fermat theorem (generalisation of Fermat's Little Theorem) *For all elements a in the group $\mathbb{Z}_n^*$ (i.e. a and n are natural numbers that are coprime):*

$$a^{J(n)} \equiv 1 \pmod{n}.$$

This theorem states that if we raise a group element (here a) to the power of the order of the group (here $J(n)$), we always obtain the neutral element for multiplication (the number 1).

The 2nd formulation of Fermat's Little Theorem is derived directly from Euler's theorem if n is a prime number.

If n is the product of two prime numbers, we can - in certain cases - use Euler's theorem to calculate the result of a modular power very quickly. We have: $a^{(p-1)*(q-1)} \equiv 1 \pmod{pq}$.

Examples:

- With $2 = 1 * 2$ and $6 = 2 * 3$ where 2 and 3 are both prime; $J(6) = 2$ because only 1 and 5 are relatively prime to 6, we obtain the equation $5^2 \equiv 5^{J(6)} \equiv 1 \pmod{6}$, without having to calculate the power.
- With $792 = 22 * 36$ and $23 * 37 = 851$ where 23 and 37 are both prime, it follows that $31^{792} \equiv 31^{J(23*37)} \equiv 31^{J(851)} \equiv 1 \pmod{851}$.

Another interesting application is a special case of determining the multiplicative inverses using the Euler-Fermat theorem (multiplicative inverses are otherwise determined using the extended Euclidean algorithm).

Example:

Find the multiplicative inverse of 1579 modulo 7351.
According to Euler-Fermat: $a^{J(n)} = 1 \pmod{n}$ for all a in $\mathbb{Z}_n^*$. If we divide both sides by a, we get: $a^{J(n)-1} \equiv a^{-1} \pmod{n}$. For the special case that the modulo is prime, we have $J(n) = p-1$. Therefore, the modular inverse is

$$a^{-1} = a^{J(n)-1} \equiv a^{(p-1)-1} \equiv a^{p-2} \pmod{p}.$$

For our example, this means:

[22] Pierre de Fermat, French mathematician, 17.8.1601 – 12.1.1665

Since the modulus 7351 is prime, $p - 2 = 7349$.
$1579^{-1} \equiv 1579^{7349} \pmod{p}$

By cleverly breaking down the exponent, we can calculate this power relatively easily (see Section 3.11: High powers):

$$7349 = 4096 + 2048 + 1024 + 128 + 32 + 16 + 4 + 1$$

$$1579^{-1} = 4716 \pmod{7351}.$$

3.9 Multiplicative order and primitive roots

The multiplicative order and the primitive root are two useful constructs (concepts) in elementary number theory.

Definition 8. *The* **multiplicative order** $\mathrm{ord}_m(a)$ *of a whole number (a mod m) (where a and m are coprime) is the smallest whole number e for which* $a^e \equiv 1 mod m$.

The following table[23] shows that in a multiplicative group (here $\mathbb{Z}_{11}^*$) not all numbers necessarily have the same order. The orders in this case are $1, 2, 5$ and 10 and we notice that:

1. The orders are all factors of 10.
2. The numbers $a = 2, 6, 7$ and 8 have the order 10 - we say that these numbers have the **maximum order** in $\mathbb{Z}_{11}^*$.

Example:
The following table shows the values a^i mod 11 for the exponents $i = 1, 2, ..., 10$ and for the bases $a = 1, 2, ..., 10$ as well as the resulting value $ord_{11}(a)$ for each a:

The table shows, for example, that the order of 3 modulo 11 has the value 5.

Definition 9. *If a and m are coprime and if* $ord_m(a) = J(m)$ *(i.e. a has maximum order), then we say that a is a* **primitive root** *of m.*

A number a is not a primitive root for every modulo m. In the above table, only $a = 2, 6, 7$ and 8 is a primitive root with respect to mod 11 ($J(11) = 10$).

Using the primitive roots, we can clearly establish the conditions for which powers modulo m have a unique inverse and the calculation in the exponents is manageable.

According to Theorem 4, the arithmetic operations of modular expressions are performed in the exponents modulo $J(n)$ rather than modulo n [24]. In $a^{e*d} = a^1 \pmod{n}$, if we wish to determine the inverses for the factor e in the exponent, we need to calculate modulo $J(n)$.

[23] Mathematica can be used to calculate this as follows:
`m=11; Table[ Mod[a^i, 11], {a, 1, m-1}, i, 1, m-1 ]`

[24] For the following example, we will adopt the usual practice for the RSA procedure of using n rather than m to denote the modulus.

Values of $a^i \bmod 11, 1 \leq a, i < 11$:

	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10	$ord_{11}(a)$
a=1	1	1	1	1	1	1	1	1	1	1	1
a=2	2	4	8	5	10	9	7	3	6	1	10
a=3	3	9	5	4	**1**	3	9	5	4	1	5
a=4	4	5	9	3	**1**	4	5	9	3	1	5
a=5	5	3	4	9	**1**	5	3	4	9	1	5
a=6	6	3	7	9	10	5	8	4	2	1	10
a=7	7	5	2	3	10	4	6	9	8	1	10
a=8	8	9	6	4	10	3	2	5	7	1	10
a=9	9	4	3	5	**1**	9	4	3	5	1	5
a=10	10	1	10	1	10	1	10	1	10	1	2

Example (with reference to the RSA algorithm):
If we calculate modulo 26, which set can e and d come from?

Solution: we have $e * d \equiv 1 \pmod{J(26)}$.

The reduced remainder set $R' = \mathbb{Z}_{26}^* = \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$.

The reduced remainder set R'' contains only the elements of R' that are relatively prime to $J(n) = 12 : R'' = \{1, 5, 7, 11\}$.

For every e in R'' there exists a d in R'' such that $a \equiv (a^e)^d \pmod{n}$. For every e in R'', there exists therefore precisely one element (not necessarily different from e) such that $e * d \equiv 1 \pmod{J(26)}$.
For all e that are relatively prime to $J(n)$ we could calculate d as follows using the Euler-Fermat theorem: For $a^{J(n)} \equiv 1 \pmod{n}$ is the same as saying $a^{J(n)-1} \equiv a^{-1} \pmod{n}$. Therefore

$$d \equiv e^{-1} \pmod{J(n)} \equiv e^{J(J(n))-1} \pmod{J(n)}.$$

The problems of factorising $n = pq$ with $q \neq p$ and finding $J(n)$ have a similar degree of difficulty and if we find a solution for one of the two problems, we also have a solution for the other[25].

The following two tables show the multiplicative orders and primitive roots modulo 45 and modulo 46.

Example:
The following table[26] shows the values $a^i \bmod 45$ for the exponents $i = 1, 2, ..., 10$ and for the bases $a = 1, 2, ..., 10$ as well as the resulting value $ord_{45}(a)$ for each a:

[25] XXX XXX

[26] Mathematica can be used to calculate this as follows (no Do loop can be used in Print and each Print command outputs a new line at the end):

```
m = 45;
Do[ Print[ Table[ Mod[a^i, m], {i, 1, 12} ],
'', '', MultiplicativeOrder[a, m, 1],
```

Values of $a^i \bmod 45, 1 \leq a, i < 13$:

$a \setminus i$	1	2	3	4	5	6	7	8	9	10	11	12	$ord_{45}(a)$	$J(45)$
1	1	1	1	1	1	1	1	1	1	1	1	1	1	24
2	2	4	8	16	32	19	38	31	17	34	23	1	12	24
3	3	9	27	36	18	9	27	36	18	9	27	36	—	24
4	4	16	19	31	34	1	4	16	19	31	34	1	6	24
5	5	25	35	40	20	10	5	25	35	40	20	10	—	24
6	6	36	36	36	36	36	36	36	36	36	36	36	—	24
7	7	4	28	16	22	19	43	31	37	34	13	1	12	24
8	8	19	17	1	8	19	17	1	8	19	17	1	4	24
9	9	36	9	36	9	36	9	36	9	36	9	36	—	24
10	10	10	10	10	10	10	10	10	10	10	10	10	—	24
11	11	31	26	16	41	1	11	31	26	16	41	1	6	24
12	12	9	18	36	27	9	18	36	27	9	18	36	—	24

$J(45)$ is calculated using Theorem 9: $J(45) = J(3^2 * 5) = 3^1 * 2 * 4 = 24$. Since 45 is not a prime, there is no multiplicative order for all values of a (e.g. for the numbers that are not relatively prime to 45 : $3, 5, 6, 9, 10, 12, ...$, because $45 = 3^2 * 5$).

Example:
Is 7 a primitive root modulo 45?
The requirement/condition $gcd(7, 45) = 1$ is fulfilled. The table shows that the number 7 is not a primitive root of 45, because $ord_{45}(7) = 12 \neq 24 = J(45)$.

Example:
The following table[27] answers the question as to whether the number 7 is a primitive root of 46. The requirement/condition $gcd(7, 46) = 1$ is fulfilled.

```
'', '', EulerPhi[m] ],
{a, 1, 12}
];
```

[27] Mathematica can be used to calculate this as follows:

```
m = 46; Do[ Print[ Table[ Mod[a^i, m], {i, 1, 23} ],
'', '', MultiplicativeOrder[a, m, 1],
'', '', EulerPhi[m] ],
{a, 1, 23} ];
```

Values of $a^i \bmod 46, 1 \leq a, i < 23$:

$a\backslash i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	ord
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	4	8	16	32	18	36	26	6	12	24	2	4	8	16	32	18	36	26	6	12	24	2	–
3	3	9	27	35	13	39	25	29	41	31	1	3	9	27	35	13	39	25	29	41	31	1	3	11
4	4	16	18	26	12	2	8	32	36	6	24	4	16	18	26	12	2	8	32	36	6	24	4	–
5	5	25	33	27	43	31	17	39	11	9	45	41	21	13	19	3	15	29	7	35	37	1	5	22
6	6	36	32	8	2	12	26	18	16	4	24	6	36	32	8	2	12	26	18	16	4	24	6	–
7	7	3	21	9	17	27	5	35	15	13	45	39	43	25	37	29	19	41	11	31	33	1	7	22
8	8	18	6	2	16	36	12	4	32	26	24	8	18	6	2	16	36	12	4	32	26	24	8	–
9	9	35	39	29	31	3	27	13	25	41	1	9	35	39	29	31	3	27	13	25	41	1	9	11
10	10	8	34	18	42	6	14	2	20	16	22	36	38	12	28	4	40	32	44	26	30	24	10	–
11	11	29	43	13	5	9	7	31	19	25	45	35	17	3	33	41	37	39	15	27	21	1	11	22
12	12	6	26	36	18	32	16	8	4	2	24	12	6	26	36	18	32	16	8	4	2	24	12	–
13	13	31	35	41	27	29	9	25	3	39	1	13	31	35	41	27	29	9	25	3	39	1	13	11
14	14	12	30	6	38	26	42	36	44	18	22	32	34	16	40	8	20	4	10	2	28	24	14	–
15	15	41	17	25	7	13	11	27	37	3	45	31	5	29	21	39	33	35	19	9	43	1	15	22
16	16	26	2	32	6	4	18	12	8	36	24	16	26	2	32	6	4	18	12	8	36	24	16	–
17	17	13	37	31	21	35	43	41	7	27	45	29	33	9	15	25	11	3	5	39	19	1	17	22
18	18	2	36	4	26	8	6	16	12	32	24	18	2	36	4	26	8	6	16	12	32	24	18	–
19	19	39	5	3	11	25	15	9	33	29	45	27	7	41	43	35	21	31	37	13	17	1	19	22
20	20	32	42	12	10	16	44	6	28	8	22	26	14	4	34	36	30	2	40	18	38	24	20	–
21	21	27	15	39	37	41	33	3	17	35	45	25	19	31	7	9	5	13	43	29	11	1	21	22
22	22	24	22	24	22	24	22	24	22	24	22	24	22	24	22	24	22	24	22	24	22	24	22	–
23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	–

$J(46)$ is calculated using Theorem 7: $J(46) = J(2*23) = 1*22 = 22$. The number 7 is a primitive root of 46, because $ord_{46}(7) = 2 = J(46)$.

Theorem 12. [28] *For a modulus $m = p$ and $a < m$: $a^i, i = 1, ..., p-1$ assumes all $J(p)$ values $1, ..., p-1$ once if and only if $ord_p(a) = J(p)$.*

3.10 Proof of the RSA procedure with Euler-Fermat

Using the Euler-Fermat theorem, we can prove the RSA [29] procedure in the group $\mathbb{Z}_n^*$.

[28] Compare the above table with the values for $a^i \bmod 11$, with $\leq a, i < 11$.

[29] The RSA procedure is the most common asymmetric cryptography procedure. Developed in 1978 by Ronald Rivest, Adi Shamir and Leonard Adleman, it can be used both for signatures and for encryption. Cryptographers always associate this procedure with the abbreviation RSA. However, please note that the abbreviation may be used in other fields to mean something completely different.

3.10.1 Basic idea of public key cryptography

The basic idea behind public key cryptography is that all participants possess a different pair of keys (P and S) and the public keys for all recipients are published. You can retrieve the public key P for a recipient from a directory just as you would look up someone's phone number in the phone book. Furthermore, each recipient has a secret key S that is needed in order to decrypt the message and that is not known to anyone else. If the sender wishes to send a message M, he encrypts it using the public key P of the recipient before sending it:

The ciphertext C is determined as $C = E(P; M)$, where E (encryption) is the encryption rule. The recipient uses his private key S to decrypt the message with the decryption rule $D : M = D(S; C)$.

In order to ensure that this system works for every message M, the following four requirements must be met:

1. $D(S; E(P; M)) = M$ for every M (invertibility)
2. All (S, P) pairs are different for all participants (i.e. lots of them are needed).
3. The time required to derive S from P is at least as high as the time required to decrypt M with no knowledge of S.
4. Both C and M can be calculated relatively easily.

The 1st requirement is a general condition for all cryptographic encryption algorithms.

The 2nd requirement can easily be met because there is a very large number of prime numbers[30] and because this can be ensured by a central office that issues certificates.

It is this last requirement that makes the procedure actually usable. This is because it is possible to calculate the powers in a linear amount of time (because there is a restriction on the length of the numbers).

Although Whitfield Diffie and Martin Hellman formulated the general method as early as 1976, an actual procedure that met all four requirements was only found later by Rivest, Shamir and Adleman.

3.10.2 How the RSA procedure works

The individual steps for implementing the RSA procedure can be described as follows (see [Eckert2001, p. 213 ff] and [Sedgewick1990, p. 338 ff]). Steps 1 to 3 constitute key generation, steps 4 and 5 are the encryption, and steps 6 and 7 are the decryption:

[30] According to the prime number theorem of Legendre and Gauss there are approximately $n/\ln(n)$ prime numbers up to the number n. This means, for example, that there are $6.5 * 10^{74}$ prime numbers under $n = 2^{256}$ $(= 1.1 * 10^{77})$ and $3.2 * 10^{74}$ prime numbers under $n = 2^{255}$. Between 2^{255} and 2^{256} there are therefore $3.3 * 10^{74}$ prime numbers with precisely 256 bits. This large number is also the reason why we cannot simply save them all.

1. Select two distincts random prime numbers[31] p and q and calculate $n = p * q$[32]. The value n is called the RSA modulus[33].

2. Select a random $e \in \{2, ..., n-1\}$ such that:
   e is relatively prime to $J(n) = (p-1) * (q-1)$. For example, we can select e such that: $\max(p, q) < e < J(n) - 1$ [34]. We can then throw away p and q.

3. Select $d \in \{1, ..., n-1\}$ with $e * d \equiv 1 \pmod{J(n)}$, i.e. d is the multiplicative inverse of e modulo $J(n)$ [35][36]. We can then throw away $J(n)$.

   $\rightarrow (n, e)$ is the public key P.

   $\rightarrow (n, d)$ is the secret key S (only d must be kept secret).

4. For encryption, the message represented as a (binary) number is divided into parts such that each part of the number is less than n.

5. Encryption of the plaintext (or the parts of it) $M \in \{1, ..., n-1\}$:

$$C = E((n, e); M) := M^e \pmod{n}.$$

6. For decryption, the ciphertext represented as a binary number is divided into parts such that each part of the number is less than n.

7. Decryption of the ciphertext (or the parts of it) $C \in \{1, ..., n-1\}$:

$$M = D((n, d); C) := C^d \pmod{n}.$$

The numbers d, e and n are usually extremely large (e.g. d and e 300 bits, n 600 bits).

3.10.3 Proof of requirement 1 (invertibility)

For pairs of keys (n, e) and (n, d) that possess fixed properties in steps 1 to 3 of the RSA procedure, the following must be true for all $M < n$:

$$M \equiv (M^e)^d \bmod n \quad \text{with} \quad (M^e)^d = M^{e*d}.$$

[31] XXX XXX

[32] XXX XXX

[33] In CrypTool the RSA modulo is denoted with a capital N.

[34] The procedure also allows us to select d freely and then calculate e. However, this has practical disadvantages. We usually want to be able to encrypt messages quickly, which is why we choose a public exponent e such that it has as few binary ones as possible. A number used often is $65537 = 2^{16} + 1$, or in binary: 10...0...01.

[35] For reasons of security, d should not be too small. On the other hand, we want to select the publicly known e to be an advantageous value that allows the exponential calculation to be performed quickly during encryption. The prime numbers 3, 17 and 65537 have proved to be particularly practical for this because their binary representations contain only a few ones.

[36] We start by determining either d or e depending on the implementation. Actually, e should be selected randomly for the encryption.

This means that the deciphering algorithm above works correctly.

We therefore need to show that: $M^{e*d} \equiv M \pmod{n}$

We will show this in 3 steps (see [Beutelspacher1996, p. 131ff]).

(a) In the first step we show that: $M^{e*d} \equiv M \pmod{p}$. This results from the requirements and from Euler's theorem. Since $n = pq$ and $J(pq) = (p-1)(q-1)$ and since e and d are selected in such a way that $e*d \equiv 1 \pmod{J(n)}$, there is a whole number k such that: $e*d = 1+k*(p-1)(q-1)$.

$$\begin{aligned} M^{e*d} &\equiv M^{1+k*J(n)} \equiv M * M^{k*J(n)} \equiv M * M^{k*(p-1)*(q-1)} \pmod{p} \\ &\equiv M * (M^{p-1})^{k*(q-1)} \pmod{p} \quad \text{based on Euler} - \text{Fermat}: \; M^{p-1} \equiv 1 \pmod{p} \\ &\equiv M * (1)^{k*(q-1)} \pmod{p} \\ &\equiv M \pmod{p} \end{aligned}$$

The requirement for using Euler's theorem was that M and p are relatively prime.

Since this is not true in general, we need to consider the case when M and p are not relatively prime. Since p is a prime number, this implies that p is a factor of M. But this means:

$$M \equiv 0 \pmod{p}.$$

If p is a factor of M, then p is also a factor of M^{e*d}. Therefore:

$$M^{e*d} \equiv 0 \pmod{p}.$$

Since p is a factor of both M and $Me*d$, it is also a factor of their difference:

$$(M^{e*d} - M) \equiv 0 \pmod{p}$$

And thus our conjecture is also true in this special case.

(b) In exactly the same way we prove that: $M^{e*d} \equiv M \pmod{q}$.

(c) We now combine the conjectures from (a) and (b) for $n = p*q$ to show that:

$$M^{e*d} \equiv M \pmod{n} \text{ for all } M < n.$$

From (a) and (b) we have $(M^{e*d} - M) \equiv 0 \pmod{p}$ and $(M^{e*d} - M) \equiv 0 \pmod{q}$. Therefore, p and q are both factors of the same number $z = (M^{e*d} - M)$. Since p and q are distinct prime numbers, their product must also be a factor of this number z. Thus:

$$(M^{e*d} - M) \equiv 0 \pmod{p*q} \;\text{ or }\; M^{e*d} \equiv M \pmod{p*q} \;\text{ or }\; M^{e*d} \equiv M \pmod{n}.$$

1st comment:
We can also condense the three steps if we use Theorem 11 (Euler-Fermat) - i.e. not the simplified theorem where $n = p$ and which corresponds to Fermat's Little Theorem:

$$(M^e)^d \equiv M^{e*d} \equiv M^{(p-1)(q-1)*k+1} \equiv (\underbrace{M^{(p-1)(q-1)}}_{\equiv M^{J(n)} \equiv 1 \pmod{n}})^k * M \equiv 1^k * M \equiv M \pmod{n}$$

2nd comment:
When it comes to signing messages, we perform the same operations but first use the secret key d, followed by the public key e. The RSA procedure can also be used to create digital signatures, because:

$$M \equiv (M^d)^e \pmod{n}.$$

3.11 High powers

RSA encryption and decryption entails calculating high powers modulo n. For example, the calculation $(100^5) \pmod 3$ exceeds the 32-bit long integer number range provided we calculate a^n by actually multiplying a with itself n times in line with the definition. In the case of extremely large numbers, even a fast computer chip would take longer than the age of the universe to calculate a single exponential. Luckily, there is an extremely effective shortcut for calculating exponentials (but not for calculating logarithms).

If the expression is divided differently using the rules of modular arithmetic, then the calculation does not even exceed the 16-bit short integer region:

$$(a^5) \equiv (((a^2 \pmod m))^2 \pmod m) * a) \pmod m.$$

We can generalise this by representing the exponent as a binary number. For example, the naive method would require 36 multiplications in order to calculate a^n for $n = 37$. However, if we write n in the binary representation as $100101 = 1 * 2^5 + 1 * 2^2 + 1 * 2^0$, then we can rewrite the expression as: $a^{37} = a^{2^5+2^2+2^0} = a^{2^5} * a^{2^2} * a^1$

Example:

$87^{43} \pmod{103}$.
Since $43 = 32+8+2+1$, 103 is prime, $43 < J(103)$ and the squares (mod 103) can be calculated beforehand
$(87^2 \equiv 50 \pmod{103}; 87^4 \equiv 50^2 \equiv 28 \pmod{103}; 87^8 \equiv 28^2 \equiv 63 \pmod{103}; 87^{16} \equiv 63^2 \equiv 55 \pmod{103}; 87^{32} \equiv 55^2 \equiv 38 \pmod{103})$, we have[37]:

$$\begin{aligned} 87^{43} &\equiv 87^{32+8+2+1} \pmod{103} \\ &\equiv 87^{32} * 87^8 * 87^2 * 87 \pmod{103} \\ &\equiv 38 * 63 * 50 * 87 \equiv 85 \pmod{103}. \end{aligned}$$

The powers $(a^2)^k$ can be determined easily by means of repeated squaring. As long as a does not change, a computer can calculate them beforehand and – if enough memory is available – save them. In order to then find a^n in each individual case, it now only needs to multiply those $(a^2)^k$ for which there is a one in the kth position of the binary representation of n. The typical effort is then reduced from 2^{600} to $2 * 600$ multiplications! This frequently used algorithm is called Square and Multiply.

[37] Using Mathematica this is calculated in less than a second on a Pentium 3:
`Mod[87^43, 87^2, 87^4, 87^8, 87^16, 87^32, 103] = {85, 50, 28, 63, 55, 38}.`

Joanne K. Rowling[38]:
It is our choices, that show what we truly are, far more than our abilities.

3.12 Examples for the application of number theory in cryptography

The results of modular arithmetic are used extensively in modern cryptography. Here we will provide a few examples from cryptography using small[39] numbers.

Enciphering a text entails applying a function (mathematical operation) to a character string (number) to generate a different number. Deciphering entails reversing this function, in other words using the distorted image that the function has created from the plaintext in order to restore the original image. For example, the sender could take the plaintext M of a confidential message and add a secret number, the key S, to obtain the ciphertext C:

$$C = M + S.$$

The recipient can reconstruct the plaintext by reversing this operation, in other words by subtracting S:

$$M = C - S.$$

Adding S reliably makes the plaintext impossible to read. However, this encryption is rather weak, because all an interceptor needs to do to calculate the key is obtain a plaintext and the associated ciphertext

$$S = C - M,$$

and can then read any subsequent messages encrypted using S.

3.12.1 One-way functions

The essential reason for this is that subtraction is just as simple an operation as addition. If the key is to be impossible to determine even with knowledge of both the plaintext and the ciphertext, we need a function that is, on the one hand, relatively easy to calculate - we don't want to have problems encrypting messages. On the other hand, the inverse function should exist (otherwise information would be lost during encryption), but should be de facto incalculable.

What are possible candidates for such a **one-way function**? We could take multiplication rather than addition, but even primary school children know that the inverse function, division, is only slightly more difficult than multiplication itself. We need to go one step higher in the hierarchy of calculation methods. It is still relatively simple to calculate the power of a number, but the corresponding two reverse functions - *taking roots* (find b in the equation $a = b^c$ when a and c are known) and *calculating logarithms* (find c in the above equation when a and b are known) are so complicated that pupils normally do not learn them at school.

[38] Harry Potter and the Chamber of Secrets, Carlsen, 1998, last chapter Dobby's reward, p 343

[39] In the RSA procedure, we call numbers small if the bit lengths are much less than 1024 bits (i.e. 305 decimal points). In practice, 1024 bits is currently the minimum length for a secure Certification Authority RSA modulus.

Although a certain structure can still be recognised for addition and multiplication, raising numbers to the power of another or calculating exponentials totally mixes up all the numbers. Knowing a few values of the function doesn't tell us much about the function as a whole (in contrast to addition and multiplication).

3.12.2 The Diffie-Hellman key exchange protocol

Whitfield Diffie, Martin E. Hellman and Ralph Merkle developed this DH key exchange protocol in Stanford in 1976.

Alice and Bob[40] use a one-way function to obtain a key S, the session key, for subsequent correspondence. This is then a secret that is only known to the two of them. Alice selects a random number a and keeps it secret. She applies a one-way function to a to calculate the number $A = g^a$ and sends it to Bob. He does the same, by selecting a secret random number b, calculating $B = g^b$ and sending it to Alice. The number g is random and can be publicly known. Alice applies the one-way function together with her secret number a to B, while Bob does the same with his secret number b and the received number A.

The result S is the same in each case because the one-way function is commutative: $(g^a)^b = (g^b)^a$. But even Bob cannot reconstruct Alice's secret number a from the data available to him, while Alice cannot determine Bob's secret number b. And a perpetrator who knows g and has intercepted both A and B cannot use this knowledge to determine a, b or S.

We will now use an example with (unrealistically) small numbers to illustrate this.

Procedure: Alice and Bob want to negotiate a secret session key S via a channel that may be intercepted.

1. They select a prime number p and a random number g and exchange this information openly.
2. Alice now selects a, a random number less than p and keeps it secret. Similarly, Bob selects b, a random number less than p and keeps it secret.
3. Alice now calculates $A \equiv g^a \pmod{p}$.
   Bob calculates $B \equiv g^b \pmod{p}$.
4. Alice sends the result A to Bob.
   Bob sends the result B to Alice.
5. In order to now determine the session key to be used by both, they both separately raise the respective results they have received to the power of their secret random number modulo p. This means:
   - Alice calculates $S \equiv B^a \pmod{p}$ and
   - Bob calculates $S \equiv A^b \pmod{p}$.

[40] Bob and Alice are the standard names used for the two authorised participants in a protocol (see [Schneier1996, p. 23]).

Even if a spy intercepts g, p, and the interim results A and B, he cannot use these to determine the session key used due to the difficulty of calculating the discrete logarithm.

Example using numbers:

1. Alice and Bob select $g = 11, p = 347$.
2. Alice selects $a = 240$, Bob selects $b = 39$ and they keep a and b secret.
3. Alice calculates $A \equiv g^a \equiv 11^{240} \equiv 49 \pmod{347}$.
   Bob calculates $B \equiv g^b \equiv 11^{39} \equiv 285 \pmod{347}$.
4. Alice sends Bob: $A \equiv 49$,
   Bob sends Alice: $B \equiv 285$.
5. Alice calculates $B^a \equiv 285^{240} \equiv 268 \pmod{347}$.
   Bob calculates $A^b \equiv 49^{39} \equiv 268 \pmod{347}$.

Alice and Bob can now communicate securely using their shared session key. Even if spies were to intercept everything transferred via the connection: $g = 11, p = 347, A = 49$ and $B = 285$, they would not be able to calculate the secret key.

Comment:
In this example using such small numbers, it would be possible, but with large numbers the discrete logarithm problem[41] [42] is extremely difficult to solve. Here, we need to solve:
For Alice: $11^x \equiv 49 \pmod{347}$, that means $\log_{11}(49) \pmod{347}$.
For Bob: $11^y \equiv 285 \pmod{347}$, that means $\log_{11}(285) \pmod{347}$.

3.13 The RSA procedure with actual numbers

Having described above how the RSA procedure works, we will now work through the steps using actual, but small, numbers.

[41] If you try to determine the discrete logarithm x that solves the equation $11^x \equiv 49 \pmod{347}$ with Mathematica by means of Solve, you obtain the em tdep message The equations appear to involve the variables to be solved for in an essentially non-algebraic way . Mathematica therefore claims not to know a direct algebraic procedure for solving the equation. Yet Mathematica is able to calculate this with the general function for the multiplicative order (here for Alice): `MultiplicativeOrder[11, 347, 49]` delivers the value 67.
Such number-theory tasks can also be solved using other tools such as the LiDIA or BC package (see URLs in appendix). The dl function in the **LC** user interface for LiDIA also delivers the value 67 for `dl(11,49,347)`.

[42] Why have the functions delivered the value 67 rather than 240 for the dl problem for Alice? The discrete logarithm is the smallest natural exponent that solves the equation $11^x \equiv 49 \pmod{347}$. Both $x = 67$ and $x = 240$ (the number selected in the example) satisfy the equation and can therefore be used to calculate the session key: $285^{240} \equiv 285^{67} \equiv 268 \pmod{347}$. If Alice and Bob had selected a primitive root modulo p as base g, then for every remainder from the set $\{1, 2, ..., p-1\}$ there is exactly one exponent from the set $\{0, 1, ..., p-2\}$ - For info: there are 172 different primitive roots for modulo 347, 32 of which are prime (not necessary). Since the number 11 selected for g in the example is not a primitive root of 347, the remainders do not take all values from the set $\{1, 2, ..., 346\}$. Thus, for a particular remainder there may be more than one exponent or even no exponent at all in the set $\{0, 1, ..., 345\}$ that satisfies the equation. `PrimeQ[347] = True; EulerPhi[347] = 346; GCD[11, 347] = 1;`
`MultiplicativeOrder[11, 347] = 173; MultiplicativeOrder[11, 347, 49] = 67`

3.13.1 RSA with small prime numbers and with a number as message

Before applying the RSA procedure to a text, we will first demonstrate it directly using a number [43].

1. Let the selected prime numbers be $p = 5$ and $q = 11$.
   Thus, $n = 55$ and $(p-1)(q-1) = 40$.
2. $e = 7$ (should lie between 11 and 40 and must be relatively prime to 40).
3. $d = 23$ (since $23 * 7 \equiv 161 \equiv 1 \pmod{40}$),

   $\rightarrow$ Public key of the recipient: $(55, 7)$,

   $\rightarrow$ Private key of the recipient: $(55, 23)$.
4. Let the message be the number $M = 2$ (so no division into blocks is required).
5. Encryption: $C \equiv 2^7 \equiv 18 \pmod{55}$.
6. The ciphertext is simply the number $C = 18$ (we therefore do not need to divide it into blocks).
7. Decryption: $M \equiv 18^{23} \equiv 18^{(1+2+4+16)} \equiv 18 * 49 * 36 * 26 \equiv 2 \pmod{55}$.

We will now apply the RSA procedure to a text, first using the upper case alphabet (26 characters), then using the entire ASCII character set as the basis for the messages.

3.13.2 RSA with slightly larger prime numbers and a text of upper case letters

We have the text ATTACK AT DAWN and the characters are coded in the following simple manner[44]:

i	11^i mod 347	
0	1	
1	11	
2	121	
3	290	
67	49	XXXgesuchter ExponentXXX
172	284	
173	1	= XXXMultiplikative Ordnung von 11 (mod 347)XXX
174	11	
175	121	
176	290	
240	49	XXXgesuchter ExponentXXX

[43] Using CrypTool v1.3 you can solve this with the Indiv.Procedures/RSA Demonstration/RSA Cryptosystem .
[44] Using CrypTool v1.3 you can solve this with the Indiv.Procedures/RSA Demonstration/RSA Cryptosystem . This is also described in the tutorial/scenario in CrypTool's online help [XXX Options specify alphabet, Number system, Block length 2 und decimal represantation XXX].

Character	Numerical value	Character	Numerical value
Blank	0	M	13
A	1	N	14
B	2	O	15
C	3	P	16
D	4	Q	17
E	5	R	18
F	6	S	19
G	7	T	20
H	8	U	21
I	9	V	22
J	10	W	23
K	11	X	24
L	12	Y	25
		Z	26

Key generation (steps 1 to 3):
1. $p = 47, q = 79$ ($n = 3713$; $J(n) = (p-1)(q-1) = 3588$).
2. $e = 37$ (should lie between 79 and 3588 and must be relatively prime to 3588.)
3. $d = 97$ (since $e * d = 1 \bmod J(n) : 37 * 97 \equiv 3589 \equiv 1 \pmod{3588}$) [45].
4. Encryption:

Text:	A	T	T	A	C	K		A	T		D	A	W	N
Number:	01	20	20	01	03	11	00	01	20	00	04	01	23	14

This 28-digit number is divided into 4-digit parts (because 2626 is still smaller than $n = 3713$):
`0120 2001 0311 0001 2000 0401 2314`

All 7 parts are encrypted using: $C \equiv M^{37} \pmod{3713}$ [46]:
`1404 2932 3536 0001 3284 2280 2235`

5. Decryption:
Ciphertext: `1404 2932 3536 0001 3284 2280 2235`
This 28-digit number is divided into 4-digit parts.
All 7 parts are decrypted using: $M \equiv C^{97} \pmod{3713}$:
`0120 2001 0311 0001 2000 0401 2314`
The 2-digit numbers are transformed into capital letters and blanks.

Using the selected values it is easy for a cryptoanalyst to derive the secret values from the public parameters $n = 3713$ and $e = 37$ by revealing that $3713 = 47 * 79$.
If n is a 768-bit number, there is, according to present knowledge, little chance of this.

[45] How to compute $d = 97$ using the *extended* gcd algorithm is shown in appendix A
[46] Using Mathematica, for example, this can be calculated usung `PowerMod[120, 37, 3713]`.
Using CrypTool v1.3 you can solve this with the Indiv. Procedures/RSA Demonstration/Factorise (IFP) .

3.13.3 RSA with slightly larger prime numbers still and a text made up of ASCII characters

In real life, the ASCII alphabet is used to code the individual characters of the message as 8-bit numbers.

The idea for this task[47] is taken from the example in [Eckert2001, p. 215].

Coded in decimal notation, the text RSA works! is as follows:

Text:	R	S	A		w	o	r	k	s	!
Number:	82	83	65	32	119	111	114	107	115	33

We will work through the example in 2 variants. XXX XXX

Key generation (steps 1 to 3):
1. $p = 251, q = 269 \quad (n = 67519;\ J(n) = (p-1)(q-1) = 67000 = 23 * 53 * 67)$ [48].
2. $e = 65537$ (should lie between 269 and 67000 and must be relatively prime to 67000) [49].
3. $d = 2473$ (since $e \equiv d^{-1} \bmod J(n) : 65537 * 2473 \equiv 162.073.001 \equiv 1 \pmod{67000}$) [50].

Variant 1: All ASCII characters are encrypted and decrypted separately (no blocks are formed).

4. Encryption:

Text:	R	S	A		w	o	r	k	s	!
Number:	82	83	65	32	119	111	114	107	115	33

The letters are not combined[51]!

Each character is encrypted using: $C = M^{65537} \pmod{67519}$ [52]:

```
58455   39103   16634   28092   58597
65768   33441   20214   40199   3240
```

5. Decryption:
Ciphertext:

```
58455   39103   16634   28092   58597
65768   33441   20214   40199   3240
```

[47] Using CrypTool v1.3 you can solve this with the Indiv.Procedures/RSA Demonstration/RSA Cryptosystem .
[48] Mathematica delivers this result, for example, with `FactorInteger[67000]={{2,3}, {5,3}, {67,1}}`. Using CrypTool v1.3 you can solve this with the Indiv.Procedures/RSA Demonstration/Factorise (IFP) .
[49] e cannot, therefore, be $2, 5$ or 67. In real life, $J(n)$ is not factorised but rather the Euclidean algorithm is used for the selected e to guarantee that $\gcd(d, J(n)) = 1$.
[50] Other possible combinations of (e, d) include: $(3, 44667)$, $(7, 19143)$, $(17, 43353)$.
[51] For secure procedures we need large numbers that assume – as far as possible – all values up to $n - 1$. If the possible value set for the numbers in the message is too small, even large prime numbers cannot make the procedure secure. An ASCII character is represented by 8 bits. If we want larger values we must combine several numbers. Two characters need 16 bits, whereby the maximum value that can be represented is 65536. The modulus n must then be greater than $2^{16} = 65536$. This is applied in variant 2. When the numbers are combined, the leading zeros are kept in binary notation (just as if we were to write all numbers with 3 digits in decimal notation above and were then to obtain the sequence `082 083, 065 032, 119 111, 114 107, 115 033`).
[52] Using Mathematica this can be calculated using `PowerMod[{82, 83, 65, 32, 119, 111, 114, 107, 115, 33}, 65537, 67519]`.

Each character is decrypted using: $M \equiv C^{2473} \bmod 67519$:
82 83 65 32 119 111 114 107 115 33

Variant 2: The ASCII characters are encrypted and decrypted two at a time as blocks.

4. Encryption:

Text:	R	S	A		w	o	r	k	s	!
Number:	82	83	65	32	119	111	114	107	115	33

Blocks are formed[53] (each ASCII character is encoded into a 8 digit binary number below):
21075 16672 30575 29291 29473

Each block is encrypted using: $C \equiv M^{65537} \pmod{67519}$ [54]:
3046 29337 62503 9978 40883

5. Decryption:
Ciphertext:
3046 29337 62503 9978 40883

Each block is decrypted using: $M \equiv C^{2473} \pmod{67519}$:
21075 16672 30575 29291 29473

4'. Encryption:
Blocks are formed: (each ASCII character is encoded into a 3 digit decimal number below):
82083 65032 119111 114107 115033
Each block is encrypted using: $C \equiv M^{65537} \pmod{67519}$ [55]:
53920 26728 12241 42791 44242

5'. Decryption:
Ciphertext:
53920 26728 12241 42791 44242

Each block is decrypted using: $M \equiv C^{2473} \pmod{67519}$:
14564 65032 51592 46588 47514

---

[53]

```
                binary representation   decimal representation
  01010010, 82  01010010 01010011       =21075
  01010011, 83
  01000001, 65  01000001 00100000       =16672
  00100000, 32
  01110111, 119 01110111 01101111       =30575
  01101111, 111
  01110010, 114 01110010 01101011       =29291
  01101011, 107
  01110011, 115 01110011 00100001       =29473
  00100001, 33:
```

[54] Using Mathematica this can be calculated using `PowerMod[{21075, 16672, 30575, 29291, 29473}, 65537, 67519]`.

[55] Using Mathematica this can be calculated using `PowerMod[{82083, 65032, 119111, 114107, 115033}, 65537, 67519]`.

3.13.4 A small RSA cipher challenge (1)

The task is taken from [Stinson1995, Exercise 4.6]: The pure solution has been published by Prof. Stinson at http://www.cacr.math.uwaterloo.ca/~dstinson/solns.html. However, it is not the result that is important here but rather the individual steps of the solution, that is, the explanation of the cryptoanalysis[56]

Two samples of RSA ciphertext are presented in Tables 4.1 and 4.2. Your task is to decrypt them. The public parameters of the system are

$n = 18923$ and $e = 1261$ (for Table 4.1) and
$n = 31313$ and $e = 4913$ (for Table 4.2).

This can be accomplished as follows. First, factor n (which is easy because it is so small). Then compute the exponent d from $J(n)$, and, finally, decrypt the ciphertext. Use the square-and-multiply algorithm to exponentiate modulo n.

In order to translate the plaintext back into ordinary English text, you need to know how alphabetic characters are encoded as elements in $\mathbb{Z}_n$. Each element of $\mathbb{Z}_n$ represents three alphabetic characters as in the following examples:

$$\begin{array}{lcl} \texttt{DOG} & \mapsto & 3 * 26^2 + 14 * 26 + 6 = 2398 \\ \texttt{CAT} & \mapsto & 2 * 26^2 + 0 * 26 + 19 = 1371 \\ \texttt{ZZZ} & \mapsto & 25 * 26^2 + 25 * 26 + 25 = 17575\,. \end{array}$$

You will have to invert this process as the final step in your program. The first plaintext was taken from The Diary of Samuel Marchbanks , by Robertson Davies, 1947, and the second was taken from Lake Wobegon Days , by Garrison Keillor, 1985.

[56] The method of solving the problem is outlined in the scenario of the online help to CrypTool and in the presentation on the website. If anyone sends us a well prepared exact method of solving the problem, we would be pleased to include it in the documentation.

TABLE 4.1: RSA ciphertext

12423	11524	7243	7459	14303	6127	10964	16399
9792	13629	14407	18817	18830	13556	3159	16647
5300	13951	81	8986	8007	13167	10022	17213
2264	961	17459	4101	2999	14569	17183	15827
12693	9553	18194	3830	2664	13998	12501	18873
12161	13071	16900	7233	8270	17086	9792	14266
13236	5300	13951	8850	12129	6091	18110	3332
15061	12347	7817	7946	11675	13924	13892	18031
2620	6276	8500	201	8850	11178	16477	10161
3533	13842	7537	12259	18110	44	2364	15570
3460	9886	8687	4481	11231	7547	11383	17910
12867	13203	5102	4742	5053	15407	2976	9330
12192	56	2471	15334	841	13995	17592	13297
2430	9741	11675	424	6686	738	13874	8168
7913	6246	14301	1144	9056	15967	7328	13203
796	195	9872	16979	15404	14130	9105	2001
9792	14251	1498	11296	1105	4502	16979	1105
56	4118	11302	5988	3363	15827	6928	4191
4277	10617	874	13211	11821	3090	18110	44
2364	15570	3460	9886	9988	3798	1158	9872
16979	15404	6127	9872	3652	14838	7437	2540
1367	2512	14407	5053	1521	297	10935	17137
2186	9433	13293	7555	13618	13000	6490	5310
18676	4782	11374	446	4165	11634	3846	14611
2364	6789	11634	4493	4063	4576	17955	7965
11748	14616	11453	17666	925	56	4118	18031
9522	14838	7437	3880	11476	8305	5102	2999
18628	14326	9175	9061	650	18110	8720	15404
2951	722	15334	841	15610	2443	11056	2186

TABLE 4.2: RSA ciphertext

6340	8309	14010	8936	27358	25023	16481	25809
23614	7135	24996	30590	27570	26486	30388	9395
27584	14999	4517	12146	29421	26439	1606	17881
25774	7647	23901	7372	25774	18436	12056	13547
7908	8635	2149	1908	22076	7372	8686	1304
4082	11803	5314	107	7359	22470	7372	22827
15698	30317	4685	14696	30388	8671	29956	15705
1417	26905	25809	28347	26277	7897	20240	21519
12437	1108	27106	18743	24144	10685	25234	30155
23005	8267	9917	7994	9694	2149	10042	27705
15930	29748	8635	23645	11738	24591	20240	27212
27486	9741	2149	29329	2149	5501	14015	30155
18154	22319	27705	20321	23254	13624	3249	5443
2149	16975	16087	14600	27705	19386	7325	26277
19554	23614	7553	4734	8091	23973	14015	107
3183	17347	25234	4595	21498	6360	19837	8463
6000	31280	29413	2066	369	23204	8425	7792
25973	4477	30989					

3.13.5 A small RSA cipher challenge (2)

The following task is a corrected version from the excellent book written by Prof. Yan [Yan2000, Example 3.3.7, p. 318]. However, it is not the result that is important here but rather the individual steps of the solution, that is, the explanation of the cryptoanalysis[57].

There are three tasks with completely different degrees of difficulty here. In each case we know the ciphertext and the public key (e, n):

(a) Known plaintext: find the secret key d using the additionally known original message.

(b) Ciphertext only: find d and the plaintext.

(c) Calculate the RSA modulus, in other words factorisation (with no knowledge of the message).

$n = 639784868795271438588314150 41, e = 17579$

Message:

```
1401202118011200,
1421130205181900,
0118050013010405,
0002250007150400
```

[57] The method of solving the problem is outlined in the scenario of the online help to CrypTool and in the presentation on the website. If anyone sends us a well prepared exact method of solving the problem, we would be pleased to include it in the documentation.

Cipher:

```
4541166789502493820925925342 3,
1659709162143202007631155220 1,
4646897927975035473263763104 4,
3287016754590374133981967137 9
```

Comments:

The original message consisted of a sentence containing 31 characters (coded with the upper case alphabet from section 3.13.2). Each group of 16 decimal numbers is then combined to form one number (the last number is filled with zeros). These numbers are raised to the power of e.

When you decrypt the message you must fill the calculated numbers with leading zeros in order to obtain plaintext.

This needs to be stressed because the type of padding is extremely important during implementation and standardisation for interoperable algorithms.

References

[Bartholome1996] A. Bartholome, J. Rung, H. Kern,
Zahlentheorie f"ur Einsteiger, Vieweg 1995, 2. Auflage 1996.

[Bauer1995] Friedrich L. Bauer,
Entzifferte Geheimnisse, Springer, 1995.

[Bauer2000] Friedrich L. Bauer,
Decrypted Secrets, Springer, 2nd edition 2000.

[Beutelspacher1996] Albrecht Beutelspacher,
Kryptologie, Vieweg, 5. Auflage 1996.

[Buchmann1999] Johannes Buchmann,
Einf"uhrung in die Kryptographie, Springer, 1999.

[Eckert2001] Claudia Eckert,
IT-Sicherheit: Konzepte-Verfahren-Protokolle, Oldenbourg, 2001.

[Graham1994] Graham, Knuth, Patashnik,
Concrete Mathemathics, a Foundation of Computer Science, 2nd edition, Addison Wesley, 1994.

[Knuth1998] Donald E. Knuth,
The Art of Computer Programming, vol 2: Seminumerical Algorithms, Addison-Wesley, 2nd edition, 1998.

[Pfleger1997] Charles P. Pfleger,
Security in Computing, Prentice-Hall, 2nd edition, 1997.

[Schneier1996] Bruce Schneier,
Applied Cryptography, Protocols, Algorithms, and Source Code in C, Wiley, 2nd edition, 1996.

[Sedgewick1990] Robert Sedgewick,
Algorithms in C, Addison-Wesley, 1990.

[Stinson1995] Douglas R. Stinson,
Cryptography - Theory and Practice, CRC Press, 1995.

[Wiles1994] Wiles, Andrew,
Modular elliptic curves and Fermat's Last Theorem, in Annals of Mathematics 141 (1995).

[Wolffenstetter1998] Albrecht Beutelspacher, J"org Schwenk, Klaus-Dieter Wolfenstetter,
Moderne Verfahren in der Kryptographie, Vieweg, 2. Auflage, 1998.

[Yan2000] Song Y. Yan,
Number Theory for Computing, Springer, 2000.

URLs / links (selection)

1. Ron Knott's Fibonacci page,
   Here, everything revolves around Fibonacci numbers.
   http://www.mcs.surrey.ac.uk/personal/R.Knott/Fibonacci/fib.html

2. CrypTool Version 1.3, 2001,
   Freeware to illustrate cryptography
   http://www.cryptool.de, http://www.cryptool.org, http://www.cryptool.com

3. Mathematica,
   Commercial mathematics package
   http://www.wolfram.com

4. LiDIA,
   Extensive library containing number-theory functions and the LC interpreter
   http://www.informatik.tu-darmstadt.de/TI/LiDIA

5. BC,
   Interpreter with number-theory functions
   http://www.maths.uq.edu.au/k̃rm/gnubc.html

6. Only after I had completed this article did I come across a website which interactively and didactically uses simple number theory to provide an extremely sophisticated description of the fundamental mathematical thought processes. It was created for a teaching project in the 11th grade of the technical grammar school (unfortunately only available in German):
   http://www.djh-freeweb.de/m̃uenchenbach

7. XXX BSI,
   Bundesamt f'ur Sicherheit und Informationstechnik
   http://www.bsi.bund.de XXX

Thanks

I would like to take this opportunity to thank Henrik Koy for making suggestions and constructively proof-reading this article.

Appendix A: the greatest common divisor (gcd) of whole numbers

The greatest common divisor of two natural numbers a and b is an important value that can be calculated very quickly. Here we make use of the fact that if a number c divides the numbers a and b (i.e. there exists an a' and a b' such that $a = a' * c$ and $b = b' * c$), then c also divides $a - \lfloor a/b \rfloor * b$. It then follows that:

$$gcd(a, b) = gcd(a - \lfloor a/b \rfloor * b, b).$$

Using this information, the algorithm for calculating the gcd of two numbers can be written as follows (in pseudo code):

```
INPUT: a,b != 0
1. if ( a < b ) then  x = a; a = b; b = x; // Swap a and b (a > b)
2. a = a - int(a/b) * b                    // a is smaller than b, the gcd(a, b)
                                           // is unchanged
3. if ( a != 0 ) then goto 1.              // a falls after each step and
                                           // the algorithm ends when a == 0.
OUTPUT "gcd(a,b) = " b    // b is the gcd of the original a and b
```

However, to other relationships can be derived from the gcd: For this, we need the set of equations for a and b:

$$\begin{aligned} a &= 1 * a + 0 * b \\ b &= 0 * a + 1 * b, \end{aligned}$$

or, in matrix notation:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} a \\ b \end{pmatrix}.$$

We summarise this information in the extended matrix:

$$\left(\begin{array}{c|cc} a & 1 & 0 \\ b & 0 & 1 \end{array}\right)$$

If we apply the above gcd algorithm to this matrix, we obtain the extended gcd algorithm:

INPUT: $a, b \neq 0$

0. $x_{1,1} := 1, x_{1,2} := 0, x_{2,1} := 0, x_{2,2} := 1$

1. $\left(\begin{array}{c|cc} a & x_{1,1} & x_{1,2} \\ b & x_{2,1} & x_{2,2} \end{array}\right) := \begin{pmatrix} 0 & 1 \\ 1 & -\lfloor a/b \rfloor * b \end{pmatrix} * \left(\begin{array}{c|cc} a & x_{1,1} & x_{1,2} \\ b & x_{2,1} & x_{2,2} \end{array}\right).$

2. `if (b != 0) then goto 1.`

`OUTPUT:` "ggT$(a,b) = a * x + b * y$: ", "ggT$(a,b) =$ " b, "$x =$" $x_{2,1}$, "$y =$" $x_{2,2}$
Since this algorithm only performs linear transformations, the same equations always apply

$$\begin{aligned} a &= x_{1,1} * a + x_{1,2} * b \\ b &= x_{2,1} * a + x_{2,2} * b, \end{aligned}$$

and we have the extended gcd equation at the end of the algorithm:

$$gcd(a,b)(= b^{\text{END}}) = a * x_{2,1} + b * x_{2,2}.$$

Example:
Using the extended gcd we can determine for $e = 37$ the multiplicative inverse number d to modulo 3588 (i.e. $37 * d \equiv 1 \pmod{3588}$):

`0.` $\left(\begin{array}{c|cc} 3588 & 1 & 0 \\ 37 & 0 & 1 \end{array}\right)$

`1.` $\left(\begin{array}{c|cc} 37 & 1 & 0 \\ 36 & 0 & -96 \end{array}\right) = \left(\begin{array}{cc} 0 & 1 \\ 1 & -(\lfloor 3588/36 \rfloor = 96) * 37 \end{array}\right) * \left(\begin{array}{c|cc} 3588 & 1 & 0 \\ 37 & 0 & 1 \end{array}\right).$

`2.` $\left(\begin{array}{c|cc} 36 & 1 & -96 \\ 1 & -1 & 97 \end{array}\right) = \left(\begin{array}{cc} 0 & 1 \\ 1 & -(\lfloor 37/36 \rfloor = 1) * 36 \end{array}\right) * \left(\begin{array}{c|cc} 37 & 1 & 0 \\ 36 & 0 & -96 \end{array}\right).$

`3.` $\left(\begin{array}{c|cc} 1 & -1 & 97 \\ 0 & 37 & -3588 \end{array}\right) = \left(\begin{array}{cc} 0 & 1 \\ 1 & -(\lfloor 36/1 \rfloor = 36) * 1 \end{array}\right) * \left(\begin{array}{c|cc} 36 & 1 & -96 \\ 1 & -1 & 97 \end{array}\right).$

`OUTPUT:`
gcd$(37, 3588) = a * x + b * y$: gcd$(37, 3588) = 1$, $x = -1$, $y = 97$.
Thus

1. 37 and 3588 are relatively prime (37 has an inverse modulo 3588).
2. $37 * 97 = (1 * 3588) + 1$ in other words $37 * 97 \equiv 1 \pmod{3588}$. and therefore the number 97 is the multiplicative inverse to 37 modulo 3588.

Appendix B: Forming closed sets

The property of closedness is always defined in relation to an operation in a set. The following shows how to construct the closed set G with respect to the operation + (mod 8) for a given initial set G_0:

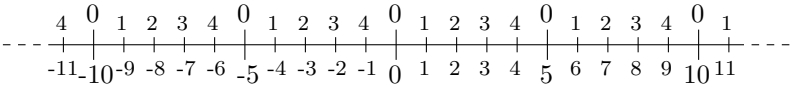
$$
\begin{aligned}
G_0 &= \{2,3\} \text{ addition of the numbers in } G_0 \text{ determines further numbers :} \\
&\quad 2+3 \equiv 5 \pmod 8 = 5 \\
&\quad 2+2 \equiv 4 \pmod 8 = 4 \\
&\quad 3+3 \equiv 6 \pmod 8 = 6 \\
G_1 &= \{2,3,4,5,6\} \text{ addition of the numbers in } G_1 \text{ determines :} \\
&\quad 3+4 \equiv 7 \pmod 8 = 7 \\
&\quad 3+5 \equiv 8 \pmod 8 = 0 \\
&\quad 3+6 \equiv 9 \pmod 8 = 1 \\
G_2 &= \{0,1,2,3,4,5,6,7\} \text{ addition of the numbers in } G_2 \textit{ does not extend the set}! \\
G_3 &= G_2 \text{ we say : } G_2 \text{ is closed for addition (mod 8).}
\end{aligned}
$$

End of forming a closed set.

Appendix C: Comments on modulo subtraction

Comment on subtraction modulo 5: $2-4 \equiv -2 \equiv 3 \bmod 2$. It is therefore not true modulo 5 that $-2 = 2$! People often make the mistake of equating this. You can show this clearly if you place the permutation $(0,1,2,3,4)$ in $\mathbb{Z}_5$, for example from -11 to $+11$, over the range of numbers in $\mathbb{Z}$.

Zahlengerade modulo 5



Zahlengerade der ganzen Zahlen

4 The Mathematical Ideas Behind Modern Cryptography

(Oyono R./ Esslinger B., September 2000, Update Nov. 2000)

4.1 One-way functions with trapdoor and complexity classes

A **one-way function** is a (one-to-one) function that can be calculated efficiently, but whose inverse is extremely complicated and practically impossible to calculate.

To put it more precisely: A one-way function is a mapping f from a set X to a set Y, such that $f(x)$ can be calculated easily for each element x of X, whereas for (almost) every y from Y it is practically impossible to find an inverse image x (i.e. an x where $f(x) = y$).

An everyday example of a one-way function is a telephone book: the function to be performed is to assign a name to the corresponding telephone number. This can be done easily due to the fact that the names are sorted alphabetically. However, the inverse function - assigning a name to a given number - is obviously difficult if you only have a telephone book available.

One-way functions play a decisive role in cryptography. Almost all cryptographic terms can be rephrased using the term one-way function. Let's take for example public-key encryption (asymmetric cryptography):

Each subscriber T to the system is assigned a private key d_T and what is known as a public key e_T. These keys must have the following property (public-key property):
For an opponent who knows the public key e_T, it is practically impossible to determine the private key d_T.

In order to construct useful public-key procedures, therefore, we look for a one-way function that is easy to calculate in one direction , but is difficult (practically impossible) to calculate in the other direction, provided that a particular piece of additional information (trapdoor) is not available. This additional piece of information allows the inverse to be found efficiently. Such functions are called **trapdoor one-way functions**. In the above case, d_T is the trapdoor information.

In this process, we describe a problem as easy if it can be solved in polynomial time as a function of the length of the input. If the length of the input is n bits, then the time for calculating the function is proportional to n^a, where a is a constant. We say that the complexity of such problems is $O(n^a)$.

The term practically impossible is slightly less precise. In general, we can say that a problem cannot be solved efficiently, if the time required to solve it increases more quickly than the polynomial time as a function of the size of the input. If, for example, the length of the input is n bits and the time required for calculating the function is proportional to 2^n, then the following currently applies: the function practically cannot be calculated for $n > 80$ (for $a = 5$ the following applies: from the length $n = 23$, 2^n is greater than n^5, after which 2^n clearly increases more quickly ($2^{23} = 8.388.608$, $23^5 = 6.436.343$)).

In order to develop a public-key procedure that can be implemented in practice, it is therefore necessary to discover a suitable trapdoor one-way function.

In order to tidy things up among this confusing multitude of possible problems and their complexities, we group problems with similar complexities into classes.

The most important complexity classes are the classes **P** and **NP**:

- The class **P**: This class contains those problems that can be solved in a polynomial amount of time.
- The class **NP**: The definition of this class looks not at the time required to solve a problem, but rather at the time required to verify a given solution. The class **NP** consists of those problems for which a given solution can be verified in a polynomial amount of time. Hereby, the term **NP** non-deterministic means polynomial and is based on a calculation model, i.e. on a computer that only exists in theory and can guess correct solutions non-deterministically then verify them in polynomial time.

The class **P** is contained in the class **NP**. A well-known unsolved problem is the question whether or not $\mathbf{P} \neq \mathbf{NP}$ is true, i.e. whether or not **P** is a true subset. An important property of the class **NP** is that it also contains what are known as **NP**-complete problems. These are problems that represent the class **NP** as follows: If a good algorithm for such a problem exists, then goodalgorithms exist for all problems from **NP**. In particular: if **P** only contained one complete problem, i.e. if a polynomial solution algorithm existed for this problem, then **P**would be equal to **NP**. In this sense, the **NP**-complete problems are the most difficult problems in **NP**.

Many cryptographic protocols are formed in such a way that the good subscribers only have to solve problems from **P**, whereas a perpetrator is faced with problems from **NP**.

Unfortunately, we do not yet know whether one-way functions actually exist. However, we can prove that one-way functions exist if and only if $\mathbf{P} \neq \mathbf{NP}$ [Balcazar1988, S.63].
Mathematicians have again and again claimed to have proven the equivalence, e.g.

http://www.geocities.com/st_busygin/clipat.html),

but so far the claims have always turned out to be false.

A number of algorithms have been suggested for public-key procedures. In many cases - although they at first appeared promising - it was discovered that they could be solved polynomially. The most famous failed applicant is the knapsack with trapdoor, suggested by Ralph Merkle [Merkle1978].

4.2 Knapsack problem as a basis for public-key procedures

4.2.1 Knapsack problem

You are given n objects $G_1, \ldots, G_n$ with the weights $g_1, \ldots g_n$ and the values $w_1, \cdots, w_n$. The aim is to carry away as much as possible in terms of value while restricted to an upper weight limit g. You therefore need to find a subset of $\{G_1, \cdots, G_n\}$, i.e. $\{G_{i_1}, \ldots, G_{i_k}\}$, so that $w_{i_1} + \cdots + w_{i_k}$ is maximised under the condition $g_{i_1} + \cdots + g_{i_k} \leq g$.

Such questions are called **NP**-complete problems (not deterministically polynomial) that are difficult to calculate.

A special case of the knapsack problem is:
Given the natural numbers $a_1, \ldots, a_n$ and $g.$, find $x_1, \ldots, x_n \in \{0, 1\}$ where $g = \sum_{i=1}^{n} x_i a_i$ (i.e. where $g_i = a_i = w_i$ is selected). This problem is also called a **0-1 knapsack problem** and is identified with $K(a_1, \ldots, a_n; g)$.

Two 0-1 knapsack problems $K(a_1, \ldots, a_n; g)$ and $K(a'_1, \ldots, a'_n; g')$ are called congruent if two coprime numbers w and m exist such that

1. $m > \max\{\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} a'_i\}$,
2. $g \equiv wg' \mod m$,
3. $a_i \equiv wa'_i \mod m$ for all $i = 1, \ldots, n$.

Comment: Congruent 0-1 knapsack problems have the same solutions. No quick algorithm is known for clarifying the question whether two 0-1 knapsack problems are congruent.

A 0-1 knapsack problem can be solved by testing the 2^n possibilities for $x_1, \ldots, x_n$. The best method requires $O(2^{n/2})$ operations, which for $n = 100$ with $2^{100} \approx 1.27 \cdot 10^{30}$ and $2^{n/2} \approx 1.13 \cdot 10^{15}$ represents an insurmountable hurdle for computers. However, for special $a_1, \ldots, a_n$ the solution is quite easy to find, e.g. for $a_i = 2^{i-1}$. The binary representation of g immediately delivers $x_1, \ldots, x_n$. In general, the a 0-1 knapsack problem can be solved easily if a permutation[58] π of $1, \ldots, n$ exists with $a_{\pi(j)} > \sum_{i=1}^{j-1} a_{\pi(i)}$. If, in addition, π is the identity, i.e. $\pi(i) = i$ for $i = 1, 2, \ldots, n$, then the sequence $a_1, \ldots, a_n$ is said to be superincreasing. The following algorithm solves the knapsack problem with a superincreasing sequence in the timeframe of $O(n)$.

```
for i = n to 1 do
    if T ≥ a_i then
        T := T − s_i
        x_i := 1
    else
        x_i := 0
if T = 0 then
    X := (x_1, ..., x_n) is the solution.
else
    No solution exists
```

Algorithm 1. Solving knapsack problems with superincreasing weights

[58] A permutation π of the numbers $1, \ldots, n$ is a change in the order in which these numbers are listed. For example, a permutation π of $(1, 2, 3)$ is $(3, 1, 2)$,, i.e. $\pi(1) = 3$, $\pi(2) = 1$ and $\pi(3) = 2$.

4.2.2 Merkle-Hellman knapsack encryption

In 1978, Merkle and Hellman [Merkle1978] specified a public-key encryption procedure that is based on defamiliarising the easy 0-1 knapsack problem with a superincreasing sequence into a congruent one with a superincreasing sequence. It is a block ciphering that ciphers an n-bit plaintext each time it runs. More precisely:

Let $(a_1, \ldots, a_n)$ be superincreasing. Let m and w be two coprime numbers with $m > \sum_{i=1}^{n} a_i$ and $1 \leq w \leq m-1$. Select $\bar{w}$ with $w\bar{w} \equiv 1 \mod m$ the modular inverse of w and set $b_i := wa_i \mod m$, $0 \leq b_i < m$ for $i = 1, \ldots, n$, and verify whether the sequence $b_1, \ldots b_n$ is not superincreasing. A permutation $b_{\pi(1)}, \ldots, b_{\pi(n)}$ of $b_1, \ldots, b_n$ is then published and the inverse permutation μ to π is defined secretly. A sender writes his/her message in blocks $(x_1^{(j)}, \ldots, x_n^{(j)})$ of binary numbers n in length, calculates

$$g^{(j)} := \sum_{i=1}^{n} x_i^{(j)} b_{\pi(i)}$$

and sends $g^{(j)}, (j = 1, 2, \ldots)$.
The owner of the key calculates

$$G^{(j)} := \bar{w} g^{(j)} \mod m, \quad 0 \leq G^{(j)} < m$$

and obtains the $x_{\mu(i)}^{(j)} \in \{0, 1\}$ (and thus also the $x_i^{(j)}$) from

$$\begin{aligned} G^{(j)} &\equiv \bar{w} g^{(j)} = \sum_{i=1}^{n} x_i^{(j)} b_{\pi(i)} \bar{w} \equiv \sum_{i=1}^{n} x_i^{(j)} a_{\pi(i)} \mod m \\ &= \sum_{i=1}^{n} x_{\mu(i)}^{(j)} a_{\pi(\mu(i))} = \sum_{i=1}^{n} x_{\mu(i)}^{(j)} a_i \mod m, \end{aligned}$$

by solving the easier 0-1 knapsack problems $K(a_1, \ldots, a_n; G^{(j)})$ with superincreasing sequence $a_1, \ldots, a_n$.

Merkle-Hellmann procedure (based on knapsack problems).

In 1982, Shamir [Shamir1982] specified an algorithm for breaking the system in polynomial time without solving the general knapsack problem. Len Adleman [Adleman1982] and Jeff Lagarias [Lagarias1983] specified an algorithm for breaking the twice iterated Merkle-Hellman knapsack encryption procedure in polynomial time. Ernst Brickell [Brickell1985] then specified an algorithm for breaking multiply iterated Merkle-Hellman knapsack encryption procedures in polynomial time. This made this procedure unsuitable as an encryption procedure. It therefore delivers a

one-way function whose trapdoor information (defamiliarisation of the 0-1 knapsack problem) could be discovered by a perpetrator.

4.3 Decomposition into prime factors as a basis for public-key procedures

4.3.1 The RSA procedure

As early as 1978, R. Rivest, A. Shamir, L. Adleman [RSA1978] introduced the most important asymmetric cryptography procedure to date.

Key generation:

Let p and q be two different prime numbers and $N = pq$. Let e be any prime number relative to $\phi(N)$, i.e. $\text{ggT}(e, \phi(N)) = 1$. Using the Euclidean algorithm, we calculate the natural number $d < \phi(N)$, such that

$$ed \equiv 1 \mod \phi(N).$$

whereby ϕ is the **Euler phi Function**.
The output text is divided into blocks and encrypted, whereby each block has a binary value $x^{(j)} \leq N$.

Public key:

$$N, e.$$

Private key:

$$d.$$

Encryption:

$$y = e_T(x) = x^e \mod N.$$

Decryption:

$$d_T(y) = y^d \mod N$$

RSA procedure (based on the factorisation problem).

Comment: The Euler phi function is defined as: $\phi(N)$ is the number of natural numbers that do not have a common factor with N $x \leq N$. Two natural numbers a and b are coprime if $\text{ggT}(a, b) = 1$.

For the Euler phi function: $\phi(1) = 1, \phi(2) = 1, \phi(3) = 2, \phi(4) = 2, \phi(6) = 2, \phi(10) = 4, \phi(15) = 8$. For example, $\phi(24) = 8$, because $|\{x < 24 : \text{ggT}(x, 24) = 1\}| = |\{1, 5, 7, 11, 13, 17, 19, 23\}|$.

If p is a prime number, then $\phi(p) = p - 1$.

If we know the various prime factors $p_1, \ldots, p_k$ of N,, then $\phi(N) = N \cdot (1 - \frac{1}{p_1}) \cdots (1 - \frac{1}{p_k})$.

In the case of $N = pq$, $\phi(N) = pq(1 - 1/p)(1 - 1/q) = p(1 - 1/p)q(1 - 1/q) = (p - 1)(q - 1)$.

n	$\phi(n)$	The natural numbers that are coprime to n and less than n.
1	1	1
2	1	1
3	2	1, 2
4	2	1, 3
5	4	1, 2, 3, 4
6	2	1, 5
7	6	1, 2, 3, 4, 5, 6
8	4	1, 3, 5, 7
9	6	1, 2, 4, 5, 7, 8
10	4	1, 3, 7, 9
15	8	1, 2, 4, 7, 8, 11, 13, 14

The function e_T is a one-way function whose trapdoor information is the decomposition into primes of N.

At the moment, no algorithm is known that can factorise two prime numbers sufficiently quickly for extremely large values (e.g. for several hundred decimal places). The quickest algorithms known today [Stinson1995] factorise a compound whole number N in a time period proportional to $L(N) = e^{\sqrt{\ln(N)\ln(\ln(N))}}$

N	10^{50}	10^{100}	10^{150}	10^{200}	10^{250}	10^{300}
$L(N)$	$1.42 \cdot 10^{10}$	$2.34 \cdot 10^{15}$	$3.26 \cdot 10^{19}$	$1.20 \cdot 10^{23}$	$1.86 \cdot 10^{26}$	$1.53 \cdot 10^{29}$

As long as no better algorithm is found, this means that values of the order of magnitude 100 to 200 decimal places are currently safe. Estimates of the current computer technology indicate that a number with 100 decimal places could be factorised in approximately two weeks at justifiable cost. Using an expensive configuration (e.g. of around 10 million US dollars), a number with 150 decimal places could be factorised in about a year. A 200−digit number should remain impossible to factorise for a long time to come, unless there is a mathematical breakthrough. For example, it would take about 1000 years to decompose a 200-digit number into prime factors using the existing algorithms; this applies even if 10^{12} operations can be performed per second, which is beyond the performance of current technology and would cost billions of dollars in development costs. However, you can never be sure that there won't be a mathematical breakthrough tomorrow.

To this date, it has not been proved that the problem of breaking RSA is equivalent to the factorisation problem. Nevertheless, it is clear that the RSA procedure will no longer be safe if the factorisation problem is solved.

4.3.2 Rabin public-key procedure (1979)

In this case it has been shown that the procedure is equivalent to breaking the factorisation problem. Unfortunately, this procedure is susceptible to chosen-ciphertext attacks.

Let p and q be two different prime numbers with $p, q \equiv 3 \mod 4$ and $n = pq$. Let $0 \leq B \leq n-1$.

Public key:

$$e = (n, B).$$

Private key:

$$d = (p, q).$$

Encryption:

$$y = e_T(x) = x(x+B) \mod n.$$

Decryption:

$$d_T(y) = \sqrt{y + B^2/4} - B/2 \mod n$$

Rabin procedure (based on the factorisation problem).

Caution: Because $p, q \equiv 3 \mod 4$ the encryption is easy to calculate (if the key is known). This is not the case for $p \equiv 1 \mod 4$. In addition, the encryption function is not injective: There are precisely four different source codes that have $e_T(x)$ as inverse image: $x, -x - B, \omega(x + B/2) - B/2, -\omega(x + B/2) - B/2$, where ω is one of the four roots of unity. The source codes therefore must be redundant for the encryption to remain unique!!!

Backdoor information is the decomposition into prime numbers of $n = pq$.

4.4 The discrete logarithm as a basis for public-key procedures

Discrete logarithms form the basis for a large number of algorithms for public- key procedures.

4.4.1 The discrete logarithm in $\mathbb{Z}_p$

Let p be a prime number and let $g \in \mathbb{Z}_p^* = \{0, 1, \ldots, p-1\}$. Then the discrete exponential function base g is defined as

$$e_g : k \longrightarrow y := g^k \mod p, \quad 1 \leq k \leq p-1.$$

The inverse function is called a discrete logarithm function $\log_g$; the following holds:

$$\log_g(g^k) = k.$$

The problem of the discrete logarithm (in $\mathbb{Z}_p^*$) is understood to be as follows:

$$\text{Given } p, g \text{ and } y, \text{ determine } k \text{ such that } y = g^k \mod p.$$

It is much more difficult to calculate the discrete logarithm than to evaluate the discrete exponential function (see 3.4.4). There are several procedures for calculating the discrete logarithm [Stinson1995]:

Name	Complexity
Baby-Step-Giant-Step	$O(\sqrt{p})$
Silver-Pohlig-Hellman	polynomial in q, the greatest prime factor of $p-1$.
Index-Calculus	$O(e^{(1+o(1))\sqrt{\ln(p)\ln(\ln(p))}})$

4.4.2 Diffie-Hellman key agreement

The mechanisms and algorithms of classical cryptography only take effect when the subscribers have already exchanged the secret key. In classical cryptography, you cannot avoid exchanging secrets without encrypting them. Transmission safety here must be achieved using non-cryptographic methods. We say that we need a secret channel for exchanging secrets. This channel can be realised either physically or organisationally.
What is revolutionary about modern cryptography is, amongst other things, that you no longer need secret channels: You can agree secret keys using non-secret, i.e. public channels.
One protocol that solves this problems is that of Diffie and Hellman

Two subscribers A and B want to agree on a joint secret key.
Let p be a prime number and g a natural number. These two numbers do not need to be secret.
The two subscribers then select a secret number a and b. from which they calculate the values $\alpha = g^a \mod p$ and $\beta = g^b \mod p$. They then exchange the numbers α and β. To end with, the two subscribers calculate the received value to the power of their secret value to get $\beta^a \mod p$ and $\alpha^b \mod p$.
Thus

$$\beta^a \equiv (g^b)^a \equiv g^{ba} \equiv g^{ab} \equiv (g^a)^b \equiv \alpha^b \mod p$$

Diffie Hellman (Key agreement).

The safety of the **Diffie-Hellman protocol** is closely connected to calculating the discrete logarithm mod p. It is even thought that these problems are equivalent.

4.4.3 ElGamal public-key encryption procedure in $\mathbb{Z}_p^*$

By varying the Diffie-Hellman key agreement protocol slightly, you can obtain an asymmetric encryption algorithm. This observation was made by Taher ElGamal.

Let p be a prime number such that the discrete logarithm in $\mathbb{Z}_p$ is difficult. Let $\alpha \in \mathbb{Z}_p^*$ be a primitive element. Let a and $\beta = \alpha^a \mod p$.

Public key:

$$p, \alpha, \beta.$$

Private key:

$$a.$$

Let $k \in \mathbb{Z}_{p-1}$ be a random number and $x \in \mathbb{Z}_p^*$ the plaintext.

Encryption:

$$e_T(x, k) = (y_1, y_2),$$

where

$$y_1 = \alpha^k \mod p$$

and

$$y_2 = x\beta^k \mod p.$$

Decryption:

$$d_T(y_1, y_2) = y_2(y_1^a)^{-1} \mod p$$

ElGamal procedure (based on the factorisation problem).

4.4.4 Generalised ElGamal public-key encryption procedure

The discrete logarithm can be generalised in any number of finite groups $(G, \circ)$. The following provides several properties of G, that make the discrete logarithm problem difficult.

Calculating the discrete exponential function Let G be a group with the operation $\circ$ and $g \in G$. The (discrete) exponential function base g is defined as

$$e_g : \ \mathrm{N} \ni k \longrightarrow g^k, \quad \text{for all } k \in \mathrm{N}.$$

where

$$g^k := \underbrace{g \circ \ldots \circ g}_{k \text{ times}}.$$

The exponential function is easy to calculate:

Lemma. *The power g^k can be calculated in at most $2 \log_2 k$ group operations.*

Proof
Let $k = 2^n + k_{n-1}2^{n-1} + \cdots + k_1 2 + k_0$ be the binary representation of k. Then $n \leq \log_2(k)$, because $2^n \leq k < 2^{n+1}$. k can be written in the form $k = 2k' + k_0$ with $k' = 2^{n-1} + k_{n-1}2^{n-2} + \cdots + k_1$. Thus

$$g^k = g^{2k'+k_0} = (g^{k'})^2 g^{k_0}.$$

We therefore obtain g^k from $g^{k'}$ by squaring and then multiplying by g. The claim is thus proved by induction to n. □

Problem of the discrete logarithm

Let G by a finite group with the operation $\circ$. Let $\alpha \in G$ and $\beta \in H = \{\alpha^i : i \geq 0\}$. We need to find a unique $a \in \mathrm{N}$ with $0 \leq a \leq \lvert H\rvert - 1$ and $\beta = \alpha^a$. We define a as $\log_\alpha(\beta)$.

Calculating the discrete logarithm A simple procedure for calculating the discrete logarithm of a group element, that is considerably more efficient than simply trying all possible values for k, is the Baby-Step-Giant-Step algorithm.

Theorem 1 (Baby-Step-Giant-Step algorithm). *Let G be a group and $g \in G$. Let n be the smallest natural number with $\lvert G\rvert \leq n^2$. Then the discrete logarithm of an element $h \in G$ can be calculated base g by generating two lists each containing n elements and comparing these lists. In order to calculate these lists, we need $2n$ group operations.*

Proof
First create the two lists
Giant-Step list: $\{1, g^n, g^{2n}, \ldots, g^{n \cdot n}\}$,
Baby-Step list: $\{hg^{-1}, hg^{-2}, \ldots, hg^{-n}\}$.
If $g^{jn} = hg^{-i}$, i.e. $h = g^{i+jn}$, then the problem is solved. If the lists are disjoint, then h cannot be represented as $g^{i+jn}, i, j \leq n,$. As all powers of g are thus recorded, the logarithm problem does not have a solution. □

You can use the Baby-Step-Giant-Step algorithm to demonstrate that it is much more difficult to calculate the discrete logarithm than to calculate the discrete exponential function. If the numbers that occur have approximately 1000 bits in length, then you only need around 2000 multiplications to calculate g^k but around $2^{500} \approx 10^{150}$ operations to calculate the discrete logarithm using the Baby-Step-Giant-Step algorithm.
In addition to the Baby-Step-Giant-Step algorithm, there are also numerous other procedures for calculating the discrete logarithm [Stinson1995].

The theorem from Silver-Pohlig-Hellmann In finite abelian groups, the discrete logarithm problem can be reduced to groups of a lower order.

Theorem 2 (Silver-Pohlig-Hellmann). *Let G be a finite abelian group with $|G| = p_1^{a_1} p_2^{a_2} \cdot \ldots \cdot p_s^{a_s}$. The discrete logarithm in G can then be reduced to solving logarithm problems in groups of the order $p_1, \ldots, p_s$.*

If $|G|$ contains a dominant prime factor p, then the complexity of the logarithm problem is approximately

$$O(\sqrt{p}).$$

Therefore, if the logarithm problem is to be made difficult, the order of the group used G should have a large prime factor. In particular, if the discrete exponential function in the group $\mathbb{Z}_p^*$ is to be a one-way function, then $p - 1$ must be a large prime factor.

Let G be a finite group with operation $\circ$, and let $\alpha \in G$, so that the discrete logarithm in $H = \{\alpha^i : i \geq 0\}$ is difficult, Let a with $0 \leq a \leq |H| - 1$ and let $\beta = \alpha^a$.

Public key:

$$\alpha, \beta.$$

Private key:

$$a.$$

Let $k \in \mathbb{Z}_{|H|}$ be a random number and $x \in G$ be a plaintext.

Encryption:

$$e_T(x, k) = (y_1, y_2),$$

where

$$y_1 = \alpha^k$$

and

$$y_2 = x \circ \beta^k.$$

Decryption:

$$d_T(y_1, y_2) = y_2 \circ (y_1^a)^{-1}$$

Generalised ElGamal procedure (based on the factorisation problem).

Elliptic curves provide useful groups for public-key encryption procedures.

References

[Adleman1982] Adleman L.: *On breaking the iterated Merkle-Hellman public key Cryptosystem.*
Advances in Cryptologie, Proceedings of Crypto 82, Plenum Press 1983, 303-308.

[Balcazar1988] Balcazar J.L., Daaz J., Gabarr´ J.: *Structural Complexity I.*
Springer Verlag, pp 63.

[Brickell1985] Brickell E.F.: *Breaking Iterated Knapsacks.*
Advances in Cryptology: Proc. CRYPTÓ84, Lecture Notes in Computer Scince, vol. 196, Springer-Verlag, New York, 1985, pp. 342-358.

[Lagarias1983] Lagarias J.C.: *Knapsack public key Cryptosystems and diophantine Approximation.*
Advances in Cryptologie, Proseedings of Crypto 83, Plenum Press.

[Merkle1978] Merkle R. and Hellman M.: *Hiding information and signatures in trapdoor knapsacks.*
IEEE Trans. Information Theory, IT-24, 1978.

[RSA1978] Rivest R.L., Shamir A. and Adleman L.: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems.*
Commun. ACM, vol 21, April 1978, pp. 120-126.

[Shamir1982] Shamir A.: *A polynomial time algorithm for breaking the basic Merkle- Hellman Cryptosystem.*
Symposium on Foundations of Computer Science (1982), 145-152.

[Stinson1995] Stinson D.R.: *Cryptography.*
CRC Press, Boca Raton, London, Tokyo, 1995.

URLs

1. http://www.geocities.com /st_busygin/clipat.html

5 Digital signatures

The aim of digital signatures is to guarantee the following two points:

- User authenticity:
  It can be checked whether a message really does come from a particular person.
- Message integrity:
  It can be checked whether the message has been changed (on route).

An asymmetric technique is used again (see encryption procedures). Participants who wish to generate a digital signature for a document must possess a pair of keys. They use their secret key to generate signatures and the recipient uses the sender's public key to verify whether the signature is correct. As before, it must be impossible to use the public key to derive the secret key.

In detail, a *Signature procedure* looks like this:
Senders use their message and secret key to calculate the digital signature for the message. Compared to hand-written signatures, therefore, digital signatures have the advantage that they also depend on the document to be signed. Signatures from one and the same participant are different unless the signed documents are completely identical. Even inserting a blank in the text would lead to a different signature. The recipient of the message would therefore detect any injury to the message integrity as this would mean that the signature no longer matches the document and is shown to be incorrect when verified.

The document is sent to the recipient together with the signature. The recipient can then use the sender's public key, the document and the signature to establish whether or not the signature is correct. In practice, however, the procedure we have just described has a decisive disadvantage. The signature is approximately as long as the document itself. To prevent an unnecessary increase in data traffic, and also for reasons of performance, we use a cryptographic hash function.

A cryptographic *hash function* maps a message of any length to a string of characters with a constant size (usually 128 or 160 bits), the hash value. It should be practically impossible, for a given number, to find a message that has precisely this number as hash value. Furthermore, it should be practically impossible to find two messages with the same hash value. In both cases the final signature procedure would display weak points.

So far, no formal proof has been found that perfectly secure cryptographic hash functions exist. However, there are several good candidates that have not yet shown any weak points in practice (e.g. SHA1 or RIPEMD160).

The hash function procedure is as follows:
Rather than signing the actual document, the sender now first calculates the hash value of the message and signs this. The recipient also calculates the hash value of the message (the algorithm used must be known), then verifies whether the signature sent with the message is a correct signature of the hash value. If this is the case, the signature is verified to be correct. This means that the message is authentic, because we have assumed that knowledge of the public key

does not enable you to derive the secret key. However, you would need this secret key to sign messages in another name.

All asymmetric encryption procedures can also be used to create digital signatures. Here, a signature is simply the message text encrypted using the sender's secret key (or hash value). The recipient uses the sender's public key to decrypt the signature and compares the obtained value with the message text (or the hash value).

Of course, you can only use encryption algorithms that allow you to exchange the encryption and decryption processes. One of these algorithms, which can be used both as an asymmetric encryption procedure and as a signature algorithm, is the RSA algorithm.

However, there are also signature procedures that cannot be used to encrypt messages (e.g. the DSA = Digital Signature Algorithm). This was particularly important in countries such as France, in which procedures are allowed for digital signatures but where you are not permitted to encrypt messages without authorisation.

5.1 Public-key certification

The aim of public-key certification is to guarantee the connection between a public key and a user and to make it traceable for external parties. In cases in which it is impossible to ensure that a public key really belongs to a particular person, many protocols are no longer secure, even if the individual cryptographic modules cannot be broken.

5.1.1 Impersonation attacks

Assume Charlie has two pairs of keys (PK1, SK1) and (PK2, SK2), where SK denotes the secret key and PK the public key. Further assume that he manages to palm off PK1 on Alice as Bob's public key and PK2 on Bob as Alice's public key (by falsifying a public key directory).

Then he can attack as follows:

- Alice wants to send a message to Bob. She encrypts it using PK1 because she thinks that this is Bob's public key. She then signs the message using her secret key and sends it.
- Charlie intercepts the message, removes the signature and decrypts the message using SK1. If he wants to, he can then change the message in any way he likes. He then encrypts the message again, but this time using Bob's genuine public key, which he has taken from a public key directory, signs the message using SK2 and forwards it to Bob.
- Bob verifies the signature using PK2 and will reach the conclusion that the signature is correct. He then decrypts the message using his secret key.

Charlie can thus listen in on communication between Alice and Bob and change the exchanged messages without them noticing. The attack also works if Charlie only has one pair of keys.

Another name for this type of attack is man-in-the-middle attack. Users are promised protection against this type of attack by publickey certification, which is intended to guarantee the authenticity of public keys. The most common certification method is the X.509 standard.

5.1.2 X.509

All participants who want to have X.509 verify that their public key belongs to a real person consult what is known as a certification authority (CA). They prove their identity to this CA (for example by showing their ID). The CA then issues them an electronic document (certificate) which essentially contains the names of the certificate- holder and the CA, the certificate-holder's public key and the validity period of the certificate. The CA then signs the certificate using its secret key.

Anyone can now use the CA's public key to verify whether a certificate is falsified. The CA therefore guarantees that a public key belongs to a particular user.

This procedure is only secure as long as it can be guaranteed that the CA's public key is correct. For this reason, each CA has its public key certified by another CA that is superior in the hierarchy. In the upper hierarchy level there is usually only one CA, which can of course then have its key certified by another CA. It must therefore transfer its key securely in another way. In the case of many software products that work with certificates (such as the Microsoft and Netscape Web browsers), the certificates of these root CAs are permanently embedded in the program right from the start and cannot be changed by users at a later stage. However, (public) CA keys, in particularly those of the root entity, can also be secured by means of making them available publicly.

6 Elliptic Curves

6.1 Elliptic curves — history

Mathematicians have been researching elliptic curves for over 100 years. In the course of time, many lengthy and mathematically complex results have been found and published in connection with elliptic curves. A mathematician would say that elliptic curves (or the mathematics behind them) are widely understood. This research was originally purely mathematical. That is to say, elliptic curves were investigated, for example, in the mathematical areas of number theory and algebraic geometry, which are generally highly abstract. Even in the recent past, elliptic curves played an important role in pure mathematics. In 1993 and 1994, Andrew Wiles published mathematical works that triggered enthusiasm far beyond the specialist audience. In these works, he proved a conjecture put forward in the 1960's. To put it short, this conjecture was concerned with the connection between elliptic curves and what are called module forms. That which is interesting for most people is that the works of Wiles also proved the famous second theorem of Fermat. Mathematicians had spent centuries (Fermat lived from 1601 to 1665) trying to find a strict proof of this theorem. Understandably, therefore, Wiles' proof got a good response. Fermat formulated his theorem as follows (written in the border of a book):

> *Cubum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos, et generaliter nullam in infinitum ultra quadratum potestatem in duos ejusdem nominis fas est dividere: cujus rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet.*

Translated freely, using the denotation of modern mathematics, this means:
No positive whole numbers x, y and z greater than zero exist such that $x^n + y^n = z^n$ for $n > 2$. I have found an amazing proof of this fact, but there is too little space in the border [of the book] to write it down.

This is truly amazing: A statement that is relatively simple to understand (we are referring to Fermat's second theorem here) could only be proved after such a long period of time, although Fermat himself claimed to have found a proof. What's more, the proof found by Wiles is extremely extensive (all of Wiles publications connected with the proof made up a book in themselves). This should therefore make it obvious that elliptic curves are generally based on highly complex mathematics.

Enough about the role of elliptic curves in pure mathematics. In 1985 Neal Koblitz and Victor Miller independently suggested using elliptic curves in cryptography. Elliptic curves have thus also found a concrete practical application. Another interesting field of application for elliptic curves is for factorising whole numbers. (For example the RAS cryptography system is based on the difficulty/complexity of finding prime factors of an extremely large number.) In this area, procedures based on elliptic curves have been investigated and partially used since 1987 (a study by H.W. Lenstra). There are also prime number tests based on elliptic curves.

Elliptic curves are used differently in the various areas. Encryption procedures based on elliptic curves are based on the difficulty of a problem known as elliptic curve discrete logarithm. The

factorisation of whole numbers uses the fact that a large number of elliptic curves can be generated for a natural composite number n with several prime factors; however, these curves are not then groups for composite n. More information about this can be found under Factorisation using elliptic curves.

6.2 Elliptic curves — mathematical basics

This section provides information about *groups* and *fields*.

6.2.1 Groups

Because the term *group* is used differently in everyday language than in mathematics, we will, for reasons of completeness, begin by introducing the essential statement of the formal definition of a group:

- A group is a non-empty set G and an operation $+$. The set G is closed under the operation $+$. Regardless of which two elements a, b from G are taken, performing the operation on them gives an element in G (i.e. $a + b = c$, and c lies in G).
- For all elements a, b and c in G: $(a + b) + c = a + (b + c)$.
- There exists an element e in G that behaves neutrally with respect to the operation $+$. That means that for all a in the set G: $a + e = e + a = a$.
- For each element a in G there exists a so-called inverse element $-a$ ($-a$ also lies in G) such that: $a + (-a) = (-a) + a = e$.

If also $a+b = b+a$ for all a, b in G then we call the group an *Abelian* group. An operation denoted as $+$ indicates an *additive* group; if the operation is denoted as $\cdot$, we speak of a *multiplicative* group.

The simplest example of an (Abelian) group is the group of whole numbers under the standard operation of addition. The set of whole numbers is denoted as $\mathbb{Z}$. $\mathbb{Z}$ has an infinite number of elements, because $\mathbb{Z} = \{..., -4, -3, -2, -1, 0, 1, 2, 3, 4, ...\}$. For example, the operation of $1 + 2$ lies in $\mathbb{Z}$, for $1 + 2 = 3$ and 3 lies in $\mathbb{Z}$. The neutral element in the group $\mathbb{Z}$ is 0. The inverse element of 3 is -3, for $3 + (-3) = 0$.

There are also *finite* groups. This means that these exists a set $\mathcal{M}$ with a fixed number of elements and an operation $+$ such that the above conditions are fulfilled. One example of this is any set $\mathbb{Z}_n$ where $\mathbb{Z}_n = \{0, 1, 2, 3, ..., n - 1\}$, n is a positive whole number and the operation is addition mod n, i.e. a and b in $\mathbb{Z}_n$ are subject to the operation $a + b \bmod n$.

Cyclic groups Cyclic groups are those groups G' that possess an element g from which the group operation can be used to generate all other elements in the group. This means that for each element a in G' there exists a positive whole number i such that if g is subject to the operation i

times (i.e. $g \cdot i$), $g + g + ... + g = a$ (additive group) or $g^i = g \cdot g \cdots g = a$ (multiplicative group). The element g is the *generator* of the cyclic group — each element in G can be generated using g and the operation.

Now to the order of an element of the group: Let a be in G. The smallest positive whole number r for which a subject to the operation with itself r times is the neutral element of the group G (i.e.: $r \cdot a = a + a + ... + a = e$ bzw. $a^r = e$), is called the *order* of a.

The order of the group is the number of elements in the set G.

6.2.2 Fields

In mathematics, a field is understood to be a set K with two operations (denoted as + and $\cdot$) which fulfils the following conditions:

- The set K forms an Abelian group together with the operation + (addition), where 0 is the neutral element of the operation s.
- The set K (without the element 0) also forms an Abelian group together with the operation $\cdot$ (multiplication).
- For all elements a, b and n in K, $n \cdot (a + b) = n \cdot a + n \cdot b$ and $(a + b) \cdot n = a \cdot n + b \cdot n$.

There are *infinite* fields, i.e. the set on which the field is based contains an infinite number of elements (e.g.: the field of real numbers). And there are also finite fields, such as $\mathbb{Z}_p = \{0, 1, 2, 3, ..., p - 1\}$, where p is a prime. $\mathbb{Z}_p$ with addition mod p and multiplication mod p is a finite field.

Characteristic of a field Let K be a field and 1 be the neutral element of K with respect to the multiplicative operation $\cdot$. For positive natural numbers n, let us understand n_1 to be $n_1 = 1 + 1 + ... + 1$ (n summands and n_1 is an element in K). If n_1 is then unequal to 0 for all $n > 0$, then we call K a field with characteristic zero. Otherwise, the characteristic of K is defined to be the smallest positive natural number p for which $p_1 = 0$ (note: p is then a prime). Comment: The field of real numbers has the characteristic 0; the field $\mathbb{Z}_p$ has the characteristic p.

6.3 Elliptic curves in cryptography

An elliptic curve is described by an equation. In order to keep it simple, we restrict our explanation to elliptic curves over

$$\mathbb{Z}_p = \{0, 1, 2, 3, ..., p - 1\}$$

where p is a prime greater than 3. $\mathbb{Z}_p$ with addition mod p and multiplication mod p is a finite field. However, we must mention that elliptic curves can be defined over any (finite) field. In particular, elliptic curves over fields with characteristic 2 are extremely interesting from a practical point of view because computers can be used to represent the elements from these fields

as bit strings. This leads to an efficient implementation of the arithmetic in such fields, which means that a computer can perform the operations of the field particularly quickly.

An elliptic curve over $\mathbb{Z}_p$ is defined by an equation of the following form:

$$y^2 \pmod p = x^3 + ax + b (\mathrm{mod}\ p)$$

(thus: equality in the field $\mathbb{Z}_p$), where a, b are in $\mathbb{Z}_p$ and $4a^3 + 27b^2$ mod p is not equal to zero. For fixed chosen numbers a and b in $\mathbb{Z}_p$, this equation has the pair of solutions

$$\mathbf{E} = \left\{ (x,y) \;\middle|\; \begin{array}{c} x \text{ and } y \text{ are in } \mathbb{Z}_p \text{ and} \\ y^2 \equiv x^3 + ax + b \pmod p \text{ and} \\ 4a^3 + 27b^2 \not\equiv 0 \pmod p \end{array} \right\},$$

i.e. the set $\mathbf{E}$ consists of all pairs x and y that are a solution (in $\mathbb{Z}_p$) of the above equation. It must be noted that the numbers a, b and p determine which pairs (x, y) lie in the set $\mathbf{E}$. This means that a, b and p specify this set. The elements (x, y) in $\mathbf{E}$ are called points on the elliptic curve. In addition, $\mathbf{E}$ has one more element O (the so-called point in infinity). The set $\mathbf{E}$ is usually called an elliptic curve. We can now define an operation (also denoted as $+$, although it is not the standard/usual addition of real numbers) on two elements in $\mathbf{E}$ such that the operation delivers an element that also lies in $\mathbf{E}$. The set $\mathbf{E}$ is therefore closed under the operation $+$. We can show that $\mathbf{E}$ is a group. The neutral element of the group $\mathbf{E}$ is the point in infinity O. Thus, for every two points (x_1, y_1) and (x_2, y_2) on the elliptic curve $\mathbf{E}$, there exists a point (x_3, y_3) on $\mathbf{E}$ such that the operation $+$ complies with the following: $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$. Under certain circumstances, these points may also be equal to the point in infinity. Thus, when we speak of a point P on an elliptic curve $\mathbf{E}$, we mean that $P = (x, y)$ and (x, y) lies in the set $\mathbf{E}$. Any two points on an elliptic curve specified by a, b and p can therefore be added and the result is a point that also lies on the same elliptic curve. We must note that $\mathbf{E}$ can have the following meanings:

- the set $\mathbf{E}$ of solutions pairs (x, y) for an equation including the point O
- the group $\mathbf{E}$ (with the operation addition of (x_1, y_1) and (x_2, y_2))
- the elliptic curve $\mathbf{E}$ (which is actually the same as the group $\mathbf{E}$)

Because these points actually refer to the same thing, it is seldom necessary to distinguish between exact meanings.

For cryptography, the important fact is that, for very large numbers, it appears to be extremely difficult to use a given point Q on an elliptic curve to determine which two points have to be added to obtain Q. For large numbers a, b and p (p, for example, has a length of more than 160 bits), the computer can easily add the point P m times after another, i.e. to determine the point $P + P + \cdots + P = Q$ in an incredibly short space of time (in a few fractions of a second) (m summands P). Rather than $P + P + \cdots + P = Q$ (m summands P) we also write $mP = Q$. If we have a point P and a point Q, which both lie on the same elliptic curve, no procedure is known that enables us — within an acceptable space of time — to determine the number m (assuming

it actually exists) for which $mP = Q$. This is referred to as the elliptic curve discrete logarithm problem (abbreviated to ECDLP).

We must note that not all elliptic curves are equally secure. This means that we must choose the parameters a and b carefully when defining a curve. For certain classes of elliptic curves, it is possible to solve the ECDLP easier than in the general case. Cryptographically unsuitable elliptic curves are called abnormal curves (these are curves over $\mathbb{Z}_p$ for which the set $\mathbf{E}$ has precisely p elements) and the supersingular curves (the curves for which we can reduce the calculation of the ECDLP to calculating the standard discrete logarithm in other finite fields, i.e. simplify the calculation). There are therefore cryptographically good and bad curves. However, for given parameters a and b we can, with some difficulty, establish whether or not the resulting elliptic curve is useful for cryptographic purposes. The curves used in cryptography are usually provided by experts. They ensure that the elliptic curves they classify as secure satisfy the current security requirements. For secure curves, the parameter p determines how long it takes to solve the ECLDP on this curve. The larger the parameter p, the longer it takes to solve the problem. Experts recommend a bit length of over 200 bits for the parameter p. This makes it clear why elliptic curves are so interesting for cryptography. Because the parameter p also determines the time required to perform the signature/encryption procedure when using elliptic curves in cryptography. The time taken to generate a pair of keys also depends on p. Thus, small values (few bits) are desirable here (in order to minimise the run times for the procedures); however, the required security must still be maintained. For example, with a length of 200 bits for p, a *good* elliptic curve is just as secure as an RSA module of over 1024 bits in length (at least according to the current state of research). The reason for this is that the quickest algorithms for solving the *elliptic curve discrete logarithm* problem have an exponential run time — unlike the subexponential run times that the best factorisation algorithms currently have (number sieve, quadratic sieve or factorisation with elliptic curves). Therefore, the parameters for cryptographic procedures based on the problem of *factorising whole numbers* must be greater than the parameters for cryptographic procedures based on the ECDL problem.

6.3.1 Digital signatures using elliptic curves

The *elliptic curve discrete logarithm problem* (ECLDP) forms the basis for elliptic-curve cryptography. Various signature procedures are based on this. What they have in common is how they use the public parameters and how these parameters are used to generate secret and public keys:

- The parameters of the elliptic curve E, i.e. a prime number p that determines over which field $\mathbb{Z}_p$ the elliptic curve $\mathbf{E}$ is defined, as well as the two numbers a and b in $\mathbb{Z}_p$.
- A point $G = (x, y)$ that lies on the elliptic curve $\mathbf{E}$.
- A prime number $r < p$ for which $rG = O$ (i.e. G added r times gives the neutral element of the group $\mathbf{E}$) and r is a factor of $\#\mathbf{E}$ (where $\#\mathbf{E}$ is the number of elements in the set $\mathbf{E}$). The point G therefore has the order r and is the generator of a cyclic subgroup of $\mathbf{E}$ with the order r.
- The number $k = \#\mathbf{E}/r$ (k is called the *cofactor*).

The parameters a, b, p, G, r and k listed above are called domain parameters. They determine on which elliptic curve $\mathbf{E}$ and in which cyclic subgroup of $\mathbf{E}$ a signature procedure has been used

The secret key s of the signature generator is a (random) whole number s in the interval $[1, r-1]$. The public key of the signature generator is a point $W = (x, y)$ on the elliptic curve $\mathbf{E}$. The public key W and secret key s are interrelated as follows: $W = sG$. This means that the domain parameters (particularly G) and the secret key s are used to calculate the public key W (by adding G s times on $\mathbf{E}$). The ECDLP is obviously used here: If W and G (as well as the other domain parameters used) are known, then it is difficult to use these to calculate s. (If the parameters are chosen correctly, this currently appears to be practically impossible).

In order to verify a signature, the recipient of the signature must know the following:

1. The signature procedure used,
2. The hash function used,
3. The domain parameters used to generate the signature
4. The public key W of the signature generator.

6.3.2 Faktorisation using elliptic curves

There are factorisation algorithms based on elliptic curves. More precisely, these procedures exploit the fact that elliptic curves can be defined over $\mathbb{Z}_n$ (n composite number). Elliptic curves over $\mathbb{Z}_n$ do not form a group, because not every point on such an elliptic curve has an inverse point. This is connected with the fact that - if n is a composite number - there exist elements in $\mathbb{Z}_n$ that do not have an inverse with respect to multiplication mod n. In order to add two points on an elliptic curve over $\mathbb{Z}_n$, we can calculate in the same way as on elliptic curves over $\mathbb{Z}_p$. Addition of two points (on an elliptic curve over $\mathbb{Z}_n$), however, fails if and only if a factor of n has been found. The reason for this is that the procedure for adding points on elliptic curves gives elements in $\mathbb{Z}_n$ and calculates the inverse elements for these (with respect to multiplication mod n) in $\mathbb{Z}_n$. The extended Euclidean algorithm is used here. If the addition of two points (that lie of an elliptic curve over $\mathbb{Z}_n$) gives an element in $\mathbb{Z}_n$ that does not have an inverse element in $\mathbb{Z}_n$, then the extended Euclidean algorithm delivers a genuine factor of n.

Factorisation using elliptic curves thus principally works as follows: You select random curves over $\mathbb{Z}_n$, as well as random points (that lie on this curve) and add them; you thus obtain points that also lie on the curve or find a factor of n. Factorisation algorithms based on elliptic curves therefore work probabilistically. The opportunity of defining large number of elliptic curves over $\mathbb{Z}_n$ allows you to increase the probability of finding two points which you can add to obtain a factor of n. These procedures are therefore highly suitable for parallelisation.

6.4 Implementing elliptic curves

CrypTool uses elliptic curves for the digital signature function.

It implements the basic algorithms for group operations, for generating elliptic curves, for importing and exporting parameters for elliptic curves over finite fields with p (p prime) elements. The algorithms have been implemented in ANSI C and comply with draft no. 8 of the IEEE P1363 work group *Standard Specifications for Public-Key Cryptography*

http://grouper.ieee.org/groups/1363.

The procedure implements the cryptographic primitives for generating and verifying signatures for the variations of Nyberg-Rueppel signatures and DAS signatures based on elliptic curves (in accordance with draft no. 8 of the IEEE P1363 work group). This was done in collaboration with the SECUDE GmbH — using the above library and the SECUDE SDK.

Index