Public Key Cryptography with a Group of Unknown Order

Richard P. Brent[1]
Oxford University

rpb@comlab.ox.ac.uk

Programming Research Group
Report PRG–TR–02–00
5 June 2000

Abstract

We present algorithms for digital signature generation and verification using public key cryptography over an arbitrary group G. The algorithms are similar to those of ElGamal, but do not require a knowledge of the group order. Forging signatures or determining the secret key requires the solution of a discrete logarithm problem over G, or the solution of other problems which appear at least as difficult. The algorithms can be modified to work over arbitrary semigroups.

1 Introduction

As first shown by Koblitz [15] and Miller [22], the well-known El Gamal algorithms [7] for encryption and digital signatures can be generalised to work over any group. However, a practical annoyance is that the algorithm for generating signatures requires a knowledge of the order $\#G$ of the group. This seems anomalous since the algorithms for encrypting and decrypting messages do not require knowledge of $\#G$. In the original El Gamal scheme, $\#G$ is not required in order to verify a signature, although it is required for verification in the standard DSA and EC-DSA schemes.

In the following we show that it is possible to create and verify signatures without a knowledge of $\#G$. The signatures are slightly longer, and the algorithms slightly slower, than in the case that $\#G$ is known, but only by small constant factors (for details see §4.1).

Groups which might be used include the group of points on an elliptic curve over a large finite field $\mathrm{GF}(q)$, the jacobian of a hyperelliptic curve over a finite field, and the class group of an imaginary quadratic number field: see [21, §8.4.2]. Other groups may be proposed in the future.

We emphasise that the security of the new algorithms seems comparable to that of the standard ElGamal algorithms if the same group is used. An imposter can not sign a message or find the secret key merely by determining $\#G$.

On the assumption that an attacker can compute $\#G$, the security of both our new algorithm and the standard El Gamal algorithm depend on $\#G$ having a large prime factor (see §5.3), and it is difficult to guarantee this without computing $\#G$. Thus, our algorithm is likely be useful in practice only if it is applied to groups whose order is very difficult to compute. This is not the case for groups of elliptic curves over finite fields $\mathrm{GF}(q)$ ($q = 2^k$ or q a large prime), because of the polynomial time algorithm of Schoof [32] and recent improvements by Atkins, Couveignes, Elkies, Lercier and Morain [1, 4, 5, 8, 18, 19, 23]. The expected run-time of the best of these algorithms is $O((\log q)^6)$.

Our notation and assumptions are described in §2, and the new algorithms for determining a cryptosystem, signing and verifying a message are given in §3. The new algorithms are compared with the standard ElGamal algorithms and some of their variants in §4. The security of the new

[1]Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

algorithms is considered in §5, and in §6 we consider applying the new algorithms with groups drawn at random from a family of groups with certain statistical properties. A generalisation of the algorithms to semigroups is mentioned in §7. Finally, some conclusions and open problems are stated in §8.

2 Notation and Assumptions

We write $\ln x$ for the natural logarithm and $\lg x$ for $\log_2 x$.

Let G be an arbitrary group with group operation "+" and identity element 0. We write elements of G as $g_0, g_1, \ldots$, and integers as $\alpha_0, \alpha_1, \ldots, \beta, \mu$. It will always be the case that $g_j = \alpha_j g_0$.

Let n be a parameter whose choice depends on the degree of security required. We assume that any event which occurs with probability $O(2^{-n/2})$ can safely be ignored, and that it is infeasible to perform $\Omega(2^{n/2})$ operations. $n = 128$ is adequate for most current purposes, but in the future it might be necessary to increase n to compensate for increases in computer power [30]. At present, $n = 160$ is a conservative choice.

Let B_k denote the set $\{0, 1, 2, \ldots, 2^k - 1\}$ of (unsigned) k-bit binary numbers. We assume that there is a mapping $\gamma : B_n \to G$ of n-bit binary numbers into G.

For encryption of messages it is necessary to split long messages into sufficiently short pieces which are regarded as binary numbers and mapped into G using a mapping such as γ. For signature generation and verification it is sufficient to work with the hash of a message rather than the message itself.

We shall be working in a cyclic subgroup $H = <g_0>$ of G generated by a certain element $g_0 \in G$. Since H is cyclic it is necessarily abelian, even if G is nonabelian. In cryptographic applications H should be finite (so that group elements can be represented in a fixed number of bits), but the algorithms work over countably infinite cyclic groups. Assuming that H is finite, we write $Q_0 = \#H$. Since all computations are performed in H, it is H and Q_0 rather than G and $\#G$ which are of interest in the following.

We assume that the discrete logarithm problem (DLP) in H is computationally difficult. A Pollard rho algorithm [25] may be able to solve it in expected time of order $Q_0^{1/2}$; in general no better algorithms are known. Better algorithms are known in special cases, e.g. if G is the multiplicative group of a finite field [9]. For more on the DLP, see §5.3.

Let $h : \mathcal{M} \cup G \to B_n$ be a good n-bit cryptographic hash function, e.g. SHA-1. Here $\mathcal{M}$ is the space of possible messages. For simplicity we assume that the same hash function can be applied both to messages and to group elements. We define $m = h(M)$, where M is the message of interest and m is its n-bit hash. In the following we assume $m \neq 0$ (if $m = 0$, just set $m \leftarrow 1$).

In §3 we present algorithms for signing and verifying messages, and also (in §3.5) for encrypting and decrypting messages. The signer/receiver is Alice, the verifier/sender is Bob, and the eavesdropper or potential forger of signatures is Eve.

Alice and Bob do not know $\#G$, although Eve may. (In the case that G is the group of an elliptic curve over a finite field, Eve could compute $\#G$ using Schoof's algorithm.)

Alice and Bob have access to a strong, cryptographically secure random number generator R which they can use to select unpredictable and (approximately) uniformly distributed numbers from B_n. We write $R(B_n)$ for a random selection from the set B_n (*not* the same choice each time).

G is public information. It is assumed that Alice and Bob have agreed on a representation of group elements and can perform the group operation, i.e. given $f, g \in G$ they can compute $f + g$ and $f - g$. (In §7 we show that the computation of $f - g$ can be avoided.)

3 The Algorithms

In this section we describe the algorithms for determining the parameters of a cryptosystem, signing a message, and verifying a signature. The security of the system is considered in §5.

3.1 Public and secret information

To determine the cryptosystem, Alice performs the following algorithm.

Algorithm I

1. Choose a random group element $g_0 \leftarrow \gamma(R(B_n))$.
2. Choose a random n-bit number $\alpha_1 \leftarrow R(B_n)$.
3. Repeat step 2 until α_1 passes a probabilistic primality test. [This is optional – see §5.2.]
4. Compute $g_1 \leftarrow \alpha_1 g_0$ in G.
5. Repeat steps 1–4 until $g_1 \neq 0$.
6. The *secret key* is α_1. [This is only known to Alice]
7. The *public key* is (g_0, g_1). [This is known to Alice, Bob and Eve]

3.2 Signing a message

Let M be a message which Alice wishes to sign. She performs the following algorithm.

Algorithm S

1. Compute $m \leftarrow h(M)$.
2. Choose a random $(2n+2)$-bit $\alpha_2 \leftarrow R(B_{2n+2})$.
3. Compute $g_2 \leftarrow \alpha_2 g_0$ and $h(g_2)$.
4. Repeat steps 2–3 until $g_2 \neq 0$ in G.
5. Compute $$\mu \leftarrow \alpha_2 + (m + h(g_2))\alpha_1 \ . \qquad (1)$$
6. If $\mu \in B_{2n+2} \backslash B_{2n+1}$ then *accept* μ; otherwise *reject* μ and return to step 2.
7. The *signature* of M is (μ, g_2).

Comment

The test at step 4 could be omitted, since $g_2 = 0$ with probability $1/Q_0$, which we assume is negligible. The reason for the test at step 6 is explained in §5.2. Step 2 will be executed twice (on average). This number can be reduced to $2^k/(2^k - 2)$ by increasing the number of bits in α_2 to $2n + k$ $(k \geq 2)$ and changing the acceptance criterion at step 6 to $\mu \in B_{2n+k} \backslash B_{2n+1}$.

3.3 Verification

To verify the signature (μ, g_2) on message M, Bob computes $m = h(M)$ and checks if the *verification condition*

$$g_2 + (m + h(g_2))g_1 = \mu g_0 \qquad (2)$$

holds in G. The signature is verified iff (2) holds.

3.4 Explanation

Suppose (1) holds. We have

$$[\alpha_2 + (m + h(g_2))\alpha_1]g_0 = \mu g_0 \ ,$$

but $\alpha_2 g_0 = g_2$, $\alpha_1 g_0 = g_1$, so (2) holds. Thus, (1) implies (2).

Conversely, if (2) holds with $g_1 = \alpha_1 g_0$, we see that g_2 is in the subgroup H generated by g_0. If $g_2 = \alpha_2' g_0$ say, then we see that $\alpha_2' = \alpha_2 \bmod Q_0$, where $Q_0 = \#H$.

The probability that a random message will pass the verification condition is $1/Q_1$, where Q_1 is the order of g_1.

3.5 Encryption, decryption and key agreement

We give the ElGamal algorithms for encryption and decryption in our notation. These do not require knowledge of Q_0, so we do not need to modify them. We assume that γ is invertible on its range (this assumption can be avoided – see §7).

For encryption of a message M (assumed short enough and encoded in B_n), Bob computes random $\alpha_2 = R(B_n)$, $g_2 = \alpha_2 g_0$, $g_3 = \alpha_2 g_1 + \gamma(M)$, and the ciphertext is (g_2, g_3).

For decryption, Alice computes $M = \gamma^{-1}(g_3 - \alpha_1 g_2)$. This works because $\alpha_2 g_1 = \alpha_1 g_2$, so

$$g_3 - \alpha_1 g_2 = \gamma(M) \ .$$

Only Alice can decrypt the ciphertext in this way, because only Alice knows the secret key α_1.

For completeness, we observe that the Diffie-Hellman key agreement protocol and related protocols can easily be expressed in terms of addition in an arbitrary group. They do not require knowledge of the group order [21, Remark 12.49]. In fact, the essential component of El Gamal encryption/decryption is agreement between Alice and Bob on a group element $\alpha_2 g_1 = \alpha_1 g_2$ using the same idea as Diffie-Hellman.

4 Comparison with ElGamal

We express the standard ElGamal algorithms for signatures in our notation to show the similarities and differences. In the standard ElGamal algorithm for signing $h(M)$, Alice does the following:

1. Choose a random $\alpha_2 \leftarrow R(B_n) \bmod Q_0$.
2. Repeat step 1 until $\mathrm{GCD}(\alpha_2, Q_0) = 1$.
3. Compute $g_2 = \alpha_2 g_0$ and $h(g_2)$.
4. Use the extended Euclidean algorithm to find $\beta < Q_0$ such that
$$m = \alpha_1 h(g_2) + \alpha_2 \beta \bmod Q_0 \ . \tag{3}$$
5. The signature is (β, g_2).

To verify a signature, Bob checks the verification condition:

$$0 \le \beta < Q_0 \ \text{ and } \ \beta g_2 + h(g_2) g_1 = m g_0 \ . \tag{4}$$

The range check on β avoids a known attack [21, Note 11.66(iv)]. The condition (4) should be compared with (2).

4.1 Signature size and efficiency comparisons

The signature (μ, g_2) generated by Algorithm S is longer than the signature (β, g_2) generated by the standard ElGamal algorithm, because μ has $2n + 2$ bits versus n bits for β. If the group elements can be represented with about n bits, our signature is about 50% longer than a standard ElGamal signature.

Performing the group operations to generate a signature with Algorithm S could take up to four times as long as for the standard ElGamal algorithm, a factor of two because α_2 has about twice as many bits as β, and a factor of two because of the possibility of repeating steps 2–6 of the algorithm. As mentioned above, the expected number of repeats can be reduced by slightly increasing the length of α_2. Thus, a practical implementation would probably take about twice as long as the standard ElGamal algorithm.

Verification of (2) should not be significantly slower than verification of (4). However, in the standard ElGamal algorithm, verification can be speeded up (by about one third) by computing an inverse mod Q_0, and this option is not available to us.

We conclude that the space and time requirements for our algorithms are greater than for the standard algorithms, but only by moderate factors (in the range 1.5 to 2), if the same group is used.

4.2 Variants of ElGamal signatures

There are many variants of the ElGamal signature scheme: see [21, Table 11.5] and [31, §20.4]. However, so far as we know, none of those in the literature give an acceptable signature without knowledge of Q_0.

For example, in [21, Table 11.5] there are six variants of (3), two of which do not require computation of an inverse mod Q_0. In our notation these two are:

$$\beta = \alpha_1 h(g_2) + \alpha_2 m \bmod Q_0 \tag{5}$$

and

$$\beta = \alpha_1 m + \alpha_2 h(g_2) \bmod Q_0 \ . \tag{6}$$

Consider using (5) *without* any reduction mod Q_0. Eve knows $h(g_2)$, m and β, so she knows a relation of the form

$$a\alpha_1 = b \bmod m \tag{7}$$

If Eve observes several messages $M^{(i)}$ she can obtain several relations of the form

$$a^{(i)}\alpha_1 = b^{(i)} \bmod m^{(i)} \ . \tag{8}$$

Using the Chinese remainder theorem, Eve can deduce the secret key α_1. Similarly for equation (6). Thus, reduction of β mod Q_0 is essential when using (5) or (6) as a signing equation.

Our proposed (1) is not susceptible to this form of attack. This is because the random integer α_2 in (1) is not multiplied by any large constant known to Eve, so Eve can not construct a useful set of modular equations.

5 Security Considerations

5.1 Forging a signature

It seems difficult for Eve to forge a signature (μ, g_2) which satisfies (2) without knowing the secret key α_1. If $h(g)$ behaves like a random function and μ is fixed, then choosing g at random in H will give a solution of

$$g + (m + h(g))g_1 = \mu g_0 \tag{9}$$

with probability about $1/Q_0$. If g is chosen then the left side of (9) is determined, so finding μ involves the solution of a DLP in H. This takes an expected time of order $Q_0^{1/2}$ by a Pollard rho method.

5.2 Finding the secret key

If Eve observes many different messages $M^{(i)}$ and associated signatures $(\mu^{(i)}, g_2^{(i)})$, she may be able to deduce some information about the secret key α_1 from the distribution of $\mu^{(i)}$ or the joint distribution of $(\mu^{(i)}, h(g_2^{(i)}))$. To avoid this possibility, we introduced the "rejection" step 6 in Algorithm S. It is easy to see that the rejection step results in $(\mu^{(i)}, h(g_2^{(i)}))$ being uniformly distributed in $(B_{2n+2} \backslash B_{2n+1}) \times B_n$ (assuming as usual a perfect random number generator and hash function). Thus, the joint distribution of $\mu^{(i)}$ and $h(g_2^{(i)})$ gives away no useful information to Eve. Without the rejection step Eve could deduce the leading bits of α_1 after a sufficiently large number of observations.

If Eve happens to observe two signatures $(\mu^{(1)}, g_2^{(1)})$ and $(\mu^{(2)}, g_2^{(2)})$ with $g_2^{(1)} = g_2^{(2)}$, she can find the secret key. There are Q_0 possible values of g_2, so a coincidence is unlikely until Eve has observed about $Q_0^{1/2}$ signatures.

If Eve finds Q_0 and Q_1 by an algorithm for determining group orders, she can obtain some information about the secret key α_1. In fact she can compute

$$\mathrm{GCD}(Q_0, \alpha_1) = Q_0/Q_1 \ .$$

Although there is only a small probability that the GCD is large, Alice can defend against this attack by choosing α_1 to be a (probable) prime (see the optional step 3 of Algorithm I). Since α_1 is an n-bit number, this ensures that $\mathrm{GCD}(Q_0, \alpha_1) = 1$, and hence $Q_0 = Q_1$, with probability $1 - O(2^{-n})$.

5.3 The discrete logarithm problem

The security of the system depends on the difficulty of solving the DLP in H. Pohlig and Hellman [24] pointed out that the DLP can be reduced to solving DLPs in groups whose orders are the prime power factors $p_j^{\beta_j}$ of $\#H$. In fact, Blake *et al* [1, §V.1] observe that it is sufficient to solve DLPs in groups whose orders are the prime factors p_j of $\#H$. Since we assume that Eve can find $\#H$ and its prime factors, the security of the system depends on the difficulty of solving the DLP in a group of order p, where p is the largest prime factor of $\#H$ (most likely p is also the largest prime factor of $\#G$).

For security we would like $p > 2^n$ with probability $1 - O(2^{-n/2})$, but this is difficult to guarantee without computing $\#G$. Thus, although Alice does not need $\#G$ to apply Algorithm S, she does appear to need $\#G$ to be assured of its security [11]. If she is willing to compute $\#G$ then she may as well use the standard ElGamal algorithm, since it is faster.

It is conceivable that there are families of groups whose orders are not easy to compute, but can be guaranteed to have at least one large prime factor (at least with very high probability).

6 Using Random Groups

Suppose we have a family $\mathcal{G}$ of groups $G_{j,k}$ with the property $\mathcal{P}$ that $2^{k-1} \le \#G_{j,k} < 2^k$ and $\#G_{j,k}$ behaves, at least so far as its largest prime factor is concerned, approximately like a random number drawn from the interval $[2^{k-1}, 2^k)$. Since the analysis is not rigorous, we shall not try to be precise about the meaning of "approximately".

A possible example is a family of groups of elliptic curves over finite fields $\mathrm{GF}(q_k)$, where q_k is a k-bit prime power. There is good evidence, both theoretical [12, Ch. 13][17] and empirical [2], that such groups satisfy property $\mathcal{P}$ if q_k is prime. There is less evidence in the case that q_k is a power of two[2].

Suppose we randomly select a group $G_{j,k} \in \mathcal{G}$. Using property $\mathcal{P}$, we can estimate the probability P that a cryptosystem based on $G_{j,k}$ and the algorithms of §3 is insecure, in the sense that it could be broken by solving the discrete logarithm problem (using the Pohlig-Hellman and Pollard rho algorithms) in $O(2^{n/2})$ group operations. From §5.3, P is the probability that the largest prime factor p of $\#H$ is $O(2^n)$, and this is essentially the same as the probability that the largest prime factor of a random k-bit integer is $O(2^n)$. Thus,

$$P \approx \rho(u) \ ,$$

where $u = k/n$ can be regarded as an *expansion factor*, and $\rho(u)$ is Dickman's function [14, §4.5.4]. For example, with an expansion factor of seven, $P \approx \rho(7) < 10^{-6}$. Table 1 gives a small table of values of $\rho(u)$ for $4 \le u \le 10$.

Table 1: Dickman's function $\rho(u)$

u	$\rho(u)$
4	4.9×10^{-3}
5	3.5×10^{-4}
6	2.0×10^{-5}
7	8.7×10^{-7}
8	3.2×10^{-8}
9	1.0×10^{-9}
10	2.8×10^{-11}

By our definition of n in §2, a probability of $2^{-n/2}$ is negligible. To ensure that P is negligible we need $\rho(u) \approx 2^{-n/2}$. Asymptotically $\ln \rho(u) \sim -u \ln u$, so we need

$$k \sim \frac{n^2}{2 \lg n} \ , \tag{10}$$

which is too large to be competitive with the RSA system [28] for $n \approx 160$.

Our analysis is conservative, because an attacker has to find the orders of about $1/\rho(k/n) \approx 2^{n/2}$ groups, with each group order computation taking at least k^6 operations with the best currently-known algorithms, before even finding a group whose discrete logarithm problem is feasible in $O(2^{n/2})$ group operations. Also, each group operation probably requires many arithmetic operations [26].

If the asymptotic analysis of the number field sieve (NFS) factorisation algorithm [16] is used to estimate the equivalent keysize k_{RSA} for RSA [28], much as in [1, §I.3], then we see that

$$k_{RSA} \sim \frac{n^3}{128(\lg n)^2} \ ,$$

so $k \ll k_{RSA}$ as $n \to +\infty$.

[2]It may be desirable to insist that $q_k - 1$ has a prime factor p_1 which is "large" in the sense that $p_1 > \sqrt{q_k} + 1$, since this implies that the group G has a large cyclic subgroup: see Lemma 1 in the Appendix.

7 Generalisation to Semigroups

Observe that the algorithms of §§3.1–3.3 do not require the existence of inverses in G. All that is required is that $+$ is an associative operation on G. Thus, it is sufficient for G to be a semigroup [3].

We can modify the algorithms of §3.5 to work if G is a semigroup. Instead of taking $g_3 = \alpha_2 g_1 + \gamma(M)$, Bob takes $\lambda = h(\alpha_2 g_1) \oplus M$, where $\oplus$ is bitwise exclusive or. The ciphertext is (g_2, λ). Alice can use the relation $\alpha_2 g_1 = \alpha_1 g_2$ to decrypt the ciphertext and obtain $M = h(\alpha_1 g_2) \oplus \lambda$.

More generally, instead of using exclusive or, M can be encrypted using an arbitrary symmetric encryption algorithm with key $h(\alpha_2 g_1)$, say $M \mapsto E = E(h(\alpha_2 g_1), M)$, and the ciphertext is (g_2, E). Alice can decrypt using $M = D(h(\alpha_1 g_2), E)$. This shows that the essential ingredient is a Diffie-Hellman key agreement between Alice and Bob [21, §12.6.1].

Why use a semigroup? It may be more difficult to solve the discrete logarithm problem in a suitable semigroup than in a group which is representable in the same number of bits, because the Pohlig-Hellman algorithm depends on the group structure. In particular, it depends on Lagrange's theorem [21, §2.171], which implies that the order of g_0 is a divisor of $\#G$.

For a system using semigroups to be practical, we need to be able to compute kg_0 in $O(\lg k)$ operations, which severely restricts the semigroups which can be used. One example is a semigroup of (possibly singular) $r \times r$ matrices over a ring.

8 Conclusions and Open Problems

It is not essential to compute the group order in order to implement public key cryptography using an arbitrary group. Our algorithms apply to any group (or semigroup) and are potentially useful provided computation of discrete logarithms in the group (or semigroup) is difficult. However, in practice it seems hard to guarantee the difficulty of the discrete logarithm problem over a group without computing the group order.

We suggest some open problems:

1. Are there families of groups which would give better performance with the same level of security as the family of groups of elliptic curves considered in §6?

2. Our algorithms only make use of an (abelian) cyclic subgroup of G. Algorithms exist for finding abelian "hidden subgroups" in polynomial time on quantum computers [33]. Thus, it is of interest to ask if there are algorithms for public key cryptography which make use of nonabelian groups G in a nontrivial way?

3. Are there suitable semigroups which would make the generalisation of §7 attractive?

8.1 Acknowledgements

Thanks to Peter Montgomery, John Pollard and Nigel Smart for their advice and assistance.

9 Appendix – a Lemma on Elliptic Curves

The following Lemma gives a simple condition under which the group of an elliptic curve over a finite field can be guaranteed to have a large cyclic subgroup. This is a necessary, but not sufficient, condition for the group order to have a large prime factor.

Lemma 1 *Let G be the group of an elliptic curve over $GF(q)$, where $q = p^k$ is a prime power and $q - 1 = cp_1$, where p_1 is prime and $p_1 > \sqrt{q} + 1$. Then G has a cyclic subgroup of order at least $\#G/c$.*

Proof. By Mordell's Theorem [27, Theorem A7.3], G can be written as

$$G \approx T_1 \oplus T_2 \ ,$$

where, writing $t_i = \#T_i$, we have $t_1|t_2$ and $t_1|q-1$. If $p_1|t_1$ then

$$\#G = t_1t_2 \geq t_1^2 \geq p_1^2 \ .$$

By Hasse's Theorem [10],

$$\#G \leq q + 1 + 2\sqrt{q} = (\sqrt{q} + 1)^2 \ ,$$

so

$$p_1 \leq \sqrt{q} + 1 \ ,$$

contradicting the condition $p_1 > \sqrt{q} + 1$. Thus $t_1|c$ and $\#G = t_1t_2 \leq ct_2$, so

$$t_2 \geq \#G/c \ .$$

□

References

[1] I. F. Blake, G. Seroussi and N. P. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series, Vol. 265, Cambridge University Press, 1999.

[2] R. P. Brent, Factorization of the tenth Fermat number, *Math. Comp.* **68** (1999), 429–451.

[3] A. H. Clifford and G. B. Preston, *The Algebraic Theory of Semigroups*, Vol. 1, second edition, Mathematical Surveys Number 7, Americal Mathematical Society, Providence, Rhode Island, 1964.

[4] J. M. Couveignes, *Quelques calculs en théorie des nombres*, thèse, Université de Bordeaux I, 1994.

[5] J. M. Couveignes, Computing l-isogenies with the p-torsion, in Proc. ANTS-II, *Lecture Notes in Computer Science* **1122**, Springer-Verlage, 1996, 59–65.

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

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

[8] N. D. Elkies, Elliptic and modular curves over finite fields and related computational issues, in *Advances in Cryptology, Proc. ASIACRYPT'98*, *Lecture Notes in Computer Science* **1514**, 1998, 21–76.

[9] D. Gordon, Discrete logarithms in GF(p) using the number field sieve, *SIAM J. on Discrete Mathematics* **6** (1993), 124–138.

[10] H. Hasse, *Beweis des Analogons der Riemannschen Vermutung für die Artinschen u. F. K. Schmidtschen Kongruenzzetafunktionen in gewissen elliptischen Fällen,* Nachr. Gesell. Wissen. Göttingen I **42** (1933), 253–262.

[11] J. Heller, *Catch–22*, Doubleday, 1961.

[12] D. Husemöller, *Elliptic Curves*, Springer-Verlag, New York, 1987.

[13] D. B. Johnson and A. J. Menezes, Elliptic curve DSA (ECDSA): an enhanced DSA, ECC Whitepapers, available from `http://www.certicom.com` .

[14] D. E. Knuth, *The Art of Computer Programming*, vol. 2, third edition, Addison Wesley, 1998.

[15] N. Koblitz, Elliptic curve cryptosystems, *Math. Comp.* **48** (1987), 203-209.

[16] A. K. Lenstra and H. W. Lenstra, Jr. (editors), *The Development of the Number Field Sieve*, Lecture Notes in Mathematics **1554**, Springer-Verlag, 1993.

[17] H. W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics (2)* **126** (1987), 649–673.

[18] R. Lercier, *Algorithmique des courbes elliptiques dans les corps finis*, thèse, École Polytechnique, Paris, 1997. Available from `http://ultralix.polytechnique.fr/~lercier/french/pub.html`.

[19] R. Lercier and F. Morain, Counting the number of points on elliptic curves over finite fields: strategies and performances, *Advances in Cryptology, Proc. EUROCRYPT'95*, *Lecture Notes in Computer Science* **921**, Springer-Verlag, 1995, 79–94.

[20] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[21] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.

[22] V. Miller, Uses of elliptic curves in cryptography, *Advances in Cryptology, Proc. CRYPTO'85*, *Lecture Notes in Computer Science* **218**, Springer-Verlag, 1986, 417–426.

[23] F. Morain, *Isogeny computations and point counting on elliptic curves*, 1998, available from `ftp://lix.polytechnique.fr/pub/submissions/morain/Preprints/ecc98.ps.gz`

[24] G. C. Pohlig and M. E. Hellman, An improved algorithm for computing logarithms over GF(q) and its cryptographic significance, *IEEE Trans. on Information Theory* 24 (1978), 106-110.

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

[26] M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning Publications, Greenwich, CT 06830, USA.

[27] H. Riesel, *Prime Numbers and Computer Methods for Factorization*, second edition, Birkhäuser, Boston, 1994.

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

[29] A. Salomaa, *Public-Key Cryptography*, second edition, Springer-Verlag, Berlin, 1996.

[30] R. S. Schaller, Moore's law: past, present and future, *IEEE Spectrum* **34**, 6 (June 1997), 52–59.

[31] B. Schneier, *Applied Cryptography*, second edition, John Wiley and Sons, New York, 1996.

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

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

[34] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, Vol. 106, Springer-Verlag, New York, 1986.