Proceedings of Symposia in Applied Mathematics
Volume 42, 1990

Odds and Ends from Cryptology and Computational Number Theory

KEVIN S. McCURLEY

ABSTRACT. Because of the enormous publicity that factoring and the RSA cryptosystem have enjoyed, the casual observer may come to believe that applications of number theory to cryptology will disappear if factoring is someday discovered to be easy. In fact, the role of number theory in complexity-based cryptology extends far beyond just the difficulty of factoring. Other papers in this collection have already discussed the roles of the discrete logarithm problem and the subset sum (knapsack) problem in cryptology. The main purpose of this paper is to describe some of the many other applications of number theory in cryptology.

1. Introduction

The heart of applications of number theory to cryptology lies in the inherent difficulty of computational problems. In some cases, the intractability of a computational problem provides a basis for security in a cryptosystem. Some notable examples of such problems are factoring (discussed in the paper by Pomerance), the knapsack problem (discussed in the paper by Odlyzko), and the discrete logarithm problem (discussed in a separate paper by the author). While these are probably the best known of the problems that have been used as the basis of security for cryptosystems (and possibly the most important), they are by no means the only ones. One of the goals of this paper is to describe some of the other problems that have been used in this manner. In doing so, we shall also present some of the cryptosystems that make use of the problems. The main emphasis of the paper is on the computational problems involved, but it is the view of the author that an understanding of the cryptographic applications will be helpful in fully appreciating the computational problem. Part of the motivation for presenting these is to suggest research topics for the interested reader.

There also exist problems in computational number theory for which the

1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 11Y16, 11T71.
This work was performed under U.S. Department of Energy contract number DE-AC04-76DP00789.

discovery of efficient algorithms has resulted in cryptologic applications. A notable example here is that of compositeness testing (discussed in a separate paper by Lenstra on primality testing), which is important for the construction of keys to be used in various cryptographic schemes. The second goal of this paper is to describe some other efficient algorithms that have proved useful in cryptologic applications.

In this paper I have confined myself to subjects that arise in cryptographic applications, but their interest is by no means limited to this arena. Computational number theory has found applications in many other areas of science and mathematics, including but not limited to Fourier analysis in communication (see [**43**]), pseudorandom number generation (see [**36**]), numerical analysis (see [**34**]), and coding theory (see [**6**]). The field of computational number theory experienced a resurgence of interest after the discovery in the mid 1970s of cryptographic applications, but we should also not forget the fact that computational number theory has interested mathematicians for centuries. Part of this interest can no doubt be traced to the fact that arithmetic occupies such a central role in mathematics.

For the reader interested in learning more about other aspects of computational number theory, there are several good general references. Two particularly good sources for information on number theoretic algorithms are the paper of Bach [**4**] and the book of Knuth [**35**, Chapter 4]. In addition, the survey [**2**] by L. M. Adleman and the author is devoted to complexity aspects of number theory and contains a list of open problems in the area.

2. Linear Algebra Algorithms in Cryptology

Just as in all other areas of applied mathematics, linear algebra plays a prominent role in the applications of number theory to cryptology. We have already seen some examples of this. First, fast integer factoring algorithms often require the solution of large sparse systems of linear equations over the finite field GF(2). Second, the fastest known discrete logarithm algorithms in finite fields require the solution of a large sparse system of linear congruences. Third, the algorithms for breaking the encryption schemes based on the subset sum problem involve algorithms for finding short vectors in lattices (otherwise known as finitely generated free modules over $\mathbb{Z}$). Yet another example of a linear algebra algorithm that arises in cryptology is the Berlekamp-Massey algorithm, which can be used to compute the minimal polynomial of a linear recurrent sequence over a finite field. In this section we shall discuss some of these applications.

In addition to the major applications that I mentioned above, there have been a few other situations in which linear algebra algorithms have come up in cryptology. In particular, the problems of solving systems of linear congruences and systems of linear diophantine equations can be used to predict the outcome of certain pseudorandom number generators [**37**], and in the cryptanalysis of certain applications of RSA to electronic monetary systems

[26]. We shall not discuss these applications further here, but we mention in passing that recent progress has been made [33] on algorithms for solving the underlying computational problems.

2.1. Sparse linear systems over finite fields. Because of the importance of solving sparse linear systems over finite fields to factoring and discrete logarithm algorithms, we shall now describe the state of knowledge in the field. There is a large body of research on the subject of algorithms for solving sparse linear systems over the field of real numbers. Such methods generally fall into two categories: direct methods such as Gaussian elimination that follow a set procedure to arrive at the answer, and indirect or iterative methods that produce a sequence of vectors that hopefully converge to the correct answer. Direct methods can generally be described in terms of arithmetic over an arbitrary field, and so can be applied directly to solving systems over finite fields. Optimized variations of Gaussian elimination that minimize fill-in of linear systems over GF(2) have been employed with great success in the implementation of algorithms for factoring (see [47], [38] for more information on the subject).

Because there is no notion of convergence in finite fields, it may at first sight appear that iterative methods for approximating the solutions of linear systems over the field of real numbers have no direct analogue for solving systems over finite fields. In fact, several of the classical algorithms that are usually viewed as iterative *do* have analogues for solving systems over finite fields. Prime examples are the conjugate gradient method and the Lanczos method. These algorithms have the property that their behavior can be described in terms of orthogonality, and if exact arithmetic is used then they will produce the exact answer after a finite number of steps. They are usually characterized as iterative methods because that is the way they are often used in practice. Experience indicates that when the methods are applied to systems over the real numbers, the sequence of vectors produced often comes very close to the exact answer long before the algorithm is guaranteed to terminate. For more information on the application of the Lanczos and conjugate gradient methods to finite fields, the reader may consult [38] or [17].

2.2. Wiedemann's coordinate recurrence method. In addition to the Gaussian elimination and the Lanczos and conjugate gradient methods for solving linear systems, another alternative was proposed by Wiedemann [62] (this method is in fact closely related to a method of A. N. Krylov; (see [27, §21]). For simplicity, we shall consider only the case of solving a system of the form $Ax = b$, where A is an $n \times n$ nonsingular matrix over $\mathrm{GF}(q)$ (the nonsquare and nonfull-rank cases are treated in [62]). Let $f(x) = \sum_{i=0}^{k} a_i x^i$ be the minimal polynomial of A, with $a_0 = 1$. Since $f(A) = 0$, it follows

that

$$A\left(-\sum_{i=1}^{k} a_i A^{i-1} b\right) = b, \tag{2.1}$$

so that if we know f, the solution to $Ax = b$ can be calculated as

$$x = -\sum_{i=1}^{k} a_i A^{i-1} b.$$

This calculation can be carried out rather easily if A is a sparse matrix, since the most difficult part of the calculation involves repeatedly multiplying a vector by the matrix A. If A has ω nonzero entries, then the calculation of $A^j b$ from $A^{j-1} b$ involves $O(n + \omega)$ arithmetic operations, so that the accumulation of the answer given by (2.1) requires $O(n(n + \omega))$ arithmetic operations in all. The storage requirements to calculate the sum in (2.1) are also quite modest, since we require storage for only three vectors of length n, namely one to store the coefficients of f, one to store the intermediate value $A^j b$, and one to accumulate the answer.

2.3. The Berlekamp-Massey algorithm. In the Wiedemann method, a necessary ingredient is the calculation of the minimal polynomial f of the matrix A, and the method that Wiedemann suggested for doing this is an adaptation of a method discovered by Berlekamp **[6]** and Massey **[41]**. Let $s_0, s_1, \ldots, s_{n-1}$ be a sequence of elements from a finite field $\mathrm{GF}(q)$. The Berlekamp-Massey algorithm determines the coefficients of the linear recurrence of shortest length that generates the sequence s. More precisely, it determines from input $s_0, \ldots, s_{n-1}$ the minimal integer $L = L_n$ and elements $a_1, \ldots, a_L$ such that

$$s_j = -\sum_{i=1}^{L} a_i s_{j-i}, \qquad L \le j \le n-1. \tag{2.2}$$

The Berlekamp-Massey algorithm has several applications, but was originally developed for the decoding of BCH codes. Another application of the Berlekamp-Massey algorithm that we shall now discuss is to the cryptanalysis of certain stream cyphers that are motivated by the one-time pad (also known as a Vernam cypher). Shannon **[59]** proved a result that may be interpreted to say that in an attack on a one-time pad in which only cyphertext is available, the cryptanalyst would be unable to distinguish the true plaintext from any other plaintext of the same length (even if the cryptanalyst has unlimited computing power). The main disadvantage of such a scheme is that the key must be essentially as long as the message, and for this reason several people have suggested using a pseudorandom sequence as a substitute for the truly random sequence. One type of pseudorandom generator that has been suggested for this purpose is that of a feedback shift register, which is a form of

electronic switching circuit. Feedback shift registers can be used to very efficiently expand a short random sequence into a much longer pseudorandom sequence, and in particular *linear* feedback shift registers produce sequences according to linear recurrence relations.

One desirable property of a pseudorandom sequence that is to be used to replace a random sequence in a one-time pad is that the period of the sequence should be very long. From this standpoint, sequences generated by linear feedback shift registers are rather good, since there is a good chance that a sequence over GF (q) that is generated by a recurrence of order L will have a minimal period of length $q^L - 1$ (see [**39**, §8.1] for more details).

Another desirable property of a pseudorandom sequence is that the next term in a sequence should not be easily predictable from previous terms in the sequence (further information on desirable properties of pseudorandom generators can be found in the paper by Lagarias). Unfortunately, random sequences that are produced by linear feedback shift registers of length L can be predicted rather easily by the Berlekamp-Massey algorithm from knowledge of the first $2L$ terms of the sequence (details are given later in this paper). For this reason, they are therefore very weak for this cryptographic application. Proposals such as in [**23**], which use a linear feedback shift register, must therefore be regarded as insecure against an attack in which the cryptanalyst is in possession of a small amount of matching cleartext and cyphertext.

Because of this weakness, interest in sequences has shifted (no pun intended!) to the use of nonlinear feedback shift registers, or nonlinear combinations of linear feedback shift registers. Various studies have been conducted on such sequences, from the point of view of correlation properties and the so-called *linear complexity profile.* These developments are beyond the scope of this paper, but good sources of information about these topics are provided by Rueppel [**55**] and recent proceedings of the Eurocrypt and Crypto Conferences.

Before we describe Wiedemann's application to solving systems of sparse linear systems, we first give a general description of the Berlekamp-Massey algorithm, using the approach of Massey. The problem is to determine the coefficients a_i as in (2.2), or equivalently to determine the generating polynomial $C(x) = 1 + \sum_{i=1}^{L} a_i x^i$ of the sequence. It can be shown (see [**39**, §8.6]) that if we have $2L$ terms of the sequence s, then we can compute the required coefficients by solving the linear system

$$\begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_{L-1} \\ s_1 & s_2 & s_3 & \cdots & s_L \\ s_2 & s_3 & s_4 & \cdots & s_{L+1} \\ \vdots & & & & \vdots \\ s_{L-1} & s_L & s_{L+1} & \cdots & s_{2L-2} \end{bmatrix} \begin{bmatrix} a_L \\ a_{L-1} \\ \vdots \\ \vdots \\ a_1 \end{bmatrix} = \begin{bmatrix} -s_L \\ -s_{L+1} \\ \vdots \\ \vdots \\ -s_{2L-1} \end{bmatrix}. \tag{2.3}$$

We could of course simply solve the system using Gaussian elimination, but

this would take approximately $L^3/3$ operations. The Berlekamp-Massey algorithm makes use of the special structure of the system to devise an algorithm with a running time of only $O(L^2)$ operations, with a small O-constant.

The algorithm proceeds inductively on n, deriving the linear recurrence of least order that will produce the sequence $s_0, \ldots, s_{n-1}$. Beginning with $n = 1$, we take $C^{(1)}(x) = 1$ and $L_1 = 0$. Assume now that we know the generating polynomial

$$C^{(n)}(x) = 1 + \sum_{j=1}^{L_n} a_j^{(n)} x^j \tag{2.4}$$

of degree L_n for the recurrence that will produce the sequence $s_0, \ldots, s_{n-1}$, and we want to determine the polynomial $C^{(n+1)}(x)$ of degreee L_{n+1} that will determine $s_0, \ldots, s_n$. We first compute the *discrepancy*

$$d_n = s_n + \sum_{j=1}^{L_n} a_j^{(n)} s_{n-j}. \tag{2.5}$$

If $d_n = 0$, then we simply take $L_{n+1} = L_n$ and $C^{(n+1)}(x) = C^{(n)}(x)$. Otherwise we take m to be the largest value for which $d_m \neq 0$ and $m < n$ (take $d_0 = 1$ to get it started), and compute

$$C^{(n+1)}(x) = C^{(n)}(x) - d_n d_m^{-1} x^{n-m} C^{(m)}(x), \tag{2.6}$$

and $L_{n+1} = \max(L_n, n + 1 - L_n)$. It can be proved (by induction) that the resulting generating polynomial corresponds to the linear recurrence of least order that generates the required sequence. Moreover, it can also be proved rather easily that the resulting algorithm uses only $O(n^2)$ field operations to compute the recurrence that generates the first n elements of the sequence. A more careful analysis of the performance of the algorithm is given in [**31**]. Further refinements on the algorithm can also be found in [**7**, §11.6 and §11.7], where a variant is described that uses only $O(n(\log n)^{1+\varepsilon})$ field operations.

It is interesting to note that the Berlekamp-Massey algorithm is closely related to the problem of calculating what are known as Padé approximations. It is well known (see [**39**, Theorem 8.40]) that the sequence s_i satisfies a kth order linear recurrence if and only if the generating function $f(x) = \sum_{i=0}^{\infty} s_i x^i$ is a rational function whose denominator is a polynomial of degree at most k. In the nth iteration of the Berlekamp-Massey algorithm, we are therefore actually calculating the denominator of a rational function whose MacLaurin series coefficients are given by the first n terms of the sequence. The Padé table of approximations to a function $f(x)$ consists of a rectangular array of rational functions R_{mn} whose numerator has degree m and denominator degree n, and whose MacLaurin series agrees

with that of f in the first $m+n+1$ terms. The sequence of polynomials generated in the Berlekamp-Massey algorithm appears in the denominators of certain entries in the Padé table. Methods for calculating the entries in the Padé table can be traced to the work of Frobenius [**28**] in 1881, where he derived recurrence relations for the numerators and denominators. The Berlekamp-Massey algorithm adapts the method of Frobenius to account for the fact that certain entries in the Padé table are not defined (see [**5**, Chapter 4] for further details).

We now return to Wiedemann's application of the Berlekamp-Massey algorithm to the problem of computing the minimal polynomial of a matrix. Suppose that we want to find the minimal polynomial of the $n \times n$ matrix A of rank n. Let $f(x) = \sum_{i=0}^{k} a_i x^{k-i}$ be the minimal polynomial of A, with $a_0 = 1$. The key observation here is that if we form a sequence of vectors by choosing a random $x^{(0)}$ and computing $x^{(m+1)} = Ax^{(m)}$, then they satisfy a linear recurrent sequence

$$x^{(j)} = -\sum_{i=1}^{k} a_i x^{(j-i)}, \tag{2.7}$$

so that the individual coordinate sequences $x_i^{(j)}$ satisfy the linear recurrence as well. It follows from the theory of linear recurrent sequences (see [**39**, Chapter 4]) that the minimal polynomial f_i of the ith coordinate sequence $x_i^{(0)}, \ldots, x_i^{(n-1)}$ is a divisor of f. If we compute several different f_i, then there is a good chance of recovering f from the least common multiple of the f_i's. In order to make the argument rigorous, we form truly random sequences from the coordinate sequences by choosing a random vector u, and forming $s_i = u^T x^{(i)}$. It is then possible to prove that the least common multiple of the minimal polynomials arising from a few choices for u will equal f with high probability. Various efficiency improvements can be made as well, but for these details we refer the interested reader to [**62**].

Although there are no apparent applications to cryptology, Wiedemann's paper raised several interesting open questions in computational number theory. The first of these is whether there exists a fast algorithm to find the inverse of a sparse matrix (i.e., in $O(n^{2+\varepsilon})$ operations). Note that there exist algorithms to compute the inverse of and *arbitrary* matrix over a field in $O(n^{2.376})$ field operations [**18**], and it has even been conjectured that $O(n^{2+\varepsilon})$ should be possible for arbitrary matrices. Another interesting question is whether it is possible to remove the nondeterminism from Wiedemann's algorithm. Wiedemann suggested a method for doing this, but it requires $O(n^2)$ space. Third, there is the question of whether there exists a fast algorithm to compute the characteristic polynomial of a sparse matrix. Finally, there is the question of whether Wiedemann's methods can be made competitive with other techniques in practice (see [**38**]). At present we have rather little practical experience with many of these algorithms.

3. Multivariate Polynomial Congruences Modulo a Composite

In 1984, Ong, Schnorr, and Shamir [49] proposed the first in a series of very efficient digital signature schemes based on the difficulty of solving a polynomial congruence modulo a composite integer. Their original scheme was the following. A trusted authority chooses an odd integer $n = pq$ that is presumed hard to factor and publishes the number n (alternatively, each user could choose their own modulus n). Each user who wants to be able to sign messages will choose a random integer s, compute $k \equiv s^2 (\mathrm{mod}\, n)$, and submit k to the trusted authority. The trusted authority then publishes a book containing all of the public keys k of users. The user who wishes to sign a message m will then produce a solution x, y to the congruence $x^2 - ky^2 \equiv m (\mathrm{mod}\, n)$. Anyone can easily verify the validity of the signature. Moreover, the user who holds the secret key s can easily produce a solution by first choosing a random integer r and then applying the extended Euclidean algorithm to calculate

$$x \equiv 2^{-1}(mr^{-1} + r) (\mathrm{mod}\, n),$$
$$y \equiv (2s)^{-1}(mr^{-1} - r) (\mathrm{mod}\, n).$$

The security of the scheme depends on a forger's apparent inability to find a solution to the congruence $x^2 - ky^2 \equiv m (\mathrm{mod}\, n)$ when k, m, and n are given, but s is kept secret. Shortly after this scheme was announced, however, it fell victim to the discovery by Pollard of a very efficient method for solving the congruence, and hence also a method to forge messages. Pollard and Schnorr [51] later proved that the congruence could be solved in random polynomial time assuming the extended Riemann hypothesis. This result was later improved by Adleman, Estes, and the author [1], who modified the algorithm in such a way as to eliminate the assumption of the extended Riemann hypothesis.

The basic idea of Pollard's is quite elegant and uses some elementary theory of binary quadratic forms. Moreover, it holds a valuable lesson for those who tend to believe that a computational problem is difficult just because the only apparent solution is difficult. Before describing his main idea, we need to make a few auxiliary observations.

Consider first the problem of finding a prime p in a given arithmetic progression m modulo n. One easy method for doing this is to simply guess a random integer $r \equiv m (\mathrm{mod}\, n)$ with $r < n^3$, and test it for primality using one of the very efficient methods described in the accompanying paper by A. K. Lenstra on primality testing. According to the prime number theorem for arithmetic progressions (see [20]), if $\gcd(m, n) = 1$, the probability that an r chosen randomly in this manner will be prime is approximately $n/(3\varphi(n) \log n)$, and this statement can be made rigorous under the assumption of the extended Riemann hypothesis. Hence we should expect to find such a prime after examining at most $O(\log n)$ integers.

Next consider the problem of representing a prime $p \equiv 1 \pmod 4$ as a sum of two squares. An efficient algorithm for this problem dates back to the time of Hermite, and was refined and analyzed further by Brillhart [**12**]. The basic idea of the algorithm is first to use one of the efficient randomized algorithms described in [**36**, §4.6.2] to find a solution to $u^2 \equiv -1 \pmod p$. From this, an algorithm that is very similar to the Euclidean algorithm will produce the desired representation of p as a sum of two squares.

From the previous observation it is now easy to describe a method for solving the congruences $x^2 \pm y^2 \equiv m \pmod n$. First, note that a solution to $x^2 - y^2 \equiv m \pmod n$ can be constructed trivially by solving the linear system $x - y \equiv m \pmod n$ and $x + y \equiv 1 \pmod n$. The case $x^2 + y^2 \equiv m \pmod n$ is somewhat more instructive. For this case we begin by using the procedure described above to construct a prime p satisfying $p \equiv m \pmod n$ and $p \equiv 1 \pmod 4$. We then represent p in the form $x^2 + y^2$. Notice now that $x^2 + y^2 = p \equiv m \pmod n$, so we are done.

Pollard's key idea for solving the congruence $x^2 - ky^2 \equiv m \pmod n$ is to reduce it to solving a congruence of the same form, but with k replaced by some k_1 with $|k_1| \leq \sqrt{4|k|/3}$ (and a new m). After a small number of such reduction steps, we eventually reach the case of solving a congruence of the form $x^2 \pm y^2 \equiv m \pmod n$, for which we have already noted that it is easy to find a solution.

The method of reduction mimics the process described previously for dealing with the case $k = -1$. We first find a prime $p \equiv m \pmod n$ for which $(k/p) = 1$. Note that of the primes $p \equiv m \pmod n$, approximately $\frac{1}{2}$ will have the additional property that $(k/p) = 1$, so that we should expect to find one rather quickly. Once this is done, we next find a solution of the congruence $u^2 \equiv k \pmod p$. If we were to follow the procedure used previously for the case $k = -1$, then we would next find a representation of p in the form $x^2 - ky^2$. The theory of binary quadratic forms tells us that p will be represented by the form $x^2 - ky^2$ if and only if this form is equivalent to the form $px^2 + 2uxy + vy^2$, where $u^2 = k + pv$. Unfortunately, there is no reason we should expect this to be the case when k is large, since the class number $h(4k)$ may be large.

While the prime p itself need not be representable by the form $x^2 - ky^2$, it is elementary to prove that a small multiple of p can be so represented, and moreover there is a simple algorithm to produce such a multiple. We construct sequences by taking $u_0 = u$, $q_1 = (u^2 - k)/p$, and for $i \geq 1$, we take

$$u_i \equiv u_{i-1} \pmod{q_i}, \qquad |u_i| \text{ minimal}, \qquad q_{i+1} = (u_i^2 - k)/q_i.$$

It is easily proved that the $|q_i|$'s are decreasing until eventually $|q_i| \leq \sqrt{4|k|/3}$

for some $i = O(\log |k|)$. We then have

$$\prod_{j=0}^{i-1}(u_j^2 - k) = (pq_1)(q_1q_2)\cdots(q_{i-1}q_i) = pt^2q_i \tag{3.1}$$

for some integer t. From the identity

$$(x^2 - ky^2)(z^2 - kw^2) = (xz + kyw)^2 - k(xw + yz)^2 \tag{3.2}$$

(which is a trivial case of composition of forms), we obtain from (3.1) that there exist integers a and b such that $a^2 - kb^2 = pt^2q_i$, so that

$$(at^{-1})^2 - k(bt^{-1})^2 \equiv mq_i \pmod{n}. \tag{3.3}$$

Now observe that if we can solve $x^2 - ky^2 \equiv q_i \pmod{n}$, then we can use (3.3) and (3.2) to solve the original equation $x^2 - ky^2 \equiv m \pmod{n}$. Hence we have reduced to the problem of solving $x^2 - ky^2 \equiv q_i \pmod{n}$, but we can further reduce this to solving $x^2 - q_iy^2 \equiv k \pmod{n}$, so we effectively reduced the original problem to one where k is replaced by the value q_i with $|q_i| \leq \sqrt{4|k|/3}$. By repeating this procedure $O(\log\log n)$ times, we eventually arrive at the problem of solving a congruence of the form $x^2 \pm y^2 \equiv m \pmod{n}$.

3.1. Modifications and extensions. After the original scheme of Ong, Schnorr, and Shamir was broken by Pollard's method, a modification was proposed based on higher-degree congruences [**50**]. One concrete proposal had its security based on the presumed difficulty of solving for x, y, z in the quadric congruence

$$(m_1 - 2kxy)^2 + 4z^2(dx^2 + k(y^2 + dz^2) - m_2) \equiv 0 \pmod{n},$$

when m_1, m_2, d, k, and n are given. This scheme was later broken in [**25**], using methods that are similar to Pollard's original idea. Yet another scheme based on a cubic equation was also proposed, but even that scheme has had doubt cast upon it in [**51**]. It remains an interesting open question to determine what types of polynomial congruences with composite moduli can be solved in random polynomial time. It is also an open question whether an efficient signature scheme can be constructed along these lines (although the previous experience will likely make people suspicious of any such proposal!).

4. The ElGamal Schemes

In 1985, ElGamal [**24**] proposed two interesting cryptographic schemes. The first of these was a public-key cryptosystem whose security is based on the difficulty of the Diffie-Hellman problem (for more on this, see the paper on the discrete logarithm problem). The other scheme was a digital signature scheme that we shall now describe. We assume that there is a public directory of keys used to validate signatures, and that a prime p and primitive root

q modulo p are publicly known. Suppose a user wishes to be able to sign messages, and assume that the message can be thought of as an integer m satisfying $0 < m < p - 1$. The user then chooses a secret key s, computes $k \equiv g^s \pmod{p}$, and lists k in the directory as his public key. In order to sign a message m, the user will produce a solution x, y to the congruence $k^x x^y \equiv g^m \pmod{p}$. Clearly, if such a signature is available, then anyone can verify it with little difficulty. Moreover, the user can easily compute such a solution by first choosing a random integer r with $\gcd(r, p-1) = 1$, computing $x \equiv g^r \pmod{p}$, and computing y from $sx + ry \equiv m \pmod{p-1}$ by the extended Euclidean algorithm. Note then that $k^x x^y \equiv g^{sx+ry} \equiv g^m \pmod{p}$.

The security of the ElGamal scheme is based on the presumed difficulty of solving the bivariate congruence $k^x x^y \equiv g^m \pmod{p}$ without knowledge of s. The problem of finding s is known as the discrete logarithm problem and is generally thought to be hard. In addition, if one fixes x, then the problem of finding y is again equivalent to the discrete logarithm problem. If we fix y, then the problem of solving for x is a crazy mixed exponential congruence for which nobody has any idea how to solve efficiently.

All of this looks very similar to the situation described for the schemes of §3 that were based on the difficulty of solving $x^2 - ky^2 \equiv m \pmod{n}$. In that case, recovering the key required the ability to factor n, and fixing either x or y and solving for the other was also known to require the ability to factor n. On the other hand, the bivariate problem turned out to be rather easy to solve, by a method that had little to do with factoring. It is an interesting open problem to see whether the same is true for the ElGamal signature congruence, namely whether it can be solved by a more efficient method than just computation of discrete logarithms.

5. Back to the Beginning: RSA

Very soon after the notion of a public-key cryptosystem was proposed by Diffie and Hellman [**22**], two such systems were proposed. One of these was the Merkle-Hellman system based on the subset sum problem (discussed in the accompanying paper by Odlyzko), and the other was the RSA scheme [**54**], named after its inventors Rivest, Shamir, and Adleman. In casual treatments of the RSA system, it is often stated that the security of the scheme depends on our inability to factor large integers, but this is in fact an exaggeration. In order to say that a system is completely secure, we need to show that it is safe against any possible attack. In the case of RSA, there are at least two attacks that we can consider. The first attack that a cryptanalyst might try is to recover the secret decryption exponent, so that he can then decrypt all past and future messages. For this attack it is known that success will require us to factor n. First note that if $n = pq$ is a product of two primes, as in the RSA cryptosystem, then knowledge of $\varphi(n)$ provides the factors p and

q, since they satisfy

$$pq = n, \qquad p + q = n + 1 - \varphi(n).$$

Moreover, if only n and $e, d < n$ are known, then this provides a multiple $ed - 1$ of $\varphi(n)$. For this case, Long ([**40**], but see also [**64**]) proved that this information provides a random polynomial-time algorithm that will produce the factors of n. By this we mean that there exists an algorithm that makes use of a random number generator, and whose running time is a random variable with expected value that is $O(\log^c n)$ for some constant c.

Another possible attack involves an attempt at decryption of a particular message. In this case, the problem that is required to be solved is to find an integer m such that $m^e \equiv c \pmod{n}$ when e, c, and n are given. For randomly chosen c and $\gcd(e, \varphi(n)) = 1$ (as in the case in the original RSA scheme), this problem may in fact be much easier than factoring the modulus since there is no proof that the ability to factor n is required.

In contrast, for $\gcd(e, \varphi(n)) > 1$, it can be shown that solving the congruence for randomly chosen c requires the ability to factor n. For example, suppose we have an algorithm that for arbitrary integers a will return an integer x such that $x^2 \equiv a \pmod{n}$ (provided such an x exists). In this case, we can select a random integer r, compute $a \equiv r^2 \pmod{n}$, and ask our algorithm for a solution x to $x^2 \equiv a \pmod{n}$. Since the algorithm cannot know ahead of time which solution was originally used to produce a, it is easy to see that there is at least a 50% chance that $\gcd(r + x, n)$ will be a nontrivial factor of n.

The fact that the problem of solving $m^e \equiv c \pmod{n}$ for arbitrary c is equivalent to factoring n when $\gcd(e, \varphi(n)) > 1$ suggests the possibility of using such an exponent in RSA encryption. One complication arises from the fact that the encryption function is no longer an automorphism of $(\mathbb{Z}/n\mathbb{Z})^*$, so that decryption of messages no longer gives a unique value. This problem can be overcome, and a method for doing so is described in [**63**]. On the other hand, the proof given above that solving $x^2 \equiv c \pmod{n}$ is equivalent to factoring points out another possible difficulty, namely the vulnerability of such a scheme to a so-called "chosen cyphertext attack." By this we mean that the user needs to be careful that he does not allow anyone to gain the decryptions of cyphertexts of their own choosing. To my knowledge, no one has yet designed an encryption scheme whose security is equivalent to factoring but is secure against such an attack. On the other hand, the corresponding problem for digital signature schemes was solved theoretically in [**30**], and the search continues for efficient cryptographic schemes that are secure against various kinds of attacks under reasonable complexity assumptions (such as the intractability of factoring). For further developments along these lines, the reader may wish to consult [**46**].

An interesting topic related to the RSA encryption function is the study of the security of individual bits of the cyphertext. In practical cryptography, it

is generally not enough to say that recovering the entire message is impossible, and what we would really like to know is that predicting *any part* of the message with better-than-expected probability is impossible. This provides a powerful motivation for studying the security of individual bits of cyphertext. During the early and middle 1980s there were several papers dealing with this subject, culminating with the paper of Alexi, Chor, Goldreich, and Schnorr [3], where they proved that if there was an efficient method for predicting the least significant bits of an RSA encrypted message with probability at least $0.5 + \varepsilon$, then this algorithm could be used to efficiently determine the entire message.

Several other cryptosystems have been proposed with the property that breaking them is thought to be hard for essentially the same reason that we think inverting the RSA function is hard, namely, the only method that we know for solving the problem of interest requires us to first solve some other problem such as factoring. The intractability of factoring may in fact turn out to have nothing whatsoever to do with breaking RSA, since we may discover a very efficient algorithm for solving the congruence. Recall that an analogous situation has already occurred with the systems described in §3.

6. The Quadratic Residuosity Problem

Recall that an integer a is a quadratic residue modulo n if $\gcd(a, n) = 1$ and there exists a solution x to the congruence $x^2 \equiv a \pmod{n}$. The quadratic residuosity problem can be stated simply as the following: Given integers a and n, decide if a is a quadratic residue modulo n. In our discussion of this problem, we shall distinguish between two cases, depending on whether n is composite.

If p is a prime, then there are at least two very efficient methods for deciding if a is a quadratic residue modulo p. The first method can be traced back to Euler, who proved that if p is an odd prime, then

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \pmod{p}, \tag{6.1}$$

where $(\div)$ is the Legendre symbol, defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a \text{ is a quadratic residue } (\bmod p), \\ -1, & \text{if } a \text{ is a quadratic nonresidue } (\bmod p), \\ 0, & \text{if } p \mid a. \end{cases}$$

Clearly we can use (6.1) with the very efficient binary method of exponentiation to solve the problem of quadratic residuosity modulo a prime. Using standard algorithms for arithmetic, this will require $O(\log^3 p)$ bit operations.

A more efficient alternative is provided by the law of quadratic reciprocity. In order to state this, we first recall the definition of the Jacobi symbol, which extends the definition of the Lengendre symbol. If n is odd with prime

factorization $n = \prod_{i=1}^{k} p_i^{e_i}$, then the Jacobi symbol is defined by

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}.$$

The law of quadratic reciprocity states that for odd a and n,

$$\left(\frac{a}{n}\right) = \begin{cases} -\left(\frac{n \bmod a}{a}\right) & \text{if } a \equiv n \equiv 3 \pmod 4, \\ \left(\frac{n \bmod a}{a}\right) & \text{otherwise.} \end{cases}$$

We now need to deal with the case that a is even. If $a = 2^e \tilde{a}$ with $\tilde{a}$ odd, we use the fact that the Legendre symbol is a multiplicative function to obtain

$$\left(\frac{a}{n}\right) = \left(\frac{2}{n}\right)^e \left(\frac{\tilde{a}}{n}\right).$$

Combining this with the result of Gauss that

$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 \pmod 8, \\ -1 & \text{if } n \equiv \pm 3 \pmod 8, \end{cases}$$

we have a complete method for evaluating the Legendre symbol (a/n). As an example, consider the problem of calculating $\left(\frac{1243}{1423}\right)$. In this case we have

$$\begin{aligned} \left(\frac{1243}{1423}\right) &= -\left(\frac{180}{1243}\right) \\ &= -\left(\frac{2}{1243}\right)^2 \left(\frac{45}{1243}\right) \\ &= -\left(\frac{28}{45}\right) \\ &= -\left(\frac{2}{45}\right)^2 \left(\frac{7}{45}\right) \\ &= -\left(\frac{3}{7}\right) \\ &= \left(\frac{1}{3}\right) \\ &= 1. \end{aligned}$$

Clearly this algorithm is closely related to the Euclidean algorithm, and it can be shown that the running time is $O(\log^2 p)$ bit operations. For a thorough analysis of algorithms for computing the Jacobi symbol, see **[56]**.

Suppose now that n is odd and composite. In this case, we know that a is a quadratic residue modulo n if and only if it is a quadratic residue modulo every prime dividing n. Clearly, if $(a/n) = -1$, then $(a/p_i) = -1$ for some i, and a is a quadratic nonresidue modulo n. On the other hand, it is clearly possible that a might be a quadratic nonresidue modulo n even though $(a/n) = 1$. This is precisely the case that is regarded by some as intractable, since the only method we know for determining quadratic residuosity in this case requires that we first factor n.

Because of our inability to solve the quadratic residuosity problem without factoring, several researchers have proposed cryptosystems whose security is based on the difficulty of determining quadratic residuosity (see for example **[8]**, **[29]**, and **[46]**). Whether it is in fact intractable (or at least equivalent to factoring in some sense) remains a very interesting question.

One peculiar connection is known to exist between this problem and the problem of determining the number of prime factors of an integer. The connection lies in the fact that if an integer n is odd and has k distinct prime factors, then the number of quadratic residues modulo n is $2^{-k}\varphi(n)$. Suppose now that we knew of an algorithm for determining whether a is a quadratic residue modulo n. We could then determine the value of k with very high probability by simply choosing random residues modulo n, seeing whether they are quadratic residues, and from this derive an estimate of the fraction that are quadratic residues.

7. Identity-based Cryptosystems

One fundamental problem in cryptology is that of identification, namely guaranteeing that the person with whom you are communicating is indeed who you think they are. In 1984, Shamir [**57**] proposed a novel approach to overcome this problem, which he referred to as "identity-based cryptosystems." In his original paper, he discussed the general notion of an identity-based scheme, and he also proposed a concrete implementation of a signature scheme using this approach. Since then, several identity-based schemes based on number-theoretic problems have been proposed in the open literature, for encryption, key distribution, and digital signatures. In this section we shall present one example from among these, namely the identity-based key distribution scheme of Okamoto and Tanaka [**48**].

We first give the motivation for devising such a scheme. In a large network, the problem of managing keys for communication between any two parties can be a daunting task. If each user has to keep keys for each user he communicates with, then the amount of memory can be prohibitive, and this precludes communication with someone for whom a key is not available. If a key distribution center is used, then there will be a significant increase in the level of communication due to the fact that users must then connect to the center before they communicate with anyone else. To overcome these problems, several approaches have been suggested in the literature, including the notion of an identity-based key-distribution scheme. In such a scheme, each user uses his or her personal identification information (such as name and address) as his or her public key. This enables people to avoid the problem of storing and maintaining large lists of public keys, and also eliminates the need for an authenticated list of keys or addresses.

There are several different schemes described in [**48**], but we shall present only the most basic one. The scheme requires a key-issuing authority whose only role is to produce legitimate keys for the future users. When the key-issuing center is set up, it chooses an RSA modulus $n = pq$, RSA exponents d and e with $ed \equiv 1 \pmod{\varphi(n)}$, and in addition, an integer g that is a primitive root modulo both p and q. The parameters e, n, and g are made public, but d, p, and q are kept secret by the center. For a user j whose identification information is I_j, the center uses the secret key d to

compute $s_j \equiv I_j^{-d} \pmod{n}$, and this is provided to the user as their secret key.

When users i and j wish to agree on a common authenticated key, user i chooses a random integer r_i with $0 < r_i < n$ and computes $x_i \equiv s_i g^{r_i} \pmod{n}$, and sends the result to user j. Similarly, user j chooses a random integer r_j, computes $x_j = s_j g^{r_j} \pmod{n}$, and sends the result to user i. Both users can now compute a common key $K \equiv g^{er_ir_j} \pmod{n}$, which user i computes as $(x_j^e I_j)^{r_i} \pmod{n}$, and user j computes as $(x_i^e I_i)^{r_j} \pmod{n}$.

The question now arises: what computational problem does a cheater have to solve in order to defeat the scheme? One possible attack involves an eavesdropper constructing the secret key after having heard the conversation between the two legitmate users. In order to do this, he must compute $g^{er_ir_j} \pmod{n}$ from n, e, g, I_i, I_j, x_i and x_j. One approach to doing so is to compute $g^{er_i} \equiv I_i x_i^e \pmod{n}$ and $g^{er_j} \equiv I_j x_j^e \pmod{n}$, and from these compute $g^{er_ir_j} \pmod{n}$. This problen is known as the Diffie-Hellman problem, and is widely thought to be hard (see the accompanying paper on the discrete logarithm problem for more information). Another approach would be to start by computing the secret keys s_i and s_j, but this is equivalent to inverting the RSA function. Moreover, even if we were able to do this, we would be no closer to computing r_i and r_j, so we would still have a Dillie-Hellman problem to solve.

Another possible attack on the system could be attempted by an impostor trying to disguise himself as user k. In this case he can send a number x of his own choosing in place of x_k, and try to compute (or guess) the secret key $K \equiv (x^e I_k)^{r_i} \pmod{n}$. This requires the impostor to compute x and r with $x^e I_k \equiv g^{er} \pmod{n}$. If we hold one of the unknowns x and r fixed, then this requires solving either an instance of the discrete logarithm problem or an RSA decryption. On the other hand, the impostor is free to choose both variables, which makes this problem look something like the ElGamal and Ong-Schnorr-Shamir problems.

As a final note we should perhaps point out a small error in [**48**]. There it is stated that for any integer x there is an integer r such that $x \equiv I_j^{-d} g^r \pmod{n}$, but in fact this is true for at most one-half of the possible values of x, since the subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$ generated by g is of order $\operatorname{lcm}\{p-1, q-1\} \leq \frac{1}{2}\varphi(n)$. This detail does not seem to affect their argument, however.

8. Miscellaneous Topics

We close with a whirlwind tour through a few of the many other problems that have turned up in applications to cryptology. Unfortunately, the one additional feature that these problems seem to have in common is that we know so little about them that relatively little can be said about them. The

purpose of such a cursory description here is to give pointers to literature for future research, and to show the diversity of problems that arise.

The first problem that we will mention sounds more combinatorial than number-theoretic, but we mention it here because no one has yet been able to exploit the underlying algebraic structure. The problem is called the "permuted kernel problem" and was proposed by Shamir [**58**] as a basis for security in an extremely efficient zero-knowledge identification scheme. The problem can be stated very simply as follows: Given an $m \times n$ matrix A with $m < n$ and an n-dimensional vector v over a finite field, find a permutation matrix P so that $APv = 0$. The problem is NP-complete, but this may be due to the fact that it is hard in a small number of cases. It would be interesting to see what sorts of algorithms could be devised for various choices of the parameters (beyond those that appeared in [**58**]).

An interesting variant of the RSA scheme as it applies to digital signatures was proposed in [**21**]. Their suggestion was to use as the signature of the message M the value $S \equiv M^{(2M+1)^{-1} (\mathrm{mod}\, \varphi(n))} (\mathrm{mod}\, n)$, where n is an RSA modulus. Such a scheme offers several practical advantages over RSA but has its security based on a different problem. For such a scheme, someone who wishes to forge a signature to the message M needs to be able to create a solution S to the congruence $S^{2M+1} \equiv M (\mathrm{mod}\, n)$. The relationship of this problem factoring or inverting RSA is not clear.

Several cryptographic schemes have been proposed that have their security based on the difficulty of solving polynomial equations in finite fields (or systems of polynomial equations). One notable example is given by Purdy in [**52**], where he proposed a one-way function for validating passwords. According to [**44**], this is used in the VMS operating system (but ignore the technical description there!). Purdy's idea was to evaluate a sparse polynomial of very large degree over a finite field. Since a power x^d can be evaluated in $O(\log d)$ field operations, a polynomial with T terms and degree d can be evaluated in time $O(T \log d)$ operations. On the other hand, all of the known algorithms for determining roots of polynomials over finite fields have running time that is at least linear in the degree d (see [**53**]). Faster methods for finding roots of sparse polynomials would therefore be of great interest. Another instance in which this would be of interest is in the Chor-Rivest knapsack-based public-key cryptosystem (discussed in the paper by Odlyzko on knapsack cryptosystems). Faster algorithms for factorization of polynomials over finite fields would be useful for speeding the decryption of messages by the legitimate receiver, and faster algorithms for factorization of trinomials of large degree would be useful for cryptanalyzing the scheme. For further details on the latter attack, see [**11**].

Another scheme based on the difficulty of solving polynomial equations appeared in [**42**], where an encryption scheme was presented for which decrypting a message requires finding a solution to a system of quadratic

polynomial equations

$$F_1(x_1, x_2, \ldots, x_n) = 0$$
$$\vdots$$
$$F_n(x_1, x_2, \ldots x_n) = 0$$

over a finite field of characteristic 2. This problem is known to be NP-hard when solved over GF(2), but rather little is known about the general case. Yet another public-key encryption scheme based on the difficulty of solving a system of polynomial equations over a finite field is given in [**60**].

In the search for sequences that are hard to predict, Damgård [**19**] has suggested using the sequence

$$(\tfrac{k}{p}), \ldots, (\tfrac{k+l}{p}),$$

where the starting point k and possibly also the modulus p are secret. For values of l that are as small as $\log p$, it can be proved that the next term cannot be predicted with any certainty, because each such sequence is equally likely to occur. For cryptographic applications we would like to say that much longer strings cannot be used to predict the next term. Thus it would be interesting to know more about the distributional properties of sequences of Legendre symbols, and about the computational problem of determining the unknown parameters from knowledge of a given sequence.

Because the various number-theoretic cryptosystems that have been proposed are computationally demanding for legitimate users when compared with traditional methods, a great deal of effort has been devoted to finding methods for speeding the low-level arithmetic required. As an example, RSA requires efficient methods for modular multiplication and exponentiation. One very simple method for speeding modular multiplication was invented by Montgomery [**45**] for speeding various factorization methods, but in fact the same method has also proved to be useful in allowing larger moduli to be used in RSA. Interest has also been focused on the general topic of addition chains for rapid exponentiation. A (somewhat out of date) survey of results in this area can be found in [**36**], and more recent heuristics can be found in [**9**].

The notion of a zero-knowledge "proof" system provides a powerful tool for the design of cryptographic protocols and is discussed at length in the accompanying paper by Goldwasser. Much of the work in this area has focused on developing foundations for cryptology, but other work has focused on developing very efficient and practical cryptographic schemes. For example, in situations where a trusted authority is required to produce a number that is the product of two large primes, but for which the primes are supposed to remain secret, the users would like to be convinced that the number n has the desired form. A partial answer to this is provided in [**61**], where they describe a method for "proving" that a composite is of the form $p^r q^s$, where

p and q are primes that are $\equiv 3 \pmod 4$, and r and s are odd. Other interesting examples of zero-knowledge protocols for number-theoretic problems are provided in [**10**] and [**16**]. Two interesting open questions in this area are the following:

> Find an efficient zero-knowledge protocol for "proving" that a number is squarefree,
>
> Find an efficient zero-knowledge protocol for "proving" that a number generates a large subgroup of $\mathbb{Z}/n\mathbb{Z}$ (or that two numbers generate all of $\mathbb{Z}/n\mathbb{Z}$).

It has been proved that any problem belonging to the complexity class NP has an interactive "proof" system, so that in theory such protocols exist. The problem here is to find protocols that are *efficient* in the sense that the prover can get by with limited computational power, and few rounds of communication are required.

One of the most famous problems in computational number theory is Gauss's class number problem, by which we mean the calculation of the class number $h(D)$ of a quadratic number field when the discriminant D is given. This has been proposed as the basis for security in a Diffie-Hellman key distribution scheme by Buchmann and Williams in [**14**], [**15**]. While the schemes they proposed are still viable, considerable progress has been made on algorithms for the calculation of class numbers. For the case of imaginary quadratic number fields of negative discriminant $-d$, Hafner and the author proved in [**32**] that there exists a probabilistic algorithm to compute $h(-d)$ in expected time $\exp(\sqrt{(2+\varepsilon)\log d \log\log d})$ operations, assuming the extended Riemann hypothesis. An algorithm for the case of real quadratic fields was proposed in [**13**], with similar running time. Thus we see that work on some of the oldest problems in computational number theory can find motivation through cryptology.

Conclusion

The rate of progress in the applications of number theory to cryptology during the last few years has been remarkable. In spite of this progress, it is also apparent that there are a number of areas where there is much work left to be done.

Acknowledgment

Thanks are due to Jim Davis and Carl Pomerance for making comments that led to improved exposition in this paper.

References

1. L. M. Adleman, Dennis R. Estes, and Kevin S. McCurley, *Solving bivariate quadratic congruences in random polynomial time*, Math. Comp. **17** (1987), 17–28.
2. L. M. Adleman and Kevin S. McCurley, *Open problems in number theoretic complexity*, in Discrete Algorithms and Complexity; Proceedings of the Japan–U.S. Joint Seminar, Kyoto, Academic Press, 1987, pp. 237–262.

3. W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, *RSA and Rabin functions: Certain parts are as hard as the whole*, SIAM J. Comput. **17** (1988), 194–209.
4. Eric Bach, *Number-theoretic algorithms*, Annual Reviews in Computer Science **4** (1990), 112–172.
5. George A. Baker, Jr. and Peter Graves-Morris, *Padé approximants. Part* 1 : *Basic theory*, volume 13 of Encyclopedia of mathematics and its applications, Addison-Wesley, Reading, MA, 1981.
6. Elwyn R. Berlekamp, *Algebraic coding theory*, McGraw-Hill, New York, 1968.
7. Richard E. Blahut, *Theory and practice of error control codes*, Addison-Wesley, Reading, MA, 1983.
8. L. Blum, M. Blum, and M. Shub, *A simple unpredictable pseudorandom number generator*, SIAM J. Comput. **15** (1986), 364–383.
9. Jurjen Bos and Mathijs Coster, *Addition chain heuristics*, in Advances in Cryptology (Proceedings of Crypto '89), Lecture Notes in Computer Science, vol. 435, Santa Barbara, Springer-Verlag, 1990, pp. 400–407.
10. Joan Boyar, Katalin Friedl, and Carsten Lund, *Practical zero-knowledge proofs: Giving hints and using deficiencies*, in Advances in Cryptology (Proceedings of Eurocrypt '89), Lecture Notes in Computer Science, Houthalen, to appear, Springer-Verlag.
11. Ernest F. Brickell and Andrew M. Odlyzko, *Cryptanalysis: A survey of recent results*, Proceedings of the IEEE **5** (1988), 578–593.
12. John Brillhart, *Note on representing a prime as a sum of two squares*, Math. Comp. **26** (1972), 1011–1013.
13. Johannes Buchmann, *A subexponential algorithm for the determination of class groups and regulators of algebraic number fields*, preprint, 1989.
14. Johannes Buchmann and Hugh C. Williams, *A key exchange system based on imaginary quadratic fields*, J. Cryptology **1** (1988), 107–118.
15. ——, *A key exchange system based on real quadratic fields*, in Advances in Cryptology (Proceedings of Crypto '89), Lecture Notes in Computer Science, vol. 435, Santa Barbara, Springer-Verlag, 1990, pp. 335–343.
16. David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graf, *An improved protocol for demonstrating possession of discrete logarithms and some generalizations*, in Advances in Cryptology (Proceedings of Eurocrypt '87), Lecture Notes in Computer Science, vol. 304, Berlin, Springer-Verlag, 1988, pp. 127–142.
17. D. Coppersmith, A. Odlyzko, and R. Schroeppel, *Discrete logarithms in GF(p)*, Algorithmica **1** (1986), 1–15.
18. D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, Journal of Symbolic Computation **9** (1990), 251–280. Extended abstract in Proc. 19th ACM Symposium on Theory of Computing (1987), 1–6.
19. Ivan Bjerre Damgård, *On the randomness of Legendre and Jacobi sequences*, in Advances in Cryptology (Proceedings of Crypto '88), Lecture Notes in Computer Science, vol. 403, Berlin, Springer-Verlag, 1990, 163–172.
20. Harold Davenport, *Multiplicative number theory*, Springer-Verlag, Berlin, second edition, 1980 (revised by Hugh L. Montgomery).
21. Wiebren de Jonge and David Chaum, *Some variations on RSA signatures and their security*, in Advances in Cryptology (Proceedings of Crypto '86), Lecture Notes in Computer Science, vol. 263, Berlin, Springer-Verlag, 1987, pp. 49–59.
22. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory **22** (1976), 472–492.
23. Jack S. Donn, *Secure your digital data*, The Electronic Engineer **5** (1972), DC-5–DC-7.
24. Taher ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory **31** (1985), 469–472.
25. Dennis R. Estes, Leonard M. Adleman, Kireeti Kompella, Kevin S. McCurley, and Gary Miller, *Breaking the Ong-Schnorr-Shamir signature scheme for quadratic number fields*, in Advances in Cryptology (Proceedings of Crypto '85), Hugh C. Williams , editor, Lecture Notes in Computer Science, vol. 218, Berlin, Springer-Verlag, 1986, pp. 3–13.
26. Jan-Hendrik Evertse and Eugéne van Heyst, *Which new RSA signatures can be computed from some given RSA signatures*?, in Advances in Cryptology (Proceedings of Eurocrypt