SIAM Review
Vol. 46, No. 4, pp. 599–634

© 2004 Society for Industrial and Applied Mathematics

A Survey of Public-Key Cryptosystems*

Neal Koblitz[†]
Alfred J. Menezes[‡]

Abstract. We give an overview of the most important public-key cryptosystems and discuss the difficult task of evaluating the merit of such systems.

Key words. cryptography, public key, elliptic curve

AMS subject classifications. 94A60, 11T71, 14G50, 68P25

DOI. 10.1137/S0036144503439190

1. Introduction.

Before the invention of public-key cryptography, a 1968 book about time-sharing systems [124] first hinted at the possibility of a new type of cryptography. The author described a new *one-way cipher* used by R. M. Needham in order to enable a computer to verify passwords without storing information that an intruder could use to impersonate a legitimate user.

> In Needham's system, when the user first sets his password, or whenever he changes it, it is immediately subjected to the enciphering process, and it is the enciphered form that is stored in the computer. Whenever the password is typed in response to a demand from the supervisor for the user's identity to be established, it is again enciphered and the result compared with the stored version. It would be of no immediate use to a would-be malefactor to obtain a copy of the list of enciphered passwords, since he would have to decipher them before he could use them. For this purpose, he would need access to a computer and even if full details of the enciphering algorithm were available, the deciphering process would take a long time. [124, pp. 91–92]

In 1974 the first detailed description of such a *one-way function* was published [97]. Speaking informally, a one-to-one function $f: X \to Y$ is "one-way" if it is easy to compute $f(x)$ for any $x \in X$ but hard to compute $f^{-1}(y)$ for most randomly selected y in the range of f.[1] In [97] the passwords and their enciphered forms were regarded as integers modulo a large prime p, and the "one-way" map from $\mathbb{Z}/p\mathbb{Z}$ to

*Received by the editors December 23, 2003; accepted for publication (in revised form) August 9, 2004; published electronically October 29, 2004.
http://www.siam.org/journals/sirev/46-4/43919.html

[†]Department of Mathematics, Box 354350, University of Washington, Seattle, WA 98195 (koblitz@math.washington.edu).
[‡]Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON N2L 3G1, Canada (ajmeneze@uwaterloo.ca).

[1]In some situations one wants a one-way function to have a stronger property, namely, that it is hard to compute any partial information about $f^{-1}(y)$ (for instance, whether it is an odd or even number) for most randomly selected y.

$\mathbb{Z}/p\mathbb{Z}$ was given by a polynomial $f(x)$ which is not hard to evaluate by computer but which takes an unreasonably long time to invert. In the paper, $p = 2^{64} - 59$ and $f(x) = x^{2^{24}+17} + a_1 x^{2^{24}+3} + a_2 x^3 + a_3 x^2 + a_4 x + a_5$, where the coefficients a_i were arbitrary 19-digit integers. At the time, one-way functions were used only to store passwords and not to send scrambled messages.

Until the late 1970s, all cryptographic message transmission was by *symmetric key*. This means that someone who has enough information to encrypt messages also has enough information to decipher messages. As a result, any two users of the system who want to communicate secretly must have exchanged keys in a safe way, perhaps using a trusted courier.

The arena for applying mathematics to cryptography expanded dramatically when Diffie and Hellman invented an entirely new type of cryptography, called *public key* [32].[2] At the heart of this concept is the idea of using a one-way function for encryption.

The functions used for encryption belong to a special class of one-way functions that remain one-way only if some information (the *decryption key*) is kept secret. Again using informal terminology, we can define a *public-key encryption function* (also called a "trapdoor" function) as a map from plaintext message units to ciphertext message units that can be feasibly computed by anyone having the *public key* but whose inverse function (which deciphers the ciphertext message units) cannot be computed in a reasonable amount of time without some additional information, called the *private key*.

This means that everyone can send a message to a given person using the same enciphering key, which can simply be looked up in a public directory whose contents can be authenticated by some means. There is no need for the sender to have made any secret arrangement with the recipient; indeed, the recipient need never have had any prior contact with the sender at all.

A possible reason for the late development of the concept of public key is that until the 1970s cryptography was used mainly for military and diplomatic purposes, for which symmetric-key cryptography was well suited. However, with the increased computerization of economic life, new needs for cryptography arose. Unlike in the military or the diplomatic corps—with rigid hierarchies, long-term lists of authorized users, and systems of couriers—in the applications to business transactions and data privacy one encounters a much larger and more fluid structure of cryptography users. Thus, perhaps public-key cryptography was not invented earlier simply because there was no real need for it until recently.

Some of the purposes for which public-key cryptography has been applied are

1. *confidential message transmission*;
2. *identification systems*, where users prove that they are authorized to have access to data or to a facility, or that they are who they claim to be;
3. *authentication*, which establishes that the message was sent by the person claimed and that it hasn't been tampered with;
4. *nonrepudiation*, which guards against people claiming not to have agreed to something that they really agreed to;

[2]It is now known that some of the ideas published in [32] and also in [98] had been developed in secret a few years before by the British intelligence services. However, they did not appear to appreciate the importance of public-key cryptography or the possibility of signatures and other applications. It was only with the publication of [32] and [98] that research in this area started to flourish.

5. *key establishment*, where two people using the open airwaves want to agree upon a secret key for use in some symmetric-key cryptosystem;
6. *electronic cash* mechanisms that ensure spender anonymity;
7. *electronic voting* schemes that ensure that votes are confidential and correctly tallied.

These tasks are performed through various types of *protocols*. The word "protocol" simply means an orderly procedure in which people send messages to one another.

The path from an academic proposal for a new type of mathematical cryptography to its practical implementation is long and arduous. First of all, mathematicians and cryptographers must become convinced that the underlying number-theoretic or combinatorial problem upon which the system's security relies is truly intractable. The only way to be more-or-less sure of this is to wait while experts try to find reasonably fast algorithms to break the cryptosystem; if they fail to do so after several years of trying, then one might believe that the problem is most likely an intrinsically difficult one. For example, most people believe that integer factorization, upon which RSA (Rivest–Shamir–Adleman) cryptography is based (see section 3), is intractable (for at least the next few years) for integers of more than 300 decimal digits.

It would be nice, of course, to be able to prove theorems that state that such a problem cannot be efficiently solved. Ideally, such a theorem would show that the currently known algorithms are close to best possible. But unfortunately no nontrivial theorems of that sort have been proved for any of the problems whose intractability is assumed in public-key cryptosystems.

There has been a lot of work recently on so-called *provable security*. However, this is a misnomer. "Provable security" results have a conditional form: "If problem X is intractable, then the cryptosystem Y is secure against attacks of type Z." Note that the intractability of the underlying mathematical problem is being assumed; moreover, there is no assurance that cryptosystem Y will not succumb to an attack of type Z', where $Z' \neq Z$.

Even if a consensus emerges that the mathematical problem at the heart of a newly proposed cryptosystem is really intractable, many other issues remain. One must evaluate different methods of choosing parameters for the system. How big must the numbers be? What are the "weak parameters" (parameters for which the supposedly difficult problem becomes much easier) that must be avoided?

In the real world, a company's credibility and large sums of money are at stake. How can businesses protect themselves against liability if a cryptosystem that was supposed to be secure is broken and thousands of credit card numbers are stolen? The answer is that various "standards bodies" affiliated with professional organizations such as ANSI, IEEE, and ISO, evaluate and make recommendations for the use of approved cryptosystems. If companies use products that adhere to these guidelines, then they are largely protected from any possible lawsuit if a system is broken. It would be extremely risky for a company to sell a product with a type of cryptography that has not been approved by the major standards bodies.

Standards bodies typically include representatives of various constituencies and professional groups, not all of whom are knowledgeable about the mathematics of cryptography. Before a cryptosystem is included in the recommendations of a standards body, a large number of people have the opportunity to raise objections either to the cryptosystem in its entirety or to the proposer's suggestions for implementation (choice of parameters, methods of generating keys, etc.). Naturally, marketers of competing cryptosystems have a strong incentive to find something wrong with the new system. And no one wants to end up in the embarrassing situation of having approved a system that is broken a few months later. So it is not surprising that standards bodies tend to be conservative and slow-moving. In the case of the most popular current public-key cryptosystems, the time lags between academic publication of a proposal for a type of cryptography and approval of specific recommendations for its practical use were roughly 15 years.

We shall discuss in most detail the public-key cryptosystems that are of greatest practical importance, but we shall also mention some other systems that are intrinsically interesting or that show some promise for the future. This survey will be selective rather than exhaustive and will reflect our own tastes and judgments.

Researchers who have a computer science background might fault us for neglecting foundational questions. Because cryptography is multidisciplinary, opinions about the importance of certain lines of work often differ sharply. In particular, some mathematicians are skeptical about the value for practical cryptography of the theoretical results that other researchers consider to be fundamental to the field. At the risk of exaggerating, we might summarize the critics' point of view as follows: *there is no such thing as a useful, nontrivial, unconditional theorem in cryptography.* The theorems that one can prove, the skeptics point out, generally have assumptions that are so strong that the desired conclusion essentially becomes an immediate consequence. Such theorems can clarify the relationship among various definitions and terms, but because they lack true mathematical depth, they cannot provide any real assurance of the security of a cryptosystem.

We do not entirely share this skeptical viewpoint, at least not in its most extreme form. However, like most mathematicians working in cryptography, we prefer a pragmatic rather than theoretical approach. Our views on the practical relevance of "provable security" results are presented in more detail in our recent article [67].

2. Notions of Security. It is a subtle and complicated matter to evaluate the security of a public-key cryptosystem. It is not enough to know that an adversary is unlikely to be able to compute the inverse of the encryption function. Most successful attacks on popular cryptosystems are more indirect than that. For example, suppose that Alice is receiving messages that have been encrypted using RSA (see below). The plaintext messages have to adhere to a certain format, and if a decrypted message is not in that form, Alice's computer transmits an error message to the sender. This seems innocuous enough. However, Bleichenbacher [14] has shown that sometimes such error messages could be used to compromise security.

A cryptographic protocol is said to be secure if an adversary cannot achieve certain well-defined goals, that is, cannot compromise the system in a certain clearly stated way. It is usually assumed that the adversary not only knows all the public keys, but also has a complete description of the algorithms used to carry out the protocol. When making a statement about the security of a protocol, one must explicitly delineate the adversary's capabilities, for example, its computational power and the nature of its interactions with legitimate parties.

A protocol is considered robust if it can withstand attacks by adversaries who are powerful and whose goals are modest. In contrast, the most obvious notion of security for a public-key encryption scheme—that an adversary who is given a public key and a ciphertext C derived with that public key is unable to determine (in a feasible amount of time) the corresponding plaintext M—is actually quite weak. In practice one might wish to prevent adversaries from meeting the less ambitious goal of being able to determine *any* information whatsoever about M from C. This stronger notion, called *semantic security*, was first studied systematically by Goldwasser and Micali

[44]. Furthermore, the adversary may be permitted access to a decryption "oracle," that is, a black box from which it can obtain the decryption of any ciphertext of its choice except, of course, the target ciphertext C. The relationships between various notions of security for public-key encryption schemes were studied in [7].

In recent years, researchers have become increasingly aware of the possibility of attacks that exploit specific properties of the implementation and operating environment. Such *side-channel attacks* utilize information leaked by the computing devices during the execution of private-key operations such as decryption and signature generation. The kind of information that can be exploited includes execution time [68], power consumption [69], electromagnetic radiation [2], induced errors [16], and error messages [14, 76]. Such information may be difficult to obtain on some devices, such as a workstation located in a secure office, but may be easy to obtain from other devices, such as a smart card which draws power from an external, untrusted source.

We should caution that most of the cryptosystems described in the remainder of the article are only *primitives*. In cryptography the term "primitive" means a basic ingredient in a cryptosystem. In practice, one generally has to modify and combine these primitives in a careful way so as to simultaneously achieve various objectives related to efficiency and strong notions of security.

3. RSA. The public-key cryptosystem that has been in practical use the longest—and is still the most popular system for electronic commerce—is RSA [98]. The basic construction is rather simple. Let the n-bit integer $N = pq$ be the product of two large primes of roughly the same size. Typically, N has about 1000 bits, and p and q each have about 500 bits.[3]

Let e and d be two integers satisfying $ed \equiv 1 \pmod{\varphi(N)}$, where $\varphi(N) = (p-1)(q-1) = N + 1 - (p+q)$ is the Euler φ-function of N, equal to the number of integers $0 \leq i < N$ that are relatively prime to N. These integers N, e, d are called, respectively, the *RSA modulus*, the *encryption exponent*, and the *decryption exponent*. The first two form the *public key* and are made publicly known. The integer d, sometimes called the *secret exponent*, is the private key known only to the person (Alice) who receives the enciphered message.

In practice, public-key encryption schemes are many times slower than their symmetric-key counterparts. Thus, RSA is typically used either to encrypt a short message (such as a credit card number) or else to encrypt a randomly chosen key k, which in turn is used with a symmetric-key encryption scheme such as the Advanced Encryption Standard (AES) to encrypt the message itself. The key k is usually quite short (e.g., 128, 192, or 256 bits for the AES), and can therefore be regarded as an integer M in the interval $[0, N-1]$.

To encrypt such a message unit M, the sender Bob computes the ciphertext C, which is the least positive residue of M^e modulo N. To decrypt C, the recipient Alice computes the least positive residue of C^d modulo N. (These operations of modular exponentiation can be carried out rapidly by means of a "repeated squaring" method.) Using Euler's theorem from elementary number theory, one can easily show that $C^d \equiv M^{ed} \equiv M \pmod{N}$.

Anyone who succeeds in factoring $N = pq$ can immediately break RSA by finding an inverse of e modulo $(p-1)(q-1)$. For many years it was conjectured that, conversely, the only way that RSA can be broken (in other words, the only way that

[3]It is easy to find random large primes by choosing random integers and performing tests on them—either very efficient "strong primality" tests or the deterministic polynomial-time primality test discovered in 2002 by M. Agrawal, N. Kayal, and N. Saxena.

the encryption function can be inverted) is to factor N. However, work by Boneh and Venkatesan [21] suggests that this conjecture might be false, that is, that the integer factorization problem might be strictly harder than the problem of inverting the RSA function $M \mapsto M^e$ modulo N.

3.1. Signatures. A particularly attractive feature of RSA is that there is a natural way to digitally sign messages. Suppose that Alice sends a message to Bob and wants to append a short "signature" S to her message that will convince Bob that it was really Alice who sent the message and that the message he received was not altered during transmission. Her whole message M might be long, consisting of a large number of message units. Alice's first step is to apply a publicly known *hash function* to M. This is a function $M \mapsto H$, where H is no longer than a single message unit. The function must be easy to compute and must satisfy two properties: (1) it must be computationally infeasible to find two different messages with the same hash value, and (2) given H, it must be computationally infeasible to find any message with hash value H.

Alice uses her RSA private key d_{Alice} to form her signature. Namely, she sets S equal to the least positive residue of $H^{d_{\text{Alice}}}$ modulo N_{Alice}. Bob, who has already computed the hash value H of the message he received and who knows Alice's public key, can check that $S^{e_{\text{Alice}}} \equiv H \pmod{N_{\text{Alice}}}$. If this congruence holds, he knows that no one but Alice could have composed the signature S (since no one else knows her decryption exponent d_{Alice}), and he also knows that the message he received could not have been tampered with (because it has the same hash value as the message that Alice sent).

3.2. Factorization Attack on RSA. The most basic attack on RSA consists of factoring the modulus $N = pq$. The integer factorization problem has been the subject of intense research, especially in the years since the invention of RSA in 1977. Let N be an n-bit integer. Most of the subexponential-time algorithms—those that take fewer than 2^{n^c} steps with $c < 1$—are of *index calculus* type. We now describe a simple index calculus algorithm to factor N.

The method is based on the elementary observation that if $x^2 \equiv y^2 \pmod{N}$, then $N = pq|(x+y)(x-y)$, and so p and q each must divide either $x+y$ or $x-y$. If x and y were formed independently of one another, then one expects that 50% of the time the two primes will divide different factors, say $p|x+y$, $q|x-y$. In that case we can factor N by using the Euclidean algorithm to compute $\gcd(N, x+y) = p$.

We start the index calculus factoring algorithm by choosing a *factor base* $\mathcal{F}$ consisting of all primes less than some bound B along with the number -1: $\mathcal{F} = \{p_0, p_1, \ldots, p_r\}$, where $p_0 = -1$, $p_1 = 2$, $p_2 = 3, \ldots$. We next choose positive integers $a < N$ (either randomly or according to some convenient criteria) and compute the least absolute residue of a^2. If this residue cannot be written as a product of numbers in our factor base, we choose another value of a. We finally arrive at a system of mod N relations of the form $a_i^2 \equiv \prod_{j=0}^{r} p_j^{\alpha_{i,j}}$, $i = 1, \ldots, s$. We try to form a product $\prod_i a_i^{2\nu_i} \equiv \prod_{i,j} p_j^{\nu_i \alpha_{i,j}}$, where $\nu_i \in \{0,1\}$, in such a way that we get a perfect square on the right. In other words, we need each prime on the right to occur to an even power; that is, $\sum_i \nu_i \alpha_{i,j}$ must be even for each $j = 0, \ldots, r$. This amounts to solving a system of $r+1$ simultaneous equations in s unknowns over the field $\mathbb{F}_2 = \{0,1\}$. Once we have such a product, we can set $x = \prod_i a_i^{\nu_i}$ and $y = \prod_j p_j^{\mu_j}$ with $\mu_j = \frac{1}{2}\left(\sum_i \nu_i \alpha_{i,j}\right)$. Then $x^2 \equiv y^2 \pmod{N}$, and there is a 50% chance that we can immediately factor N. If we fail to factor N, we find another solution to the simultaneous equations over $\mathbb{F}_2$, and try again.

Example 1. *Let $N = 319$, and choose $\mathcal{F} = \{-1, 2, 3, 5, 7, 11, 13\}$. After squaring some 2-digit numbers, we find that we can take a_i, $1 \le i \le 7$, equal to* $17, 18, 19, 25, 27, 33, 36$ *because of the following relations mod* 319:

$$17^2 \equiv -2 \cdot 3 \cdot 5, \quad 18^2 \equiv 5, \quad 19^2 \equiv 2 \cdot 3 \cdot 7, \quad 25^2 \equiv -13,$$

$$27^2 \equiv 7 \cdot 13, \quad 33^2 \equiv 2^2 \cdot 3 \cdot 11, \quad 36^2 \equiv 2^2 \cdot 5.$$

The exponents of the p_j in $\prod_i a_i^{2\nu_i}$ are the left sides of the following system of congruences mod 2:

$$\begin{aligned} \nu_1 + \nu_4 &\equiv 0, \\ \nu_1 + \nu_3 + 2\nu_6 + 2\nu_7 &\equiv 0, \\ \nu_1 + \nu_3 + \nu_6 &\equiv 0, \\ \nu_1 + \nu_2 + \nu_7 &\equiv 0, \\ \nu_3 + \nu_5 &\equiv 0, \\ \nu_6 &\equiv 0, \\ \nu_4 + \nu_5 &\equiv 0. \end{aligned}$$

One solution for the ν-vector is $(0, 1, 0, 0, 0, 0, 1)$*, but that leads only to the trivial congruence* $10^2 \equiv 10^2 \pmod{N}$*. We have better luck with the solution* $(1, 1, 1, 1, 1, 0, 0)$*, which gives* $(17 \cdot 18 \cdot 19 \cdot 25 \cdot 27)^2 \equiv (2 \cdot 3 \cdot 5 \cdot 7 \cdot 13)^2 \pmod{N}$*, i.e.,* $112^2 \equiv 178^2 \pmod{N}$*. We now immediately compute* $\gcd(319, 112 + 178) = 29$*, and factor* $319 = 29 \cdot 11$.

It can be shown that the time required to factor an n-bit integer by the above index calculus factorization method is of order $2^{n^{1/2+\epsilon}}$ for any $\epsilon > 0$. (More precisely, the number of steps is $\exp(O(\sqrt{n \log n}))$.) Throughout the 1980s modifications and generalizations were introduced that improved upon the performance of index calculus methods; however, no one was able to reduce the exponent of n below $1/2 + \epsilon$. Even when Lenstra developed an exciting and conceptually very different factorization method based on elliptic curves [74], asymptotically his method required roughly the same amount of time as the index calculus algorithms. Some people wondered whether the exponent $1/2+\epsilon$ might be best possible for a general integer factorization algorithm.

However, in the 1990s ideas of Pollard [95] led to a major breakthrough in factorization, called the *number field sieve*. By carrying over index calculus to algebraic number fields, it was possible to factor an arbitrary n-bit integer in time bounded by $2^{n^{1/3+\epsilon}}$ for any $\epsilon > 0$. (More precisely, $\exp(O(\sqrt[3]{n \log^2 n}))$.) The number field sieve is at present the fastest method for factoring an RSA modulus; the current record is a number of 576 bits.

The reduction of the exponent of n from $1/2 + \epsilon$ to $1/3 + \epsilon$ has important consequences in the long run. It means that even modest improvements in hardware and software can significantly increase the size of the numbers that can be factored. For this reason the current recommendation for implementation of RSA is to use numbers of at least $n = 1024$ bits.

A recent research trend has been to design special-purpose hardware on which factoring algorithms such as the number field sieve might be faster or more cost-effective than on conventional general-purpose computers. Among the noteworthy proposals are Shamir's TWINKLE machine (see [72]), Bernstein's circuits [10], and the TWIRL machine of Shamir and Tromer [108]. Shamir and Tromer [108] estimate that the relation-generation stage of the number field sieve for factoring a 1024-bit RSA modulus can be completed in less than a year by a machine that would cost about \$10 million to build, and that the linear algebra stage is easier. Such special-purpose hardware has yet to be built (unless it has been built in secret), so it remains to be seen if this work will have any impact on the size of RSA moduli used in practice.

3.3. Other Algorithmic Attacks on RSA. Most successful attacks on RSA are not based on factoring the modulus N and do not result from the implementer's use of insufficiently large N. Rather, they exploit subtle features of the particular way in which RSA is used. We give two examples; for a more thorough treatment, see [15].

First of all, suppose that Alice chooses a small value for her decryption exponent d in order to speed up the decryption of messages sent to her and the signing of messages that she sends. (Recall that both tasks require her to raise an integer to the dth power modulo her n-bit modulus N.) If d is much smaller than N, this is a very bad idea. Namely, Wiener [123] showed that if d has fewer than $n/4$ bits (more precisely, if $d < \frac{1}{3}N^{1/4}$), then an unauthorized person knowing only the public key can efficiently compute d. Boneh and Durfee [17] raised the exponent of N to 0.292, and they conjectured that if $d < N^{1/2}$, then there should be an efficient algorithm to determine d. Thus, Alice should always choose d with more than $n/2$ bits; preferably, d should have n bits.

On the other hand, Alice can probably get away with choosing her public exponent e (which is used to encrypt messages and also to verify signatures) to be small. In fact, most implementations of RSA use $e = 3$ or $e = 2^{16} + 1 = 65537$. But Håstad [49] found a flaw when e is small and Alice wants to broadcast the same message M to a large number of users with their different public keys N_i and small public exponents. Suppose, for instance, that all of the public exponents are $e = 3$. Then an eavesdropper Eve who knows the ciphertext C_i sent to three different recipients can recover the message M. To see this, suppose that $M < N_i$, $i = 1, 2, 3$ (otherwise M has to be broken up into smaller message units). Eve knows the residue of M^3 modulo each N_i, since that is precisely C_i. Using the Chinese remainder theorem, she can then compute the residue of M^3 modulo the product $N_1N_2N_3$. But that residue is equal to M^3 itself, since $M^3 < N_1N_2N_3$. Once Eve knows the actual value of M^3, she can trivially extract the cube root to find M.

This difficulty—along with some others—can be avoided by *padding* messages, that is, by inserting a short sequence of random symbols in message units before sending them (in such a way that the recipient can easily delete the added symbols before reading the text). Of course, a different random sequence must be inserted each time Alice sends a message.

3.4. Side-Channel Attacks. We give an example of a power analysis attack on the RSA signature scheme. Suppose that a smart card generates signatures using the repeated squaring method for exponentiation. That is, if the binary representation of the decryption exponent is $d = \sum_{i=0}^{l} d_i 2^i$, then the smart card computes $S = H^d \bmod N$ as follows:

```
1. $S \leftarrow 1$.
2. For $i$ from 0 to $l$ do
          If $d_i = 1$ then $S \leftarrow S \cdot H \bmod N$.
          $H \leftarrow H^2 \bmod N$.
3. Return($S$).
```

Because modular squaring and modular multiplication are usually implemented as different routines (since the former is faster than the latter), it can be expected that the power consumed by the smart card while performing a squaring has different characteristics than when a multiplication is performed. These differences can sometimes be visualized by plotting the power trace which shows the power consumed during each clock cycle. Hence, examination of the power trace of the operation $H^d \bmod N$ can reveal the sequence of multiplication and squaring operations and thus the private key d.

One way to counteract this attack is to insert dummy operations as follows so that one squaring and one multiplication are performed during each iteration of the main loop:

```
1. $S_0 \leftarrow 1$.
2. For $i$ from 0 to $l$ do
       $S_1 \leftarrow S_0 \cdot H \bmod N$.
       $S_0 \leftarrow S_{d_i}$.
       $H \leftarrow H^2 \bmod N$.
3. Return($S_0$).
```

This countermeasure decreases efficiency and, moreover, may still allow other side-channel attacks. The development of cost-effective and efficient countermeasures to side-channel attacks is an ongoing research problem that is being tackled by both cryptographers and engineers.

3.5. Deployment. RSA is the most widely deployed public-key cryptosystem today. A common everyday use of RSA is in the Secure Sockets Layer (SSL) protocol that is used by popular browsers such as Netscape and Internet Explorer for secure web transactions such as credit card payments. SSL is used to assure an individual user (called a *client*) of the authenticity of the web site (called the *server*) he or she is visiting, and to establish a secure communications channel for the remainder of the session. Web pages that are protected with SSL have addresses that start with "https". Web pages with addresses that start simply with "http" are not protected.

When a client first visits a secured web page (e.g., https://www.nsa.gov), the server transmits its *certificate* to the client. Such a certificate has two components, a *data part* containing the server's identifying information and RSA public key, and a *signature part* which is the RSA signature of a *certifying authority* that vouches for the data part. It is assumed that the certifying authority has carefully verified the server's identity before issuing the certificate. Upon receipt of the certificate, the client verifies the signature using the certifying authority's public key, which is preinstalled in the browser. A successful verification confirms the authenticity of the server and of its RSA public key. Note that while the server is authenticated to the client, there is no authentication of the client to the server. SSL does have client-to-server authentication capability, but this is seldom used in practice because it is difficult to implement a system to certify the public keys of individual users on a large scale.

Next, the client selects a random session key, encrypts it with the server's RSA public key, and transmits the resulting ciphertext to the server. The server decrypts the session key, which is then used with a symmetric-key cryptosystem to encrypt and authenticate all sensitive data exchanged for the remainder of the session.

The establishment of a secure link is indicated by a closed padlock in the Netscape and Internet Explorer browsers. Clicking on this icon reveals the server's certificate and information about the certifying authority.

4. Knapsack. The *knapsack problem*, also known as the *subset sum problem*, is the following: Given an n-tuple $\{v_i\}$ of positive integers and an integer V, find an n-bit integer $N = (\epsilon_{n-1}\epsilon_{n-2}\cdots\epsilon_1\epsilon_0)_2$, $\epsilon_i \in \{0,1\}$, such that $\sum_{i=0}^{n-1}\epsilon_i v_i = V$, if such an N exists. Note that there may be no solution N or many solutions, or there might be a unique solution, depending on the n-tuple $\{v_i\}$ and the integer V.

A special case of the knapsack problem is the *superincreasing knapsack*. This is the case when the v_i, arranged in increasing order, have the property that each one is greater than the *sum* of all of the earlier v_i. For example, if $v_i = 2^i$, then the problem is trivial; the unique solution is $N = V$.

It is known that the general knapsack problem is NP-hard.[4] However, any superincreasing knapsack problem is easy to solve. Namely, we look down the v_i, starting with the largest, until we get to the first one that is $\le V$. We include the corresponding i in our subset I (in other words, we take $\epsilon_i = 1$), replace V by $V - v_i$, and then continue down the list of v_i until we find one that is less than or equal to this difference. Continuing in this way, either we eventually obtain a subset of $\{v_i\}$ which sums to V, or else we reach a step where we have $V - \sum_{i\in I} v_i$ equal to a positive integer less than all of the remaining v_i (or equal to a positive integer when there are no remaining v_i), in which case there is no solution.

We now describe how to construct the basic knapsack cryptosystem of Hellman and Merkle [50]. We suppose that our plaintext message units are n-bit integers M. Each user chooses a superincreasing n-tuple $\{v_0,\ldots,v_{n-1}\}$, an integer m which is greater than $\sum_{i=0}^{n-1} v_i$, and an integer a prime to m, $0 < a < m$. This is done by some random process. The user then computes $b = a^{-1} \bmod m$ (that is, b is the least positive integer such that $ab \equiv 1 \pmod{m}$), and also computes the n-tuple $\{w_i\}$ defined by $w_i = av_i \bmod m$ (that is, w_i is the least positive residue of av_i modulo m). The user keeps the numbers v_i, m, a, and b all secret, but publishes the n-tuple of w_i. That is, the enciphering key is $K_E = \{w_0,\ldots,w_{n-1}\}$. The deciphering key is $K_D = (b,m)$ (which, along with the enciphering key, enables one to determine $\{v_0,\ldots,v_{n-1}\}$).

Someone who wants to send a message $M = (\epsilon_{n-1}\cdots\epsilon_1\epsilon_0)_2$ to a user with enciphering key $\{w_i\}$ computes $C = f(M) = \sum_{i=0}^{n-1}\epsilon_i w_i$ and transmits that integer. To decipher the message, the recipient first finds the least positive residue V of bC modulo m. Since $bC = \sum \epsilon_i b w_i \equiv \sum \epsilon_i b a v_i \equiv \sum \epsilon_i v_i \pmod{m}$, it follows that $V = \sum \epsilon_i v_i$. (Here we are using the fact that both $V < m$ and $\sum \epsilon_i v_i \le \sum v_i < m$ to convert the congruence modulo m to equality.) It is then easy to find the unique solution $(\epsilon_{n-1}\cdots\epsilon_0)_2 = M$ of the superincreasing knapsack problem.

Note that an eavesdropper who knows only $\{w_i\}$ is faced with the knapsack problem $C = \sum \epsilon_i w_i$, which is *not* a superincreasing problem, because the superincreasing property of the n-tuple of v_i is destroyed when v_i is replaced by the least positive residue of av_i modulo m. Thus, at first glance, the unauthorized person seems to be faced with a much harder problem.

For a while, many people were optimistic about the possibilities for the Merkle–Hellman knapsack. Encryption and decryption are fast—much faster than in RSA. Moreover, they hoped that, since the problem of solving a knapsack is NP-hard, the system should be secure.

[4]This means that any problem $\mathcal{P}$ in a very broad class can be reduced to the general knapsack problem. Roughly speaking, any algorithm for the knapsack can be modified to get an equally efficient algorithm for $\mathcal{P}$. If there were a polynomial-time algorithm for the knapsack problem, then any such problem $\mathcal{P}$ would also be solvable in polynomial time, and the famous $P \neq NP$ conjecture would be false.

However, there was a fallacy in that reasoning. The type of knapsack problem $C = \sum \epsilon_i w_i$ that must be solved, while not a superincreasing knapsack, is nevertheless of a very special type, namely, it is obtained from a superincreasing problem by a simple modular multiplication. In 1982, Shamir [107] found an algorithm to solve this type of knapsack problem that is polynomial in n. Thus, the original Merkle–Hellman cryptosystem is completely insecure.

One way around Shamir's algorithm is to make the knapsack system a little more complicated by using a sequence of transformations of the form $x \mapsto ax \bmod m$ for different a and m. However, Brickell [22] generalized Shamir's attack to all such "low-density" knapsacks (see also [23] and [88]).

A few years later Chor and Rivest [27] developed a type of knapsack cryptosystem that did *not* use low-density knapsacks and remained unbroken for a decade. The Chor–Rivest system is based on the multiplicative structure of the finite field $\mathbb{F}_{p^m}$ of p^m elements, where one might choose, for example, $p = 197$, $m = 24$. Alice's public key $\{w_1, \ldots, w_n\}$ that Bob uses to encipher an n-bit message $\{\epsilon_1, \ldots, \epsilon_n\}$ is obtained as follows (here we give a simplified version of the actual construction). Alice represents elements of $\mathbb{F}_{p^m} = \mathbb{F}_p[X]/f(X)$, where $f(X)$ is a fixed irreducible polynomial of degree m, as polynomials of degree less than m. Let g be a generator of $\mathbb{F}_{p^m}^*$, and let π be a fixed secret permutation of $\{1, \ldots, n\}$. Then for $i = 1, \ldots, n$ Alice lets w_i, $1 \le w_i < q - 1$, be integers such that $g^{w_i} = X + \pi(i)$ in $\mathbb{F}_p[X]/f(X)$. Bob encrypts a message by setting $C = \sum \epsilon_i w_i$, and Alice decrypts by factoring the polynomial $g^C = \prod (X + \pi(i))^{\epsilon_i}$ in $\mathbb{F}_p[X]$. Although the cryptosystem seemed much harder to attack than the low-density knapsack systems, in 1998 it was broken by Vaudenay [119].

Shamir's complete breaking of the original Merkle–Hellman knapsack in 1982 was a jolting experience for the nascent academic cryptographic research community of the time. A promising public-key system, which was more efficient than RSA and seemingly more secure as well (since RSA is not based on an NP-hard problem), was totally demolished by Shamir's paper four years after it was invented. And subsequent attempts to rescue the knapsack idea fared no better.

There are at least two general lessons to be learned here. First, when a proposer's original version of a cryptosystem is successfully attacked, often it is futile to thwart this attack by tweaking the system a little—by changing the parameters or inserting a new layer of complexity. The success of the attack possibly indicates a fundamental weakness in the system, in which case modified versions will succumb to variants or generalizations of the original attack.

A second lesson is to be skeptical of theoretical arguments for the security of a system. Concepts of complexity theory such as NP-hardness do not necessarily have direct relevance to cryptography. Since the time of the ancient Greeks, our model of mathematical elegance has been a rigorously proved theorem. However, in cryptography such theoretical results—often appearing with the name "provable security"[5]—are sometimes less convincing than a decade or two of computational experience attempting unsuccessfully to break a system.

5. Discrete Logarithm Cryptosystems.

Another type of public-key cryptographic system is based on the discrete logarithm problem (DLP) in a finite field. Let $\mathbb{F}_q$ denote the field of q elements, and let $g \in \mathbb{F}_q^*$ be a fixed element, not necessarily a generator. The *DLP* in $\mathbb{F}_q^*$ to the base g is the following problem: Given $y \in \mathbb{F}_q^*$,

[5]Lars Knudsen once commented, "If it's provably secure, then it probably isn't."

find an integer x such that $y = g^x$ (or, if y is not in the subgroup generated by g, determine that no such integer exists; but in cryptographic applications y is always a power of g).

5.1. The Diffie–Hellman Key Exchange. The Diffie–Hellman key exchange [32] works as follows. Suppose that Alice and Bob want to agree upon a key, perhaps for use in some symmetric-key cryptosystem. This must be done using open communication channels. That is, an eavesdropper Eve knows everything that Alice sends to Bob and everything that Bob sends to Alice.

Alice and Bob first agree on a finite field $\mathbb{F}_q$ and a base element g of order N, where $N|q-1$. Their communication is public, so Eve also has this information. Next, Alice secretly chooses a random positive integer $k_{\text{Alice}} < N$, computes $g^{k_{\text{Alice}}} \in \mathbb{F}_q^*$, and sends this to Bob. Meanwhile, Bob does likewise: he sends $g^{k_{\text{Bob}}} \in \mathbb{F}_q^*$ to Alice, while keeping k_{Bob} secret. The agreed upon key will then be the element

$$g^{k_{\text{Alice}}k_{\text{Bob}}} \in \mathbb{F}_q^*,$$

which Bob can compute by raising the field element he received from Alice to his secret k_{Bob}-th power, and Alice can compute by raising the field element she received from Bob to the k_{Alice}-th power. This works because in $\mathbb{F}_q^*$ we have

$$g^{k_{\text{Alice}}k_{\text{Bob}}} = \left(g^{k_{\text{Alice}}}\right)^{k_{\text{Bob}}} = \left(g^{k_{\text{Bob}}}\right)^{k_{\text{Alice}}}.$$

In this way Alice and Bob have arrived at a common randomly generated element of the subgroup of $\mathbb{F}_q^*$ generated by g. If they want their key to be a large integer or sequence of bits, they can agree upon a simple function from $\mathbb{F}_q$ to the integers that will convert the shared key to the desired form.

The problem facing the adversary Eve is the so-called *Diffie–Hellman problem*: Given g, g^{k_A}, $g^{k_B} \in \mathbb{F}_q^*$, find $g^{k_A k_B}$. It is easy to see that anyone who can solve the DLP in $\mathbb{F}_q^*$ can then immediately solve the Diffie–Hellman problem as well. The converse is not known. That is, it is conceivable (though thought to be unlikely) that someone could invent a way to solve the Diffie–Hellman problem without being able to find discrete logarithms. In other words, breaking the Diffie–Hellman key exchange has not been *proven* to be equivalent to solving the DLP. For partial results supporting the conjectured equivalence of the two problems, see [19] and [79]. In practice it is probably safe to assume that the Diffie–Hellman key exchange is secure provided that the discrete logarithm problem is intractable.

Because of the Pohlig–Hellman algorithm [93], the order N of the base element should be either prime or "almost prime" (the product of a prime and a very small integer).

5.2. The Digital Signature Algorithm (DSA). In 1991 the U.S. government's National Institute of Standards and Technology proposed a digital signature standard using a Digital Signature Algorithm (DSA) based on the DLP in a prime field $\mathbb{F}_p$.

To set up the scheme, each user Alice proceeds as follows:

1. she chooses a prime N of about 160 bits (using a random number generator and a primality test);
2. she then chooses a second prime p that is congruent to 1 modulo N and has at least 1000 bits;
3. she chooses a generator g of the cyclic subgroup of $\mathbb{F}_p^*$ of order N (by computing $g_0^{(p-1)/N} \bmod p$ for a random integer g_0; if this number is not equal to 1, it will be a generator);

4. she takes a random integer x in the range $0 < x < N$ as her secret key, and sets her public key equal to $y = g^x \bmod p$.

Now suppose that Alice wants to sign a message. Using a hash function that takes positive integer values less than N, she computes the hash value H of her message. She next picks a random integer k in the same range $0 < k < N$, computes $g^k \bmod p$, and sets r equal to the least nonnegative residue modulo N of the latter number (that is, g^k is first computed modulo p, and the result, regarded as an integer in $\{0, 1, \ldots, p-1\}$, is then reduced modulo the smaller prime N). Finally, Alice finds an integer s such that $sk \equiv H + xr \pmod{N}$. (This just involves multiplying the number on the right by the inverse of k modulo N.) Her signature is the pair (r, s) of integers modulo N.

To verify the signature, the recipient Bob computes the hash value H and then $u_1 = s^{-1}H \bmod N$ and $u_2 = s^{-1}r \bmod N$. He then computes $g^{u_1}y^{u_2} \bmod p$. If the result agrees modulo N with r (as it should, since $g^{u_1+xu_2} = g^k$), he is satisfied. He accepts the signature because he is confident that only someone who knew Alice's secret key x—presumably, this means only Alice—could have formed the signature. He also knows that the message has not been tampered with, since the hash value H of the message he received is the same as the hash value of the message that Alice sent.

The only way known to forge a DSA signature is to find discrete logs in $\mathbb{F}_p^*$. This requires roughly the same amount of time as factoring a positive integer that has the same size as p. In fact, the fastest method available at present to solve the DLP in $\mathbb{F}_p$ is a variant of the same technique—the number field sieve—that can factor the largest integers [46].

The DSA has the advantage that signatures are fairly short, consisting of two numbers of 160 bits (the magnitude of N). By comparison, the RSA signature in section 3.1 is about three times as long. The security of the system depends upon intractability of the DLP in the multiplicative group of the rather large field $\mathbb{F}_p$. Although to break the system it would suffice to find discrete logs in the smaller subgroup generated by g, in practice this seems to be no easier than finding arbitrary discrete logarithms in $\mathbb{F}_p^*$. Thus, the DSA seemed to have attained both a high level of security and low signature storage and implementation time. However, recently the DSA has been superseded by the ECDSA, which is a similar system based on the group of an elliptic curve rather than a finite field. This signature scheme will be described in the next section.

6. Elliptic Curve Cryptography.

Elliptic curves have been extensively studied for almost two centuries, and there is a vast literature on the topic. Research into number-theoretic questions concerning elliptic curves was originally pursued mainly for aesthetic reasons. But in recent decades such questions have become important in several applied areas, including coding theory, pseudorandom number generation, and integer factorization.

In 1985, Koblitz [61] and Miller [84] independently proposed using the group of points on an elliptic curve defined over a finite field in discrete log cryptosystems. The primary advantage that elliptic curve systems have over systems based on the multiplicative group of a finite field (and also over systems based on the intractability of integer factorization) is the absence of a subexponential-time algorithm (such as those of index calculus type) that could find discrete logs in these groups. Consequently, one can use an elliptic curve group that is smaller in size while maintaining the same level of security. The result is potentially smaller key sizes, bandwidth savings, and faster implementations, features which are especially attractive for security applications where computational power and integrated circuit space are limited, such as smart cards and wireless devices.

6.1. Background on Elliptic Curves. Assume first that $\mathbb{F}$ is a field of characteristic not equal to 2 or 3. An *elliptic curve* E over $\mathbb{F}$ is an equation

$$y^2 = x^3 + ax + b, \tag{1}$$

where $a, b \in \mathbb{F}$ and $4a^3 + 27b^2 \neq 0$ (the latter condition ensures that the cubic on the right does not have multiple roots). If $\mathbb{K}$ is a field containing $\mathbb{F}$, then the set of $\mathbb{K}$-points of E, denoted $E(\mathbb{K})$, consists of all solutions $(x, y) \in \mathbb{K} \times \mathbb{K}$ of (1) together with a special point ∞ called the *point at infinity.*

It is well known that $E(\mathbb{K})$ is an (additively written) abelian group with the point ∞ serving as its identity element. The rules for group addition are summarized below.

Addition Formulas for the Curve (1). If $P = (x_1, y_1) \in E$, then $-P = (x_1, -y_1)$. If $Q = (x_2, y_2) \in E$, $Q \neq -P$, then $P + Q = (x_3, y_3)$, where

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned}$$

and

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q, \\[2ex] \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$$

There is a nice classical way—called the *chord and tangent construction*—to visualize the group law on an elliptic curve defined over the real numbers. We illustrate with the elliptic curve $y^2 = x^3 - x$, pictured in Figure 1.

To add two points P and Q, we draw a chord between them and find its third point of intersection with the curve. The point R symmetric to this point with respect to the x-axis is the sum $P + Q$. If $Q = P$, then instead of a chord we take the tangent line to the curve at P.

For k a positive integer and P a point we use the notation kP to denote P added to itself k times.

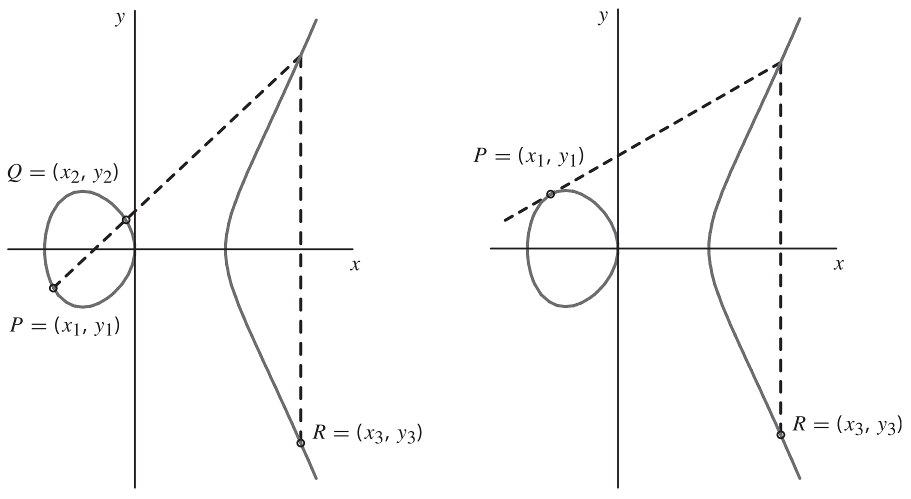
If $\mathbb{F}$ is a field of characteristic 3, then we have an equation similar to (1) but with an x^2-term which cannot be eliminated by a linear change of variables. The formulas for point addition are similar to the ones above.

Elliptic curves defined over a finite field are of two types. Most are what are called *ordinary* or *nonsupersingular* curves, but a small number are *supersingular.* If $\mathbb{F}$ is a field of characteristic 2, then a supersingular elliptic curve E is an equation

$$y^2 + cy = x^3 + ax + b,$$

where a, b, $c \in \mathbb{F}$, $c \neq 0$, together with the point at infinity ∞; and a nonsupersingular elliptic curve E is an equation

$$y^2 + xy = x^3 + ax^2 + b,$$



(a) Addition: $P + Q = R$. (b) Doubling: $P + P = R$.

Fig. 1 *Geometric addition and doubling of elliptic curve points.*

where a, $b \in \mathbb{F}$, $b \neq 0$, together with the point at infinity ∞. In both cases, $E(\mathbb{K})$ for any $\mathbb{K} \supset \mathbb{F}$ is an abelian group with the point ∞ serving as the identity. The addition formulas for the two types of curves in characteristic 2 are similar to the ones given above for equation (1).

If E is defined over a finite field $\mathbb{F}_q$, then $E(\mathbb{F}_q)$ is a finite abelian group of rank 1 or 2; in other words, either it is cyclic or else a product of two cyclic groups. We have $E(\mathbb{F}_q) \cong C_{n_1} \oplus C_{n_2}$, where C_n denotes a cyclic group of order n, n_2 divides n_1, and furthermore $n_2 | q-1$. A well-known theorem of Hasse (see [110, p. 131]) states that the cardinality $\#E(\mathbb{F}_q) = q+1-t$, where $|t| \leq 2\sqrt{q}$. We call $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$ the *Hasse interval.* The curve E is said to be *supersingular* if $t^2 = 0, q, 2q, 3q$, or $4q$; otherwise the curve is *nonsupersingular.*

When q is a power of 2, this agrees with the definition given earlier. In that case $\#E(\mathbb{F}_q)$ is odd if E is supersingular and even if E is nonsupersingular.

A result of Waterhouse [122] states that if q is a prime, then for each t satisfying $|t| \leq 2\sqrt{q}$ there exists at least one elliptic curve E defined over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = q+1-t$. If q is a power of 2, then for each even t satisfying $|t| \leq 2\sqrt{q}$ there exists at least one (nonsupersingular) elliptic curve E defined over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = q+1-t$. Schoof [105] derived a formula for the number of isomorphism classes of elliptic curves defined over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = q+1-t$ for each t satisfying $|t| \leq 2\sqrt{q}$.

Example 2 (elliptic curve over $\mathbb{F}_{11}$). *Consider the elliptic curve $E : y^2 = x^3 + 2x + 4$ defined over $\mathbb{F}_{11}$. Then $\#E(\mathbb{F}_{11}) = 17$, and $E(\mathbb{F}_{11})$ is cyclic. A generator of $E(\mathbb{F}_{11})$ is $P = (0,2)$. The points in $E(\mathbb{F}_{11})$, expressed as multiples of P, are shown below:*

$$\begin{array}{lllll}
P = (0,2), & 2P = (3,2), & 3P = (8,9), & 4P = (6,1), & 5P = (9,5), \\
6P = (7,3), & 7P = (2,4), & 8P = (10,10), & 9P = (10,1), & 10P = (2,7), \\
11P = (7,8), & 12P = (9,6), & 13P = (6,10), & 14P = (8,2), & 15P = (3,9), \\
16P = (0,9), & 17P = \infty. & & &
\end{array}$$

6.2. Elliptic Curve Cryptosystems. Discrete log cryptosystems were first described in the setting of the multiplicative group of the integers modulo a prime p. Such systems can be modified to work in the group of points on an elliptic curve.[6] For instance, the Diffie–Hellman key exchange can be adapted for elliptic curves as follows. First note that a random point on an elliptic curve E can serve as a key, since Alice and Bob can agree in advance on a method to convert it to an integer (for example, they can take the image of its x-coordinate under some agreed upon simple map from $\mathbb{F}_q$ to the natural numbers).

So suppose that E is an elliptic curve over $\mathbb{F}_q$, and P is a publicly known point on the curve. Alice secretly chooses a random integer k_A and computes the point k_AP, which she sends to Bob. Likewise, Bob secretly chooses a random k_B, computes k_BP, and sends it to Alice. The common key is $Q = k_Ak_BP$. Alice computes Q by multiplying the point she received from Bob by her secret k_A; Bob computes Q by multiplying the point he received from Alice by his secret k_B. An eavesdropper who wanted to spy on Alice and Bob would have to determine $Q = k_Ak_BP$ knowing P, k_AP, and k_BP, but not k_A or k_B. The eavesdropper's task is called the *elliptic curve Diffie–Hellman problem* (ECDHP).

It is not hard to modify this Diffie–Hellman key exchange protocol for the purpose of message transmission, using an idea of ElGamal [33]. Suppose that the set of message units has been imbedded in E in some agreed upon way, and Bob wants to send Alice a message $M \in E$. As in Diffie–Hellman, Alice has already randomly generated a secret key k_A and computed her public key k_AP. Bob now chooses another secret random integer l and sends Alice the pair of points $(lP, M + l(k_AP))$. (Notice that ElGamal encryption is *probabilistic* rather than deterministic.) To decipher the message, Alice multiplies the first point in the pair by her secret k_A and then subtracts the result from the second point in the pair.

We next describe the elliptic curve digital signature algorithm (ECDSA), which is analogous to the DSA in section 5.2.

ECDSA Key Generation. E is an elliptic curve defined over $\mathbb{F}_q$, and P is a point of prime order N in $E(\mathbb{F}_q)$; these are system-wide parameters. For simplicity, we shall suppose that q is a prime, although the construction can easily be adapted to a prime power q as well. Each user Alice constructs her keys by selecting a random integer x in the interval $[1, N-1]$ and computing $Q = xP$. Alice's public key is Q; her private key is x.

ECDSA Signature Generation. To sign a message having hash value H, $0 < H < N$, Alice does the following:

1. She selects a random integer k in the interval $[1, N-1]$.
2. She computes $kP = (x_1, y_1)$ and sets r equal to the least nonnegative residue of x_1 mod N (where x_1 is regarded as an integer between 0 and $q-1$). (Note: If $r = 0$, then she must go back to step 1 and select another k.)
3. She computes k^{-1} mod N and sets s equal to the least nonnegative residue of $k^{-1}(H + xr)$ mod N. (Note: If $s = 0$, then she must go back to step 1.)

The signature for the message is the pair of integers (r, s).

ECDSA Signature Verification. To verify Alice's signature (r, s) on a message, Bob should do the following:

1. Obtain an authenticated copy of Alice's public key Q.

[6]Or, indeed, in any finite group. However, such a discrete log cryptosystem is worth considering only if there is reason to believe that it is safe from attack; see section 6.3.

2. Verify that r and s are integers in the interval $[1, N-1]$, and compute the hash value H of the message.
3. Compute $u_1 = s^{-1}H \bmod N$ and $u_2 = s^{-1}r \bmod N$.
4. Compute $u_1P + u_2Q = (x_0, y_0)$ and, regarding x_0 as an integer between 0 and $q-1$, set v equal to the least nonnegative residue of $x_0 \bmod N$.
5. Accept the signature if and only if $v = r$.

Notice that if Alice generated her signature correctly, then $u_1P + u_2Q = (u_1 + xu_2)P = kP$ because $k \equiv s^{-1}(H + xr) \pmod N$, and so $v = r$.

To obtain a security level similar to that of DSA, the parameter N should have about 160 bits. If this is the case, then DSA and ECDSA signatures have the same bitlength (320 bits). The main advantage of ECDSA over DSA is that operations are performed in a much smaller field $\mathbb{F}_q$.

Instead of using the same elliptic curve for everyone, we could fix the underlying finite field $\mathbb{F}_q$ for all users and let each select her own elliptic curve E and point $P \in E(\mathbb{F}_q)$. In this case, the coefficients of the defining equation for E, the point P, and the order N of P must also be included in a user's public key. If the underlying field $\mathbb{F}_q$ is fixed, then hardware or software can be built to optimize computations in that field. At the same time, there are an enormous number of choices of elliptic curves E over the fixed $\mathbb{F}_q$.

6.3. Security. The basis for the security of elliptic curve cryptosystems such as ECDSA and ElGamal encryption is the apparent intractability of the following *elliptic curve discrete logarithm problem* (ECDLP): Given an elliptic curve E defined over $\mathbb{F}_q$, a point $P \in E(\mathbb{F}_q)$ of order N, and a point $Q \in E(\mathbb{F}_q)$, determine the integer x, $0 \le x \le N-1$, such that $Q = xP$, provided that such an integer exists.

The Pohlig–Hellman algorithm [93] reduces the determination of x to the determination of x modulo each of the prime factors of N. Hence, in order to achieve the maximum possible security level, N should be prime. The best general-purpose algorithm known to date for the ECDLP is the Pollard-ρ method [94], which takes fewer than $N^{1/2+\epsilon} = 2^{(1/2+\epsilon)n}$ steps if N is an n-bit prime. We now describe this method.

Given P and Q in a cyclic order-N subgroup $G \subset E(\mathbb{F}_q)$, we want to find x such that $Q = xP$. First, partition $G = S_1 \cup S_2 \cup S_3$ randomly into three sets of roughly equal size. Select $X_0 = a_0P + b_0Q$ with random a_0, b_0.

Construct a recursive sequence of points,

$$X_{i+1} = \begin{cases} Q + X_i & \text{if } X_i \in S_1, \\ 2X_i & \text{if } X_i \in S_2, \\ P + X_i & \text{if } X_i \in S_3, \end{cases}$$

and recursive sequences of integers,

$$a_{i+1} = \begin{cases} a_i & \text{if } X_i \in S_1, \\ 2a_i & \text{if } X_i \in S_2, \\ a_i + 1 & \text{if } X_i \in S_3 \end{cases}$$

and

$$b_{i+1} = \begin{cases} b_i + 1 & \text{if } X_i \in S_1, \\ 2b_i & \text{if } X_i \in S_2, \\ b_i & \text{if } X_i \in S_3. \end{cases}$$

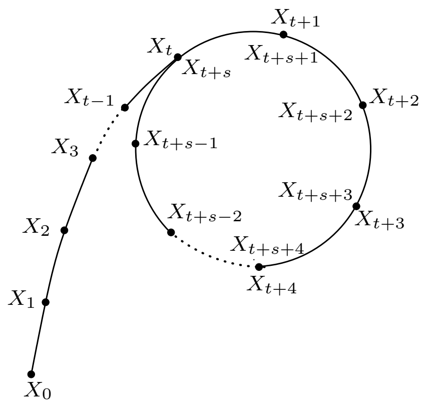


Fig. 2 *ρ-like shape of the sequence $\{X_i\}$ in the Pollard-ρ method, where t = tail length and s = cycle length.*

Then $X_i = a_iP + b_iQ$ for all i. The idea is that this sequence eventually becomes periodic. Figure 2 shows how the ρ-method got its name.

Once we find i and j such that $X_i = X_j$ we have

$$X_i = a_iP + b_iQ = (a_i + xb_i)P = X_j = (a_j + xb_j)P,$$

and hence

$$a_i + xb_i \equiv a_j + xb_j \pmod{N},$$

from which $x \bmod N$ can immediately be determined (except in the very unlikely event that $b_i \equiv b_j \pmod{N}$).

In order to greatly reduce storage, in practice one looks for a match between X_i and X_{2i}. This slightly increases the running time, but reduces the storage almost to zero. It was a crucial observation (due to Pollard) that the search for a match between X_i and X_j—which would require storage of order $O(\sqrt{N})$—can be replaced for a search for a match between X_i and X_{2i}. Otherwise, the ρ-method would have been no better than an earlier deterministic matching method of D. Shanks called "baby step–giant step" that takes roughly the same amount of time and requires $O(\sqrt{N})$ storage.

Assuming that the above map from X_i to X_{i+1} behaves like a random mapping, a match can be found by the time i reaches $O(\sqrt{N})$. Much research has been devoted to improving the Pollard-ρ method (see, for example, [116]). The general form of the estimate for the number of steps remains $O(\sqrt{N})$ even after all the modifications. Thus, the aim of this work is to reduce the constant in $O(\sqrt{N})$.

For certain elliptic curves ECDLP algorithms have been found that are faster than Pollard-ρ. The Weil and Tate pairings can be used to embed the group $E(\mathbb{F}_q)$ in the multiplicative group of the field $\mathbb{F}_{q^k}$ for some integer k (see [80] and [36]). This reduces the ECDLP in $E(\mathbb{F}_q)$ to the DLP in $\mathbb{F}_{q^k}^*$. A necessary condition for a cyclic subgroup of $E(\mathbb{F}_q)$ of order N to be embedded in $\mathbb{F}_{q^k}^*$ is that N divide $q^k - 1$.

Once the ECDLP has been replaced by the DLP in $\mathbb{F}_{q^k}^*$, we can hope to use an index calculus algorithm with subexponential running time $2^{n^{1/3+\epsilon}}$, where $n = \log_2(q^k)$. See Coppersmith [29] for the case of even q, and Gordon [46] and Schirokauer

[103] for the case when q is a prime and $k = 1$. No algorithm with this running time is known when q is odd and $k > 1$, but we adopt the "optimistic" supposition that the above time estimate can be achieved for the DLP in $\mathbb{F}_{q^k}$ for all q and $k \geq 1$. Even with this supposition, k must be less than $\log^2 q$, since otherwise the index calculus algorithm for $\mathbb{F}_{q^k}$ will take fully exponential time in $\log q$.

For the very special class of supersingular curves, it is known that $k \leq 6$. For these curves the reduction using the Weil and Tate pairing gives a subexponential-time algorithm for the ECDLP. However, a randomly generated elliptic curve has an exponentially small probability of being supersingular; and, as shown in [64] (see also [6]), for most randomly generated elliptic curves we have $k > \log^2 q$.

In addition, if the elliptic curve is defined over a prime field $\mathbb{F}_p$ and $E(\mathbb{F}_p)$ happens to have cardinality exactly equal to p, then Satoh and Araki [101], Semaev [106], and Smart [113] showed how to imbed the elliptic curve group into the additive group of integers mod p and thereby solve the ECDLP very quickly.

No subexponential-time algorithm is known for the ECDLP except for the special classes discussed above. Miller [84] (see also [112]) discusses the index calculus method (see section 3.2) as it might apply to elliptic curve groups. He comments that unlike in the case of $\mathbb{F}_q^*$, where there are natural candidates for the factor base (prime numbers of small size or small degree irreducible polynomials), there appear to be no likely candidates in $E(\mathbb{F}_q)$. When q is a prime the most natural ones might come from reduction modulo q of points of small height in $\widetilde{E}(\mathbb{Q})$, $\mathbb{Q}$ the field of rational numbers, for some "lifting" $\widetilde{E}$ of E. (The height of a point is related to the number of bits needed to represent the point.) However, Miller points out that there are very few points of small height in $\widetilde{E}(\mathbb{Q})$. Furthermore, even if such a factor base can be found, finding an efficient method for lifting a point in $E(\mathbb{F}_q)$ to a point in $\widetilde{E}(\mathbb{Q})$ looks hopeless.

In 1998, Silverman [111] proposed a clever variant on index calculus attacks. His method reversed the order of the stages in index calculus, and for that reason he called it "xedni calculus" ("index" spelled backwards). This technique to solve the ECDLP was analyzed in [57] and found to be far slower than the Pollard-ρ method.

In certain cases when $q = 2^m$ with composite extension degree $m = ln$, it is possible to solve the ECDLP faster by means of the so-called *Weil descent* method than by Pollard-ρ. The idea of Weil descent, which is due to G. Frey, is to convert the DLP on an elliptic curve over $\mathbb{F}_{2^{ln}}$ to the DLP on the Jacobian of a genus-g curve defined over $\mathbb{F}_{2^l}$. This approach has been investigated systematically in [40, 81, 58, 78, 51, 82]. The vast majority of elliptic curves cannot be attacked using these methods, and one can avoid Weil descent entirely by working over prime fields or fields of 2^m elements with m prime.

Recently Gaudry [39] used an index-calculus approach to solve the ECDLP on a curve defined over a field of order $q = p^m$ where m is composite. His method is asymptotically faster than the Pollard-ρ method when m is divisible by a small number greater than 2. For example, if $3|m$, then the running time of Gaudry's algorithm is $O(p^{10m/21+\epsilon})$, whereas the Pollard-ρ method has a running time of $O(p^{m/2+\epsilon})$.

Strictly speaking, the security of elliptic curve cryptographic systems is usually based on the assumed intractability of a problem that is slightly weaker than the ECDLP. For example, security of the elliptic curve Diffie–Hellman key agreement protocol relies on the presumed intractability of the elliptic curve Diffie–Hellman problem (ECDHP; see section 6.2). Clearly ECDHP polynomial-time reduces to ECDLP. Boneh and Lipton [19] proved a partial converse: if the ECDLP cannot be solved in subexponential time, then neither can ECDHP.

6.4. Selecting an Appropriate Elliptic Curve. By an "appropriate" elliptic curve, we mean an elliptic curve E defined over a finite field $\mathbb{F}_q$ where the ECDLP in $E(\mathbb{F}_q)$ resists all known attacks. In particular, the following conditions should be satisfied:

(i) To resist the Pollard-ρ attack, $\#E(\mathbb{F}_q)$ should be divisible by a sufficiently large prime N (for example, $N > 2^{160}$).

(ii) To resist the Weil and Tate pairing attacks, N should not divide $q^k - 1$ for all $1 \le k \le C$, where C is large enough so that it is computationally infeasible to find discrete logarithms in $\mathbb{F}^*_{q^C}$. ($C = 20$ suffices in practice.)

(iii) If q is prime, then $\#E(\mathbb{F}_q)$ must not equal q.

Below we give an overview of three techniques for selecting an appropriate curve.

Using Hasse's Theorem. Here one uses a curve over $\mathbb{F}_q$ that is actually defined over a much smaller subfield $\mathbb{F}_{q_0}$.

If E is an elliptic curve defined over $\mathbb{F}_{q_0}$, then E can be viewed as an elliptic curve over any extension $\mathbb{F}_{q_0^m}$ of $\mathbb{F}_{q_0}$, and $E(\mathbb{F}_{q_0})$ is a subgroup of $E(\mathbb{F}_{q_0^m})$. Hasse's theorem enables one to compute $\#E(\mathbb{F}_{q_0^m})$ from $\#E(\mathbb{F}_{q_0})$ as follows. Let $t = q_0 + 1 - \#E(\mathbb{F}_{q_0})$. Then $\#E(\mathbb{F}_{q_0^m}) = q_0^m + 1 - \alpha^m - \beta^m$, where α and β are complex numbers determined from the factorization of $1 - tT + q_0T^2 = (1 - \alpha T)(1 - \beta T)$.

This method is most commonly used when q is a power of 2. In that case we first pick an elliptic curve over a small field $\mathbb{F}_{2^\ell}$, compute $\#E(\mathbb{F}_{2^\ell})$ (which is easy to do by exhaustive counting), and then use Hasse's theorem to determine $\#E(\mathbb{F}_q)$ for $q = 2^{m\ell}$ for m in an appropriate range (in practice we want $m\ell > 160$). If conditions (i) and (ii) above (with $q = 2^{m\ell}$) are not satisfied for any m in the desired range, then another curve is selected and the process is repeated. Since the number of elliptic curves over $\mathbb{F}_{2^\ell}$ is relatively small, it may not be possible to construct the desired curve using this method.

Koblitz [65] observed that if one uses k of small Hamming weight (that is, its binary expansion has mostly zero-bits) when computing kP, then one gets doubling of points "almost 3/4 for free" for some *anomalous* curves over $\mathbb{F}_{2^l}$.[7] In [114] Solinas showed how to compute kP very efficiently for arbitrary k on an anomalous curve defined over $\mathbb{F}_2$.

The Complex Multiplication Method. The method of complex multiplication (CM) allows the choice of an elliptic curve order *before* the curve is explicitly constructed. Thus, orders can be generated so as to satisfy conditions (i)–(iii); a curve is constructed only when these conditions are met. For elliptic curves over $\mathbb{F}_p$, the CM method is also called the *Atkin–Morain method* (see [86]); over $\mathbb{F}_{2^m}$, it is called the *Lay–Zimmer method* (see [70]).

The CM method generates elliptic curves of a special sort: the values of $|t|$ are very close to their upper limit $2\sqrt{q}$, and the curves have complex multiplication by a small discriminant. While it is conceivable that this feature may render these curves cryptographically insecure, it must be stressed that no attack is currently known that takes advantage of this structure.

Choosing a Curve at Random. Another approach to selecting an appropriate elliptic curve E over $\mathbb{F}_q$ is to select random coefficients $a, b \in \mathbb{F}_q$ of the equation of E (subject to the constraint that $4a^3 + 27b^2 \ne 0$ if q is odd, and $b \ne 0$ if q is even). One then computes $\#E(\mathbb{F}_q)$ and factors this number. This process is repeated until conditions (i)–(iii) are satisfied.

[7] An elliptic curve over $\mathbb{F}_q$ is said to be *anomalous* if $t = 1$ or, equivalently, if $\#E(\mathbb{F}_q) = q$.

In the case of elliptic curves over a prime field $\mathbb{F}_q$, a theorem of Lenstra [74] shows that if the coefficients a and b are selected uniformly at random, then the orders of the resulting elliptic curves are roughly uniformly distributed in the Hasse interval $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$. Similar results for the case of elliptic curves over $\mathbb{F}_{2^m}$ can be deduced from the work of Waterhouse [122] and Schoof [105].

With condition (i) in mind, we shall say that a positive integer u is *B-almost prime* if it is divisible by a prime greater than u/B. For fixed B and large q, it is reasonable to assume that the probability of B-almost primality of the order of a randomly chosen elliptic curve over $\mathbb{F}_q$ is roughly equal to the probability of B-almost primality of a random integer of the same order of magnitude as q. If q is even, then one considers random *even* integers of the same order of magnitude as q. For example, if $q = 2^{173}$ and we want an elliptic curve whose order is divisible by a prime $N > 2^{160}$ (so $B = 2^{13}$), we expect to try about 14 curves before finding one whose order is B-almost prime.

In 1985, Schoof [104] found a polynomial-time algorithm for computing the number of $\mathbb{F}_q$-points on an elliptic curve defined over $\mathbb{F}_q$ in the case when q is odd; the algorithm was later extended to the case of even q in [63]. Schoof's algorithm has a worst-case running time of $O((\log q)^8)$ bit operations and is rather inefficient in practice. However, in recent years much faster point-counting algorithms have been developed, including the Schoof–Elkies–Atkin (SEA) algorithm (see [13, Chapter VII] and [56]) for elliptic curves over prime fields, and Satoh's algorithm [100, 35, 102] and the AGM method (see [38]) for elliptic curves over characteristic two finite fields.

6.5. A Signature Scheme Based on the Weil Pairing.

We conclude this section by describing a signature scheme that, unlike ECDSA, depends in an essential way on properties of elliptic curves that do not have analogues in the multiplicative group of a finite field. It is one of the very few elliptic curve cryptosystems that were not developed by analogy with earlier cryptosystems.[8] The signature scheme uses elliptic curves that have the unusual property that, while the Diffie–Hellman problem (see section 6.2) is hard, the *decisional* Diffie–Hellman problem—the problem of determining whether or not the discrete log of a point is equal to the product of the discrete logs of two other points—is easy.

Let us consider the simple equation

$$y^2 = x^3 - x$$

over the prime field $\mathbb{F}_p$, $p \neq 2$. When $p \equiv 3 \pmod 4$ this curve is supersingular—it is an easy exercise to show that the curve has exactly $p+1$ points (including ∞). We get a nonsupersingular curve when $p \equiv 1 \pmod 4$. In the nonsupersingular case, a formula for the number of points was discovered by Gauss. Namely, write $p = A^2 + B^2$ as a sum of two squares, with A and B chosen so that A is odd and $A+B \equiv 1 \pmod 4$. (This is a very easy computational task.) Then the curve has $p+1-2A$ points. If this number is almost prime (that is, equal to a prime number times a small factor), then the curve is suitable for the ECDSA and other cryptographic applications.

But it is the other case—when $p \equiv 3 \pmod 4$—that can be used for the signature scheme described below. Since -1 is a nonsquare in $\mathbb{F}_p$, by adjoining a square root i of -1 we get the field of p^2 elements $\mathbb{F}_{p^2}$. It is not hard to see that our curve has

[8]Other cryptographic protocols that also make crucial use of the Weil and Tate pairings include the three-party one-round key agreement protocol of Joux [60] and the identity-based public-key encryption scheme of Boneh and Franklin [18]. The idea of this type of use of the pairings first appeared in [99].

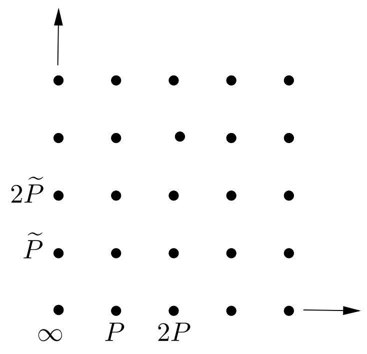


Fig. 3 *N^2 points of order dividing N in $E(\mathbb{F}_{p^2})$. The horizontal axis is in $E(\mathbb{F}_p)$.*

$(p+1)^2$ points over this larger field. Moreover, if $P = (u, v) \neq \infty$ is an $\mathbb{F}_p$-point, then the map $P \mapsto \widetilde{P} = (-u, iv)$ takes P to an $\mathbb{F}_{p^2}$-point having the same order as P. (Notice that if (u, v) satisfies the equation $y^2 = x^3 - x$, then so does $(-u, iv)$.) If P has order N, then there are N^2 $\mathbb{F}_{p^2}$-points of order dividing N, namely, all linear combinations $kP + \ell\widetilde{P}$, $0 \leq k, \ell < N$ (see Figure 3).

There is a bilinear pairing $\langle\ ,\ \rangle$, called the *Weil pairing*, on this set of N^2 points of order N, such that $\langle P, \widetilde{P}\rangle = \zeta$, where ζ is a primitive Nth root of unity in $\mathbb{F}_{p^2}$. (Notice that since N divides $p+1$, it also divides p^2-1, and so $\mathbb{F}_{p^2}$ contains a primitive Nth root of unity.) This pairing is not hard to compute [85].

As mentioned in section 6.3, in the early 1990s it was noticed that in this situation it is easy to transform the ECDLP to the DLP in the field $\mathbb{F}_{p^2}$. Namely, if $Q = xP$ is the $\mathbb{F}_p$-point whose discrete logarithm x you want to find, it follows from the bilinearity of the Weil pairing that $\langle Q, \widetilde{P}\rangle = \langle xP, \widetilde{P}\rangle = \zeta^x$. Thus, the problem of finding the discrete log of Q to the base P on the curve is equivalent to the problem of finding the discrete log of $\langle Q, \widetilde{P}\rangle$ to the base ζ in $\mathbb{F}_{p^2}$. This means that for an n-bit prime p the ECDLP on this curve can be solved in time $2^{n^{1/3+\epsilon}}$ using the number field sieve instead of the Pollard-ρ method, which would take time $2^{(\frac{1}{2}+\epsilon)n}$. (Here we are again supposing that the number field sieve will be improved for extension fields such as $\mathbb{F}_{p^2}$ so as to achieve the same order of running time as for prime fields; currently it is slower on extension fields.) For adequate security one would need to use roughly 500-bit primes p rather than 160-bit primes p as in the case of nonsupersingular elliptic curves. For this reason one might think that these supersingular curves are of no use in cryptography.

But interestingly, it is precisely the supersingular curves that are needed for the "short signature" scheme devised by Boneh, Lynn, and Shacham [20]. This signature scheme relies in an essential way on the Weil pairing, which is not an analogue of anything that is available for the multiplicative group of a finite field.

Here is how the Boneh–Lynn–Shacham signature scheme works on the curve $y^2 = x^3 - x$ over $\mathbb{F}_p$, $p \equiv 3 \pmod 4$. Assume that p has been chosen so that the DLP in $\mathbb{F}_{p^2}^*$ is intractable. However, p is not so large as to make it difficult to perform arithmetic and compute the Weil pairing in $\mathbb{F}_{p^2}$. Suppose that Alice wants to sign a message to Bob that has hash value H, which is taken to be a point in the subgroup of $E(\mathbb{F}_p)$ generated by P (unlike in the ECDSA, where the hash value is an integer less than N). As in other elliptic curve systems, Alice's secret key is a random integer x, and her public key is the multiple $Q = xP$ of the base point P. Then Alice's signature is simply the point $S = xH$. To verify the signature, Bob computes $\widetilde{Q}$ (the image of Q under the map that takes a point (u, v) to $(-u, iv)$) and the two pairings $\langle H, \widetilde{Q}\rangle$ and $\langle S, \widetilde{P}\rangle$; and he accepts the signature if these two elements of $\mathbb{F}_{p^2}$ are equal. Because of the bilinearity of the Weil pairing, if Alice formed the signature correctly, then both are equal to $\langle H, \widetilde{P}\rangle^x$. Bob accepts the signature because he is confident that only Alice would have been able to find the point S whose discrete log to the base H is equal to the discrete log of Q to the base P.

While the implementation of the Boneh–Lynn–Shacham signature requires only arithmetic in $\mathbb{F}_{p^2}$, forging a signature requires the ability to solve the Diffie–Hellman problem on the elliptic curve. Namely, given P, Q, and H, the forger must find a point S whose discrete log to the base P is equal to the product of the discrete logs of H and Q. The only way known to do this is to solve the DLP in $\mathbb{F}_{p^2}^*$, and we are assuming that that is not feasible.

There is a relatively small set of elliptic curves on which the Boneh–Lynn–Shacham signature scheme can be implemented. If the group of points in which we are working has order N, then the Weil pairing takes values in the Nth roots of unity, which lie in an extension field $\mathbb{F}_{q^k}$ such that $N|q^k - 1$. For most elliptic curves, the multiplicative order k of q modulo N has the same order of magnitude as N, and so it is infeasible to do arithmetic in the gigantic extension field $\mathbb{F}_{q^k}$—in fact, in practice it is impossible even to store an element of such a field. In contrast, for the supersingular curve $y^2 = x^3 - x$ over $\mathbb{F}_p$ with $p \equiv 3 \pmod 4$, we have $k = 2$, and the Weil pairing computations take place in $\mathbb{F}_{p^2}$.

7. Other Systems Based on Discrete Logarithms.

In principle, one can construct a public-key cryptographic system based on the DLP in any group, provided that the DLP is difficult enough to provide security. In addition to the multiplicative group of finite fields and elliptic curves defined over finite fields, several other groups have been considered.

7.1. Hyperelliptic and Other Curves.

For a curve defined over the complex numbers, the genus can be interpreted as the number of "handles" in the corresponding surface. An elliptic curve can be represented as a torus (donut-shaped surface), and a genus-5 curve has the appearance shown in Figure 4.

If the genus g of a curve is greater than 1, then the points on the curve do not have a natural group law. However, one can consider formal sums of points modulo the equivalence relation determined by the divisors of functions. These *divisor classes* are the elements of the *Jacobian* of the curve, which has a natural group structure. These groups generalize the group of points on an elliptic curve.

In general, it is a complicated matter to find good sets of divisor class representatives and efficient algorithms for the group law on the divisor classes. However, *hyperelliptic curves*—those whose equation has a quadratic polynomial in y on the left side and a polynomial in x of degree $2g+1$ on the right—are much easier to work with. It was hyperelliptic curve Jacobians that were proposed for use in cryptography in 1989 [62].

The Jacobian of a hyperelliptic curve is closely analogous to the ideal class group of an imaginary quadratic number field. In fact, the rules for the group law on a hyperelliptic Jacobian are very similar to the classical rules developed by Gauss for composition of binary quadratic forms. This is why hyperelliptic curves are convenient to work with. On the other hand, the similarity with class groups also explains

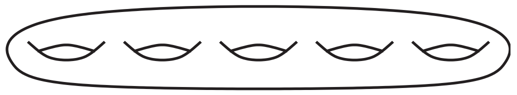


Fig. 4 *A Riemann surface with* 5 *handles.*

why it turned out that for high-genus curves one has subexponential-time index calculus algorithms for the DLP on these groups (see [47, 1]). Even when the genus is just moderately large, namely, $g \geq 3$, Gaudry [37] and Thériault [117] showed that index calculus methods to solve the DLP are asymptotically faster than the Pollard-ρ method. The running time for Gaudry's algorithm on the Jacobian of a genus-g curve over $\mathbb{F}_q$ is of order $q^{2g/(g+1)+\epsilon}$, whereas Thériault's algorithm has a running time of $O(q^{(4g-2)/(2g+1)+\epsilon})$. Since the running time of the Pollard-ρ method is $q^{g/2+\epsilon}$, it follows that for $g \geq 3$ a hyperelliptic cryptosystem would require greater keylengths than elliptic curve cryptosystems for the same level of security. Hyperelliptic cryptosystems for $g = 2$ have no known security disadvantage compared to elliptic curve systems. While there have been relatively few practical implementations of hyperelliptic cryptosystems, recent work (see [5, 92]) suggests that genus-2 hyperelliptic curves are only slightly less efficient that their elliptic curve counterparts.

7.2. Class Groups. We mentioned that the Jacobian group of a hyperelliptic curve is in some ways similar to the ideal class group of a quadratic number field. Such ideal class groups themselves have been studied for use in cryptographic protocols in a series of papers by Buchmann, Williams, and others [26, 25, 11, 24]. In the case of imaginary quadratic fields they use the usual class group. However, in the case of real quadratic number fields the class group is not suitable, so they instead work with what they call an *infrastructure*, where the composition law gives only an approximation to a group, not a true group.

7.3. XTR. A special case of the Diffie–Hellman system, proposed by Lenstra and Verheul [73] (see also [45]), has aroused considerable interest; they called their system XTR, which stands for "efficient compact subgroup trace representation." As in section 5, one works in the subgroup G of prime order N in the multiplicative group of the finite field $\mathbb{F}_q$. We take $q = p^6$, where p is a prime that is $\equiv 2 \pmod 3$, and N is chosen so that it divides the factor $p^2 - p + 1$ of $p^6 - 1$ (in which case the subgroup G is not contained in any proper subfield of $\mathbb{F}_q$). In practice N should have roughly the same bitlength as p; it is recommended that both have about 170 bits, in which case q has over 1000 bits.

It is not hard to set up the parameters N and p. For example, if r is chosen so that both $r^2 - r + 1$ and $r^2 + 1$ are prime, then one can set $N = r^2 - r + 1$ and $p = N + r = r^2 + 1$ (in which case obviously $p^2 - p + 1 \equiv r^2 - r + 1 \equiv 0 \pmod N$).

A crucial innovation in [73] is that the elements of G can be represented using only $2 \log_2 p$ bits rather than $\log_2 q = 6 \log_2 p$ bits. Moreover, exponentiation can be done directly with these short representations. This leads to greater efficiency and short key sizes comparable to those in elliptic curve cryptography. Since the mathematics is relatively simple (as in RSA), Lenstra and Verheul feel that their system "may be regarded as the best of two worlds, RSA and ECC [elliptic curve cryptography]" [73, p. 2].

The group used in XTR is closely related to the group of points on a certain supersingular elliptic curve defined over $\mathbb{F}_{p^2}$. In fact, the Weil pairing embedding in

[80] maps this elliptic curve group to precisely the XTR group. Since the publication of [80], the existence of this reduction map from a supersingular elliptic curve to the multiplicative group of a finite field has traditionally been considered to be a security weakness arguing against the use of supersingular curves. Thus, some cryptographers have doubted the wisdom of using a system that is closely related to a version of elliptic curve cryptography that had been rejected. However, Verheul [120] has pointed out that the Weil pairing embedding reduces the elliptic curve DLP to the XTR group DLP, not vice versa; and he believes that the DLP in their group is likely to be strictly harder than the DLP on the curve. In addition, the Boneh–Lynn–Shacham signature scheme described in section 6.5 shows that supersingular elliptic curves should not be automatically ruled out for use in cryptography. By the same token, XTR also deserves serious study and consideration.

7.4. Connection between the DLP and Integer Factorization. At first glance it might seem that RSA has no relation to cryptographic systems that are based on the DLP. However, the integer factorization problem and the DLP are more directly related than one might have thought. Suppose that we want to factor an RSA modulus $N = p \cdot q$, and suppose that we have an algorithm $\mathcal{A}$ that finds discrete logarithms in the multiplicative group $G = (\mathbb{Z}/N\mathbb{Z})^*$ of integers modulo N that are prime to N. We claim that, with little additional effort, we can use $\mathcal{A}$ to find the factors of N.

Namely, let g be a random integer in G. Let k be the order of g modulo N, and let k_1 and k_2 be the orders of g modulo the two prime factors of N; note that $k = \mathrm{lcm}(k_1, k_2)$. Because N has not yet been factored, we do not know the values of k, k_1, or k_2.

Choose an exponent m that is significantly larger than N—for example, of order N^2—and compute $y = g^m$ in G. Now apply the algorithm $\mathcal{A}$ to find a discrete logarithm x of y to the base g in G. Since x and m are both discrete logs of y to the base g, it follows that $m - x$ is a multiple of k. Because m was chosen to be large, we may assume that $m - x \neq 0$.

Let 2^ℓ be the highest power of 2 dividing $m - x$. We now compute $g^{(m-x)/2^\ell}$, $g^{(m-x)/2^{\ell-1}}, \ldots, g^{(m-x)/2}$, $g^{m-x} = 1$, and we let u denote the last number in this sequence that is not equal to 1 modulo N (we take $u = 1$ if $g^{(m-x)/2^\ell} = 1$). Let 2^i be the highest power of 2 dividing k_1, and let 2^j be the highest power dividing k_2. It is easy to see that if $i \neq j$, then u will be a nontrivial square root of 1: $u^2 \equiv 1 \pmod{N}$, $u \not\equiv \pm 1 \pmod{N}$. In that case we can immediately factor N by taking $\gcd(N, u \pm 1)$.

On the other hand, if $i = j$, then we choose a different value of g and start over. One can check that for randomly chosen g there is at least a 50% chance that $i \neq j$. This gives us a probabilistic method of factoring N, given an algorithm $\mathcal{A}$ for the DLP in $(\mathbb{Z}/N\mathbb{Z})^*$.

In the two cases of xedni calculus (see section 6.3) and quantum computation (see section 10), this reduction of integer factorization to the DLP in $(\mathbb{Z}/N\mathbb{Z})^*$ has been used to show that a technique originally developed for the DLP can also be used to factor integers.

8. NTRU. In 1996, a cryptosystem developed by three mathematicians at Brown University was presented at the "rump session" of the annual Crypto conference in Santa Barbara (see [52]). It is fundamentally different from both RSA and elliptic curve cryptography, and it has some efficiency advantages over them. On the other hand, a history of successful attacks on various versions of NTRU makes many people hesitant to endorse its use. Whether or not it is ever approved for practical use by the industrial standards groups, its construction is clever, interesting, and worthy of careful study. We shall describe the version of the NTRU encryption scheme in [52].

Three integers N, p, and q are public parameters for the system. Here p is odd, prime to q, and much smaller than q. For example, the values $N = 107$, $p = 3$, $q = 64$ were suggested in the original proposal at Crypto '96. In general, it is believed that the larger N is, the harder the system is to attack.

We work with N-tuples of integers, regarded as polynomials modulo $X^N - 1$. This means that two such N-tuples $f = \sum_{i=0}^{N-1} f_i X^i$ and $g = \sum_{i=0}^{N-1} g_i X^i$ are multiplied using the convolution $f * g = \sum_{k=0}^{N-1} (f * g)_k X^k$ with $(f * g)_k = \sum_{i+j \equiv k \pmod N} f_i g_j$. We shall be reducing the coefficients of such N-tuples modulo p and also modulo q. Let $\mathcal{L}(d, d')$ denote the set of polynomials of degree less than N having d coefficients equal to 1, d' coefficients equal to -1, and the rest equal to 0. Let $S_f = \mathcal{L}(d_1, d_1 - 1)$, $S_g = \mathcal{L}(d_2, d_2)$, and $S_\varphi = \mathcal{L}(d_3, d_3)$ for some choice of three integers $d_i < N/2$. A message unit M will be an N-tuple of integers between $-(p-1)/2$ and $(p-1)/2$ (also regarded as a polynomial of degree less than N).

To form her private key Alice randomly selects $f \in S_f$ and $g \in S_g$, where f must have inverses modulo p and modulo q. Let f_p and f_q, respectively, denote such inverses. The polynomials f and g are kept secret. Alice's public key consists of the polynomial $h = f_q * g \bmod q$.

To encipher a message unit M, Bob randomly generates $\varphi \in S_\varphi$ and computes $C = p\varphi * h + M \bmod q$. (This is another example of probabilistic rather than deterministic encryption.) When Alice receives C, she uses her secret f to compute $a \equiv f * C \pmod q$, where she chooses the coefficients of a in the interval from $-q/2$ to $q/2$. She then reduces these coefficients (regarded as ordinary integers) modulo p and computes $f_p * a \bmod p$. We claim that with high probability this is the message M. To see this, note that modulo q we have $a \equiv f * (p\varphi * h + M) \equiv f * p\varphi * f_q * g + f * M \equiv p\varphi * g + f * M$. If the parameters were chosen carefully, usually all of the coefficients of the polynomial $p\varphi * g + f * M$ are between $-q/2$ and $q/2$, in which case the mod q value of this polynomial is actually the true value. But if Alice knows the true value of $p\varphi * g + f * M$, she need only reduce modulo p to get $f * M \bmod p$, and then apply f_p to get $f_p * f * M \equiv M \pmod p$.

The "moderate security" version of NTRU that was presented at Crypto '96 (with the above values for N, p, q) was broken by Coppersmith and Shamir [30], who used lattice-basis reduction methods [71] to find short vectors in a lattice that arises when one tries to find the plaintext from the NTRU ciphertext and public key. Subsequently there have been other successful attacks on certain versions of NTRU (see, for example, [59] and [54]). In response, the inventors of NTRU have adopted new parameters and padding schemes that they believe can resist all known attacks. On their web site (www.ntru.com) they offer valuable cash prizes to anyone who can break their "challenges" with N-parameter equal to 251, 347, and 503.[9]

In the first few years after NTRU was proposed, a common criticism was that it did not have a signature scheme. In 2001 an NTRU signature scheme was proposed at Eurocrypt [53], but both that scheme and a revised version were broken soon after (see [42, 43]). A new revised signature scheme is now available on the NTRU web site, but at present the prospects for commercial adoption of an NTRU-based signature scheme are unclear.

[9]For the integer factorization challenges posed by RSA, see www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html. For the ECDLP challenges posed by Certicom (the main marketer of elliptic curve cryptography), see www.certicom.com.

9. Cryptosystems Based on Other Algebraic Structures. For certain algebraic structures natural questions arise that seem to be very difficult to answer. Are two given elements in a nonabelian group conjugate to one another? Can a given multivariate polynomial be expressed as a sum of polynomial multiples of a given set of polynomials? We now look at cryptographic systems whose security relies upon the presumed intractability of such problems.

9.1. Noncommutative Structures—Braid Groups. Let G be a group. This means that we have an associative operation $\circ$ on elements of G that has an identify element and inverses. (Often the symbol $\circ$ is suppressed, and we write $g \circ h$ as gh.) This operation is not necessarily commutative, and in this subsection we suppose that G is a nonabelian group. If the group G arises in a natural way, or if it is given abstractly by generating elements and relations that they satisfy, it might be very difficult to determine whether two elements described in different ways are equal. This problem—called the *word problem* in group theory—is known to be algorithmically undecidable [87]. Another question that in general is very difficult is the *conjugacy problem*, which asks whether, given two group elements a and b, there exists $c \in G$ such that $cac^{-1} = b$. The *conjugacy search problem* supposes that a and b are conjugate and asks us to find an element c such that $cac^{-1} = b$.

Some early attempts to construct a cryptosystem using the word problem were due to Magyarik and Wagner [75] and to Van et al. [118]. In the latter paper a one-way function was constructed by successively inserting relations in the middle of words, starting from a word formed by two elements of G. But the resulting system was too cumbersome to be practical (the same was true of the cryptosystem in [75]).

A more recent and more extensively studied cryptographic system based on the structure of nonabelian groups is the braid group cryptosystem [3, 4]. Following [12], we first describe the classical braid group. Let E denote the Euclidean plane, and let F_nE denote the set of n-tuples of *distinct* points of E:

$$F_nE = \Big\{(z_1, \ldots, z_n) \;\Big|\; z_i \in E,\ z_i \neq z_j \text{ if } i \neq j\Big\}.$$

Let B_nE denote the set of equivalence classes of elements of F_nE, where two n-tuples in F_nE are equivalent if one is a permutation of the other. Then the nth braid group is defined to be the fundamental group $G = \pi_1 B_nE$. That is, G is the set of equivalence classes of continuous maps of a circle to B_nE, where two such "loops" are equivalent if one can be continuously deformed into the other.

More concretely, choose a base point $z^0 = (z_1^0, \ldots, z_n^0) \in F_nE$. Any element of $\pi_1 B_nE$ is represented by a loop in B_nE that can be lifted to a path in F_nE that starts at z^0 and ends at a point obtained by permuting the coordinates of z^0. That is, an element of the braid group is represented by a continuous function $f(t) = (f_1(t), \ldots, f_n(t))$, $0 \le t \le 1$, such that $f(0) = z^0$ and $f(1)$ is a permutation of z^0. The union of the graphs of $f_i(t)$, $i = 1, \ldots, n$, in $E \times [0,1]$ is called a *geometric braid.* Figure 5, taken from [12, p. 6], shows a geometric braid for $n = 4$.

Two braids $\mathcal{A}$ and $\mathcal{A}'$ are equivalent if there is a continuous sequence of braids $\mathcal{A}(s)$, $0 \le s \le 1$, such that $\mathcal{A}(0) = \mathcal{A}$ and $\mathcal{A}(1) = \mathcal{A}'$. The identity braid is given by the constant function $f(t) = z^0$, and the inverse of a braid given by $f(t)$ is the "backwards" braid $g(t)$ given by $g(t) = \sigma^{-1} f(1-t)$, where σ is the permutation such that $f(1) = \sigma(z^0)$. The composition of two braids corresponding to functions f and g is given by the function $h(t)$ defined as follows: $h(t) = f(2t)$ for $0 \le t \le 1/2$, $h(t) = \sigma(g(2t-1))$ for $1/2 \le t \le 1$.

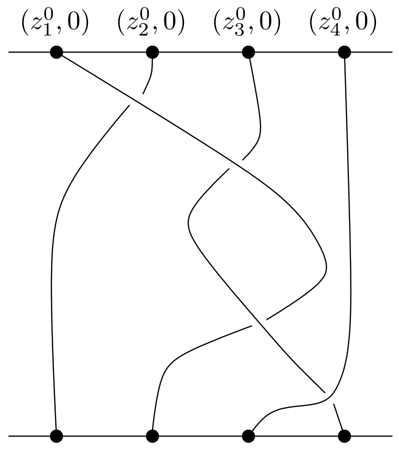


Fig. 5 *A geometric braid for $n = 4$.*

We now describe a key exchange system whose security is based on the presumed intractability of the conjugacy search problem in the braid group (see [4]). Here is how it works. Alice selects m elements $a_1, \ldots, a_m$ of G. These elements are publicly known. She then randomly generates a secret sequence $j_1, \ldots, j_\ell$ of indices between 1 and m, and sets $A = a_{j_1} \cdots a_{j_\ell}$. Bob similarly selects $b_1, \ldots, b_m$ and a secret element $B = b_{k_1} \cdots b_{k_\ell}$. Next, Alice conjugates Bob's publicly known b_i by her secret element A: $x_i = A^{-1} b_i A$; and Bob does likewise: $y_i = B^{-1} a_i B$. Alice sends the m-tuple $(x_1, \ldots, x_m)$ to Bob, and Bob sends $(y_1, \ldots, y_m)$ to Alice. The shared key is the commutator $A^{-1}B^{-1}AB$, which Alice calculates as follows:

$$A^{-1} y_{j_1} \cdots y_{j_\ell} = A^{-1}B^{-1} a_{j_1} \cdots a_{j_\ell} B = A^{-1}B^{-1}AB;$$

and which Bob calculates as follows:

$$(B^{-1} x_{k_1} \cdots x_{k_\ell})^{-1} = (B^{-1}A^{-1} b_{k_1} \cdots b_{k_\ell} A)^{-1} = (B^{-1}A^{-1}BA)^{-1}.$$

This is a clever method of arriving at a shared key. Unfortunately, it seems to be vulnerable to certain types of attacks. Hughes [55] has shown that the existence of a map, called the Burau representation, from the braid group to a general linear group often enables one to use linear algebra to find the secret A and B. The attack is complicated by the fact that the Burau representation is not faithful—in other words, many different braids are mapped to the same matrix—but Hughes shows that the parameters suggested in [4] are insecure.

9.2. Hidden Monomials and Polly Cracker. In this subsection we describe two different public-key cryptosystems that are based on commutative algebra. The first one is an example of various systems developed by Patarin [90, 91] after he broke a simpler version due to Imai and Matsumoto (see [89]). This cryptosystem, which has not yet been broken, is based on the observation that a system of n linear equations in n unknowns is easy to solve, while a system of n quadratic equations is not.

Let q be a power of 2, and let $\{\beta_1, \ldots, \beta_n\}$ be a basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. An element of $\mathbb{F}_{q^n}$ will be written in boldface, and the corresponding n-vector over $\mathbb{F}_q$ with respect to the basis $\{\beta_1, \ldots, \beta_n\}$ will be denoted by underlining. Both plaintext and ciphertext message units will be n-vectors over $\mathbb{F}_q$, denoted $\underline{x}$ and $\underline{y}$, respectively.

To set up her "hidden monomial" cryptosystem, Alice first chooses two secret affine transformations,

$$\underline{u} = A\underline{x} + \underline{c}, \qquad \underline{v} = B\underline{y} + \underline{d}, \tag{2}$$

where A and B are fixed invertible $n \times n$-matrices over $\mathbb{F}_q$ and $\underline{c}$ and $\underline{d}$ are fixed n-vectors. Alice next chooses an integer h of the form

$$h = q^{\alpha_1} + q^{\alpha_2} - q^{\beta_1} - q^{\beta_2}$$

that is prime to $q^n - 1$, and she computes h' such that $hh' \equiv 1 \pmod{q^n - 1}$. She also chooses secret nonzero elements $\mathbf{r}, \mathbf{s} \in \mathbb{F}_{q^n}$. The enciphering function from $\underline{x}$ to $\underline{y}$ is based on the following relation between the corresponding $\mathbf{u}$ and $\mathbf{v}$ (here $\mathbf{v}$ is nonzero):

$$\mathbf{u}^h = \mathbf{r} + \frac{\mathbf{s}}{\mathbf{v}}, \qquad \text{so that} \qquad \mathbf{u} = \left(\mathbf{r} + \frac{\mathbf{s}}{\mathbf{v}}\right)^{h'}. \tag{3}$$

Equivalently, the relation between $\mathbf{u}$ and $\mathbf{v}$ is

$$\mathbf{u}^{q^{\alpha_1}} \mathbf{u}^{q^{\alpha_2}} \mathbf{v} = \mathbf{u}^{q^{\beta_1}} \mathbf{u}^{q^{\beta_2}} (\mathbf{r}\mathbf{v} + \mathbf{s}).$$

Alice uses this relation to set up her public key as follows. Notice that for any fixed k the map from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^n}$ given by $\mathbf{u} \mapsto \mathbf{u}^{q^k}$ is $\mathbb{F}_q$-linear, and so is given by a matrix with respect to the basis $\{\beta_1, \dots, \beta_n\}$. Similarly, for any fixed k, $1 \le k \le n$, the map given by $\mathbf{u} \mapsto \beta_k \mathbf{u}$ is $\mathbb{F}_q$-linear. Thus, the left-hand side $\mathbf{u}^{q^{\alpha_1}} \mathbf{u}^{q^{\alpha_2}} \mathbf{v}$ of the above relation can be expressed in terms of the basis as a sum $\sum p_j(u_1, \dots, u_n, v_1, \dots, v_n)\beta_j$, where each p_j is a polynomial of total degree 3 in the coordinates of $\mathbf{u}$ and $\mathbf{v}$. This polynomial is linear in the v_i and is of total degree 2 in the u_i. Similarly, the right-hand side $\mathbf{u}^{q^{\beta_1}} \mathbf{u}^{q^{\beta_2}} (\mathbf{r}\mathbf{v} + \mathbf{s})$ can be written in the same way, where again the coefficients of the β_j are polynomials of degree 1 in the v_i and degree 2 in the u_i. Alice can easily compute the coefficients of the polynomials on the left and right. Finally, she uses her affine relations (2) to transform these polynomials into polynomials in the plaintext $\underline{x}$ and ciphertext $\underline{y}$ that are quadratic in the x_i and linear in the y_i. By equating the polynomial coefficients of each β_j, Alice arrives at a set of n polynomial relations among the $2n$ variables $x_1, \dots, x_n, y_1, \dots, y_n$.

Alice's public key consists of these n polynomial relations of total degree 3 connecting the coordinates of the plaintext and ciphertext. She keeps the matrices A, B, the vectors $\underline{c}, \underline{d}$, and the constants $\mathbf{r}, \mathbf{s}$ all secret; and if she wants, she can also keep her basis $\{\beta_1, \dots, \beta_n\}$ and the integer h secret as well. The only information that Bob needs is the coefficients of the polynomial relations between $\underline{x}$ and $\underline{y}$.

When he sends a message, Bob must find the ciphertext $\underline{y}$ from $\underline{x}$. Since the degree-3 polynomials are linear in the y_i, this involves solving a system of n linear equations in n unknowns, so Bob can quickly find the ciphertext. An eavesdropper, who knows only the ciphertext and the public key, is faced with the difficult task of solving a system of n quadratic equations in n unknowns. Alice, of course, can find $\underline{x}$ from $\underline{y}$ in a much easier way. Namely, she uses the affine relation $\underline{v} = B\underline{y} + \underline{d}$ to determine $\underline{v}$ from $\underline{y}$; then she goes directly from $\mathbf{v}$ to $\mathbf{u}$ using her "hidden monomial" relation (3); and, finally, she goes from $\underline{u}$ to $\underline{x}$ by inverting the affine map $\underline{u} = A\underline{x} + \underline{c}$.

The above cryptosystem is a special case of a broad class of constructions due to Patarin. Most of his systems remain unbroken, but there has not yet been enough analysis of their security for one to be completely confident. There are also questions of efficiency that remain to be resolved. For example, in the system just described the public key is large, consisting of $O(n^4)$ coefficients in $\mathbb{F}_q$.

Other efforts at constructing public-key cryptosystems using commutative algebra have been based on such hard problems as ideal membership (determining whether a given polynomial belongs to the ideal generated by a fixed set of polynomials) and Groebner basis (finding a certain best possible set of generating polynomials for an ideal). We give a simple example of such a system (called "Polly Cracker" by Fellows) [34], which, however, has recently been successfully attacked [41].

Let $\mathbb{F}_q$ be a finite field, and let $T = \{t_i\}_{i=1}^n$ be a set of variables. Alice wants to be able to receive messages $M \in \mathbb{F}_q$ from Bob. Her secret key is a random vector $y \in \mathbb{F}_q^n$, and her public key is a set of polynomials $B = \{q_j\}$ in $\mathbb{F}_q[T]$ such that $q_j(y) = 0$ for all j. To send the message M, Bob generates an element $p = \sum h_j q_j$ of the ideal $J \subset \mathbb{F}[T]$ generated by B, and sends her the polynomial $C = p + M$. When Alice receives the ciphertext polynomial C, she finds M by evaluating it at y: $C(y) = p(y) + M = M$.

Note that it is very easy for Alice to construct a pair

$$(\text{private key } = y, \quad \text{public key } = B).$$

Namely, she generates a random y, chooses arbitrary polynomials $\widetilde{q}_j$, and sets $q_j = \widetilde{q}_j - \widetilde{q}_j(y)$. But it is an open question whether she can choose the keys in such a way as to avoid attacks such as [41].

10. Quantum Cryptography and Quantum Computation. Quantum cryptography has a very different flavor from all of the types of public-key cryptography discussed above, because it is based not on a mathematical one-way function, but rather on a process which is known to be one-way by some basic laws of physics. The idea was first proposed by Brassard and Bennett in the early 1980s [9, 8], and at present there is at least one web site (www.magiqtech.com) claiming to have commercial products for quantum key distribution.

Suppose that Alice and Bob want to agree upon a secret key—a random sequence of bits—for use in a symmetric-key cryptosystem. Here is how they can use quantum mechanics to do that, while at the same time determining whether or not an unauthorized person (Eve) has been eavesdropping on their communications. Alice randomly chooses a polarization (that is, a line in space and one of the two directions along that line) for each photon that she sends to Bob. According to a basic principle of quantum mechanics, if Bob measures a photon along the same line that Alice chose, the photon will keep its polarization. If he measures the photons along randomly chosen lines, on the average he will measure the correct direction only half the time.

In order to determine the key, for each photon, Bob chooses a line along which to measure its polarization. He then sends Alice a list of the lines he used. Alice informs him which of the lines agree with the ones she used for the polarization. Bob knows the correct direction of polarization of that subsequence of photons. Some of the photons in this subsequence are used to form the sequence of bits for the shared key. In order to see whether or not an adversary is eavesdropping, Alice and Bob compare the photons in the subsequence that are not being used for the key. If Eve has been measuring the polarizations, she will have altered many of the values by measuring them, and Bob and Alice will immediately detect the discrepancy. On the other hand, if the sequences they compare are in agreement, they can be confident that no one has been intercepting their communications, and their key is secure.

There is a second application of quantum mechanics to cryptography that is of a very different sort: quantum computation. The idea, which was first developed in detail by Shor [109], is to construct a computing device that performs quantum mechanical experiments in order to test different alternatives simultaneously. In certain situations, such a device can sift through an exponential number of possibilities in polynomial time.

It is a challenging task to develop a quantum algorithm for the types of problems used in cryptography. The most important problems for which this has been done are integer factorization and discrete logarithms (including elliptic curve discrete logarithms [96]). Actually, the basic type of problem that a quantum device can tackle is the discrete logarithm. The proof that it can also factor integers [109] is based on the technique in section 7.4 for using a discrete log algorithm in order to factor an integer. So far no one has found quantum algorithms for hidden monomial, NTRU, or braid group cryptosystems. If quantum computing ever becomes practical, it will then be necessary to have cryptosystems available for real-world use (and approved by the industrial standards bodies) that are not based on integer factorization or discrete logarithms.

11. Further Reading.

There are many books devoted to algorithmic number theory and public-key cryptography. A starting point for obtaining the relevant mathematical background is the book by Koblitz [66]. Detailed treatments of topics in algorithmic number theory such as primality testing and integer factorization are given by Cohen [28] and Crandall and Pomerance [31]. Comprehensive books on cryptography include those by Menezes, van Oorschot, and Vanstone [83], Stinson [115], and Mao [77]. The mathematics behind elliptic curve cryptography is well explained by Washington [121]. See Hankerson, Menezes, and Vanstone [48] for an extensive coverage of implementation aspects of elliptic curve cryptography.

REFERENCES

[1] L. Adleman, J. DeMarrais, and M. Huang, *A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields*, in Algorithmic Number Theory: First International Symposium, Lecture Notes in Comput. Sci. 877, Springer-Verlag, Berlin, 1994, pp. 28–40.

[2] D. Agrawal, B. Archambeault, J. Rao, and P. Rohatgi, *The EM side-channel(s)*, in Cryptographic Hardware and Embedded Systems—CHES 2002, Lecture Notes in Comput. Sci. 2523, Springer-Verlag, Berlin, 2002, pp. 29–45.

[3] I. Anshel, M. Anshel, and D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Res. Lett., 6 (1999), pp. 1–5.

[4] I. Anshel, M. Anshel, and D. Goldfeld, *New key agreement protocol in braid group cryptography*, in Topics in Cryptography—CT-RSA 2001, Lecture Notes in Comput. Sci. 2020, Springer-Verlag, Berlin, 2001, pp. 13–27.

[5] R. Avanzi, *Aspects of hyperelliptic curves over large prime fields in software implementations*, in Cryptographic Hardware and Embedded Systems—CHES 2004, Lecture Notes in Comput. Sci. 3156, Springer-Verlag, Berlin, 2004, pp. 148–162.

[6] R. Balasubramanian and N. Koblitz, *The improbability that an elliptic curve has subexponential discrete log problem under the Menezes–Okamoto–Vanstone algorithm*, J. Cryptology, 11 (1998), pp. 141–145.

[7] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, *Relations among notions of security for public-key encryption schemes*, in Advances in Cryptology—CRYPTO '98, Lecture Notes in Comput. Sci. 1462, Springer-Verlag, Berlin, 1998, pp. 26–45.

[8] C. H. Bennett and G. Brassard, *Quantum cryptography: Public key distribution and coin tossing*, in Proceedings of the IEEE International Conference on Computer Systems and Signal Processing, Bangalore, India, 1984, pp. 175–179.

[9] C. H. Bennett, G. Brassard, S. Breidbart, and S. Wiesner, *Quantum cryptography, or unforgeable subway tokens*, in Advances in Cryptology—CRYPTO '82, Plenum, New York, 1983, pp. 267–275.

[10] D. Bernstein, *Circuits for Integer Factorization: A Proposal*, preprint, 2001.

[11] I. Biehl, J. Buchmann, and C. Thiel, *Cryptographic protocols based on the discrete logarithm problem in real quadratic number fields*, in Advances in Cryptology—CRYPTO '94, Lecture Notes in Comput. Sci. 839, Springer-Verlag, Berlin, 1994, pp. 56–60.

[12] J. Birman, *Braids, Links, and Mapping Class Groups*, Princeton University Press, Princeton, NJ, 1975.

[13] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, Cambridge, UK, 1999.

[14] D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1*, in Advances in Cryptology—CRYPTO '98, Lecture Notes in Comput. Sci. 1462, Springer-Verlag, Berlin, 1998, pp. 1–12.

[15] D. Boneh, *Twenty years of attacks on the RSA cryptosystem*, Notices Amer. Math. Soc., 46 (1999), pp. 203–213.

[16] D. Boneh, R. DeMillo, and R. Lipton, *On the importance of checking cryptographic protocols for faults*, in Advances in Cryptology—EUROCRYPT '97, Lecture Notes in Comput. Sci. 1233, Springer-Verlag, Berlin, 1997, pp. 37–51.

[17] D. Boneh and G. Durfee, *Cryptanalysis of RSA with private key d less than $N^{0.292}$*, IEEE Trans. Inform. Theory, 46 (2000), pp. 1339–1349.

[18] D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, SIAM J. Comput., 32 (2003), pp. 586–615.

[19] D. Boneh and R. Lipton, *Algorithms for black-box fields and their applications to cryptography*, in Advances in Cryptology—CRYPTO '96, Lecture Notes in Comput. Sci. 1109, Springer-Verlag, Berlin, 1996, pp. 283–297.

[20] D. Boneh, B. Lynn, and H. Shacham, *Short signatures from the Weil pairing*, in Advances in Cryptology—ASIACRYPT 2001, Lecture Notes in Comput. Sci. 2248, Springer-Verlag, Berlin, 2001, pp. 514–532.

[21] D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring*, in Advances in Cryptology—EUROCRYPT '98, Lecture Notes in Comput. Sci. 1403, Springer-Verlag, Berlin, 1998, pp. 59–71.

[22] E. Brickell, *Breaking iterated knapsacks*, in Advances in Cryptology—CRYPTO '84, Lecture Notes in Comput. Sci. 196, Springer-Verlag, Berlin, 1985, pp. 342–358.

[23] E. Brickell and A. Odlyzko, *Cryptanalysis: A survey of recent results*, Proc. IEEE, 76 (1988), pp. 578–593.

[24] J. Buchmann and S. Hamdy, *A survey on IQ cryptography*, in Public-Key Cryptography and Computational Number Theory, Walter de Gruyter, Berlin, 2001, pp. 1–15.

[25] J. Buchmann, R. Scheidler, and H. C. Williams, *A key exchange protocol using real quadratic fields*, J. Cryptology, 7 (1994), pp. 171–199.

[26] J. Buchmann and H. C. Williams, *A key exchange system based on imaginary quadratic fields*, J. Cryptology, 1 (1988), pp. 107–118.

[27] B. Chor and R. Rivest, *A knapsack-type public key cryptosystem based on arithmetic in finite fields*, IEEE Trans. Inform. Theory, 34 (1988), pp. 901–909.

[28] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, Berlin, 1993.

[29] D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Trans. Inform. Theory, 30 (1984), pp. 587–594.

[30] D. Coppersmith and A. Shamir, *Lattice attacks on NTRU*, in Advances in Cryptology—EUROCRYPT '97, Lecture Notes in Comput. Sci. 1233, Springer-Verlag, Berlin, 1997, pp. 52–61.

[31] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer-Verlag, Berlin, 2001.

[32] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, 22 (1976), pp. 644–654.

[33] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory, 31 (1985), pp. 469–472.

[34] M. Fellows and N. Koblitz, *Combinatorial cryptosystems galore!*, Contemp. Math., 168 (1994), pp. 51–61.

[35] M. Fouquet, P. Gaudry, and R. Harley, *An extension of Satoh's algorithm and its implementation*, J. Ramanujan Math. Soc., 15 (2000), pp. 281–318.

[36] G. Frey and H. Rück, *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves*, Math. Comp., 62 (1994), pp. 865–874.

[37] P. Gaudry, *An algorithm for solving the discrete log problem on hyperelliptic curves*, in Advances in Cryptology—EUROCRYPT 2000, Lecture Notes in Comput. Sci. 1807, Springer-Verlag, Berlin, 2000, pp. 19–34.

[38] P. Gaudry, *A comparison and a combination of SST and AGM algorithms for counting points of elliptic curves in characteristic* 2, in Advances in Cryptology—EUROCRYPT 2003, Lecture Notes in Comput. Sci. 2656, Springer-Verlag, Berlin, 2003, pp. 311–327.

[39] P. Gaudry, *Index Calculus for Abelian Varieties and the Elliptic Curve Discrete Logarithm Problem*, preprint, 2004.

[40] P. Gaudry, F. Hess, and N. Smart, *Constructive and destructive facets of Weil descent on elliptic curves*, J. Cryptology, 15 (2002), pp. 19–34.

[41] W. Geiselmann and R. Steinwandt, *Cryptanalysis of Polly Cracker*, IEEE Trans. Inform. Theory, 48 (2002), pp. 2990–2991.

[42] C. Gentry, J. Jonsson, M. Szydlo, and J. Stern, *Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt* 2001, in Advances in Cryptology—ASIACRYPT 2001, Lecture Notes in Comput. Sci. 2248, Springer-Verlag, Berlin, 2001, pp. 1–20.

[43] C. Gentry and M. Szydlo, *Analysis of the revised NTRU signature scheme R-NSS*, in Advances in Cryptology—EUROCRYPT 2002, Lecture Notes in Comput. Sci. 2332, Springer-Verlag, Berlin, 2002, pp. 299–320.

[44] S. Goldwasser and S. Micali, *Probabilistic encryption*, J. Comput. System Sci., 29 (1984), pp. 270–299.

[45] G. Gong and L. Harn, *Public-key cryptosystems based on cubic finite field extensions*, IEEE Trans. Inform. Theory, 45 (1999), pp. 2601–2605.

[46] D. M. Gordon, *Discrete logarithms in $GF(p)$ using the number field sieve*, SIAM J. Discrete Math., 6 (1993), pp. 124–138.

[47] J. L. Hafner and K. S. McCurley, *A rigorous subexponential algorithm for computation of class groups*, J. Amer. Math. Soc., 2 (1989), pp. 839–850.

[48] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, New York, 2003.

[49] J. Håstad, *Solving simultaneous modular equations of low degree*, SIAM J. Comput., 17 (1988), pp. 336–341.

[50] M. E. Hellman and R. C. Merkle, *Hiding information and signatures in trapdoor knapsacks*, IEEE Trans. Inform. Theory, 24 (1978), pp. 525–530.

[51] F. Hess, *The GHS attack revisited*, in Advances in Cryptology—EUROCRYPT 2003, Lecture Notes in Comput. Sci. 2656, Springer-Verlag, Berlin, 2003, pp. 374–387.

[52] J. Hoffstein, J. Pipher, and J. Silverman, *NTRU: A ring-based public key cryptosystem*, in Algorithmic Number Theory: Third International Symposium, Lecture Notes in Comput. Sci. 1423, Springer-Verlag, Berlin, 1998, pp. 267–288.

[53] J. Hoffstein, J. Pipher, and J. Silverman, *NSS: An NTRU lattice-based signature scheme*, in Advances in Cryptology—EUROCRYPT 2001, Lecture Notes in Comput. Sci. 2045, Springer-Verlag, Berlin, 2001, pp. 211–228.

[54] N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J. Silverman, A. Singer, and W. Whyte, *The impact of decryption failures on the security of NTRU encryption*, in Advances in Cryptology—CRYPTO 2003, Lecture Notes in Comput. Sci. 2729, Springer-Verlag, Berlin, 2003, pp. 226–246.

[55] J. Hughes, *A linear algebraic attack on the AAFG1 braid group cryptosystem*, in Proceedings of the 7th Australian Conference on Information Security and Privacy ACISP 2002, Lecture Notes in Comput. Sci. 2384, Springer-Verlag, Berlin, 2002, pp. 176–189.

[56] T. Izu, J. Kogure, M. Noro, and K. Yokoyama, *Efficient implementation of Schoof's algorithm*, in Advances in Cryptology—ASIACRYPT '98, Lecture Notes in Comput. Sci. 1514, Springer-Verlag, Berlin, 1998, pp. 66–79.

[57] M. Jacobson, N. Koblitz, J. Silverman, A. Stein, and E. Teske, *Analysis of the xedni calculus attack*, Des. Codes Cryptogr., 20 (2000), pp. 41–64.

[58] M. Jacobson, A. Menezes, and A. Stein, *Solving elliptic curve discrete logarithm problems using Weil descent*, J. Ramanujan Math. Soc., 16 (2001), pp. 231–260.

[59] E. Jaulmes and A. Joux, *A chosen ciphertext attack against NTRU*, in Advances in Cryptology—CRYPTO 2000, Lecture Notes in Comput. Sci. 1880, Springer-Verlag, Berlin, 2000, pp. 20–35.

[60] A. Joux, *A one round protocol for tripartite Diffie-Hellman*, in Algorithmic Number Theory: Fourth International Symposium, Lecture Notes in Comput. Sci. 1838, Springer-Verlag, Berlin, 2000, pp. 385–393.

[61] N. Koblitz, *Elliptic curve cryptosystems*, Math. Comp., 48 (1987), pp. 203–209.

[62] N. Koblitz, *Hyperelliptic cryptosystems*, J. Cryptology, 1 (1989), pp. 139–150.

[63] N. Koblitz, *Constructing elliptic curve cryptosystems in characteristic* 2, in Advances in Cryptology—CRYPTO '90, Lecture Notes in Comput. Sci. 537, Springer-Verlag, Berlin, 1991, pp. 156–167.

[64] N. Koblitz, *Elliptic curve implementation of zero-knowledge blobs*, J. Cryptology, 4 (1991), pp. 207–213.

[65] N. Koblitz, *CM-curves with good cryptographic properties*, in Advances in Cryptology—CRYPTO '91, Lecture Notes in Comput. Sci. 576, Springer-Verlag, Berlin, 1992, pp. 279–287.

[66] N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd ed., Springer-Verlag, New York, 1994.

[67] N. Koblitz and A. Menezes, *Another Look at "Provable Security,"* preprint, 2004.

[68] P. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, in Advances in Cryptology—CRYPTO '96, Lecture Notes in Comput. Sci. 1109, Springer-Verlag, Berlin, 1996, pp. 104–113.

[69] P. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*, in Advances in Cryptology—CRYPTO '99, Lecture Notes in Comput. Sci. 1666, Springer-Verlag, Berlin, 1999, pp. 388–397.

[70] G. Lay and H. Zimmer, *Constructing elliptic curves with given group order over large finite fields*, in Algorithmic Number Theory: First International Symposium, Lecture Notes in Comput. Sci. 877, Springer-Verlag, Berlin, 1994, pp. 250–263.

[71] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovasz, *Factoring polynomials with integer coefficients*, Math. Ann., 261 (1982), pp. 513–534.

[72] A. K. Lenstra and A. Shamir, *Analysis and optimization of the TWINKLE factoring device*, in Advances in Cryptology—EUROCRYPT 2000, Lecture Notes in Comput. Sci. 1807, Springer-Verlag, Berlin, 2000, pp. 35–52.

[73] A. K. Lenstra and E. R. Verheul, *The XTR public key system*, in Advances in Cryptology—CRYPTO 2000, Lecture Notes in Comput. Sci. 1880, Springer-Verlag, Berlin, 2000, pp. 1–19.

[74] H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2), 126 (1987), pp. 649–673.

[75] M. Magyarik and N. Wagner, *A public key cryptosystem based on the word problem*, in Advances in Cryptology—CRYPTO '84, Lecture Notes in Comput. Sci. 196, Springer-Verlag, Berlin, 1985, pp. 19–36.

[76] J. Manger, *A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1 v2.0*, in Advances in Cryptology—CRYPTO 2001, Lecture Notes in Comput. Sci. 2139, Springer-Verlag, Berlin, 2001, pp. 230–238.

[77] W. Mao, *Modern Cryptography: Theory and Practice*, Prentice-Hall, Upper Saddle River, NJ, 2003.

[78] M. Maurer, A. Menezes, and E. Teske, *Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree*, LMS J. Comput. Math., 5 (2002), pp. 127–174.

[79] U. M. Maurer and S. Wolf, *The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms*, SIAM J. Comput., 28 (1999), pp. 1689–1721.

[80] A. Menezes, T. Okamoto, and S. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Trans. Inform. Theory, 39 (1993), pp. 1639–1646.

[81] A. Menezes and M. Qu, *Analysis of the Weil descent attack of Gaudry, Hess and Smart*, in Topics in Cryptology—CT-RSA 2001, Lecture Notes in Comput. Sci. 2020, Springer-Verlag, Berlin, 2001, pp. 308–318.

[82] A. Menezes, E. Teske, and A. Weng, *Weak fields for ECC*, in Topics in Cryptology—CT-RSA 2004, Lecture Notes in Comput. Sci. 2964, Springer-Verlag, Berlin, 2004, pp. 366–386.

[83] A. Menezes, P. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.

[84] V. Miller, *Uses of elliptic curves in cryptography*, in Advances in Cryptology—CRYPTO '85, Lecture Notes in Comput. Sci. 218, Springer-Verlag, Berlin, 1986, pp. 417–426.

[85] V. Miller, *Short Programs for Functions on Curves*, manuscript, 1986.

[86] F. Morain, *Building cyclic elliptic curves modulo large primes*, in Advances in Cryptology—EUROCRYPT '91, Lecture Notes in Comput. Sci. 547, Springer-Verlag, Berlin, 1991, pp. 328–336.

[87] P. S. Novikov, *On the algorithmic unsolvability of the word problem in group theory*, Trudy Mat. Inst. im. Steklov., 44 (1955), pp. 1–143.

[88] A. Odlyzko, *The rise and fall of knapsack cryptosystems*, in Cryptology and Computational Number Theory, Proc. Sympos. Appl. Math. 42, AMS, Providence, RI, 1990, pp. 75–88.

[89] J. Patarin, *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt '88*, in Advances in Cryptology—CRYPTO '95, Lecture Notes in Comput. Sci. 963, Springer-Verlag, Berlin, 1995, pp. 248–261.

[90] J. Patarin, *Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms*, in Advances in Cryptology—EUROCRYPT '96, Lecture Notes in Comput. Sci. 1070, Springer-Verlag, Berlin, 1996, pp. 33–48.

[91] J. Patarin, *Asymmetric cryptography with a hidden monomial*, in Advances in Cryptology—CRYPTO '96, Lecture Notes in Comput. Sci. 1109, Springer-Verlag, Berlin, 1996, pp. 45–60.

[92] J. Pelzl, T. Wollinger, J. Guajardo, and C. Paar, *Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves*, in Cryptographic Hardware and Embedded Systems—CHES 2003, Lecture Notes in Comput. Sci. 2779, Springer-Verlag, Berlin, 2003, pp. 351–365.

[93] S. Pohlig and M. Hellman, *An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance*, IEEE Trans. Inform. Theory, 24 (1978), pp. 106–110.

[94] J. Pollard, *Monte Carlo methods for index computation mod p*, Math. Comp., 32 (1978), pp. 918–924.

[95] J. Pollard, *Factoring with cubic integers*, in The Development of the Number Field Sieve, Lecture Notes in Math. 1554, Springer-Verlag, Berlin, 1993, pp. 4–10.

[96] J. Proos and C. Zalka, *Shor's discrete logarithm quantum algorithm for elliptic curves*, Quantum Inf. Comput., 3 (2003), pp. 317–344.

[97] G. Purdy, *A high-security log-in procedure*, Comm. ACM, 17 (1974), pp. 442–445.

[98] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Comm. ACM, 21 (1978), pp. 120–126.

[99] R. Sakai, K. Ohgishi, and M. Kasahara, *Cryptosystems based on pairings*, in Proceedings of the 2000 Symposium on Cryptography and Information Security, Okinawa, 2000.

[100] T. Satoh, *The canonical lift of an ordinary elliptic curve over a prime field and its point counting*, J. Ramanujan Math. Soc., 15 (2000), pp. 247–270.

[101] T. Satoh and K. Araki, *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*, Comment. Math. Univ. St. Paul., 47 (1998), pp. 81–92.

[102] T. Satoh, B. Skjernaa, and Y. Taguchi, *Fast computation of canonical lifts of elliptic curves and its application to point counting*, Finite Fields Appl., 9 (2003), pp. 89–101.

[103] O. Schirokauer, *Discrete logarithms and local units*, Philos. Trans. Roy. Soc. London Ser. A, 345 (1993), pp. 409–423.

[104] R. Schoof, *Elliptic curves over finite fields and the computation of square roots mod p*, Math. Comp., 44 (1985), pp. 483–494.

[105] R. Schoof, *Nonsingular plane cubic curves*, J. Combin. Theory Ser. A, 46 (1987), pp. 183–211.

[106] I. Semaev, *Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p*, Math. Comp., 67 (1998), pp. 353–356

[107] A. Shamir, *A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem*, in Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, New York, 1982, pp. 145–152.

[108] A. Shamir and E. Tromer, *Factoring large numbers with the TWIRL device*, in Advances in Cryptology—CRYPTO 2003, Lecture Notes in Comput. Sci. 2729, Springer-Verlag, Berlin, 2003, pp. 1–26.

[109] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), 1484–1509.

[110] J. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, New York, 1986.

[111] J. Silverman, *The xedni calculus and the elliptic curve discrete logarithm problem*, Des. Codes Cryptogr., 20 (2000), pp. 5–40.

[112] J. Silverman and J. Suzuki, *Elliptic curve discrete logarithms and the index calculus*, Advances in Cryptology—ASIACRYPT '98, Lecture Notes in Comput. Sci. 1514, Springer-Verlag, Berlin, 1998, pp. 110–125.

[113] N. Smart, *The discrete logarithm problem on elliptic curves of trace one*, J. Cryptology, 12 (1999), pp. 193–196.

[114] J. Solinas, *Efficient arithmetic on Koblitz curves*, Des. Codes Cryptogr., 19 (2000), pp. 195–249.

[115] D. Stinson, *Cryptography: Theory and Practice*, 2nd ed., Chapman & Hall/CRC, Boca Raton, FL, 2002.

[116] E. Teske, *Speeding up Pollard's rho method for computing discrete logarithms*, in Algorithmic Number Theory: Third International Symposium, Lecture Notes in Comput. Sci. 1423, Springer-Verlag, Berlin, 1998, pp. 541–554.

[117] N. Thériault, *Index calculus attack for hyperelliptic curves of small genus*, in Advances in Cryptology—ASIACRYPT 2003, Lecture Notes in Comput. Sci. 2894, Springer-Verlag, Berlin, 2003, pp. 75–92.

[118] D. L. Van, A. Jeyanthi, R. Siromoney, and K. G. Subramanian, *Public key cryptosystems based on word problems*, in Proceedings of the ICOMIDC Symposium on Mathematics of Computation, Ho Chi Minh City, 1988.

[119] S. Vaudenay, *Cryptanalysis of the Chor-Rivest cryptosystem*, in Advances in Cryptology—CRYPTO '98, Lecture Notes in Comput. Sci. 1462, Springer-Verlag, Berlin, 1998, pp. 243–256.

[120] E. Verheul, *Evidence that XTR is more secure than supersingular elliptic curve cryptosystems*, in Advances in Cryptology—EUROCRYPT 2001, Lecture Notes in Comput. Sci. 2045, Springer-Verlag, Berlin, 2001, pp. 195–210.

[121] L. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, Boca Raton, FL, 2003.

[122] W. Waterhouse, *Abelian varieties over finite fields*, Ann. Sci. École Norm. Sup. (4), 2 (1969), pp. 521–560.

[123] M. Wiener, *Cryptanalysis of short RSA secret exponents*, IEEE Trans. Inform. Theory, 36 (1990), pp. 553–558.

[124] M. V. Wilkes, *Time-Sharing Computer Systems*, Elsevier, New York, 1968.