Discrete logarithms: The past and the future

Andrew Odlyzko
AT&T Labs - Research
amo@research.att.com

July 19, 1999

Abstract

The first practical public key cryptosystem to be published, the Diffie-Hellman key exchange algorithm, was based on the assumption that discrete logarithms are hard to compute. This intractability hypothesis is also the foundation for the presumed security of a variety of other public key schemes. While there have been substantial advances in discrete log algorithms in the last two decades, in general the discrete log still appears to be hard, especially for some groups, such as those from elliptic curves. Unfortunately no proofs of hardness are available in this area, so it is necessary to rely on experience and intuition in judging what parameters to use for cryptosystems. This paper presents a brief survey of the current state of the art in discrete logs.

1. Introduction

Many of the popular public key cryptosystems are based on discrete exponentiation. If G is a group, such as the multiplicative group of a finite field or the group of points on an elliptic curve, and g is an element of G, then (writing the group multiplicatively) g^n is the discrete exponentiation of base g to the power n. This operation shares many properties with ordinary exponentiation, so that, for example,

$$g^{(n+m)} = g^n * g^m \ .$$

The inverse operation is, given h in G, to determine n (if it exists) such that $h = g^n$. The number n, usually taken in the range $0 \le n < |\langle g \rangle|$, where $|H|$ is the order of H, and $\langle g \rangle$ is the subgroup generated by g, is called the discrete logarithm of h to base g, since it again shares many properties with the ordinary logarithm. For example, if we use the notation $n = \log_g(h)$ when $h = g^n$, then, assuming for simplicity that G is cyclic and is generated by g,

$$\log_g(h * j) \equiv \log_g(h) + \log_g(j) \pmod{|G|} \ .$$

Discrete logs have a long history in number theory. Initially they were used primarily in computations in finite fields (where they typically appeared in the closely related form of Zech's logarithm). However, they were rather obscure, just like integer factorization. Unlike the latter, they could not even invoke any famous quotes of Gauss (cf. [BachS]) about their fundamental importance in mathematics. The status of discrete logs started to grow in the 20th century, as more computations were done, and as more thought went into algorithmic questions. It appears that they started to play an important role in cryptography already in the 1950s, long before public key systems appeared on the scene, as cryptosystems based on shift-register sequences displaced those based on rotor machines. Discrete logs occur naturally in that context as tools for finding where in a shift register sequence a particular block occurs. The main impetus for the intensive current interest in discrete logs, though, came from the invention of the Diffie-Hellman (DH) method [DiffieH].

The DH key-exchange algorithm was the first practical public key technique to be published, and it is widely used. The basic approach is that if Alice and Bob wish to create a common secret key, they agree on a group $\langle g \rangle$, and then Alice chooses a random integer a, while Bob chooses a random integer b. Alice then computes g^a and sends it to Bob over a public channel, while Bob computes g^b and sends that to Alice. Now Alice and Bob can both compute

$$g^{a \cdot b} = (g^a)^b = (g^b)^a \ ,$$

while an eavesdropper who happens to have overheard the exchange, and thus knows g, g^a, and g^b, will hopefully not be able to compute the secret g^{ab}.

If the discrete log problem for the group $\langle g \rangle$ is easy, an eavesdropper can compute either a or b, and can find out what g^{ab} is. It is an important open question whether determining g^{ab} knowing just g, g^a, and g^b is as hard as the discrete log problem in general. (See [MaurerW] for the latest references on this topic, which will not be covered here. For references on another important subject, namely that of bit security of the discrete log, which will also not be dealt with here, see [BonehV, HastadN].) However, a fast discrete log algorithm would definitely destroy the utility of the widely used Diffie-Hellman protocol. This factor has stimulated an outpouring of research on the complexity of discrete logs.

This paper is a brief survey of the current state of the art in algorithms for discrete logs. There are many cryptosystems based on discrete exponentiation other than the DH key exchange algorithm. Starting especially with the Taher ElGamal proposal [ElGamal], many schemes have been proposed, including the official U.S. Digital Signature Algorithm (DSA). However, they will not be covered here,

and I refer readers to [MenezesVOV, Schneier] for more information about them. Even in the area of complexity of the discrete log problem there have been several general surveys [Lebedev, McCurley, Odlyzko1, Odlyzko2, SchirokauerWD], as well as several more recent papers on specialized subfields. Therefore in this paper I will only give pointers to the latest results, and present some high level observations about the current status and likely future of the discrete log problem.

2. Why discrete logs?

Almost everything that public key cryptography provides, such as digital signatures and key exchange, can be accomplished with RSA and its variants. However, cryptosystems based on discrete exponentiation remain of interest for three main reasons:

(a) Patent issues. The Diffie-Hellman patent expired in 1997, while the RSA patent has until the fall of 2000 to run. Therefore anyone interested in using public key cryptography in the United States (which is the only place where these patents were applied for and issued) can save money and also avoid licensing negotiations.

(b) Technical advantages. In many cases where algorithms of comparable functionality exist, say one over the finite field of integers modulo a prime p, and another using a composite integer n of the same size, breaking the discrete log modulo p appears to be somewhat harder than factoring the integer n. Further, elliptic curve cryptosystems appear to offer the possibility of using much smaller key sizes than would be required by RSA-type cryptosystems of comparable security.

Some other advantages of discrete log cryptosystems come from their limitations. It is widely believed that the U.S. Digital Signature Algorithm is based on discrete logs because it is harder to use it for encryption than if it were based on RSA (and thus on integer factorization). This helps enforce export control regulations on strong encryption without weakening the digital signature methods that are less stringently controlled. On the other hand, many people like the DH algorithm, since the session key it generates is evanescent. In the simplest application of RSA to key generation, Alice creates a session key and transmits it to Bob using Bob's public key. An eavesdropper who can coerce Bob afterwards into revealing his private key can then recover the full text of the communication exchanged by Alice and Bob. In contrast, if Alice and Bob use DH to generate the session key, destroy it after the session ends, and do not store their communication, then neither coercion nor cryptanalysis will enable the eavesdropper to find out what information was exchanged.

(c) They are different. Cryptographers have learned by bitter experience that it is unwise to put all eggs in a single basket. It is desirable to have a diversity of cryptosystems, in case one is broken.

It is an unfortunate fact that discrete logs and integer factorization are so close that many algorithms developed for one problem can be modified to apply to the other. For security, it would be better to have much more diversity. However, more than two decades after the publication of the first two practical public key systems, the DH and the RSA algorithms, the only public key cryptosystems that are trusted and widely deployed are based on the presumed difficulty of the same two problems those schemes relied upon. Interestingly enough, the earlier discovery of public key cryptography in the classified community in the early 1970s [GCHQ] also produced essentially the same two algorithms. There have been many attempts to find public key schemes based on other principles, but so far most have led to systems that were broken, and the ones that are still standing are often regarded with suspicion.

3. General attacks

This section discusses some general algorithms for discrete logs that assume little knowledge of the group.

For most commonly encountered cyclic groups $G = \langle g \rangle$, there is an efficient method for producing a unique canonical representation for an element. (There are exceptions, though, such as some class groups, in which equivalence of two representations is hard to prove.) For such a group, there are several methods for computing the discrete log in a number of operations that is about $|G|^{1/2}$. The first and best known of these is the Shanks "baby-steps, giant-steps" technique. If n is the order of g (or even an upper bound for $|\langle g \rangle|$), we let

$$m = \lceil n^{1/2} \rceil \tag{3.1}$$

and compute

$$h, hg^{-1}, hg^{-2}, \ldots, hg^{-(m-1)} \tag{3.2}$$

and

$$1, g^m, g^{2m}, \ldots, g^{(m-1)m} \ . \tag{3.3}$$

If $h \in \langle g \rangle$, then $h = g^{i+jm}$ for some $0 \le i, j \le m-1$, and the entries hg^{-i} and g^{jm} in the two lists (3.2) and (3.3) will be equal (provided both are in their canonical formats). Checking for equality in two sorted lists of m entries each can be done in linear time (assuming that the representations of

elements are compact enough). Hence the running time of the algorithm is dominated by the arithmetic required to compute the two lists (3.2) and (3.3) and the time to sort them.

Shanks' algorithm is deterministic. If one is willing to give up on determinism, one can replace sorting by hashing, speeding up the process. On the other hand, there is no easy way to reduce space requirements (other than by increasing the running time), which are of order $m \sim |\langle g \rangle|^{1/2}$. (For other examples of computational tradeoffs in cryptographic searches, see [AmiraziziH]].)

There are two other general algorithms for the discrete log problem that run in time $O(|\langle g \rangle|^{1/2})$ and very little space. Both methods are randomized and are due to Pollard [Pollard1]. We sketch a version of one of those methods here. Let $x_0 = 1$ and for $i \geq 0$ let

$$x_{i+1} = \begin{cases} x_i h, & \text{if} \quad x_i \in S_1 \ , \\ x_i^2, & \text{if} \quad x_i \in S_2 \ , \\ x_i g, & \text{if} \quad x_i \in S_3 \ , \end{cases}$$

where S_1, S_2, and S_3 are a partition of the set of canonical representations of elements of $\langle g \rangle$ into three sets of roughly equal size. Then

$$x_i = h^{a_i} g^{b_i}$$

for some integers a_i, b_i. If we find that $x_i = x_{2i}$ for some $i \geq 1$, then we obtain an equation of the form

$$h^{a_i} g^{b_i} = h^{a_{2i}} g^{b_{2i}} \ ,$$

which yields a linear equation for $\log_g(h)$.

It is easy to reduce a general discrete log problem in a cyclic group $\langle g \rangle$ with order whose factorization is known, to the case where the element g has prime order. If $|\langle g \rangle| = n_1 \cdot n_2$, with n_1 and n_2 relatively prime, then solutions to the discrete log problem for the cyclic groups $\langle g^{n_1} \rangle$ and $\langle g^{n_2} \rangle$ can be easily combined to yield a solution to the discrete log problem in $\langle g \rangle$. A further simple reduction shows that solving the discrete log problem in a group of prime order allows one to solve the problem in groups with orders that are powers of that prime.

The Shanks method and the kangaroo method of Pollard can also be used to compute the discrete logarithm of h in about $m^{1/2}$ steps when this discrete log is known to lie in an interval of length at most m. Hence cryptosystem designers have to be careful not to limit the range in which discrete logs lie.

The running times of the Shanks and Pollard algorithms have not been improved to any substantial extent. Only improvements by constant factors have been obtained [Pollard2, Teske, VanOorschotW]. There has been progress, on the other hand, in obtaining fast parallel versions [Pollard2, VanOorschotW],

in which the elapsed time for the computation shrinks by a factor that is linear in the number of processors used. (For the latest applications of these techniques to elliptic curve discrete logs, see [EscotSST]. For a state of the art survey on parallel integer factorization, see [Brent].) However, the basic processing for any of these algorithms still requires a total of about $p^{1/2}$ steps, where p is the largest prime dividing the order of g. This lack of progress in several decades is very important, since it has led to the assumption that in the absence of other structure in a cyclic group $G = \langle g \rangle$ of prime order, it will require on the order of $|G|^{1/2}$ operations to compute a discrete log in G. Many modern public key cryptosystems based on discrete logs, such as the U.S. Digital Signature Algorithm (DSA) [MenezesVOV, Schneier], rely on the Schnorr method [Schnorr], which reduces the computational burden normally imposed by having to work in a large finite field by working within a large multiplicative subgroup Q of prime order q. The assumption is that the discrete log problem in Q cannot be solved much faster than $q^{1/2}$ steps. For q of order 2^{160}, as in DSA, this is about 10^{24} group operations. Since group operations are typically considerably more intensive than the basic instructions of ordinary computers (see [EscotSST] for careful computational experience with elliptic curve operations), it is reasonable to estimate that 10^{24} group operations might require at least 10^{26} ordinary computer instructions. A mips-year (MY, discussed in greater detail in Section 7) is equivalent to about $3 \cdot 10^{13}$ instructions, so breaking DSA, say, with the Pollard or Shanks algorithms would require over 10^{12} MY, which appears to be adequate for a while at least. (See Table 3 for estimates of computational power likely to be available in the future for cryptanalytic efforts. Several people, including Len Adleman and Richard Crandall, have observed that all the instructions executed by digital computers in history are on the order of Avogadro's number, about $6 \cdot 10^{23}$. The largest factoring projects so far have used around 10^{17} operations, and other large distributed projects have accumulated on the order of 10^{20} operations.)

Unfortunately, there is no proof that algorithms faster than those of Shanks and Pollard will not be invented. It would not require a subexponential technique to break the DSA. A method that runs in time $q^{1/4}$ would already destroy it. It is only the complete lack of progress in this area over a quarter of a century that has provided a subjective feeling of comfort to cryptosystem designers and led them to choose a security parameter close to the edge of what is feasible. It is not only improvements of the Shanks and Pollard methods that could be a threat. Note that the security of DSA is based on the assumption that the only attacks are either those that work in the multiplicative subgroup of order q without exploiting any special properties of this group, or else by methods such as the index-calculus ones (discussed in sections 4 and 7) which work with the full group modulo p. There is no proof that some algebraic relations could not be exploited to find an improved algorithm.

There do exist lower bounds on the complexity of the discrete log problems. If practically no knowledge of the group is available, Babai and Szemeredi [BabaiS] have proved a lower bound of order p, where p is the largest prime dividing the order of a cyclic group G. Babai and Szemeredi assume that encodings of group elements are not unique, and that an oracle has to be consulted to determine whether two elements are equal, as well as to perform group operations. Clearly their bound does not cover the operations of the Shanks and Pollard algorithms, which run in time $p^{1/2}$, not p.

Weaker but more realistic lower bounds have also been obtained by Nechaev [Nechaev] and Shoup [Shoup]. (See also [SchnorrJ].) They show that in certain models of computation, (basically, in Shoup's case, ones in which group elements do have unique encodings, but arbitrary ones, with no structure, and in which the algorithm does not have access to the encodings of elements, and has to consult an oracle to perform group operations) it does require on the order of $p^{1/2}$ group operations to compute the discrete log of an element, where p is the largest prime dividing the order of the group. However, it is not clear just how much these bounds mean because of the restrictions on operations that their models allow. Thus even slight structure in the group can potentially lead to much faster algorithms. The index calculus methods are the most prominent collection of algorithms that have successfully used additional knowledge of the underlying groups to provide subexponential algorithms.

4. Index calculus methods

The basic idea, which goes back to Kraitchik [McCurley], is that if

$$\prod_{i=1}^{m} x_i = \prod_{j=1}^{n} y_j \tag{4.1}$$

for some elements of $GF(q)^*$, then

$$\sum_{i=1}^{m} \log_g x_i \equiv \sum_{j=1}^{n} \log_g y_j \pmod{q-1} . \tag{4.2}$$

If we obtain many equations of the above form (with at least one of them involving an element z such as g, for which $\log_g z$ is known), and they do not involve too many x_i and y_j, then the system can be solved. This is similar to the situation in integer factorization, discussed in greater detail in [Lenstra], in which one needs to find a linear dependency among a system of linear equations modulo 2. For more details and latest references on index calculus methods for discrete logarithms, see [SchirokauerWD, Schirokauer3].

Progress in index calculus algorithms has come from better ways of producing relations that lead to equations such as (4.1). The simplest possible approach (for discrete logs modulo a prime p) is to

take a random integer a, compute $u \equiv g^a \pmod{p}$, $1 \leq u \leq p-1$, and check whether

$$u = \prod p_i \ , \tag{4.3}$$

where the p_i are all primes satisfying $p_i < B$ for some bound B. (When the above congruence holds, we say that u is smooth with smoothness bound B.) For most values of a, u will not be smooth, and so will be discarded. However, even with this primitive approach one can obtain running time bounds of the form

$$\exp((c + o(1))(\log p)^{1/2}(\log\log p)^{1/2}) \quad \text{as} \quad p \to \infty \tag{4.4}$$

for some constant c.

The very first analyses of the asymptotic running time of index calculus algorithms appeared in the 1970s, and were of the form (4.4). (Most of these analyses were for integer factorization methods.) All the progress in the 1970s and 1980s was in obtaining better values of c, and for a long time $c = 1$ was the record value, both for discrete logs modulo primes and for integer factorization. For fields $GF(q)$ with $q = p^n$ for small p, Coppersmith's algorithm [Coppersmith1] offered running time

$$\exp((C + o(1))(\log q)^{1/3}(\log\log q)^{2/3}) \quad \text{as} \quad q \to \infty \tag{4.5}$$

for a positive constant C. (To be precise, C in Coppersmith's algorithm was a "variable constant", with precise value bounded above and below by two positive constants, and exact value depending on the relation of n to the nearest powers of p.) However, for prime fields no methods faster than (4.4) were known, and for some fields $GF(q)$ with $q = p^n$ in which both p and n grew, even bounds of the form (4.4) were not available. This lack of progress led to fairly wide speculation that running times for integer factorization and for discrete logs in prime fields could not be improved beyond (4.4) with $c = 1$. However, in 1988 Pollard found a new approach for factoring integers. This method was developed into the special number field sieve (SNFS) by Hendrik Lenstra, and later into the general number field sieve (GNFS) through a collaboration of several researchers (see [LenstraL] for details). Initially there was wide skepticism as to whether this method would be practical, but those doubts have been dispelled. The key point is that the lack of progress over several years did not come from a fundamental limit on computational complexity. A single clever idea stimulated many further ingenious developments, and led to a quantum jump in the algorithmic efficiency of integer factorization and discrete log algorithms.

The first version of the GNFS for discrete logs was developed by Gordon [Gordon]. Gordon's algorithm was improved by Schirokauer [Schirokauer1] (see also [Schirokauer2, SchirokauerWD]).

Adleman [Adleman] (see also [AdlemanH, Schirokauer3]) has invented the function field sieve, which can be regarded as a generalization and often an improvement of the Coppersmith algorithm [Coppersmith1] for fields of small characteristic. As a result, we now possess a variety of discrete log algorithms with running times of the form (4.5). There are still fields $GF(q)$ with $q = p^n$ for which no running time bound of this form is known to hold, and it is an interesting research topic to close the gap and obtain a uniform running time estimate for all finite field discrete logs of the form (4.5).

For fields $GF(q)$ with $q = p^n$ where n is large, the running time bound of (4.5) holds with $C = (32/9)^{1/3} = 1.5262\ldots$. For n small, in general we know only that (4.5) holds with $C = (64/9)^{1/3} = 1.9229\ldots$. For special primes p, which initially were just the primes of the Cunningham form with $p = r^n + a$, where r and a are small, and n large, but which recently have been shown to include numerous other families of primes (see [Semaev3, Semaev4]), versions of the number field sieve for the fields $GF(p)$ run in times of the form (4.5) with $C = 1.5262\ldots$ or even less.

Subexponential index calculus algorithms have been developed for a variety of discrete log problems. (See [Jacobson, MuellerST] for recent examples.) The one notable example where they have not been made to work is for elliptic curve discrete logs, a subject we will return to in a later section. First, though, it is worth noting that most of the recent progress in index calculus algorithms has come from exploitation of algebraic properties of finite fields.

All recent algorithms for discrete logs that are claimed to run in time of the form (4.5) for some constant C are heuristic, in that there is no proof they will run that fast. If one is willing to settle for running times of the form (4.4), then it is possible to obtain rigorous probabilistic algorithms [LovornBP]. However, there is still no rigorous deterministic discrete log algorithm for any large class of typically hard finite fields.

5. Smoothness

Index calculus algorithms depend on a multiplicative splitting of elements (integers, ideals, or polynomials) into elements drawn from a smaller set, typically consisting of elements that are in some sense "small". Elements that do split this way are called smooth, and a fundamental problem in the analysis of index calculus algorithms is to estimate the probability that some process (typically sieving) will produce smooth elements. In almost all cases, the heuristic assumption is made that the elements that arise in the sieving process behave like random elements of that order. That assumption has been verified extensively by computations of smooth elements, as well as by the success of the integer fac-

torization and discrete log algorithms that depend on it. For recent results on smoothness of integers, see [HildebrandT], and on smoothness of algebraic integers, see [BuchmannH]. The latest results on smoothness of polynomials are in [PanarioGF]. (See also [GarefalakisP1, GarefalakisP2] for more general results on polynomial factorization.) Smoothness estimates of this type are also crucial for the the few rigorous proofs of running times of probabilistic algorithms.

(The paper of Soundararajan, mentioned in [Odlyzko2] and several other papers on discrete logarithms, will not be published. It was partially anticipated by the papers of Manstavicius [Manstavicius1, Manstavicius2], and is now largely superseded by the more recent [PanarioGF].)

6. Linear systems over finite fields

Index calculus algorithms require solutions of large sets of linear equations over finite fields. For a long time in the 1970s and early 1980s this step was regarded as a major bottleneck, affecting the asymptotic running time estimates of algorithms such as the continued fraction method and the quadratic sieve. Fortunately the linear systems of equations produced by all index calculus algorithms are sparse. This makes possible development of algorithms that take advantage of this sparsity and operate faster than general ones. The introduction of structured Gaussian elimination [Odlyzko1] (designed to produce smaller linear systems to be solved by other methods) and of the finite field versions of the Lanczos and conjugate gradient algorithms [CoppersmithOS, Odlyzko1], and the subsequent discovery of the Wiedemann algorithm [Wiedemann] led to a reduction in the estimates of the difficulty of the equation solving phase. However, practice lagged behind theory for a long time. Although large scale simulations with the structured Gaussian elimination had been described in [Odlyzko1], it was only after large sets of equations arising from real discrete log problems were solved using combinations of structured Gaussian elimination and the Lanczos and conjugate gradient algorithms [LaMacchiaO] that these methods came into wide use.

The main advances in linear algebra for index calculus algorithms in the 1990s came from the parallelization of the Lanczos and Wiedemann algorithms by Coppersmith [Coppersmith2, Coppersmith3]. Currently the most widely used parallel method is Montgomery's version of the Lanczos algorithm [Montgomery], where it is used after structured Gaussian elimination reduces the matrix to manageable size. These parallelization methods essentially speed up the basic algorithms over the field of two elements (the only case that is needed for integer factorization) by factors of 32 or 64 (depending on the word length of the computer) and are very effective. There are concerns that linear equations might

again become a major bottleneck for current algorithms as larger integers are factored. While sieving can be done on a network of distributed machines, each with modest memory requirements and minor communication needs, linear equation solutions require a closely coupled system of processors with a large memory. Still, those requirements are not too onerous. For example, the recent factorization of a 140 decimal digit integer [CavallarLRLLMMZ] required finding a linear dependency among a system of almost 5 million equations in about that many unknowns, and this was accomplished in about 100 hours on a single fast processor using 810 MB of memory. (This set was generated by structured Gaussian elimination from about 66 million equations in almost 56 million unknowns.) Since the world is increasingly becoming dependent on big centralized Web servers, there is a proliferation of fast multiprocessor computers with tens of gigabytes of memory. The number of entries in the matrix grows roughly linearly in the size of the matrix (since structured Gaussian elimination is used in ways that partially preserve sparsity), and the running time is about quadratic in the size. Thus running time is likely to be more of a problem than storage space. However, this difficulty can probably be overcome by further parallelization, using multiple processors.

In discrete logs, the linear algebra is a more serious problem, since solutions have to be carried out not modulo 2, but modulo large primes. Hence the parallelizations of Coppersmith and Montgomery do not provide any relief, and the original structured Gaussian elimination, Lanczos, and conjugate gradient methods as implemented in [LaMacchiaO1] are still close to best possible. (See [Lambert] for a careful analysis and some improvements.) The difficulty of discrete log problems is not as extreme as it might first appear, though, since most matrix entries are still small, and so storage requirements do not balloon inordinately. Further, most arithmetic operations avoid full multiprecision operations by multiplying a large integer by a small one. Still, the running time is likely to be higher than for integer factorization by much more than the factor of 64 that comes just from the inability to apply the parallelizations of Coppersmith and Montgomery. More research on fast linear algebra modulo large primes would definitely be useful.

For completeness, it is also worth mentioning some rigorous analyses of sparse matrix algorithms over finite fields [Kaltofen, Teitelbaum, Villard].

7. State of the art in index calculus algorithms

How large are the discrete log problems that can be handled? The McCurley challenge problem [McCurley] to compute the discrete log modulo a prime of 129 decimal digits has been solved [WeberD],

but the prime involved was of a special form, so that the special number field sieve could be used. (McCurley posed his challenge before the invention of the number field sieve.) In fields of characteristic 2, Gordon and McCurley [GordonM] have solved the discrete log problem completely for $GF(2^{401})$, and partially (without completing the linear algebra step) for $GF(2^{503})$.

For prime fields $GF(p)$ in which p does not have any special structure, the record is held by Weber [Weber] for an attack with the general number field sieve on a prime of 85 decimal digits, and by Joux and Lercier (May 26, 1998 email announcement [NMBRTHRY]) on a prime of 90 decimal digits with the Gaussian integer method of [CoppersmithOS].

As in other survey papers, it is appropriate to warn that to obtain a proper estimate of security of discrete log problems it is better to consider what has been done in integer factorization. Much more effort has been devoted to that subject than to discrete logs, and most of the leading algorithms are similar. Thus although discrete logs in prime fields do appear harder than factoring integers of the same size, it is prudent to disregard this difference when choosing a cryptosystem. The current record in factoring a generally hard integer is that of the 140 decimal digit challenge integer from RSA Data Security, Inc., RSA-140, which was accomplished with the general number field sieve [CavallarLRLLMMZ]. Among Cunningham integers, the record is the factorization (by the same group as the one that factored RSA-140) of a 211 decimal digit integer by the special number field sieve (email announcement of April 25, 1999 to [NMBRTHRY]).

The factorization of RSA-140 required about 2,000 MY (mips-years, the conventional measure of computing power that is still widely used in estimating integer factorization projects [Odlyzko3]). This is not a huge amount. For comparison, the distributed.net project [Distributed] has available to it the spare capacity of around 100,000 computers, which amount to around 10^7 mips. Thus all these machines can provide around 10^7 MY in a year. (The SETI@home project [SETI] had as of the middle of 1999 about 750,000 computers participating in the analysis of data from the search for extraterrestrial intelligence, and was growing rapidly. Other large projects use the software system [Entropia].) Thus if somebody asks how large an integer can be factored, a good first answer is to ask the questioner in return how many friends that person has. There is a huge and growing amount of idle computing power on the Internet, and harnessing it is more a matter or social and political skills than technology.

Tables 1 and 2 reproduce the running time estimates of [Odlyzko3] for the gnfs (general number field sieve) and the snfs (special number field sieve). These estimates are somewhat conservative, since incremental improvements to the gnfs and snfs in the five years since [Odlyzko3] was written have

Table 1: Computing required to factor integers with current version of gnfs

bits of n	MY required
512	$3 \cdot 10^4$
768	$2 \cdot 10^8$
1024	$3 \cdot 10^{11}$
1280	$1 \cdot 10^{14}$
1536	$3 \cdot 10^{16}$
2048	$3 \cdot 10^{20}$

Table 2: Computing required to factor integers with the snfs

bits of n	MY required
768	$1 \cdot 10^5$
1024	$3 \cdot 10^7$
1280	$3 \cdot 10^9$
1536	$2 \cdot 10^{11}$
2048	$4 \cdot 10^{14}$

made them more efficient (so that, for example, gnfs currently requires probably only around 10^4 MY to factor a 512 bit integer).

There are some technical issues (such as not all machines in the distributed.net project having enough memory to run algorithms that factor 512 bit integers using gnfs), but the general conclusion is that 512 bit RSA is already very insecure. Integers of 512 bits can be factored today, even in small, covert experiments run within a single modest sized company, and even more so within a large organization. The estimates made in [Odlyzko3] for computing power that might be available in the future are presented in Table 3 below. They still seem reasonable.

The preceding tables show that even with current algorithms, within a few years it will be possible for covert efforts (involving just a few people at a single institution, and thus not easily monitored) to crack 768 bit RSA moduli in a year or so. However, given the record of improvements in index calculus

Table 3: Computing power available for integer factorization (in MY)

year	covert attack	open project
2004	10^8	$2 \cdot 10^9$
2014	$10^{10} - 10^{11}$	$10^{11} - 10^{13}$

algorithms, it seems imprudent (as was argued in [Odlyzko3]) to assume that the current version of gnfs is the best that will be available for a long time. At the least, it seems a reasonable precaution to assume that future algorithms will be as efficient as today's snfs, in which case even 1024 bit RSA moduli might be insecure for anything but short-term protection.

8. Elliptic curve discrete logs

So far little has been said about elliptic curve cryptosystems. However, from a practical point of view, they currently are the most important issue in discrete logs. The first elliptic curve schemes were proposed independently by Neal Koblitz and Victor Miller in 1985. Since that time a variety of other systems have been proposed, and some are being deployed, and many more are under active consideration. (See [KoblitzMV] for a recent survey.) The attraction of elliptic curves is that in general no attacks more efficient than those of Pollard and Shanks are known, so key sizes can be much smaller than for RSA or finite field discrete log systems for comparable levels of security. (See [EscotSST] for a detailed account of the latest square root attacks on the Certicom [Certicom] challenges.) The lack of subexponential attacks on elliptic curve cryptosystems offers potential reductions in processing power, storage, message sizes, and electrical power. It is often claimed that properly chosen elliptic curve cryptosystems over fields with sizes of 160 bits are as secure as RSA or finite field discrete log systems with moduli of 1024 bits.

Although elliptic curve cryptosystems are becoming more widely accepted, they are still regarded with suspicion by many. The main concern is that they have not been around long enough to undergo careful scrutiny. This concern is magnified by the deep mathematics that is required to work with them, which reduces the pool of people qualified to examine them.

There are some negative results, such as those of Joe Silverman and Suzuki [SilvermanS] (extending earlier remarks of Victor Miller) which show that certain extensions of index calculus methods will not work on elliptic curves. Joe Silverman's xedni calculus, which provided an intriguing approach to elliptic curve discrete logs has also been shown recently to be unlikely to work efficiently [JacobsonKSST]. There is a natural relation between multiplication and addition in a finite field, which is what makes the index calculus methods work in that setting. There is no such natural relation between the group of points of an elliptic curve and another group, and this appears to be the main reason efficient discrete log methods for elliptic curves have not been found. However, that does not guarantee an attack will not be found.

While there has been no general attack, there has been a worrying series of attacks on special curves. For example, super-singular curves were once thought to be very promising in cryptography, but Menezes, Okamota, and Vanstone showed their discrete log problems could be solved efficiently. More recently, the "anomalous" elliptic curves were shown to have extremely efficient discrete log algorithms. What worries elliptic curve skeptics is that there is much mathematical structure that could potentially be exploited. After all, much of recent progress in index calculus algorithms has come from exploitation of algebraic relations. Further, it has also been shown by Adleman, De Marrais, and Huang [AdlemanDH] (see also [Enge]) that on high genus curves, there do exist efficient discrete log algorithms.

9. The future

The most worrisome long-term threat to discrete log cryptosystems that we can foresee right now comes from quantum computers. Shor [Shor] showed that if such machines could be built, integer factorization and discrete logs (including elliptic curve discrete logs) could be computed in polynomial time. This result has stimulated an explosion in research on quantum computers (see [LANL] for the latest results). While there is still some debate on whether quantum computers are feasible, no fundamental obstructions to their constructions have been found, and novel approaches are regularly suggested. The one comforting factor is that all experts agree that even if quantum computers are eventually built, it will take many years to do so (at least for machines on a scale that will threaten modern public key systems), and so there will be advance warning about the need to develop and deploy alternate systems. (Unfortunately, as the Y2K problem shows, even many years' advance warning is often insufficient to modify deeply embedded systems.)

There are also threats to discrete log and RSA cryptosystems from other hardware approaches. DNA computers do not appear to offer much help. More conventional devices seem more promising. Adi Shamir has proposed the TWINKLE optoelectronic device [Shamir] for faster sieving. There are serious doubts about the feasibility of Shamir's original proposal. However, several people have observed that a much more practical and scalable device can be constructed by abandoning all the attention-catching optical parts of the design, and building a similar device in silicon, using digital adders.

Special purpose devices for factoring and discrete logs may be helpful, just as the Deep Crack device (designed by Paul Kocher, following an earlier proposal from Mike Wiener) was helpful in

demonstrating conclusively that DES is of marginal security. However, they are unlikely to have a major impact, since there is huge computing power available in the idle time of the computers on the Internet, and that power can be harnessed easily. (An earlier special purpose sieving device [PomeranceST] was made obsolete by the arrival of massive distributed computing over the Internet.) Given the rapid growth in such computing power and the relatively slow increase in the running time of the number field sieve with the size of the modulus, cryptosystem designers are already (or should be) building in generous safety margins.

The bottom line is that for general cyclic groups with no structure, no substantial progress has been made in about 25 years. Furthermore, the problems that arise there are close to the fundamental ones of computational complexity, and no sophisticated approaches that require esoteric mathematics have shown any promise of providing better algorithms. Hence cryptologists are comfortable with the assumption that the Pollard and Shanks techniques are close to best possible. They are usually willing to apply the Schnorr technique of working in a multiplicative group of order q where q is a prime of at least 160 bits. (For secrets that need to be preserved for decades, though, it is prudent to increase q to something like 200 bits.) Among index calculus methods, there has been consistent progress, with occasional quantum jumps interspersed with a stream of smaller incremental improvements. However, even the quantum jumps in asymptotic efficiency have not resulted in sudden dramatic increases in the problems that could be solved. Hence system designers are comfortable with choosing key sizes for RSA and finite field discrete log cryptosystems that take into account the current state of the art and add in a generous extra margin of safety to compensate for expected growth in computing power as well as improvements in algorithms. This means that keys have to be of at least 1024 bits even for moderate security, and at least 2048 bits for anything that should remain secure for a decade. For elliptic curves, there are still some doubts. They are extremely attractive, and no general subexponential attacks are known. However, if they are to be used, it might be prudent to build in a considerable safety margin against unexpected attacks, and use key sizes of at least 300 bits, even for moderate security needs.

The main reason for providing generous safety margins is that unexpected new mathematical insights are the greatest potential threat to both discrete logs and integer factorization. There is a long history (see [Odlyzko3]) of overestimates of the difficulty of factoring integers, for example, and most of these overestimates have come from not anticipating new algorithms. Several examples of this phenomenon were cited earlier in this paper. Another way to bring this point out is to note that the number of people who have worked seriously on integer factorization and discrete logs is not all that high. Fur-

thermore, an inordinately large fraction of the really novel ideas (such as the rho and $p-1$ methods, the basic number field sieve, and lattice sieving) have come from a single individual, John Pollard. This suggests that the area has simply not been explored as thoroughly as is often claimed, and that more surprises might still be in store for us.

Acknowledgements: I thank Ross Anderson, Ian Blake, Richard Brent, Don Coppersmith, Richard Crandall, Martin Hellman, Neal Koblitz, Scott Kurowski, Arjen Lenstra, Hendrik Lenstra, Oliver Schirokauer, Igor Shparlinski, Joe Silverman, Nigel Smart, Edlyn Teske, Paul Van Oorschot, and Dan Werthimer for corrections and helpful comments.

References

[Adleman] L. M. Adleman, The function field sieve, pp. 108–121 in *Algorithmic Number Theory: First Intern. Symp., ANTS-I,* L. M. Adleman and M.-D. Huang, eds., Lecture Notes in Math. #877, Springer, 1994.

[AdlemanDH] L. M. Adleman, J. De Marrais, and M.-D. A. Huang, A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields, pp. 28–40 in *Algorithmic Number Theory: First Intern. Symp., ANTS-I,* L. M. Adleman and M.-D. Huang, eds., Lecture Notes in Math. #877, Springer, 1994.

[AdlemanH] L. M. Adleman and M.-D. A. Huang, Function field sieve method for discrete logarithms over finite fields, *Information and Computation,* to appear.

[AmiraziziH] H. R. Amirazizi and M. E. Hellman, Time-memory-processor trade-offs, *IEEE Trans. Inform. Theory 34* (1988), 505–512.

[BabaiS] L. Babai and E. Szemeredi, On the complexity of matrix group problems I, pp. 229–240 in *Proc. 25-th Found. Computer Sci. Symp.*, IEEE Press, 1984.

[BachS] E. Bach and J. Shallit, *Algorithmic Number Theory. Vol. I: Efficient Algorithms,* MIT Press, 1996.

[BonehV] D. Boneh and R. Venkatesan, Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. pp. 129–142 in *Advances in Cryptology - CRYPTO '96,* N. Koblitz, ed., *Lecture Notes in Computer Science #1109,* Springer, 1996.

[Brent] R. P. Brent, Some parallel algorithms for integer factorization, *Proc. Euro-Par '99,* Lecture Notes in Computer Sci., Springer, 1999, to appear. Available at ¡ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/rpb193.ps.gz¿.

[BuchmannH] J. A. Buchmann and C. S. Hollinger, On smooth ideals in number fields, *J. Number Theory* 59 (1996), 82–87.

[CavallarLRLLMMZ] S. Cavallar, W. Lioen, H. te Riele, B. Dodson, A. Lenstra, P. Leyland, P. Montgomery, B. Murphy, and P. Zimmermann, Factorization of RSA-140 using the number field sieve, to be published.

[Certicom] Certicom elliptic curve challenge. Details and current status available at ⟨http://www.certicom.com⟩.

[Coppersmith1] D. Coppersmith, Fast evaluation of logarithms in fields of characteristic two, *IEEE Trans. Inform. Theory* 30 (1984), 587–594.

[Coppersmith2] D. Coppersmith, Solving linear equations over $GF(2)$: block Lanczos algorithm, *Linear Algebra Appl.* 192 (1993), 33–60.

[Coppersmith3] D. Coppersmith, Solving homogeneous linear equations over $GF(2)$ via block Wiedemann algorithm, *Math. Comp.* 62 (1994), 333–350.

[CoppersmithOS] D. Coppersmith, A. Odlyzko, and R. Schroeppel, Discrete logarithms in $GF(p)$, *Algorithmica* 1 (1986), 1–15.

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

[Distributed] distributed.net, "The largest computer on Earth," ⟨http://www.distributed.net/⟩.

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

[Enge] A. Enge, Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time, to be published. Available at ⟨http://www.cacr.math.uwaterloo.ca⟩.

[Entropia] Entropia.com, Inc. software for massive distributed computations. See ⟨http://entropia.com⟩.

[EscotSST] A. E. Escot, J. C. Sager, A. P. L. Selkirk, and D. Tsapakidis, Attacking elliptic curve cryptosystems using the parallel Pollard rho method, *CryptoBytes (The technical newsletter of RSA Laboratories),* 4 (no. 2) (1998), pp. 15–19. Available at ⟨http://www.rsa.com/rsalabs/pubs/cryptobytes/⟩.

[GarefalakisP1] T. Garefalakis and D. Panario, Polynomials over finite fields free from large and small degree irreducible factors, to be published.

[GarefalakisP2] T. Garefalakis and D. Panario, The index calculus method using non-smooth polynomials, to be published.

[GCHQ] Several reports on GCHQ's secret discovery of non-secret (public key) cryptography by C. Cocks, J. H. Ellis, and M. Williamson, available at ⟨http://www.cesg.gov.uk/pkc.htm⟩.

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

[GordonM] D. M. Gordon and K. McCurley, Massively parallel computation of discrete logarithms, pp. 312–323 in *Advances in Cryptology - CRYPTO '92*, E. F. Brickell, ed., *Lecture Notes in Computer Science #740*, Springer, 1992.

[HastadN] J. Håstad and M. Näslund, The security of individual RSA bits, pp. 510–519 in *Proc. 39-th Found. Comp. Sci. Symp.*, IEEE, 1998.

[HildebrandT] A. Hildebrand and G. Tenenbaum, Integers without large prime factors, *J. Theor. Nombres Bordeaux* 5 (1993), 411–484.

[Jacobson] M. J. Jacobson, Jr., Applying sieving to the computation of quadratic class groups, *Math. Comp.* 68 (1999), 859–867.

[JacobsonKSST] M. J. Jacobson, Jr., N. Koblitz, J. H. Silverman, A. Stein, and E. Teske, Analysis of the Xedni calculus attack, *Designs, Codes and Cryptography*, to appear. Available at ⟨http://www.cacr.math.uwaterloo.ca⟩.

[Kaltofen] E. Kaltofen, Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems, *Math. Comp.* 64 (1995), 777–806.

[KoblitzMV] N. Koblitz, A. J. Menezes, and S. Vanstone, The state of elliptic curve cryptography, *Designs, Codes, and Cryptography*, to appear.

[LaMacchiaO1] B. A. LaMacchia and A. M. Odlyzko, Solving large sparse linear systems over finite fields, pp. 109–133 in *Advances in Cryptology: CRYPTO '90*, A. Menezes, S. Vanstone, eds., *Lecture Notes in Computer Science #537*, Springer, 1991. Available at ⟨http://www.research.att.com/∼amo⟩.

[LaMacchiaO2] B. A. LaMacchia and A. M. Odlyzko, Computation of discrete logarithms in prime fields, *Designs, Codes, and Cryptography,* 1 (1991), 46–62. Available at ⟨http://www.research.att.com/∼amo⟩.

[Lambert] R. Lambert, Computational aspects of discrete logarithms, Ph.D. thesis, Dept. Electrical Comp. Eng., Univ. of Waterloo, 1996.

[LANL] Quantum Physics e-print archive, ⟨http://xxx.lanl.gov/archive/quant-ph⟩.

[Lebedev] A. Lebedev, The discrete logarithm problem, manuscript in preparation.

[Lenstra] A. K. Lenstra, Integer factoring, *Designs, Codes, and Cryptography,* to appear.

[LenstraL] A. K. Lenstra and H. W. Lenstra, Jr., eds., *The Development of the Number Field Sieve*, Lecture Notes in Mathematics #1554, Springer, 1993.

[LovornBP] R. Lovorn Bender and C. Pomerance, Rigorous discrete logarithm computations in finite fields via smooth polynomials, pp. 221–232 in *Computational Perspectives on Number Theory (Chicago, 1995), AMS/IS Stud. Adv. Math.* **7**, Amer. Math. Soc., 1998.

[McCurley] K. S. McCurley, The discrete logarithm problem, pp. 49–74 in *Cryptography and Computational Number Theory,* C. Pomerance, ed., *Proc. Symp. Appl. Math.* **42**, Amer. Math. Soc., 1990, pp. 49–74.

[Manstavicius1] E. Manstavicius, Semigroup elements free of large prime factors, pp. 135–153 in *New trends in probability and statistics, Vol. 2 (Palanga, 1991),* VSP, Utrecht, 1992. MR 93m:11091.

[Manstavicius2] E. Manstavicius, Remarks on elements of semigroups that are free of large prime factors, *Liet. Mat. Rink.* 32 (1992), 512–525 (Russian). English translation in *Lithuanian Math. J.* 32 (1992), 400–409. MR 94j:11093.

[MaurerW] U. Maurer and S. Wolf, Lower bounds on generic algorithms in groups, pp. 72–84 in *Advances in Cryptology - EUROCRYPT '98*, K. Nyberg, ed., *Lecture Notes in Computer Science #1403,* Springer, 1998.

[MenezesVOV] A. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography,* CRC Press, 1996.

[MuellerST] V. Müller, A. Stein, and C. Thiel, Computing discrete logarithms in real quadratic congruence function fields of large genus, *Math. Comp.* 68 (1999), 807–822.

[Nechaev] V. I. Nechaev, On the complexity of a deterministic algorithm for a discrete logarithm, *Math. Zametki* 55 (1994), 91–101. English translation in *Math. Notes* 55 (1994), 165–172.

[NMBRTHRY] Victor Miller's number theory mailing list archive, available at ⟨http://www.listserv.nodak.edu⟩.

[Montgomery] P. L. Montgomery, A block Lanczos algorithm for finding dependencies over $GF(2)$, pp. 106–120 in *Advances in Cryptology–EUROCRYPT '95*, L. C. Guillou and J.-J. Quisquater, eds., *Lecture Notes in Computer Science* **921**, Springer, 1995.

[Odlyzko1] A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, pp. 224–314 in *Advances in Cryptology: Proceedings of Eurocrypt '84*, T. Beth, N. Cot, I. Ingemarsson, eds., *Lecture Notes in Computer Science* **209** Springer-Verlag, 1985. Available at ⟨http://www.research.att.com/∼amo⟩.

[Odlyzko2] A. M. Odlyzko, Discrete logarithms and smooth polynomials, pp. 269–278 in *Finite Fields: Theory, Applications and Algorithms*, G. L. Mullen and P. Shiue, eds., Contemporary Math. #168, Amer. Math. Soc., 1994. Available at ⟨http://www.research.att.com/∼amo⟩.

[Odlyzko3] A. M. Odlyzko, The future of integer factorization, *CryptoBytes (The technical newsletter of RSA Laboratories)*, 1 (no. 2) (1995), pp. 5–12. Available at ⟨http://www.rsa.com/rsalabs/pubs/cryptobytes/⟩ and ⟨http://www.research.att.com/∼amo⟩.

[PanarioGF] D. Panario, X. Gourdon, and P. Flajolet, An analytic approach to smooth polynomials over finite fields, pp. 226–236 in *Algorithmic Number Theory: Third Intern. Symp., ANTS-III*, J. P. Buhler, ed., Lecture Notes in Math. #1423, Springer, 1998.

[Pollard1] J. M. Pollard, Monte Carlo methods for index computations $\bmod p$, *Math. Comp. 32* (1978), 918–924.

[Pollard2] J. M. Pollard, Kangaroos, Monopoly and discrete logarithms, *J. Cryptology*, to appear.

[PomeranceS] C. Pomerance and J. W. Smith, Reduction of huge, sparse matrices over finite fields via created catastrophes, *Experimental Math. 1* (1992), 89–94.

[PomeranceST] C. Pomerance, J. W. Smith, and R. Tuler, A pipeline architecture for factoring large integers with the quadratic sieve algorithm, *SIAM J. Comput. 17* (1988), 387–403.

[RSADSI] RSA Data Security factoring challenge. Details and current status available at ⟨http://www.rsadsi.com∼amo⟩.

[Schirokauer1] O. Schirokauer, Discrete logarithms and local units, *Phil. Trans. Royal Soc. London*, A405 (1993), 409–423.

[Schirokauer2] O. Schirokauer, Using number fields to compute logarithms in finite fields, *Math. Comp.* (1999), to appear.

[Schirokauer3] O. Schirokauer, manuscript in preparation.

[SchirokauerWD] O. Schirokauer, D. Weber, and T. Denny, Discrete logarithms: The effectiveness of the index calculus method, pp. 337–362 in *Algorithmic Number Theory: Second Intern. Symp., ANTS-II*, H. Cohen, ed., Lecture Notes in Math. #1122, Springer, 1996.

[Schneier] B. Schneier, *Applied Cryptography*, 2nd ed., Wiley, 1995.

[Schnorr] C. P. Schnorr, Efficient signature generation by smart cards, *J. Cryptology 4* (1991), 161–174.

[SchnorrJ] C. P. Schnorr and M. Jakobsson, Security of discrete log cryptosystems in the random oracle + generic model, to be published.

[Semaev1] I. A. Semaev, An algorithm for discrete logarithms over an arbitrary finite field, *Diskret. Mat.* 7 (1995), 99–109 (Russian). English translation in *Discrete Math. Appl.* 5 (1995), 107–116.

[Semaev2] I. A. Semaev, A generalization of the number field sieve, pp. 45–63 in *Probabilistic Methods in Discrete Mathematics (Petrozavodsk, 1996)*, VSP, 1997.

[Semaev3] I. A. Semaev, An algorithm for evaluation of discrete logarithms in some nonprime finite fields, *Math. Comp.* 67 (1998), 1679–1689.

[Semaev4] I. A. Semaev, Special prime numbers and discrete logs in prime finite fields, to be published.

[SETI] SETI@home distributed computing project. See ⟨http://setiathome.ssl.berkeley.edu⟩.

[Shamir] A. Shamir, Factoring large numbers with the TWINKLE device, to be published. Available at ⟨http://jya.com/twinkle.eps⟩

[Shor] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* 26 (1997), 1484–1509. Available at ⟨http://www.research.att.com/∼shor⟩.

[Shoup] V. Shoup, Lower bounds for discrete logarithms and related problems, pp. 256–266 in *Advances in Cryptology–EUROCRYPT '97*, W. Fumy, ed., Lecture Notes in Computer Science #1233, Springer, 1997.

[SilvermanS] J. H. Silverman and J. Suzuki, pp. 110–125 in *Advances in Cryptology - ASIACRYPT '98*, K. Ohta and D. Pei, eds. *Lecture Notes in Computer Science #1514*, Springer, 1998.

[Soundararajan] K. Soundararajan, Asymptotic formulae for the counting function of smooth polynomials, unpublished manuscript.

[Teitelbaum] J. Teitelbaum, Euclid's algorithm and the Lanczos method over finite fields, *Math. Comp.* 67 (1998), 1665–1678.

[Teske] E. Teske, Speeding up Pollard's rho method for computing discrete logarithms, pp. 541–554 in *Algorithmic Number Theory: Third Intern. Symp., ANTS-III*, J. P. Buhler, ed., Lecture Notes in Math. #1423, Springer, 1998.

[VanOorschotW] P. C. Van Oorschot and M. J. Wiener, Parallel collision search with cryptanalytic applications, *J. Cryptology*, 12 (1999), 1–28.

[Villard] G. Villard, Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems, *Proc. ISSAC'97.*

[Weber] D. Weber, Computing discrete logarithms with quadratic number rings, pp. 171–183 in *Advances in Cryptology - EUROCRYPT '98*, K. Nyberg, ed., *Lecture Notes in Computer Science #1403*, Springer, 1998.

[WeberD] D. Weber and T. F. Denny, The solution of McCurley's discrete log challenge, pp. 458–471 in *Advances in Cryptology–CRYPTO '98*, H. Krawczyk, ed., Lecture Notes in Computer Science #1462, Springer, 1998.

[Wiedemann] D. H. Wiedemann, Solving sparse linear equations over finite fields, *IEEE Trans. Inform. Theory* 32 (1986), 54–62.