Computational Number Theory and Cryptography

Preda Mihăilescu and Michael Th. Rassias

Abstract This is a succinct survey of the development of cryptography with accent on the public key age. The paper is written for a general, technically interested reader. We also review some fundamental mathematical ideas of computational number theory that play an important role in present time cryptography.

Keywords Computational number theory • Cryptography • Elliptic curves over finite fields • Diffie-Hellman algorithm

2000 Mathematics Subject Classification: 11Y11, 11G05, 11Y16, 11Y40, 68Q17, 68Q25

Introduction

Cryptography is the collection of methods and approaches for concealing information in communications from the access by unwished or unauthorized parties. A logical art for dealing with this problem is known from early Antiquity and

This paper substantially improves and extends the former article of the authors, that appeared in [MR].

P. Mihăilescu
Mathematisches Institut der Universität Göttingen, Germany
e-mail: preda@uni-math.gwdg.de

M.Th. Rassias (✉)
Department of Mathematics, ETH-Zürich, Rämistrasse 101, 8092 Zürich, Switzerland
e-mail: michail.rassias@math.ethz.ch

N.J. Daras (ed.), *Applications of Mathematics and Informatics in Science and Engineering*, Springer Optimization and Its Applications 91, DOI 10.1007/978-3-319-04720-1_22,

it developed along the centuries, mostly in the frame in which two parties, say nobleman and general, or concealed lovers, communicated in writing by sending each other messages, which could only be understood when knowing some additional data—*secret keys*—and the details for the procedure of encrypting and decrypting the messages—*algorithm*. Algorithms were often assembled from a collection of useful basic ideas, known by tradition.

Traditional Secret Key Cryptography

Transposing the alphabet of a spoken language into a sequence of numeric codes is always useful for discussing cryptographic ideas. Suppose thus that the Latin alphabet $a, b, \ldots, z$ is encoded in ascending order by the numbers $0, 1, \ldots, 24$. The idea of permuting the letters cyclically by a constant σ was purportedly used by Caesar in the Gallic wars—hence the name of *Caesar* code. For instance, for $\sigma = 3$, the word ATHENS becomes DWKHQV. For decryption, use $\sigma = 25 - 3 = 22$. One can improve the security of this code, by using context specific keys, key sequences, and other well-defined combination—such variations were investigated in the 16th century by the French diplomat Blaise de Vigenère. The purpose was to counter the obvious weakness of the Caesar code with respect to *frequency* attacks: provided a sufficiently large cipher code, and knowing that letters like e, a, m occur much more frequently than z, h, q, one can easily determine the value of σ, thus compromising the whole encryption. Since these ideas can in addition be combined with some commonly known text modifications, the bag of tricks for artisanal cryptography offered *sufficient* variety for satisfying the needs until the advent of the 20th century. In parallel with the development of new, particular algorithms of encryption, the analysis of methods for discovering both keys and the particularities of an encryption procedure—like for instance, the frequency analysis mentioned for the Caesar cipher—developed itself into the science of *cryptanalysis*. Today, cryptanalysis and cryptography are regarded as the two complementary aspects of the science of *cryptology*. While the creation of private codes and keys could be considered to some extent as a playful, even enjoyable undertaking, which requires some rigor though, for preventing countermeasures of the cryptanalyst, the classical encryption has one more important limitation: the peers need to be in anticipated agreement regarding both of the encryption algorithm and the keys. This leads to several consequences: the first is that one would wish the algorithm to be so strong, that it suffices to exchange the keys while keeping the same algorithm over longer periods. The second is that one needs well-trained and faithful couriers for the keys.

In order to illustrate the methods and challenges of classical ciphers, we propose to the reader to try and decrypt the following small text, which was encrypted by a scheme developed by the 10-year-old daughter of the first author starting from a children's game encryption, found in a book, and which they use for discussing within a gang of good friends. The cipher text is:

$$TGGMCITGWKKNKVCTZCOKNUKECFGOZCCFCWK$$

It is obtained by a combination of the ideas discussed above.

The Advent of Computing Machines . . .

In the 20th century, military confrontations became more devastating, and the power of data processing with the help of machines increased without precedent. However, until World War II and even later, the basic scheme for secure communication remained the same: the secret communicators shared some common algorithm, which could eventually be performed by a machine, and they were using some *shared secret key*, for the diffusion of which many lives were put in danger. One of the most documented episodes of warfare use of cryptography was the German development of their encryption machine Enigma II on the basis of a simpler, earlier version Enigma I, which had been a commercial product before the war.[1] Little did they know that this unfortunate combination of economic and military application had led to the fact that a team of young Polish mathematicians from Poznan were in possession of a means for breaking Enigma I. When hired by the British authorities, the work for breaking the enhanced version was, against the German expectation, an achievable one, and the breaking of Enigma had its important consequences for the outcome of the war [10, 19].

. . . and of Personal Computers and Networked Communications

The advent of computers brought on the one hand the massive improvement of computational capacities, then, in the early 1970s, and on the other as the US-army built the ARPA-net, the advent of networked communications, an ancestor of the Internet. In light of this progress, cryptography was led into simplifying the definition of its object and tasks. Some very useful principles have been established, which stay to hold. First, it was understood that there is little security in the use of proprietary, secret algorithms—the choice of cryptographers going in the direction of simple, publicly known and well-understood and cryptanalized algorithms. As long as the bag of tricks is known, it can even happen more easily that a flaw escapes in the design of a proprietary algorithm. As a consequence the assumed gain of security obtained from keeping a secret of the cryptographic procedure is counterbalanced by the insecurity stemming from the lack of reliable cryptanalysis. In simple words, the modern attitude to security is resumed in the

[1]The development of Enigma I during the early days of mechanical office machine, shows that there has always existed an important requirement for cryptography also in business.

paradigm publically known and cryptanalized algorithm and secret keys. As a consequence, the protection of keys becomes the center of the security concerns and is offered the due attention: the system is as secure as the keys are. In addition, the new approach to cryptography promises that, due to the collective scrutiny of the cryptographic community, in time the most efficient and reliable algorithms are naturally selected, while weaknesses and possible attacks eventually show up in the processes. An algorithm is more reliable when it has longly resisted public scrutiny by the community, and *not* when it is based on sophisticated *secret tricks*.

We mentioned that in early times of cryptography, secret keys were transported by couriers, which brought their life in danger for this purpose, while even later, in times of telegraphic transmission of keys, the problem of building secure channels for key transmission was a crucial one. In the seventies, the first networked system of computers became conceivable. It physically realized by the American army, in the form of the ARPA net, which first connected between 1972–1974 a number of Universities on the East and West Coast, for research and experimental purposes. The notion of remote computer-communication became tangible for the users of the net.

Under these conditions, it became obvious that the old systems for secure key distribution could no longer satisfy the needs of security for this technological advance and some new ideas were called for, in order to solve the problem in a simple, time-efficient, and reliable way.

The idea was provided by the concept of public-key cryptography, which was born in Stanford from the joint work of W. Diffie and R. Hellman who studied public key infrastructures, and R. Merkle who studied secret key distribution. Here is the way Diffie and Hellman presented the problem in [8], which mentions the joint work with Merkle: *In turn, such applications (fast computers) create a need for new types of cryptographic systems, which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature.*

Public Key Cryptography Arises

The idea was remarkably simple and elegant. Its natural properties were strikingly reflected 30 years later, when it became publicly known and verified, that J. Ellison, an engineer and cryptographer working for MI5s General Communication Head-Quarters GCHQ, had developed exactly the same concepts and schemes as Diffie, Hellman and Merkle, yet seven years earlier. The research was only declassified after the year 2000; it was a matter of academic debate, if a person working for secret services, outside the academic community should be granted credit for scientific developments. Beyond these it is in any way remarkable that the same ideas could be developed twice in a totally independent way.

Traditionally, a protected communication was established by using secret key cryptography. In a wide area communication network, in which numerous peers could communicate over large distances, the chances for establishing a common

secret key prior to communication are low, so there was demand for a procedure which would allow a pair of peers A and B—Alice and Bob, as cryptographers often used to name them—to dispose of a *shared secret key*, without any prior communication, either direct or by means of a parallel, secured channel. Only some public known data base Δ, and algorithm could be accepted as premise for achieving the purpose.

The concept of *public key cryptography*, introduced by the three authors mentioned above, is simply described by the following: If X is a peer who wants to engage in secure network communication, he should start by generating a set of data, which is bundled into his own *secret key* S_X. A subset of this data, bundled in the *public key* P_X will be made public to all peers he might be wishing to communicate with—it will be, for instance, part of the data base Δ, or it may be transmitted over any unsecured channel. The two keys should enjoy the following two properties:

1. Both keys can be used for encrypting texts according to some algorithm yet to be defined.
2. Messages encrypted by S_X can be decrypted by P_X and vice versa. Moreover, the keys should be sufficiently random: the chances for two peers generating accidentally the same secret key should be close to zero.
3. It should be computationally unfeasible to derive S_X from P_X.

For ascertaining the third condition, one usually derives the public key P_X from the secret one, by using some kind of *trap-door* function f. Under this term, one understands an invertible function, such that the value of f is very easy to compute, but the inverse is computable in theory, but infeasible in practice, provided the data is sufficiently large. A typical such example is the map $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which associates two primes p, q to their product $n = p \cdot q$. This can be computed very efficiently even for quite large primes. However, the inverse problem, of factoring n is assumed. There is no proof for the fact that there cannot exist some fast—e.g. *polynomial* algorithm, thus one whose run-time is a polynomial in the number $m = \log_2(n)$ of bits of the input number n—for factoring integers. However, the problem is one of the most intensively researched ones in algorithmic number theory; after decades of collective work, the most efficient algorithm for factoring, the Number Field Sieve (NFS) requires the order of

$$e^{cm^{1/3}}$$

binary operations, to be *very hard*.

On the basis of the premises 1.–3., if Alice and Bob want to communicate, then Alice sends to Bob messages encrypted by P_B, which she may retrieve from the public key repository. However, only Bob can decrypt the message, so the communication is secured. Based on this idea, a further useful application emerged: it is often useful to be able to certify the ownership of some message, to *sign* the message in a unique and non-repudiable way. In this case, secrecy is less of a concern than ownership is. The solution consists of associating a short cryptographic

hash-value H to the message, which is encrypted by the secret key S_A. Any receiver will then be able to regenerate the hash value on his own, decrypt the encrypted hash with P_A, and then compare the two results. If they match, Bob has a proof that it was Alice who sent the message.

Within the next 20 years the public key cryptography and the academic paradigm of cryptology spread out and reached probably even most of the banks and diplomatic transmissions, which traditionally used to consider the use of private algorithm as a particularly welcome increase of security.[2]

Classical Public Key Cryptosystems

In the next two years after the abstract definition of public key cryptography, two major algorithms that implement this idea and are still in use today, were invented.[3]

The first one was using the *discrete logarithm* problem in the multiplicative group of finite fields as a trap door. If p is some large prime and $g \in \mathbb{F}_p^\times$ generates the multiplicative group modulo p and $a = g^c \in \mathbb{F}_p^\times$, then it is easy to compute

$$f(x) = b = a^x, \text{ for arbitrary } x.$$

However, to recover x from b, the Discrete Logarithm Problem in finite fields, is a computationally hard problem—thus adequate for a trap door function. For the factoring problem, there is no proof that no faster algorithms can be found—however, the best one discovered until today has a comparable asymptotic complexity to the number field sieve for factoring integers, mentioned above.

Diffie and Hellman proposed an algorithm for exchange of a shared secret over an insecure channel, and is widely known as the *Diffie-Hellman key exchange algorithm*. It functions as follows: If p and $g \in \mathbb{F}_p^\times$ are like before—these being public data—then Alice and Bob start by choosing some random one-time keys A_R, B_R, which are elements of $\mathbb{Z}/((q-1)\cdot\mathbb{Z})$. Then Alice sends to Bob $M_A = g^{A_R}$ and receives from Bob $M_B = g^{B_R}$. The reader can verify that by using the private data and the data received, both Alice and Bob may retrieve $S = g^{A_R \cdot B_R}$, which is the data from which the common secret key is extracted. However an eavesdropper, who is always called Eve in cryptography, would only know g^{A_R} and g^{B_R}, but not A_R or B_R. The system can be broken by breaking the Discrete Logarithm. But, does the converse also hold? This is not known. The particular, more special

[2]This fact was reflected again in the fact that the producers of cryptographic machinery were involved in customer tailoring algorithms for this purpose. In the late nineties, manufacturers of cryptographic hardware still had only a precious few customers insisting on the "privilege" of purchasing machines which run according to some unique and "secret" algorithm.

[3]It is also noteworthy that, after J. Ellis had defined the abstract notion of public key cryptosystems, in a similar way to Diffie, Hellman and Merkle, the same algorithms were discovered in MI5 too, by C. Crook and M. Williamson, only in the reverse order.

problem in which one should retrieve $g^{xy} \in \mathbb{F}_p^\times$ from g^x, g^y has received the name DH-Problem, for obvious reasons. More recently, variants of the Diffie–Hellman key exchange have been proposed, which can be *proved* to be equivalent to the DH problem: i.e., they can be broken if and only if the DH problem is broken. The key exchange algorithm does not offer the possibility to generate signatures; however, J. L. Massey and J. K. Omura proposed in 1983, a variant based also on the discrete logarithm trap door function, which allows also public key encryption, and thus signatures.

We should mention that in general public key algorithms are much slower than secret key encryption. Therefore it is most likely that one would use them for establishing a shared secret key, after which a communication session can be encrypted with a common agreed secret key algorithm, using the established key. For this purpose the original Diffie–Hellman algorithm is sufficient. Incidentally this two-step approach to encryption is the core idea in the SSL/TLS protocol, developed between 1992–2002, and which is currently used in all confidential https communications on the Internet—for instance, when you book an electronic flight ticket, or buy a book from Amazon.

The first proper public key *encryption* algorithm was provided one year later, in 1977, by R. Rivest, A. Shamir, and L. Adleman at MIT. Their algorithm, widely known as RSA after the initials of their names, uses the problem of factoring integers as a trap door. A secret key consists of $S_A = \{p, q, d\}$, where p, q are two large primes satisfying some additional randomness conditions and

$$0 < d < (p-1)(q-1), \quad \text{with} \quad (d, pq(p-1)(q-1)) = 1$$

is a random number; if $e \in \mathbb{N}$ is such that

$$ed \equiv 1 \bmod (p-1)(q-1),$$

the public key consists only of $P_A = (n, e)$, with $n = p \cdot q$. In some instances, e is a fixed number for the whole system, so d will be determined by the holder of the secret key using the same defining congruence. With these prerequisites, if M is a short message it will be identified with a number in $\mathbb{Z}/(n \cdot \mathbb{Z})$ and its public key encryption $M_e \equiv M^e \bmod n$ can be computed in the open, but can only be decrypted by Alice, the holder of d, since

$$M \equiv M_e^d = M^{ed} \bmod n.$$

Conversely, if Alice encrypts M with d, then anyone can recover M and upon doing so will have a proof Alice having produced the encryption: indeed, only the owner of the secret key could produce this encryption, which can thus act as a private signature of Alice.

Despite initial attempts of the NSA to inhibit the publicizing of the ideas of public key encryption and RSA, these were brought to the public already in 1977 by Martin Gardner in his widely read column “Mathematical Games” in the Scientific

American magazine and were eventually published in the communications of the ACM [31]: the way to public key cryptography was open!

In the same year 1978, R. McEliece proposed a somewhat different cryptosystem, which was inspired from coding theory. The trap door function is drawn in this case from general linear codes, a context in which the parameters of a linear code are specially adapted to the purpose of public key cryptography. The resulting algorithm has an advantage compared to the number theory-based algorithm mentioned above and some further, based on elliptic curves, that we shall discuss below, since it is faster. However, the keys may be as large as $1MB$, which compares poorly to the $128B$ required by RSA for a comparable level of security.[4]

Cryptanalysis

In 1978, Hellman and Merkle invented a public key cryptosystem that did not rely on number theory, but rather on the NP-complete *knapsack* problem.

The first major success of public key cryptography was that the expectation became true, and the domain of cryptanalysis—concerned with the analysis of possible attacks against cryptographic schemes—became a flourishing academic domain of investigation. One of the most spectacular successes was due to the development of the *lattice reduction* algorithm by A. Lenstra, H. Lenstra Jr., and L. Lòvasz, the LLL-algorithm. Given a lattice $\mathscr{L} \subset \mathbb{Z}^n$, there exists a base consisting of the shortest vectors. Classical algorithms for finding such a base are known from the work of Charles Hermite. Only, in the case when the base is presented by an initial generating system of very large vectors, the process is exponential. The algorithm was developed from techniques used by Lòvasz in integer programming; the idea was to use an approximate Gram-Schmidt-orthogonalization which provides some *close* to minimal vectors in $\mathscr{L}$. The advantage is that the algorithm runs in polynomial time and has therefore a wide variety of applications both in cryptography and in number theory itself. One of the first applications of LLL was in showing that the keys of the knapsack cryptosystem could be cracked in polynomial time: in order to do so, one had only to solve a particularly simple *subfamily* of problems belonging to the knapsack family. This result showed the advantage of public academic scrutiny of cryptographic schemes, since it had only taken five years to reveal the weaknesses of one of them. But it also blocked the way for applications of the knapsack. Some improved versions have been presented, that could never be attacked—but they never made it to public applications.

The most important effect of cryptanalysis was less visible. The community quickly developed its own language and defined a variety of subtle *attack scenarios*,

[4] One compares the security of two fundamentally different algorithms, by estimating the parameter sets required, such that breaking the given algorithms by means of the best state-of-the-art algorithm would require comparably large amounts of time.

in which the eavesdropper *Eve* was offered increasing levels of advantages: thus Eve can simply tap a wire communication, but she might also collect large amounts of data signed by Alice, or even induce her into signing a chosen suite of messages. Thus, possible attacks could be investigated for these various levels of disclosure. The procedure is very fruitful, since the algorithms to which no attack is found, even under the most generous premises for Eve, is for good reasons assumed to offer reliable security.

Later, the encryption hardware began being regarded as a point of attack, as it was observed that physical measurements on a chip while it is computing an RSA encryption, for instance, may reveal some bits of the secret key. Additional measures were then developed to protect from these *side channel attacks*. This way, well-defined attack scenarios are used for checking the security of various cryptosystems and protocols.

The development of cryptography is triggered by the two opposite demands, for efficiency and for security. It occurred more than once, that the wish for efficiency led to the use of some extreme key configurations. These provided particularly efficient arithmetic, thus effective computation of the cryptographic scheme. However, as in the case of the knapsack problem, the question could have been asked, if by restricting to particular families of the key space, one did not move into a particular instance of the general, hard problem, to which the trap door function was associated. The question was first answered by the observation that no algorithms are currently known that could take advantage of the particular family of keys used. But eventually, an attack was discovered, which discarded the use of certain keys, or even whole cryptographic schemes. As an example, it is for instance, useful to have a universal, short public exponent e for the RSA scheme. This had been used in practice in the late 1980s. But M. Hagstad and then D. Coppersmith showed that if e is too small, it is easy to gather sufficiently many messages signed by the same key S_A, and then use simple arithmetic in order to crack that key. Therefore, the smallest fixed key currently allowed by standards is $e = 2^{16} + 1$, and this may change with the growth of computing and storage capacities.

We have already discussed the fact that for the number theoretical public key systems introduced so far, an efficient attack of the underlying number theoretic problem (factoring or discrete logarithm) breaks the schemes. Conversely however, it is not known if general attacks can be found that break the scheme *without* offering an efficient general solution for the inversion of the trap door function. Such questions about *provable* security became actual in the late nineties. We have already mentioned that by modelling the DH-problem, which is a particular form of the discrete logarithm, the best results available in this direction were obtained by U. Maurer [23] and V. Shoup, and various coauthors, e.g. in [6].

Dickman's Theorem and the Trap Door Functions

In the thirties of the last century, J. Dickson considered the question of estimating the largest prime factors of some random integer n. Using heuristic estimates on the repartition of primes, he found for instance that if $p|n$ is the largest prime dividing n, then $p = O(n^{\ln 2})$. More generally, an integer $n > 1$ is defined to be y-smooth if none of its prime factors exceeds y. The function

$$\psi(x, y) = \sharp\{\, 1 \leq n \leq x \;:\; n \text{ is } y\text{-smooth}\,\}$$

counts the smooth numbers less than x. With these definitions, Dickman also proved that for all $u > 0$ there is a real number $\rho(u)$ such that

$$\psi(x, x^{1/u}) \sim \rho(u)x.$$

The function $\rho(u)$ was described in terms of a differential equation, in which u was fixed for $x \to \infty$.

Half a century later, the gap was filled by Canfield, Erdős and Pomerance [5], who proved that

Theorem 1 (Canfield, Erdős and Pomerance). *For all real sequences with $u \to \infty$ under the constraint $u < (1 - \epsilon) \ln x / \ln \ln x$, one has*

$$\psi(x, x^{1/u}) = xu^{-u+o(u)} \tag{1}$$

As a consequence one concludes that with probability $P > 1/2$ one out of

$$L_n[1/2, 1] := e^{\sqrt{\log(n) \log\log(n)}}$$

random integers belonging to the interval $(0, n)$ will be y-smooth, for $y = O(L[1/2, 1])$. Bounds of the type

$$L_n[c, d] = e^{d \log(n)^c \log\log(n)^{1-c}},\ 0 < c < 1$$

are called *subexponential* for obvious reasons: they grow much faster than any polynomial in $m = \log(n)$ but substantially slower than e^m. All the state-of-the-art, subexponential algorithms for solving either a variant of the discrete logarithm problem, or for factoring integers, take advantage in some way of this consequence or variants thereof.

We exemplify here the ideas on the instance of the *quadratic sieve method*, which is a classical fast algorithm for factoring integers. It has its origin in the following simple observation of Fermat: if m is a composite integer, then the congruence

$$x^2 \equiv c \bmod m$$

will have at least four solutions, and there are x, y such that $x \not\equiv \pm y \bmod m$, but $x^2 \equiv y^2 \bmod m$. Then $(x + y, m)$ is a non-trivial factor of m. Theorem 1 helps find such pairs x, y, as follows: for numbers $x(i)$ in some interval $\lceil \sqrt{n} \rceil + i, 0 \leq i \leq B$, one computes the remainder[5]

$$r(i) = x(i)^2 \text{ rem } m$$

and retains only those values of x, for which r is a B-smooth number. After gathering sufficiently many such relations, one may hope that the product of some $r(i)$ is a square: namely, that there is an index subset $J \subset [0, B]$ such that

$$\prod_{i \in J} r(i) = R^2, \ R \in \mathbb{Z}.$$

Letting then

$$X = \prod_{i \in J} x(i),$$

we obtain the congruence

$$X^2 \equiv R^2 \bmod m.$$

If in addition, $X \not\equiv \pm R \bmod m$, which should happen with probability $\geq 1/2$, then $(X \pm R, m)$ is a non-trivial factor of m. The method relies on some empirical assumptions on the repartition of factors of $r(i)$: namely, that the distribution of these residues is such that one may apply the relation (1) for estimating the probability that one of these numbers is B-smooth. These allow to establish an *optimal* bound

$$B \sim \exp(\sqrt{\log(m) \log\log(m)}) = L_n[1/2, 1].$$

In our case $B = L(n; 1/2)$ and the quadratic sieve runs in time polynomial in B-experience having so far confirmed the underlying heuristical assumptions.

The following nice example is taken from the book of R. Crandall and C. Pomerance [7]: let $m = 1649$, with $41 = \lceil \sqrt{m} \rceil$. We find

$$41^2 \equiv 32 \bmod m; \quad 42^2 \equiv 115 \bmod m; \quad 43^2 \equiv 200 \bmod m.$$

Since $32 \cdot 200 = 2^{5+3} \cdot 5^2 = 80^2$, we let $R = 80$ and

$$X = 41 \cdot 43 = 42^2 - 1 \equiv 114 \bmod m,$$

finding that $114^2 \equiv 80^2 \bmod m$ and eventually $17 = (114 - 80, 1649)$, which is a non-trivial factor.

[5] In computational algebra, the notation x rem y stands for the unique representative of the equivalence class of x mod y which lays in the interval $[0, y)$.

For the discrete logarithm problem in $\mathbb{F}_p^\times$, which consists of determining x such that $g^x \equiv b \bmod p$, one uses smooth numbers as follows: Fix a smoothness bound y and let $q_1, \ldots, q_r < y$ be all the primes up to y. For random values of m, one computes $u = g^m \text{ rem } p$ and keeps only those values of u which are y-smooth. After collecting sufficiently many relations, one will then be able to compute the discrete logarithms l_i such that $q_i \equiv g^{l_i} \bmod p$. Next, one tries random values of k searching for ones that make $v = bg^{-k} \text{ rem } p$ be a y-smooth number. The precomputed values l_i will then help determine $x = k + \log_p(v)$ from the prime decomposition of v. This algorithm also relies on heuristic assumptions, on the basis of which the running time is $L_p[1/2, \sqrt{2}]$.

At the end of the 1980s, John Pollard found a way for applying the idea of the quadratic sieve to integers in number fields rather than $\mathbb{Q}$. The method was first applied to the factorization of the Fermat number $F_9 = 2^{2^9} + 1$. In the following years, it was generalized and improved by a series of mathematicians, starting with A. Lenstra and M. Manasse. The resulting *number field sieve* is currently the asymptotically fastest factoring method and it runs in time $O(L_n[1/3, c])$, for some constant $c < 2$.

Similar methods are known for the discrete logarithm method: they use number fields in case of larger characteristics, and function fields for small characteristics. Like in the case of factoring, their running time is also $O(L_n[1/3, c])$. Current records reach as high as 7–800 binary digits for factoring composite of general form and $\sim$ 5–600 for the discrete logarithm in prime fields. During more than one decade, the discrete logarithm was hardest in finite fields $\mathbb{F}_{p^\ell}$ for which $\ell \sim \log(p)$: these orders of magnitude could not be attacked by either number or function field sieves.

Recently A. Joux from INRIA Nancy developed a series of new ideas for improving discrete logarithms in finite non-prime extensions. There are several versions and applications of these ideas. First, they succeed in filling in the gap that existed between the function field and the number field sieve, by providing algorithms in the order of $L_p[1/3, c]$, and also for the case of extension fields with $\ell \sim \log(p)$. They allow to solve the discrete logarithm problem in *quasi-polynomial* for field $\mathbb{F}_{p^\ell}$ when $\ell \sim p$; the result has been presented at Eurocrypt 2013 and is published in [1]. The ideas find another application in discrete logarithm in the fields of characteristic two extension degree $\mathbb{F}_{2^{q \cdot k}}$ with q a prime and k an integer related to q. Joux also announced a variant of his method to yield an algorithm for discrete logarithms in general fields of characteristic two, running in $L_q[1/4, c]$, where q is the size of the field [18]. This would be the first known algorithm of this efficiency. The developments in this field are still quite fluid, but certainly within the following months to a few years, some important and efficient versions of discrete logarithm algorithms in a variety of fields will be well described and understood.

Elliptic Curves

The cryptographic schemes discussed so far use multiplicative groups $(\mathbb{Z}/n\cdot\mathbb{Z})^*$ or $\mathbb{F}_q^\times$ and related trap door function. Having (computational) access to a larger family of well-understood abelian groups would certainly enlarge the possibilities for cryptographic and algorithmic applications.

In 1984 René Schoof made the way by opening the discovery of a polynomial time algorithm for counting the number of points on an elliptic curve over a finite field. This brought the groups of algebraic geometry in the realm of applications and algorithms. Within one year, H. W. Lenstra Jr. proposed an important variant of Pollard's rho-method for factoring, based on elliptic curves: the *elliptic curve method* or ECM. Also, V. Miller and N. Koblitz proposed independently the use of elliptic curves for cryptography. The ECM method has a run-time comparable to the quadratic sieve, but it behaves particularly well for numbers m which have some small prime factors, i.e. sensibly smaller than $\sqrt{m}$: the run time is namely estimated to be $L_p[1/2,\sqrt{2}]$, where p is the smallest prime dividing m.

We recall that an ordinary elliptic curve over a finite field $\mathbb{F}_q = \mathbb{F}_{p^\ell}$ of characteristic $p > 3$ is the set of solutions

$$E_q(a,b) := \{P = (X,Y) \,:\, Y^2 = X^3 + aX + b, X, Y \in \mathbb{F}_q\} \subset \mathbb{F}_q^2\,.$$

There is an abelian addition $\oplus$ defined on this curve, which has the *point at infinity* $\mathscr{O}$ as neutral element. The neutral element can be understood as arising when the addition law, which is based on rational functions, leads to a division by zero. The formally correct definition is obtained by embedding the curve in a projective space. The curve is ordinary, if it is not singular and not supersingular, two conditions that can be verified in terms of $q, b, \mathbb{F}_q$. Thus

$$\mathscr{E}_q(a,b) = (E_q(a,b), \oplus)$$

becomes an abelian group. The classical theorem of Hasse gives the following bounds for the size of this finite group:

$$|\mathscr{E}_q(a,b) - q + 1)| < 2\sqrt{q}. \tag{2}$$

An elliptic curve can be defined in a similar way over the algebraic closure $\overline{\mathbb{F}}_q$. Its N-torsion is

$$\mathscr{E}[N] = \{P \in \mathscr{E} \,:\, [N]P = \mathscr{O}\},$$

where $[N]P$ denotes the N-fold addition of P to itself. The torsion subgroup is—with one exception—a two-dimensional free $\mathbb{Z}/(N\cdot\mathbb{Z})$-module, and a vector space, for prime N. If $\zeta \in \overline{\mathbb{F}}_q$ is a primitive $N-$th root of unity, there is a non-degenerate bilinear, skew symmetric pairing:

$$\langle\cdot,\cdot\rangle : \mathscr{E}[N] \times \mathscr{E}[N] \to \langle\zeta\rangle, \tag{3}$$

the Weil pairing. In particular, if P, Q are linear independent torsion points, then

$$\langle P, [x]Q\rangle = (\langle P, Q\rangle)^x\,, \tag{4}$$

an identity in the multiplicative group $(\mathbb{F}_q[\zeta])^\times$.

The idea of the ECM factoring method of Lenstra adapts an older algorithm of Pollard, which was designed to work in multiplicative groups, to the larger family of elliptic curves. It can be described briefly as follows: if n is a number to be factored, one draws random numbers a, b such that a point $P = (X, Y)$ is known with

$$Y^2 = X^3 + aX + b, 0 \leq X, Y < n.$$

Assume now that n has a prime divisor p such that $m := |E_p(a, b)|$ is a B-smooth integer for some fixed, not too large integer B. If $K = B!$, then in the process of computing the multiple $[K]P$ by additions and doublings on the curve modulo n, one will *most probably* encounter a factorization of n: some denominator will be divisible by p (point at infinity!), but not by all the primes dividing n. Lenstra proved that for uniform randomly distributed a, b, the numbers m are close to being uniformly distributed in the Hasse interval (2). Theorem 1 then implies that by choosing $B = L_p[1/2, 1]$ random curves, one will find a curve for which m is B-smooth with probability $> 1/2$. This explains the main steps of the algorithm and of its proof. The interested reader may use Silverman's [34] and Washington's [35] textbooks for a detailed rigorous introduction to elliptic curves and their applications to cryptography.

Counting Points

The idea of Schoof is both elegant and important, beyond even the immediate algorithmic and cryptographic applications: it opened a new area of research for practical algorithms for counting points on finite abelian varieties. This research area is still growing, while the main domain of application goes beyond the limits of cryptography, since at least a decade. The algorithms are more and more used for larger computations related to mathematical questions such as the Birch Swinnerton-Dyer conjecture, and other properties of L-series. See also [30] for an elementary theoretical application of point counting.

Initially, Schoof [32] started from the following simple remark: if

$$E_p(a, b)\ :\ Y^2 = X^3 + aX + b$$

is an elliptic curve defined over the finite field $\mathbb{F}_p$, of which one assumes that it is ordinary, then Riemann's conjecture for elliptic curves implies that, in the endomorphism ring of the curve $\mathrm{End}(E_p, \overline{\mathbb{F}}_p)$ defined over the algebraic closure of $\mathbb{F}_p$, the Frobenius verifies the quadratic equation

$$\Phi^2 - t\Phi + p = 0. \tag{5}$$

Since E_p is fixed by Φ, we have

$$|E_p(a,b)| = p - t + 1$$

for the number of points fixed by the Frobenius. Counting the points is thus equivalent to determining the value of the *trace of the Frobenius* t; since the Hasse inequality (2) states that

$$t < 2\sqrt{p},$$

it suffices to determine the remainder $t \mod \ell$ for a set of small primes with:

$$L = \prod \ell > 2\sqrt{p}.$$

Therefore, the core step of the algorithm consists in modeling the ℓ-torsion $E_p[\ell]$ into an algebra

$$\mathbb{B} = \mathbb{F}_p[X,Y]/\left(\psi_\ell(X),\ Y^2 - (X^3 + aX + b)\right),$$
$$P = \left(X + (\psi_\ell(X)),\ Y + (Y^2 - (X^3 + aX + b))\right) \in \mathbb{B}.$$

in which $\psi_\ell(X)$ is the ℓ-division polynomial which has as roots all the x-coordinates of ℓ-division points. Therefore, any such point enjoys the properties which define the *generic* ℓ-torsion point $P \in \mathbb{B}$. It is then a straightforward computation, to determine $t \mod \ell$ from the identity

$$\Phi^2 P + pP = t\Phi P.$$

The seminal idea of Schoof, to determine the parameters of the Riemann ζ-function from projections in torsion spaces, and thus counting points on varieties over finite fields was both improved for simple varieties, such as elliptic curves, and extended to more general abelian varieties. In the first case, the primary thing to do was to reduce the size of the algebra $\mathbb{B}$—which can be done by finding smaller factors of $\psi_\ell(X) \mod p$.

The breakthrough in this direction was indicated by Noam Elkies (cf. [9, 33]), who brought modular forms in the game, thus showing how to find in half of the cases some factors $f(X)|\psi_\ell(X)$ of linear degree, compared to the quadratic degree in ℓ of the division polynomial. The ℓ-torsion $E_p[\ell] \cong \mathbb{F}_\ell^2$ as a vector space; fixing two linear independent points $P, Q \in E_p[\ell]$, we see that $G := \mathrm{Gal}\,(\mathbb{B}/\mathbb{F}_p)$ acts on the vector space $E_p[\ell]$ by acting on the base P, Q. We obtain herewith a representation $\rho : G \to \mathrm{GL}_2(\mathbb{F}_\ell)$, with respect to which $\rho(\Phi)$ verifies the same quadratic equation. Let δ be the discriminant of the quadratic polynomial in (5), which is the same as the characteristic polynomial of the image of $\rho(\Phi) \in \mathrm{GL}_2(\mathbb{F}_\ell)$. Then, according to the value of the Legendre symbol $\left(\frac{\delta}{\ell}\right) \in \{1, 0, -1\}$, the matrix $\rho(\Phi)$ is diagonalizable, has normal upper triangular form or has eigenvalues in $\mathbb{F}_{\ell^2}$.

In the first case, there are two *eigenpoints* P, Q of the Frobenius and the orbit of their x coordinates under multiplication on the curve is galois invariant. We obtain herewith the *eigenpolynomials*

$$f_P(X) = \prod_{k=1}^{(\ell-1)/2} (X - ([k]P)_x) \mid \psi_\ell(X), \quad \text{where}$$

$$\deg(f_P) = (\ell - 1)/2, \quad \text{and} \quad \deg(\psi_\ell) = (\ell^2 - 1)/2,$$

together with a new algebra $\mathbb{B}'$, obtained by replacing ψ_ℓ with f_P. For the computation of F_P, Elkies considered the function field $\mathbb{C}[[j(q)]]$. Some classical arguments on Eisenstein series and $\Gamma_0(\ell)$-modular forms, imply that for each j-invariant j_m of an ℓ-isogenous curve to E_p—or also, for each zero of the modular equation $\Phi_\ell(X, j(q))$—there is a polynomial $f_j(X) \in C[[j(q)]][X]$ which has the x-coordinates of the kernel of the respective isogeny as zeroes. The polynomials can be constructed in the function field by manipulations of q-expansions and they have the useful property that all the coefficients are algebraic integers. The insight of Elkies was to show that one can substitute for j_m the value of some zero $\Phi_\ell(X, j(E_p))$ mod p and reduce the coefficients of $f_j(X)$ modulo p, thus obtaining some eigenpolynomial corresponding to the value of j_m. Indeed, if $\mathbf{E}$ is any curve over $\overline{\mathbb{Q}}$ which reduces to E_p at some prime ideal above p, then its j-invariant reduces to the one of E_p and so do the invariants of its ℓ-isogenies. Therefore, if the modular equation has linear factors j_m over $\mathbb{F}_p$, by inserting these in the expression for $f_j(X)$, upon reduction at the same prime, the coefficients of the polynomial f_j map to the ones of some eigenpolynomial. Using improved algorithms for manipulation of series [4], one can compute the eigenpolynomials in time $O(\log^3(p))$, the running time being dominated by the computation of zeroes of $\Phi_\ell(X, j(E_p))$ mod p. Further improvements can be achieved by using the galois structure of the resulting algebras [25]. The galois theory of finite, commutative algebras has wider applications in algorithmic context and was generalized in [27].

For curves defined over finite fields of small characteristic p, it is possible to project (5) in the p^N-torsion group. Using different flavors of cohomology combined with Newton iterations, various authors starting with T. Satoh, K. Kedlaya and A. Lauder developed in this way, the most efficient point counting algorithms for elliptic curves. Some of them are generalized to super elliptic curves, elliptic surfaces, etc. However, this approach works best only for very small characteristics.

Cryptography

The elliptic curve-based cryptographic schemes which have survived scrutiny and became part of current standards on public key cryptography are essentially variants

of the Diffie-Hellman key exchange scheme and are based on the difficulty of solving the discrete logarithm problem: find x such that

$$[x]P = Q, \quad \text{for} \quad P, Q \in E_p(a, b)$$

being points on an elliptic curve, such that Q is known to generate a cyclic group of high order. Unlike in finite fields, the discrete logarithm problem on elliptic curves is not known to allow any sub-exponential time solutions. The best known methods have run time $O(\sqrt{p})$, where p is the characteristic of the (prime) field over which E_p is defined. As a consequence, one can work in much smaller groups than in the case of the multiplicative groups of finite fields, still achieving the same estimated security of a scheme, with respect to state-of-the-art attacks. This advantage led to a new wave of interest for elliptic curve cryptography in connection with the security of mobile phones.

The Weil pairing requires certain caution though. One may in principle use the identity (4) in order to reduce the discrete logarithm problem on the elliptic curve to one in the multiplicative group of the field $\mathbb{F}_r := \mathbb{F}_q[\zeta]$. Since discrete logarithms in multiplicative groups allow for subexponential algorithms, being thus much more efficient, the size of this extension $\mathbb{F}_r$ plays an important role and the reduction might cause problems when $\mathbb{F}_r$ is not too large. The use of the Weil pairing for the discrete logarithm on *supersingular* elliptic curves was pointed out for the first time by Gerhard Frey. The problem came to light when Frey was asked to estimate a software using these curves—on which a particularly efficient implementation of the group laws is possible—for its security. He showed that for these specific curves, the Weil pairing reduced the elliptic curve logarithm problem to one in finite fields of critically small size—$\mathbb{F}_r = \mathbb{F}_{q^k}$ for $k \in \{2, 3, 6\}$, thus leading to serious security problems. The idea was taken over by A. J. Menezes, P. C. van Oorschot and S. A. Vanstone and is currently known in the literature under the name of *MOV attack*. The attack is in general inefficient, but discarded the use of supersingular curves for cryptographic purposes, for the reasons mentioned above. Interestingly, more than a decade later, due to the increasing demand for efficient cryptography using short bandwidth, in application to securing cell phone communications, the supersingular curves found a revival. Recently, some research is invested in finding good combinations of finite fields and supersingular curves, such that on the one hand time savings can be made in the arithmetic, and on the other hand the field $\mathbb{F}_r = \mathbb{F}_{q^6}$ is intractable for the number field sieve discrete logarithm. This example shows that there still is a certain volatility about development of practical cryptographic system, which however overlaps the reliable overall results of cryptanalysis.

A further example where efficiency is sought at the critical border line of the MOV attacks are the so-called Koblitz curves, defined over fields $\mathbb{K} = F_{p^\ell}$ of small characteristic and having $a, b \in \mathbb{F}_p$. Since p is small, it is of course likely that the field $\mathbb{F}_r \supset \mathbb{K}$ required for a MOV attack is a not very large extension of $\mathbb{K}$, even when the curves are not supersingular.

In the last years, D. Boneh and A. Joux developed the idea of *identity-based* cryptography. In order to cope with increasing demand of various cryptographic keys, the idea is to provide the possibility in some limited networks for the user to have access to his secret key, essentially by means of his own identity. The most spread implementation of this idea also uses Weil pairing, and is thus called *pairing-based cryptography*. The recent developments in discrete logarithms for fields of small characteristics described above have thus an important impact, requiring significant increases in the size of the keys used.

Despite standardization, which made cryptographic developments obsolete on the Internet, there are thus reasons why research in this particular area is still very fertile. We recommend the detailed and lively survey of Heß et. al. [16]. The interested reader is referred to [20–22, 24, 26, 28] for further reading.

Key Management and Biometry

Since the security of a cryptosystem relies on its keys, it is an important task to manage these keys in a secure and efficient way. In a public-key environment, one discerns the following essentially distinct aspects:

A. *Managing secret keys*. Since these are data without meaning for humans, they should necessarily be stored on some electronic media, thus leading to the security concern that only the authorized key possessor should have access to the use of these keys.
B. *Trusting public keys*. We have seen that in the public key setting, Alice needs to use some public key of Bob. This can either be provided by Bob during the communication, or read from a common, public data base. But in both cases, since the key is obtained over the network, Alice wishes to be certain that the public key received really belongs to Bob. Otherwise, Eve might for instance provide an own key, while convincing Alice that she obtained the public key of Bob. In this way Eve would be in the position of decrypting messages that Alice had encrypted in the assumption they should only be accessible to Bob, the rightful owner of the secret key belonging to the public one that she received.

There are various solutions for solving both of the above problems. For the first, keys can be stored on some card device, that needs to be activated by some password. Alternatively, the same principle can be replicated on any variety of secure storage media, including an encrypted hard disk. Alternatively, the user may have access to secure *applications* that manage keys locally on his behalf. In this case, the activation password will be application-dependent.

For the second problem, the key idea is called *certification*. Some *trusted authority*, which has verified the physical identity of Bob matching to his pubic key, will add a signature on this public key, made with the secret key of the authority. The signature put by the trusted authority upon Bob's key is also called a *certificate*. The trusted authority's public key will be accessible in a non-forgeable way, so Alice

can verify the signature, thus gaining trust for the fact that Bob's key is genuine. In practice, in order to generate a chain of trust reaching from Bob to Alice it may sometimes be necessary to build up a chain of certificates: trusted authority T_1 certifies Bob's key, T_2 certifies the one of T_1, reaching to T_k which is the last authority the key of which is unconditionally trusted by Alice.

Public Key Infrastructure

The principle is very useful and works well in local networks belonging to an environment which has an own hierarchy of trust which can be naturally mapped to the certificate hierarchy. Such are, for instance, large enterprises, administrations and government institutions. Since auxiliary problems of secure key generation, certificate production and verification, secure storage, etc. follow from these key management problems, producing professional solutions to the key management problem of large intranets became a market and the typical software solutions are called Public Key Infrastructures (PKI), being systems that allow to implement all the above-mentioned functionalities within the intranet of some institution.

Note that in this case the fact of having a common institutional frame is a major help, since it allows to distribute the trust according to well-defined rules that belong to the institution and are very likely to exist independently of the cryptographic setting. It is, however, not always the case that secured communication needs to be established within a closed intranet. In that case, although numerous major companies offer the facility of key generation and distribution, thus offering themselves like some kind of trusted authority for the customer, the level of trust that can be offered to such commercial solutions is rather low and would not suffice for offering reliable confidentiality.

The Open System Approach

An alternative idea was invented by Paul Zimmerman, who has developed a public domain software for secure mail exchange, called Pretty Good Cryptography, and which is meanwhile available also as professional software. Zimmerman's idea of trust in an open network is strikingly simple: it is likely to assume that the communicating peers—Alice and Bob—can agree upon some commonly trusted instance, say Tim. In that case Bob can either already hold a certificate signed by Tim, or one signed by a person that holds a certificate signed by Tim, and so on. If this chain of verifications breaks up, then Bob will be able to provide Alice with a set of certificates that convince her that Tim *indirectly* trusts Bob. Otherwise, Bob will have to ask Tim for a certificate which shall be provided a posteriori. In this way the ring of certificates of each peer grows dynamically, by request and need. While the trust system here is perfectly non-hierarchical and symmetric—now peer has an

unconditional level of trust, some other problems must be taken into consideration. For instance, the fact that the trust chain can be quite unreliable, especially when growing too long. Instance A may trust B within a certain frame, and B may trust C, but at the end A might not have a sufficient level of trust in C at all, and would not have signed a certificate if directly asked for one.

These elementary concerns have not been mentioned here with the aim of an exhaustive discussion, but rather in order to raise the awareness about the multiple facets of the problem of secure key management, while indicating the most important approaches for a solution, with their known advantages and disadvantages.

Passwords and Biometry

We have mentioned that in the case of problem A above, Alice may end up having a multitude of secret keys distributed through various applications she may work with on a permanent basis. And the access to her secret keys will be granted by some password, that should sufficiently identify her. This and other contexts in which access is granted based on passwords leads to new issues. First, in order to grant the password with sufficient security, there should exist both a minimal dynamics—requiring periodical password changes—as a sufficient randomness in the passwords themselves, which is seldom granted when using passwords that can be memorized by humans. Add to this the expectation that the password of the same peer, for different applications or environments should differ—so that the compromising of one password does not put in danger the whole range of domains accessed by Alice. We see that the access control by means of passwords poses problems itself.

The identification of persons by means of their physical body or dynamics—called *biometric recognition*—is a specialty that grew from forensic needs developing itself in the computer era into a self-contained branch of computer science at the intersection of image processing, pattern recognition and security. Whether the biometry of concern is provided by fingerprints, iris or face traits, voice or writing patterns, biometric recognition has always the following specific characteristics:

a. Identification is a *stochastic process* and not a deterministic one, as for instance in the case of a password verification by means of some one-way function. Since the biometrics of a person are sampled at two distinct places in time and space, they will not be identical. Due to this and a series of additional factors of incertitude introduced by the physical and computer-processing, identification will always be subject to error. The standard way to measure these average errors is by overlapping the two possible error sources: *false accept*, when another person is falsely accepted for Alice and *false reject* when Alice's identity is not accepted on the basis of her biometry and she is rejected. The equal error rate (EER) is the optimal performance of a system in which the two error rates are identical.

b. Unless the data caption system has a reliable method for distinguishing live, natural biometrics from artificial counterfacts, impersonation attacks are possible.
c. Biometrics are unique, so a biometric trait once compromised for a certain type of application, is irreplaceable and ulterior use of that biometrics has lost its security.

Despite these quite restrictive conditions of use, biometric identification has the important advantage of commodity: it can make the necessity of multiple, dynamic passwords obsolete. As a consequence, biometry is already in use for access control applications of low security sensitiveness: access to lounges, clubs, hotel rooms or as replacement for visitor's cards. It can also replace the login password for personal computers. When it comes to security applications, neither the potential uses nor the attacks are so well delimited and classified as is the case in cryptography. Consequently the security claims one encounter in the vast literature of the field does not offer the reliability expected from the context of cryptography. One should therefore recall as a rule of thumb the fact that the probability of a successful attack against a biometric system is quite well approximated by the EER of the system. Since EER of one in a million are seldom—being claimed for some systems using iris recognition, it can be seen that biometric identification is practical and comfortable, but yet not acceptable in conjunction with cryptographic applications. The use of multiple biometrics—including multifinger recognition—is therefore an area of active research, in which one of the subtle issues to consider is the fact that it should not be possible to uncouple the individual biometrics.

Quantum Technology and Other Cryptosystems

The main intensively used public key cryptography methods rely on the number theoretic problems described above. There have been numerous interesting attempts to use the large list of NP complete problems in order to derive some trap door function—the knapsack problem is only one of the most famous ones. We can hardly go into the detail necessary in order to pay justice both to the interest of the attempts and the reasons for their failure or restricted use.

Before discussing below several alternative cryptosystems which survived the scrutiny of cryptananlyst and are still discussed as possible alternatives, we turn here our attention to the contribution of physics.

The Advent of Quantum Theory

Since the early 1980s the Canadian mathematicians G. Brassard and C. Crépeau suggested the use of quantum effects for security applications: the simple idea

was that Eve could not tap a quantum communication wire, without destroying the information content transmitted, so security would be provided by a *self-destruction mechanism* introduced by quantum mechanics in the confidential information transmitted. The physical and cryptographical aspects of the idea have been in active research ever since. Unlike the mathematical systems already described, or also others that follow, which can be conceived and analyzed on paper, after which their practical realization reduces to quite a simple task of programming, the difficulties encountered in this case were and remain of physical nature. In the first decade of this century, several practical implementations of quantum[6] cryptography have been announced, reaching over distances of up to 100 km. It is thus the distance and the stability of quantum transmission via fiber-optics which is the bottleneck for this system.

In the nineties of the last century, various ingenious experiments and ideas for alternative computing infrastructures were imagined or even tested. One may mention along these lines, L. Adleman's—one of the inventors of RSA—experiments for computing with bacteria.[7] Perhaps the most persisting future projection in this context is the concept of *quantum computing*; in this case there is a physical idea behind, which is stable enough in order to lead to formal mathematical models of computations that might be performed on quantum computers; one can use for a start the short introduction given in [10]. Using existing models of quantum computers, mathematicians since more than a decade have been developing algorithms that run according to the given model. It is, for instance, known that quantum computers *can* invert all the trap door functions used in the cryptographic schemes described above, in polynomial time. Developing models for quantum computing is an ongoing area of intensive research activity in which some of the most eminent theoretical mathematicians and physicists find appealing questions. For instance, the Fields medalist Michael H. Freedman leads the Q-Section of Microsoft where he applies topological methods to quantum computation (cf. [3, 11–15]).

The quantum computers information unit is a *qubit*; unlike a bit, a qubit can, simply speaking, carry any superposition of the states 0 and 1. The calculation on a quantum computer with n qubits ends with measurement of all the states, collapsing each qubit into one of the two pure states. It is the fact that computations happen in a state of superposition of all quantum states which leads to the distinct superior capacities of quantum computers. Somehow similar to the case of quantum cryptography, there is a major physical problem in the realization of quantum computers, and that is realizing *stable* qubits, stability being with respect to the influence of the environment and in particular other qubits. There are persistent announcements of small progress in the technology of quantum computing, keeping

[6]The reader should not confuse *quantum cryptography* with *quantum computing*, where quantum effects are wished to help computations, not only secure information transmission: the physical challenges are even larger in the latter case.

[7]The idea showed to be in principle feasible, but never reached more than the representation of the decimal digits on such "computers."

the hope alive that one might live the day when first experimental quantum computers carrying more than 3–4 qubits will be routinely available. For instance, in order to factor an RSA key of the currently standard length of 1024 bits, a quantum computer should have in the order of magnitude of 1024 qubits. With this prerequisite however, the number would be factored within milliseconds.

Alternative Cryptosystems

Public key cryptography is sensibly slower than secret key encryption, by a factor of roughly 1,000, as a thumb rule. This led to the wish to design some fast asymmetric schemes that do not use the kind of arithmetics that are the bottlenecks for the DH and RSA systems.

A successful solution in this respect was invented by three number theorists: J. Hoffstein, J. Pipher and J.H. Silverman [17]; they designed the cryptosystem NTRU (Number Theorists are Us), which uses arithmetic in a ring of truncated polynomials, such that decryption—the slower operation—can be done in $O(n \log(n))$ rather than $O(n^2 \log(n))$ or more operations, as is the case for RSA. Here, the constant n is roughly the key size. In the case of NTRU, this is slightly larger for comparable security; for instance, a comparable security to the one provided by RSA keys of 1024 bits may require NTRU bits of 4000 bits. This key increase is affordable, for the performance advantage gained. The security of the system is based on the problem of finding shortest vectors in large lattices. While the best methods for solving this problem continuously improve, this fact can be easily compensated for, by accordingly small increases of the key sizes. The system NTRU has been developed a lot during the last 15 years and was accepted five years ago also as an IEEE standard.

Recently, Dan Bernstein gave a new revival to McElieces algorithms, by developing a variant which is technically improved for efficiency and uses, among others, some algorithms for polynomial simultaneous evaluation and interpolation, which developed in part after the original invention of the cryptosystem. Bernstein refers to his variant as Mcbits [2] and uses the argument that unlike the number theoretical cryptosystems, this scheme is resistant to the state-of-the-art models of quantum computing. One may of course argue that the day when quantum computers become routinely available, it should be expectable that quantum encryption is available too, thus making mathematical cryptography somehow obsolete. The practical bottleneck of Mcbits in present days is the size of the keys, with ranges to several megabytes. It is otherwise an efficient algorithm which can be taken into consideration in environments in which communicating large keys is less of a bottleneck than the computation time for encryption/decryption.

A further family of interesting public key schemes uses non-commutative groups—such as for instance *braid groups*, (e.g. cf. [29]). Their developers also make a point out of the fact that the scheme is resistant to quantum computing.

Conclusion

Cryptography was born in the early ages as a skill of mental combinations put at the service of privacy and military protection. It developed over time into a highly mathematized discipline, which unites the science of concealing with the analysis of attacks into one single unit, *cryptorology*. While the last decades of research and the development of computers have offered satisfactorily wide methods for solving the elementary needs of security, it seems that the prognoses for the future are more captivated by the advent of physical solutions offered by quantum mechanics, both to the cryptanalysis of the most widely spread public key schemes but also, constructively, for the implementation of new, purely physical cryptosystems.

Acknowledgements We would like to express our thanks to Professor Joseph Silverman for his useful remarks on the manuscript.

References

1. R. Bărbulescu, P. Gaudry, A. Joux, É. Thomé, *A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic*, http://eprint.iacr.org/2013/400
2. D. J. Bernstein, Tung Chou, Peter Schwabe, *McBits: fast constant-time code-based cryptography,* CHES 2013, to appear.
3. M. Bordewich, M. H. Freedman, L. Lovász and D. Welsh, *Approximate counting and quantum computation*, Combinatorics, Probability and Computing, **14**(2005), 737–754.
4. A. Bostan, F. Morain, B. Salvy and É Schost: *Fast algorithms for computing isogenies between elliptici curves*, Math. Comp. **77** (2008), 1755–1778.
5. E. R. Canfield, P. Erdős, C. Pomerance, *On a problem of Oppenheim concerning Factorisatio Numerorum*, J. Number Theory 17 (1983) 1–28.
6. R. Cramer and V. Shoup, *Signature Schemes based on strong RSA assumptions*, Extended abstract in Proc. ACM CCS 1999.
7. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, Springer, 2004.
8. Whitfield Diffie and Martin Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory; Nov. 1976.
9. N. D. Elkies, *Elliptic and modular curves over finite fields and related computational issues*, Computational Perspectives on Number Theory: Proc. Conf. in honor of A. O. L. Atkin (D. A. Buell and J. T. Teitelbaum, eds.), AMS/International Press, 1998, 21–76.
10. *Cryptanalysis of ENIGMA in Wikipedia* : http://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma
11. M. H. Freedman, *Complexity classes as mathematical axioms*, Annals of Math., **170**(2009), 995–1002.
12. M. H. Freedman, A. Kitaev and Z. Wang, *Simulation of topological field theories by quantum computers*, Commun. Math. Phys., **227**(2002), 587–603.
13. M. H. Freedman, A. Kitaev, M. J. Larsen and Z. Wang, *Topological quantum computation*, Bull. Amer. Math. Soc., **40**(2003), 31–38.
14. M. H. Freedman, M. J. Larsen and Z. Wang, *Density representations of braid groups and distribution of values of Jones invariants*, Commun. Math. Phys. **228**(2002), 177–199.
15. M. H. Freedman, M. J. Larsen and Z. Wang, *A modular functor which is universal for quantum computation*, Commun. Math. Phys., **227**(2002), 605–622.

16. F. Heß, A. Stein, S. Stein and M. Lochter, *The Magic of Elliptic Curves and Public Key Cryptography*, Jahresbericht Deutsch Math.-Ver. **1**14 (2012), 59–88.
17. J. Hoffstein, J. Pipher and J.H. Silverman: *An Introduction to Mathematical Cryptography*, Springer (2008)
18. A. Joux: *A new index calculus algorithm with complexity* $L(1/4 + o(1))$ *in very small characteristic*, http://eprint.iacr.org/2013/095
19. D. Kahn: *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*, Scribner (1997).
20. N. Koblitz, *Elliptic curve cryptosystems*, Math. Comp., **48**(1987), 203–209.
21. N. Koblitz, *Course in Number Theory and Cryptography*, Springer-Verlag, New York, 1994.
22. H. W. Lenstra, *Factoring integers with elliptic curves*, Annals Math., **126**(3)(1987), 649–673.
23. U. Maurer: *Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms*. Advances in Cryptology - Crypto '94, Springer-Verlag, (1994), 271–281.
24. R. J. McEliece (January and February 1978), *A Public-Key Cryptosystem Based On Algebraic Coding Theory*, DSN Progress Report. 42–44: 114. Bibcode:1978DSNPR..44..114M.
25. Mihăilescu, P., Morain, F., and Schost, E.: *Computing the eigenvalue in the Schoof-Elkies-Atkin algorithm using Abelian lifts*. In ISSAC '07: Proceedings of the 2007 international symposium on Symbolic and algebraic computation (New York, NY, USA, 2007), ACM Press, pp. 285–292.
26. P. Mihăilescu and M. Th. Rassias, *Public key cryptography, number theory and applications*, Newsletter of the European Mathematical Society, **86**(2012), 25–30.
27. P. Mihăilescu and V. Vuletescu, *Elliptic Gauss sums and applications to point counting.* J. Symb. Comput. 45, **8**(2010), 825–836.
28. V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology: Proc. of Crypto '85, Lecture Notes in Computer Science, **218**(1986), Springer-Verlag, New York, pp. 417–426.
29. A. Myasnikov, V. Shpilrain and A. Ushakov, *Group-based Cryptography*, Advanced Courses in Math. CRM Barcelona, Birkhäuser Verlag (2008)
30. M. Th. Rassias, *On the representation of the number of integral points of an elliptic curve modulo a prime number*, http://arxiv.org/abs/1210.1439
31. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining signatures and public key cryptography*, Communications of the ACM, **21** (1978), 121–126.
32. R. Schoof, *Elliptic Curves over Finite Fields and Computation of Square Roots mod* p, Math. Comp. **4**3(1985), 483–494.
33. R. Schoof, *Counting Point on Elliptic Curves over Finite Fields*, Journal de Th. des Nombres Bordeaux,
34. J. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics **106**, Springer-Verlag, New York, 1986.
35. L. C. Washington, *Elliptic Curves-Number Theory and Cryptography*, CRC Press, London, New York, 2008.