Generation Methods of Elliptic Curves

by

Harald Baier and Johannes Buchmann

August 27, 2002

An evaluation report for the

Information-technology Promotion Agency, Japan

Contents

Chapter 1

Introduction

1.1 Preface

Let q be a prime power, and let E be an elliptic curve over the field $\mathbb{F}_q$ of q elements. As usual we associate to E a finite set called *the set of rational points of E over $\mathbb{F}_q$*. We denote this set by $E(\mathbb{F}_q)$. We will explain these terms in Chapter 2. Once we know that $E(\mathbb{F}_q)$ actually is a finite Abelian group, we may define the discrete logarithm problem in $E(\mathbb{F}_q)$ as usual. However, since the use of elliptic curves in cryptography, various algorithms to solve the discrete logarithm problem in the group of rational points of an elliptic curve have been found. Hence, in order to keep the discrete logarithm problem intractable, we have to choose the elliptic curve diligently.

As of today the security of an elliptic curve cryptosystem is determined by the cardinality of $E(\mathbb{F}_q)$. Thus in order to decide whether a group of rational points is suitable for use in cryptography, we have to know its group order. It turns out that in general this is a burdensome and nontrivial task. The following methods are known to find a suitable group.

The first approach, mostly referred to as the *random approach,* first chooses a random curve E. Using point counting algorithms, the group order of $E(\mathbb{F}_q)$ is determined. Once the cardinality is known, we can decide whether the group is suitable for use in cryptography or not. If it turns out that the curve does not yield a secure cryptosystem, a new elliptic curve is chosen.

The second method makes use of the theory of complex multiplication. It is therefore referred to as the *complex multiplication method.* We abbreviate this method by CM-method. Its proceeding is quite different from the random approach. In the complex multiplication method one first searches for candidates of a suitable group cardinality. This can be done without knowing the corresponding elliptic curves. Once a suitable cardinality is found, the elliptic curve is determined using complex multiplication.

Finally, let $q = p^n$ be a prime power with $n > 1$. In addition, let m be a positive divisor of n, $m \neq n$. If E is defined over $\mathbb{F}_{p^m}$ and if we know the group order of $E(\mathbb{F}_{p^m})$, a theorem of Weil may be used to get $|E(\mathbb{F}_q)|$. The use of Weil's theorem was first proposed by Koblitz ([Kob92]). Thus we refer to this method as the *Koblitz approach.* We remark that in [X9.62] this approach is called the Weil method.

In this document we report on these methods to find a suitable group. We give both theoretical and practical run times of all methods, and we compare the advantages and disadvantages of either algorithm.

This report is organized as follows: The subsequent section lists the notation we use in this document. In Chapter 2 we introduce the security conditions we have to impose on an elliptic curve group. Next, in Chapter 3 we discuss in detail all known algorithms to find such a group. Finally, Chapter 4 deals with implementation issues.

1.2 Notation

In this evaluation report we use a notation similar to [P1363].

p	a rational prime
q	a power of p, i.e. $q = p^n$
$\mathbb{F}_q$	the finite field of q elements
E	an elliptic curve over a finite field
(a, b)	the parameters of an elliptic curve
$E(\mathbb{F}_q)$	the group of rational points of E over the field $\mathbb{F}_q$
$\|E(\mathbb{F}_q)\|$	the group order of $E(\mathbb{F}_q)$
O	the point at infinity
G	a base point of an elliptic curve cryptosystem
r	the cryptographic prime factor
k	the cofactor

In the framework of the CM-method we make use of the following symbols.

Δ	an imaginary quadratic discriminant
$\mathcal{O}_\Delta$	the imaginary quadratic order of discriminant Δ
$h(\Delta)$	the class number of discriminant Δ
$h_c(\Delta)$	the crossover class number

As usual we describe the complexity of an algorithm in terms of its *bit-complexity*. The bit-complexity estimates the number of basic operations a processor has to perform when executing an algorithm. Throughout this document we estimate the bit-complexity of an algorithm as its 'ordinary' bit-complexity, that is we assume that a schoolbook implementation of the algorithm is used. From a practical point of view this is more reasonable than using theoretical bit-complexities of optimized algorithms. Often the optimized variants are not implemented, as their complexity is only asymptotically superior to the ordinary algorithm.

Furthermore by log we denote the natural logarithm, that is the logarithm to the base e. In addition, the logarithm to the base 2 is written as $\log_2$.

Chapter 2

Elliptic Curves in Cryptography

In this chapter we review the security conditions we have to impose on an elliptic curve group for use in cryptography. As usual we distinguish two different cases. First, in Section 2.1 we list the requirements if $q = p$ is a large prime. Second, in Section 2.2 we turn to elliptic curves defined over a finite field of characteristic 2.

2.1 Elliptic Curve Groups over $\mathbb{F}_p$

Let $q = p$ be a prime, $p \geq 5$. An *elliptic curve* over $\mathbb{F}_p$ is a pair $E = (a, b) \in \mathbb{F}_p^2$ with $4a^3 + 27b^2 \neq 0$. A *point* on E is a solution $(x, y) \in \mathbb{F}_p^2$ of $y^2 = x^3 + ax + b$ or the point at infinity O obtained by considering the projective closure of this equation. The set of points on E over $\mathbb{F}_p$ is denoted by $E(\mathbb{F}_p)$. It carries a group structure with the point at infinity acting as the identity element. It is called the *group of rational points of E over $\mathbb{F}_p$*.

In the scope of this report we call the elliptic curve group $E(\mathbb{F}_p)$ cryptographically strong if it satisfies the following conditions which make the cryptosystems, in which $E(\mathbb{F}_p)$ is used, secure and efficient.

We first consider security. If $E(\mathbb{F}_p)$ is used in a cryptosystem, the security of this cryptosystem is based on the intractability of the discrete logarithm problem in $E(\mathbb{F}_p)$. Several discrete logarithm algorithms are known. To make their application impossible, we require that $E(\mathbb{F}_p)$ satisfies the following conditions.

1. We have $|E(\mathbb{F}_p)| = k \cdot r$ with a prime $r > 2^{160}$ and a positive integer k.
2. The primes r and p are different.
3. The order of p in the multiplicative group $\mathbb{F}_r^\times$ of $\mathbb{F}_r$ is at least B, where $B \geq 20$.

The first condition excludes the application of generic discrete logarithm algorithms. Their running time is roughly the square root of the largest prime factor of the group order (see for example [vOW99]). We make use of the bound 2^{160} as proposed in [X9.62], as this bound is consensus in the cryptographic community. The second condition makes the anomalous curve attack impossible (see [SA98], [Sem98], [Sma99]).

The last condition excludes the attacks of Menezes, Okamoto, Vanstone ([MOV91]), and the attack of Frey, Rück ([FR94]). Both methods reduce the discrete logarithm problem in $E(\mathbb{F}_p)$ to the discrete logarithm problem in a finite extension field of $\mathbb{F}_p$. The degree of this extension over $\mathbb{F}_p$ is at least the order of p in $\mathbb{F}_r^\times$, where in general equality holds, as shown in [BK98]. The third condition is based on the assumption that the discrete logarithm problem in a finite field of order of magnitude p^B is intractable. The bound $B = 20$ is explicitely given in [SEC1]. In [X9.62] the standard requires $B \geq 21$ (Annex A.1.1 of [X9.62]). Furthermore, we point out that the German Information Security Agency [GIS01] requires $B \geq 10^4$. However, if the first condition holds and if $B \geq 20$, an attacker will have to compute a discrete logarithm in a finite field of order of magnitude at least 2^{3200}. This is currently not possible, and following Lenstra/Verheul ([LV01]) will stay impossible for at least the next 40 years. We therefore consider $B \geq 20$ to be a good choice.

Let us now consider efficiency. Suppose that an elliptic curve E over a prime field $\mathbb{F}_p$ satisfies the security conditions. If this curve is used in a cryptosystem, the efficiency of this system depends on the efficiency of the arithmetic in $\mathbb{F}_p$. So p should be as small as possible. It follows from a theorem of Hasse that

$$(\sqrt{|E(\mathbb{F}_p)|} - 1)^2 \leq p \leq (\sqrt{|E(\mathbb{F}_p)|} + 1)^2 \ . \tag{2.1}$$

Hence, we try to make $|E(\mathbb{F}_p)|$ as small as possible. Now the first security condition implies

$$|E(\mathbb{F}_p)| = k \cdot r \tag{2.2}$$

with a prime number $r > 2^{160}$ and a positive integer k, the so called *cofactor*. The security of the cryptosystem, in which $E(\mathbb{F}_p)$ is used, is based on the intractability of the discrete logarithm problem in the subgroup of order r in $E(\mathbb{F}_p)$. This security is independent of k. Therefore, k can be as small as possible. An explicit bound of k is given in [SEC1]. We therefore refine the first security condition as follows:

4. We have $|E(\mathbb{F}_p)| = k \cdot r$ with a prime number $r > 2^{160}$ and a positive integer $k \leq 4$.

We remark that this requirement is stricter than the notion of trialdivision with a bound 255, as proposed in [X9.62], Annex A.3.2 . However, our practical results give evidence that curves with $k \leq 4$ are found in reasonable time.

We explain an additional security condition required by the German Information Security Agency GISA ([GIS01]). The third condition implies that the endomorphism ring $\mathrm{End}(E(\overline{\mathbb{F}}_p))$ of the elliptic curve over the algebraic closure of $\mathbb{F}_p$ is an imaginary quadratic order. The GISA requires the following.

5. The class number of the maximal order which contains $\mathrm{End}(E(\overline{\mathbb{F}}_p))$ is at least 200.

The reason for this condition is that among all curves over a prime field only very few have endomorphism rings with small class numbers. So those curves may be subject to specific attacks. However, no such attacks are known. As this condition is not considered in any international cryptographic standard, we do not take it into account.

To summarize we say that an elliptic curve group $E(\mathbb{F}_p)$ is *cryptographically strong* if it satisfies the conditions from Table 2.1. They are labelled (O1) - (O3), where 'O' stands for odd.

(O1)	$\lvert E(\mathbb{F}_p)\rvert = k \cdot r$, $r > 2^{160}$ prime, $k \leq 4$
(O2)	$p \neq r$
(O3)	$p^s \not\equiv 1 \bmod r$, $1 \leq s < 20$

Table 2.1: Security conditions for an elliptic curve group $E(\mathbb{F}_p)$

2.2 Elliptic Curve Groups over $\mathbb{F}_{2^n}$

In this section let $q = 2^n$. An *elliptic curve* over $\mathbb{F}_{2^n}$ is a pair $E = (a, b) \in \mathbb{F}_{2^n}^2$ with $b \neq 0$. A *point* on E is a solution $(x, y) \in \mathbb{F}_{2^n}^2$ of $y^2 + xy = x^3 + ax^2 + b$ or the point at infinity O. Again the set of points on E over $\mathbb{F}_{2^n}$ is denoted by $E(\mathbb{F}_{2^n})$ and again it carries a group structure with the point at infinity acting as the identity element.

The security and efficiency conditions on $E(\mathbb{F}_{2^n})$ are similar to the requirements of Section 2.1. We summarize them in Table 2.2. They are labelled (E1) - (E3), where 'E' stands for even.

(E1)	$\lvert E(\mathbb{F}_{2^n})\rvert = k \cdot r$, $r > 2^{160}$ prime, $k \leq 4$
(E2)	$2^{ns} \not\equiv 1 \bmod r$, $1 \leq s < 20$
(E3)	n is prime

Table 2.2: Security conditions for an elliptic curve group $E(\mathbb{F}_{2^n})$

We remark that the anomalous curve condition is $r \neq 2$ in this case. As (E1) requires r to be a large prime, the anomalous curve condition follows from (E1). In addition we point out that the primality of n is not required by any standard. However, recent results in the framework of the Weil descent make the condition (E3) necessary (see [GHS02a], [GHS02b]).

Finally, we remark that in contrast to the GISA we do not require that E may not be defined over $\mathbb{F}_2$. Hence we allow the use of the two elliptic curves $(0, 1)$ and $(1, 1)$ as proposed by Koblitz ([Kob92]).

Chapter 3

Generation Methods

In this chapter we discuss various methods to find a cryptographically strong elliptic curve group. The task we have to solve is as follows: Let r_0 and k_0 be positive integers with $r_0 \geq 2^{160}$ and $k_0 \leq 4$. In addition to the requirements of Chapter 2 we have to find an elliptic curve group whose order factors as $k \cdot r$ with $r \geq r_0$ and $k \leq k_0$. Thus the integers r_0 and k_0 serve as bounds for r and k, respectively, to define an individual security and efficiency level.

Before actually investigating the generation methods we turn to the important question of primality testing.

3.1 Primality Tests

Testing integers for primality is an important task in public key cryptography. However, as primality proving is rather slow *probabilistic* primality tests are used in practice. The term 'probabilistic' means that the primality test may output a wrong answer.

Throughout the different cryptographic standards the Miller-Rabin test is proposed for use in practice (e.g. [X9.62], Annex A.2.1). Let i be an integer and T a positive integer. If the Miller-Rabin test says that i is composite, the answer is true. However, if the Miller-Rabin test claims i to be prime, the answer is wrong with a probability at most $1/4$. Thus in order to decrease the error probability, the Miller-Rabin test is performed independently T times for the input i. A common number of independent tests is $T = 50$, as proposed in [X9.62], Annex A.2.1 . Then the probability of accepting a composite i as prime number is at most 2^{-100}. This error bound is sufficient for practical applications. We write $\texttt{isPrime}(i, 50)$ to denote the Miller-Rabin test. $\texttt{isPrime}(i, 50)$ returns `false`, if i is shown to be composite within at most 50 tests. It returns `true` otherwise.

Finally, the bit-complexity of $\texttt{isPrime}(i, 50)$ is $O(\log^3 i)$ ([Coh95]). We remark that very recently a deterministic polynomial time primality test was published ([AKS02]). However, its applicability in practice is not yet clear.

3.2 Finding Suitable Elliptic Curve Groups over $\mathbb{F}_p$

In this section we present two methods to find a cryptographically strong elliptic curve group over a finite prime field. First, the random approach is discussed in Section 3.2.1. Second, we present the CM-method in Section 3.2.2. We remark that the Koblitz approach is not applicable in the case of a finite prime field. In Section 3.2.3 we show how to find a point of order r. Finally, in Section 3.2.4 we compare the random approach to the CM-method.

Before turning to the algorithms we describe the algorithm `isStrongP`(r_0, k_0, p, N). It requires positive integers r_0 and k_0 as input with $r_0 \geq 2^{160}$ and $k_0 \leq 4$, respectively. In addition the algorithm gets a prime p and a positive integer N. The algorithm implements the requirements (O1) - (O3) of Table 2.1, where we substitute 2^{160} by r_0 and 4 by k_0 in (O1). It returns a prime r if $N = k \cdot r$ is the order of a cryptographically strong elliptic curve group over $\mathbb{F}_p$ with $r \geq r_0$ and $k \leq k_0$. Otherwise `isStrongP` returns 0.

Algorithm 3.1: `isStrongP`(r_0, k_0, p, N)

Input: Positive integers r_0 and k_0 with $r_0 \geq 2^{160}$ and $k_0 \leq 4$.
A prime p and the order N of a group of rational points of an elliptic curve over $\mathbb{F}_p$.

Output: A prime r if $N = k \cdot r$ is the order of a cryptographically strong elliptic curve group over $\mathbb{F}_p$ with $r \geq r_0$ and $k \leq k_0$, and 0 otherwise.

```
1: //check if N is in the Hasse interval
2: if |N − (p + 1)| > 2√p then
3:    return (0);
4: r ← 0; k ← 0; //initialize both r and k with 0
5: //check condition (O1) by trialdivision
6: for i ← 1; i ≤ k0; i ← i + 1 do
7:    if i | N AND isPrime(N/i, 50) = true AND N/i ≥ r0 then
8:       r ← N/i; k ← i; break;
9: if r = 0 then
10:   return (0);
11: //check condition (O2)
12: if p = r then
13:   return (0);
14: //check condition (O3)
15: p_r ← 1 mod r;
16: for i ← 1; i ≤ 19; i ← i + 1 do
17:   p_r ← p · p_r mod r;
18:   if p_r = 1 then
19:      return (0);
20: return (r);
```

We estimate the bit-complexity of `isStrongP`(r_0, k_0, p, N). First, the computation of $\sqrt{p}$ in line 2 and the trialdivision in line 7 are negligible. As we have $N = O(p)$, the Miller-Rabin test in line 7 is of bit-complexity $O(\log^3 p)$. Finally, the multiplication and reduction modulo r in line 17 are of bit-complexity at most $O(\log^3 p)$, too. Thus in all `isStrongP`(r_0, k_0, p, N) is of bit-complexity $O(\log^3 p)$.

3.2.1 Random Approach

We present an algorithm to randomly generate a cryptographically strong elliptic curve group $E(\mathbb{F}_p)$. We denote this algorithm by `randomApproachP`(r_0, k_0). Its input are positive integers r_0 and k_0 with $r_0 \geq 2^{160}$ and $k_0 \leq 4$. The algorithm outputs a prime p, positive integers r and k, and an elliptic curve E such that $|E(\mathbb{F}_p)| = k \cdot r$ and such that `isStrongP`$(r_0, k_0, p, k \cdot r)$ returns r.

The first task is to find a prime p. As of today no attacks on elliptic curve cryptosystems are known which exploit special properties of some field $\mathbb{F}_p$. Thus the choice of the prime p is not critical. However, we have to consider the boundary conditions $r \geq r_0$ and $k \leq k_0$. We write b for the bitlength of $k_0 \cdot r_0$. We propose to choose p such that $k_0 \cdot r_0 \leq p \leq 2^b$. The method `getPrime`$(r_0, k_0)$ returns such a prime. The user may choose his own implementation of `getPrime`. For instance, one may want to use primes in the interval $[k_0 \cdot r_0, 2^b]$ which are generated by some pseudorandom number generator as described in FIPS 186 ([FIPS186]).

Once p is known the further proceeding is as follows: Choose parameters a and b with $4a^3 + 27b^2 \not\equiv 0 \bmod p$, determine the order of the group of rational points of the curve (a, b) over $\mathbb{F}_p$, and finally check if this group is cryptographically strong.

We first explain how to choose a and b. It is common to choose parameters verifiably at random. This method is e.g. explained in [X9.62], Annex A.3.3.2 . The basic idea is to make use of the one-way property of a cryptographic hash function. By h we denote such a hash function and by l the length in bits of the output of h. We assume $l \geq 160$ (e.g. SHA-1 or RIPEMD-160). In order to generate a curve verifiably at random one first chooses a bitstring of length at least l. We write SEED for this string. Once SEED is known the value h(SEED) is used to compute a and b deterministically by a publicly known algorithm. Thus if we provide SEED, the hash function h, and the deterministic algorithm to compute (a, b) from h(SEED), any entity may verify that a and b actually are computed using SEED. The one-way property of h guarantees that the parameters actually are chosen at random. In this report we write `getParametersP`(p, SEED) for any algorithm which returns an elliptic curve E defined over $\mathbb{F}_p$ verifiably at random.

If the curve $E = (a, b)$ is chosen we have to determine the group order of $E(\mathbb{F}_p)$. Currently, the best known algorithm for this task is the SEA-algorithm. The SEA-algorithm is due to Schoof, Elkies and Atkin (see for instance [Mül95], [BSS99]). We write `SEA`(p, E). It requires a prime p and an elliptic curve E defined over $\mathbb{F}_p$ as input. The algorithm returns $|E(\mathbb{F}_p)|$. We denote the result of `SEA`(p, E) by N. If `isStrongP`$(r_0, k_0, p, N) \neq 0$ we are done. Otherwise we have to invoke `getParametersP`(p, SEED), `SEA`(p, E), and `isStrongP`(r_0, k_0, p, N) until we succeed.

We point out two methods for speeding up `randomApproachP`. First, one may use an *early-abort-strategy*. The fundamental idea of the SEA-algorithm is to write $N = p + 1 - t$, where t is called the *trace* of E over $\mathbb{F}_p$. The SEA-algorithm computes the trace t modulo some small primes p_i. Then t is recovered using the Chinese Remainder Theorem. If we know $t \bmod p_i$ for such a small prime, we can check if $p_i \mid N$. Thus if this is true and if in addition $p_i > k_0$, the condition $k \leq k_0$ is false for the current chosen curve. This early-abort-strategy has no cryptographic implications.

A second enhancement is due to the fact that $|E'(\mathbb{F}_p)| = p + 1 + t$, where E' denotes a twisted elliptic curve of E over $\mathbb{F}_p$ (the term twisted elliptic curve is defined in Section 3.2.2). Thus

Algorithm 3.2: `randomApproachP`(r_0, k_0)

Input: Positive integers r_0 and k_0 with $r_0 \geq 2^{160}$ and $k_0 \leq 4$.
Output: Primes p and r, and a positive integer k.
An elliptic curve E over $\mathbb{F}_p$ with $|E(\mathbb{F}_p)| = k \cdot r$ and such that `isStrongP`$(r_0, k_0, p, k \cdot r) = r$.

1: $p \leftarrow$ `getPrime`(r_0, k_0);
2: **while** `true` **do**
3: $E \leftarrow$ `getParametersP`(p, SEED);
4: $N \leftarrow$ `SEA`(p, E);
5: $r \leftarrow$ `isStrongP`(r_0, k_0, p, N);
6: **if** $r \neq 0$ **then**
7: return $(p, E, r, N/r)$;

if $E(\mathbb{F}_p)$ turns out to fail the test `isStrongP`, we may test $E'(\mathbb{F}_p)$ without performing the SEA-algorithm. However, this approach is not covered by the above mentioned algorithm to choose a curve verifiably at random. Nevertheless, if E is chosen verifiably at random, it is easy to extend the above algorithm to allow the use of E', too.

Depending on the implemented SEA-algorithm the bit-complexity of `randomApproachP`(r_0, k_0) may be shown to be $O(\log^{5+\epsilon} k_0 \cdot r_0)$ up to $O(\log^7 k_0 \cdot r_0)$ where $\epsilon > 0$ (see [BSS99], [Bai02b]).

3.2.2 Complex Multiplication Approach

In this section we discuss the CM-method to find an elliptic curve group over $\mathbb{F}_p$. It is out of the scope of this report to present in detail the theory of complex multiplication. We remark that none of the relevant standards comprises a detailed algorithm if a security level r_0 and k_0 is defined in advance. We therefore sketch the algorithm `cryptoCurve` as developed in [Bai02a]. A rather abstract description of an algorithm using the CM-method may be found in the standards of IEEE ([P1363]) or of ANSI ([X9.62], [X9.63]).

The central term in the framework of the CM-method is that of an *imaginary quadratic discriminant*. We denote such a discriminant by Δ. It is a negative integer with $\Delta \equiv 0, 1 \bmod 4$. By $\mathcal{O}_\Delta$ we denote the *imaginary quadratic order* of discriminant Δ, that is we have $\mathcal{O}_\Delta = \mathbb{Z}[\frac{\Delta+\sqrt{\Delta}}{2}]$. In addition we write $h(\Delta)$ for the class number of $\mathcal{O}_\Delta$. If p is a prime number then p is said to be a norm in $\mathcal{O}_\Delta$ if integers t, y exist such that

$$t^2 - \Delta y^2 = 4p \; . \tag{3.1}$$

If p is a norm in $\mathcal{O}_\Delta$, using complex multiplication, elliptic curves $E_{1,p}$ and $E_{2,p}$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O}_\Delta$ and

$$|E_{1,p}(\mathbb{F}_p)| = p + 1 - t, \quad |E_{2,p}(\mathbb{F}_p)| = p + 1 + t \tag{3.2}$$

can be constructed as follows (see [AM93], [BSS99], [Bai02a]).

Let $H \in \mathbb{Z}[X]$ be the minimal polynomial of $j(\frac{\Delta+\sqrt{\Delta}}{2})$ where j is the elliptic modular function. The degree of H is $h(\Delta)$. Modulo p the polynomial H splits into linear factors. Let j_p be a zero of H mod p that is, j_p is an integer such that $H(j_p) \equiv 0 \bmod p$. We assume $\Delta < -4$ in

what follows (the cases $\Delta = -3$ and $\Delta = -4$ are not covered in this report). Then we have $j_p \notin \{0; 1728\}$. Let s_p be a quadratic nonresidue mod p. With

$$\kappa_p = \frac{j_p}{1728 - j_p}, \quad (a_p, b_p) = (3\kappa_p, 2\kappa_p) \tag{3.3}$$

we have

$$\{E_{1,p}, E_{2,p}\} = \{(a_p, b_p), (a_p s_p^2, b_p s_p^3)\}. \tag{3.4}$$

The elliptic curves $E_{1,p}$ and $E_{2,p}$ are said to be twisted elliptic curves over $\mathbb{F}_p$. After this construction it is not known which of the curves is $E_{1,p}$ and which is $E_{2,p}$. However by choosing points on each curve and testing whether their order is a divisor of $p + 1 + t$ or $p + 1 - t$, the curves $E_{1,p}$ and $E_{2,p}$ can be identified.

The crucial observation is that we can decide whether one of the groups $E_{1,p}(\mathbb{F}_p)$ or $E_{2,p}(\mathbb{F}_p)$ is cryptographically strong before we actually construct those curves. We only need to know the prime number p and its representation (3.1). Then we know the group orders of $E_{1,p}(\mathbb{F}_p)$ and $E_{2,p}(\mathbb{F}_p)$ from (3.2). Using those orders and algorithm `isStrongP` we can check the security conditions (O1), (O2), and (O3).

In general most of the time is spent to compute the polynomial H. The reason is that the coefficients of H become rather large, even for a discriminant of a small class number. However, as explained in [P1363], [X9.62], [X9.63], and [Bai02a] depending on the value $\Delta \bmod 24$ one may use alternative polynomials whose coefficients are very small compared to H. Although working with these polynomials accelerates the CM-method significantly in practice, the bit-complexity of the CM-method is invariant. We remark that Enge and Morain recently proposed further alternative polynomials speeding up the CM-method ([EM02]).

Let h_0 be a positive integer (its meaning will become clear soon). In [Bai02a] the algorithm `cryptoCurve`(r_0, k_0, h_0) is described. It implements the above proceeding. The input parameter h_0 allows to choose an individual lower bound of the class number of the imaginary quadratic discriminant in use. Its output is a discriminant Δ with $h(\Delta) \geq h_0$, a prime p of bitlength $\lfloor \log_2 k_0 \cdot r_0 \rfloor + 1$, a prime r with $r \geq r_0$, a positive integer $k \leq k_0$, and an elliptic curve group $E(\mathbb{F}_p)$ of order $k \cdot r$ such that `isStrongP`$(r_0, k_0, p, r \cdot k)$ returns r. In addition the algorithm returns a base point $G \in E(\mathbb{F}_p)$ of order r. An algorithm to find such a point G is described in Section 3.2.3.

In `cryptoCurve` one may choose a prime p in advance, too. It is shown in [Bai02a] that the bit-complexity of `cryptoCurve`(r_0, k_0, h_0) is at most $O(\log^4 r_0 k_0 (\log r_0 k_0 + h_0^2 \log h_0 \log\log h_0) + h_0^6 \log h_0)$, if p is not given. As explained in Section 1.2 the term 'at most' means that we do not assume to work with optimized algorithms.

3.2.3 Finding a Point of Large Prime Order

We explain how to find a base point G. Let a prime p and an elliptic curve E over $\mathbb{F}_p$ be given. As usual we write $|E(\mathbb{F}_p)| = k \cdot r$ with a prime r. We then choose a random element $x_0 \in \mathbb{F}_p$, that is we uniformly take a non-negative integer x_0, $x_0 < p$. If $x_0^3 + ax_0 + b$ is a square in $\mathbb{F}_p$, we denote by y_0 a square root of $x_0^3 + ax_0 + b$ in $\mathbb{F}_p$. Otherwise we choose new random values x_0 until we succeed.

We may assume $1 \leq y_0 \leq p - 1$. y_0 may be computed using Shank's RESSOL algorithm ([Coh95]) or the algorithm in Annex D.1.4 of [X9.62]. One may flip a coin to choose a root

$1 \leq y_0 \leq \frac{p-1}{2}$ or $\frac{p+1}{2} \leq y_0 \leq p-1$. Then (x_0, y_0) is a point in $E(\mathbb{F}_p) \setminus \{O\}$. If $k \cdot (x_0, y_0) \neq O$, then $G := k \cdot (x_0, y_0)$ is a point of order r due to a theorem of Lagrange. However, in order to recover a false input, we propose to compute $r \cdot G$. If $r \cdot G \neq O$, an error message is output.

The computation of G is dominated by drawing a square root in $\mathbb{F}_p$. The bit-complexity of this procedure is $O(\log^4 p)$ ([Coh95]). In addition, the verification of the order of G is of bit-complexity $O(\log^4 p)$, too. Thus the whole computation of G and verifying its order is of bit-complexity $O(\log^4 p)$.

3.2.4 Comparison of Both Generation Methods

In this section we compare the random approach to the CM-method to find a cryptographically strong elliptic curve group over $\mathbb{F}_p$. We compare the security and performance implications of both generation methods.

Let us first turn to security. The main advantage of the random approach is that *every* cryptographically strong elliptic curve group over $\mathbb{F}_p$ is computed with approximately the same probability. Thus the generated curves are not special in any sense. Contrary the CM-method is only applicable if discriminants of reasonable small class numbers are in use, say discriminants of class number at most 1000 (see [Bai02a]). Then the generated curves are special in the sense that their endomorphism ring has a class number at most 1000. Thus not every cryptographically strong elliptic curve group may be output by the CM-method. However, as no attack makes use of this property, we do not consider a small class number to imply cryptographic weakness.

Next we discuss the practical performance. We present practical data for the case $k_0 = 1$, that is we search for an elliptic curve group of prime order. We write b for the bitlength of r_0. We stated in Section 3.2.1 that the bit-complexity of the random approach only depends on b. However, as explained in Section 3.2.2 the bit-complexity of the CM-method depends on the class number of the imaginary quadratic discriminant in use, too. In [Bai02b] we investigate in detail for which class number both approaches have the same run time in practice for some given, fixed b. We call this class number the *crossover class number* and denote it by $h_c(b)$.

In order to determine $h_c(b)$ we first measured the run time of `randomApproachP`(r_0, k_0). We point out that we implemented the early-abort-strategy and the use of a twisted curve as explained in Section 3.2.1. We then invoked `cryptoCurve`(r_0, k_0, h_0) for various class number bounds h_0 to get the crossover class number $h_c(b)$ (for details we refer to [Bai02b]). All tests are performed on an ordinary PC (Athlon XP1600+ running Linux 2.4.10 at 1.4 GHz and having 1GByte main memory) using freely available software. The result is given in Table 3.1.

b	160	170	180	190	200	210
Run time in minutes	3.63	4.87	7.97	10.3	13.1	16.7
$h_c(b)$	750	820	960	1040	1090	1200

Table 3.1: Average run time of the random approach to find a cryptographically strong elliptic curve group $E(\mathbb{F}_p)$ of prime order. b denotes the bitlength of p. For each b we performed 100 tests. In addition, crossover class numbers $h_c(b)$ are given.

The CM-method is supposed to be faster in practice if $h(\Delta) < h_c(b)$ for the discriminant Δ used in the CM-method. We conclude from Table 3.1 that the crossover class number is rather large. Thus even if one respects the additional requirement of the GISA that the class number of the fundamental discriminant corresponding to Δ is at least 200, the CM-method is superior to the random approach for bitlengths of cryptographic interest. Finally, we expect the crossover class number to increase if polynomials proposed by Enge and Morain ([EM02]) are used in the CM-method.

3.3 Finding Suitable Elliptic Curve Groups over $\mathbb{F}_{2^n}$

In this section we present three methods to find a cryptographically strong elliptic curve group over a finite field of characteristic 2. First, in Section 3.3.1 we describe the random approach. Then in Section 3.3.2 we present the CM-method. Finally, in Section 3.3.3 we turn to the Koblitz approach.

We first describe an algorithm to check the conditions (E1), (E2), and (E3) of Section 2.2. The algorithm is called `isStrong2`(r_0, k_0, n, N), which is very similar to algorithm `isStrongP`. It requires positive integers r_0 and k_0 as input with $r_0 \geq 2^{160}$ and $k_0 \leq 4$, respectively. In addition the algorithm gets a prime n and a positive integer N. As in the case of odd characteristic we substitute 2^{160} by r_0 and 4 by k_0 in (E1). `isStrong2`(r_0, k_0, n, N) returns a prime r if $N = k \cdot r$ is the order of a cryptographically strong elliptic curve group over $\mathbb{F}_{2^n}$ with $r \geq r_0$ and $k \leq k_0$. Otherwise it returns 0. We point out that $|E(\mathbb{F}_{2^n})|$ is even for a cryptographically strong elliptic curve group $E(\mathbb{F}_{2^n})$.

Algorithm 3.3: `isStrong2`(r_0, k_0, n, N)

Input: Positive integers r_0 and k_0 with $r_0 \geq 2^{160}$ and $2 \leq k_0 \leq 4$.
A prime n and the order N of a group of rational points of an elliptic curve over $\mathbb{F}_{2^n}$.

Output: A prime r if $N = k \cdot r$ is the order of a cryptographically strong elliptic curve group over $\mathbb{F}_{2^n}$ with $r \geq r_0$ and $k \leq k_0$, and 0 otherwise.

1: //check if N is in the Hasse interval
2: **if** $|N - (2^n + 1)| > 2\sqrt{2^n}$ **then**
3: return (0);
4: $r \leftarrow 0$; $k \leftarrow 0$; //initialize both r and k with 0
5: //check by trialdivision if cofactor is at most k_0; if not, return 0
6: **for** $i \leftarrow 2; i \leq k_0; i \leftarrow i + 1$ **do**
7: **if** $i \mid N$ AND `isPrime`$(N/i, 50) =$ `true` AND $N/i \geq r_0$ **then**
8: $r \leftarrow N/i$; $k \leftarrow i$; break;
9: **if** $r = 0$ **then**
10: return (0);
11: //check condition (E2)
12: $q_r \leftarrow 1 \bmod r$;
13: **for** $i \leftarrow 1; i \leq 19; i \leftarrow i + 1$ **do**
14: $q_r \leftarrow 2^n \cdot q_r \bmod r$;
15: **if** $q_r = 1$ **then**
16: return (0);
17: return (r);

Once we know that the bit-complexity of $\texttt{isStrongP}(r_0, k_0, p, N)$ is $O(\log^3 p)$, it is obvious that the bit-complexity of $\texttt{isStrong2}(r_0, k_0, n, N)$ is $O(n^3)$.

3.3.1 Random Approach

In this section we describe an algorithm to randomly generate a cryptographically strong elliptic curve group $E(\mathbb{F}_{2^n})$. We denote this algorithm by $\texttt{randomApproach2}(r_0, k_0, n)$. Its input are positive integers r_0 and k_0 with $r_0 \geq 2^{160}$ and $2 \leq k_0 \leq 4$, and a prime n, $n = \lceil \log_2 k_0 \cdot r_0 \rceil$. The algorithm outputs positive integers r and k, and an elliptic curve E over $\mathbb{F}_{2^n}$ such that $|E(\mathbb{F}_{2^n})| = k \cdot r$ and such that $\texttt{isStrong2}(r_0, k_0, n, k \cdot r)$ returns r.

Similar as in the case $q = p$ the proceeding is as follows: Choose parameters a and b in $\mathbb{F}_{2^n}$ with $b \neq 0$, determine the order of the group of rational points of the curve (a, b) over $\mathbb{F}_{2^n}$, and finally check if this group is cryptographically strong.

Again we recommend to choose a and b verifiably at random. The algorithm is very similar to the case $q = p$. A detailed explanation may be found for example in [X9.62], Annex A.3.3.1. We write $\texttt{getParameters2}(n, \text{SEED})$ for any algorithm which returns an elliptic curve E defined over $\mathbb{F}_{2^n}$ verifiably at random.

If the curve $E = (a, b)$ is chosen we have to determine the group order of $E(\mathbb{F}_{2^n})$. Currently, the best known algorithm for this task is a variant of an algorithm due to Satoh ([Sat99]). This variant is proposed by Fouquet, Gaudry, and Harley ([FGH00], [FGH01]). It uses a combination of the early-abort-strategy in the SEA-algorithm and the Satoh-algorithm. We denote their method by $\texttt{SFGH}(n, E)$. It requires a prime n and an elliptic curve E as input. It returns $|E(\mathbb{F}_{2^n})|$, if the early-abort-strategy does not show cryptographic weakness, and 0 otherwise. We denote the result of $\texttt{SFGH}(n, E)$ by N. If $N \neq 0$ and $\texttt{isStrong2}(r_0, k_0, n, N) \neq 0$ we are done. Otherwise we have to invoke $\texttt{getParameters2}(n, \text{SEED})$, $\texttt{SFGH}(n, E)$, and $\texttt{isStrong2}(r_0, k_0, n, N)$ until we succeed.

Algorithm 3.4: $\texttt{randomApproach2}(r_0, k_0, n)$

Input: Positive integers r_0 and k_0 with $r_0 \geq 2^{160}$ and $2 \leq k_0 \leq 4$.
A prime n with $n = \lceil \log_2 k_0 \cdot r_0 \rceil$.

Output: A prime r and a positive integer k.
An elliptic curve E over $\mathbb{F}_{2^n}$ with $|E(\mathbb{F}_{2^n})| = k \cdot r$ and such that $\texttt{isStrong2}(r_0, k_0, n, k \cdot r) = r$.

1: **while** `true` **do**
2: $E \leftarrow \texttt{getParameters2}(n, \text{SEED})$;
3: $N \leftarrow \texttt{SFGH}(n, E)$;
4: **if** $N \neq 0$ **then**
5: $r \leftarrow \texttt{isStrong2}(r_0, k_0, n, N)$;
6: **if** $r \neq 0$ **then**
7: return $(E, r, N/r)$;

We remark that one may again use a twisted elliptic curve of E over $\mathbb{F}_{2^n}$ to speed up `randomApproach2`. The bit-complexity of Satoh's algorithm may be shown to be $O(n^{3+\epsilon})$ for some $\epsilon > 0$ ([Sat99]). Although `SFGH` uses a mixed strategy, we assume that `SFGH` has

the same bit-complexity. Thus the bit-complexity of algorithm `randomApproach2`(r_0, k_0, n) is $O(n^{4+\epsilon})$.

3.3.2 Complex Multiplication Approach

In this section we discuss the CM-method to find an elliptic curve group over $\mathbb{F}_{2^n}$. In order to implement this approach efficiently, we assume that a database of discriminants of class numbers $m \cdot n$ is to our disposal, where $m \in \mathbb{N}$. For instance, such a database was computed in the framework of [Bai02a].

Let Δ be an imaginary quadratic discriminant of class number $h(\Delta)$, $n \mid h(\Delta)$. We first have to investigate the following two norm equations:

$$t'^2 - \Delta y'^2 = 8\,, \tag{3.5}$$
$$t^2 - \Delta y^2 = 2^{n+2}\,. \tag{3.6}$$

In order to find an elliptic curve having the desired properties, we have to ensure that Equation (3.5) has no integer solution (t', y'), while Equation (3.6) has a solution $(t, y) \in \mathbb{Z}^2$. If this is true, using complex multiplication, elliptic curves $E_{1,2^n}$ and $E_{2,2^n}$ over $\mathbb{F}_{2^n}$ with endomorphism ring $\mathcal{O}_\Delta$ and

$$|E_{1,2^n}(\mathbb{F}_{2^n})| = 2^n + 1 - t, \quad |E_{2,2^n}(\mathbb{F}_{2^n})| = 2^n + 1 + t \tag{3.7}$$

can be constructed as explained below (see [LZ94], [X9.62]). We set $N_1 = |E_{1,2^n}(\mathbb{F}_{2^n})|$ and $N_2 = |E_{2,2^n}(\mathbb{F}_{2^n})|$ in what follows.

As in Section 3.2.2 let $H \in \mathbb{Z}[X]$ denote the minimal polynomial of $j(\frac{\Delta+\sqrt{\Delta}}{2})$. Modulo 2 the polynomial H splits into pairwise different polynomials of degree n, all of which are irreducible in $\mathbb{F}_2[X]$. Let j_{2^n} be a zero of H in $\mathbb{F}_{2^n}$. We have $j_{2^n} \neq 0$. If $N_1 \equiv 0 \bmod 4$ we set

$$E_{1,2^n} = (0, j_{2^n}^{-1}), \quad E_{2,2^n} = (1, j_{2^n}^{-1})\,. \tag{3.8}$$

If $N_1 \equiv 2 \bmod 4$ we define

$$E_{1,2^n} = (1, j_{2^n}^{-1}), \quad E_{2,2^n} = (0, j_{2^n}^{-1})\,. \tag{3.9}$$

As n is odd, it is well known that this definition is correct (e.g. [LZ94]). The elliptic curves $E_{1,2^n}$ and $E_{2,2^n}$ in Equations (3.8) and (3.9) are called twisted elliptic curves over $\mathbb{F}_{2^n}$.

As in the case $q = p$ we can check if one of the numbers N_1 or N_2 is the order of a cryptographically strong elliptic curve group before we actually construct the corresponding curves. Furthermore, again we can use alternative polynomials whose coefficients are very small compared to H. The bit-complexity of the CM-method for fields of even characteristic is the same as in the case of a finite prime field.

3.3.3 Koblitz Approach

In this section we explain the Koblitz approach. The Koblitz approach bases on a theorem of Weil. In the scope of this report, it works as follows.

In all, there are two elliptic curves defined over $\mathbb{F}_2$ as introduced in Section 2.2. By K_1 we denote the elliptic curve $(0,1)$, by K_2 the curve $(1,1)$. It is easy to see that $K_1(\mathbb{F}_2) = \{(0,1),(1,0),(1,1),O\}$ and $K_2(\mathbb{F}_2) = \{(0,1),O\}$. Thus we have $|K_1(\mathbb{F}_2)| = 4$ and $|K_2(\mathbb{F}_2)| = 2$. For both groups we write the group order as $2+1-t_i$, where $i \in \{1,2\}$. We then associate to each curve a polynomial $F_i := 2X^2 - t_iX + 1$. Let $\frac{1}{\alpha_i}$ be a (complex) root of F_i, and let m be a positive integer. Then a theorem of Weil states that $|K_i(\mathbb{F}_{2^m})| = 2^m + 1 - (\alpha_i^m + \overline{\alpha_i}^m)$, where $\overline{\alpha_i}$ is the complex conjugate of α_i.

It is easy to see that $\alpha_1 = \frac{-1+\sqrt{-7}}{2}$ and $\alpha_2 = \frac{1+\sqrt{-7}}{2}$. For instance, we have

$$\begin{aligned} |K_1(\mathbb{F}_{2^{163}})| &= 2^2 \cdot 653 \cdot 6521 \cdot 34101072914026637 \cdot 201295412327271978497234 33 \,, \\ |K_2(\mathbb{F}_{2^{163}})| &= 2 \cdot 5846006549323611672814741753598448348329118574063 \,, \end{aligned}$$

where we write both integers with respect to their prime factorization. The group $K_2(\mathbb{F}_{2^{163}})$ obviously respects the requirements (E1) and (E3). In addition, condition (E2) is satisfied, too. Hence according to our definition $K_2(\mathbb{F}_{2^{163}})$ is a cryptographically strong elliptic curve group. In general, once we know n we can easily determine whether one of the groups $K_i(\mathbb{F}_{2^n})$ is cryptographically strong. The elliptic curves K_i are called *Koblitz curves*.

For all primes n, $163 \le n \le 500$, we determined if the groups $K_1(\mathbb{F}_{2^n})$ and $K_2(\mathbb{F}_{2^n})$ are cryptographically strong, respectively. We implemented a C++ program and used the computer algebra system LiDIA ([LiDIA]). The result is given in Table 3.2. It is in conformance with the results of Solinas ([Sol97]). We see that there are only 6 exponents n for which the group $K_1(\mathbb{F}_{2^n})$ is secure. The same is true for $K_2(\mathbb{F}_{2^n})$.

$K_1(\mathbb{F}_{2^n})$: n	233	239	277	283	349	409
$K_2(\mathbb{F}_{2^n})$: n	163	283	311	331	347	359

Table 3.2: Exponents n, $163 \le n \le 500$, yielding a cryptographically strong elliptic curve group $K_i(\mathbb{F}_{2^n})$.

We remark that the endomorphism ring of both K_1 and K_2 is $\mathcal{O}_{-7}$. Thus its class number is equal to 1. Finally, we point to a security issue of elliptic curve groups generated by the Koblitz approach. Let $K(\mathbb{F}_{2^n})$ be the group in use. Then a method due to Gallant, Lambert, and Vanstone ([GLV00]) is faster by a factor $\sqrt{2n}$ than the standard square root attacks.

3.3.4 Finding a Point of Large Prime Order

We explain how to find a base point G for elliptic curve groups $E(\mathbb{F}_{2^n})$. The proceeding is similar to the case $q = p$ as described in Section 3.2.3. The following algorithm may be found in [X9.62].

Let a prime n and an elliptic curve $E = (a,b)$ over $\mathbb{F}_{2^n}$ be given. As usual we write $|E(\mathbb{F}_{2^n})| = k \cdot r$ with a prime r. We then choose a random element $x_0 \in \mathbb{F}_{2^n}$. If $x_0 = 0$, the corresponding point $(0, b^{2^{n-1}})$ is of order 2 in $E(\mathbb{F}_{2^n})$. Hence this point is not a suitable choice. We therefore assume $x_0 \neq 0$. We set $\alpha = x_0^3 + ax_0^2 + b$. If $\alpha = 0$, we set $P = (x_0, 0)$. For fields of characteristic 2 it is common to solve quadratic equations of the form $z^2 + z = \beta$ (an algorithm is given in Annex D.1.6 of [X9.62]). Thus we set $\beta = \alpha \cdot x_0^{-2}$, $z = y \cdot x_0^{-1}$, and test if the equation

$z^2 + z = \beta$ has a solution $z_0 \in \mathbb{F}_{2^n}$. If not, we choose a new random value x_0. Otherwise we set $P = (x_0, z_0 \cdot x_0)$. Then P is a point in $E(\mathbb{F}_{2^n}) \setminus \{O\}$. If $k \cdot P \neq O$, then $G := k \cdot (x_0, y_0)$ is a point of order r due to a theorem of Lagrange. As in the case $q = p$ we propose to compute $r \cdot G$. If $r \cdot G \neq O$, an error message is output. If $k \cdot P = O$ a new element x_0 is chosen.

The computation of G involves solving a quadratic equation in $\mathbb{F}_{2^n}$ and computing scalar multiplies of a point. The bit-complexity depends on the representation of the finite field $\mathbb{F}_{2^n}$. For instance, if we choose a normal basis, solving a quadratic equation is for free (only XOR and squaring).

3.3.5 Comparison of the Three Generation Methods

We compare the three generation methods described in the previous sections.

First, we discuss the random approach. We cite timings from [FGH01] in Table 3.3. The authors of [FGH01] do not clearly state what they mean by a secure elliptic curve. However, they use the term almost prime in the context of the corresponding group order. We therefore assume that the timings in Table 3.3 give a reasonable estimation for run times to find a cryptographically strong elliptic curve group in the sense of this report.

n	163	193	197	233	239
Run time in seconds	5	10	10	21	22

Table 3.3: Average run time of the random approach to find an elliptic curve group $E(\mathbb{F}_{2^n})$ of almost prime order. The timings come from [FGH01] and are measured on an Alpha EV6 running at 750 MHz.

We point out that the timings are measured on a quite different platform compared to the PC used in Section 3.2.4. Nevertheless the run times give evidence that the SFGH with early-abort-strategy is very fast in practice. Again we point to the advantage of the random approach that every cryptographically strong elliptic curve group is chosen with approximately the same probability. Thus again the selected groups are not special in any sense.

Second, we do not have current run times of the CM-method. However, we do not expect the CM-method to be significantly faster than the random approach as we have to choose the class number of the discriminant to be a multiple the field degree n. As $n \geq 160$ the class number is at least 160, too. We therefore recommend not to use the CM-method in the case $q = 2^n$.

Third, we remark that the current record of counting the number of rational points of a general group $E(\mathbb{F}_{2^n})$ is hold by a group of INRIA ([Har02]). They succeeded to determine the group order of a randomly chosen curve for $n = 32003$. The run time on the DEC Alpha EV6 was about 27 hours. This shows that the random approach is very fast in case of fields of characteristic 2, even for large fields.

Finally, we turn to the Koblitz approach. As stated in Section 3.3.3, if we assume $160 \leq n \leq 500$ there are only 12 elliptic curve groups suitable for use in cryptography. This is a very restricted choice. In addition, as mentioned at the end of Section 3.3.3, computing discrete logarithms in such groups is faster by a factor $\sqrt{2n}$ than the standard general discrete logarithm algorithms. Nevertheless all Koblitz groups respect the requirements (E1), (E2),

and (E3). They may thus be regarded as cryptographically secure from a current point of view.

Chapter 4

Implementation Issues

The most important operation in the framework of elliptic curve cryptography is the *scalar multiplication*. The scalar multiplication is the problem of computing the point $s \cdot P$, if an integer s and a rational point $P \in E(\mathbb{F}_q)$ are given for some elliptic curve E defined over $\mathbb{F}_q$.

Before actually discussing algorithms to perform a scalar multiplication, Section 4.1 deals with the problem of how to implement the arithmetic in a finite field. First, in Section 4.1.1 we describe methods to efficiently implement the arithmetic in $\mathbb{F}_p$. In addition, we present a class of primes p for this purpose. The problem of how to efficiently implement arithmetic in finite fields of characteristic 2 is addressed in Section 4.1.2.

There are a lot of publications dealing with proposals of speeding up the scalar multiplication on a certain class of elliptic curves. It is out of the scope of the evaluation report at hand to discuss all investigations in detail. Instead we give an overview of the currently methods in use. A survey of fast algorithms for implementing the scalar multiplication in a general group is given in [Gor98]. We discuss in detail methods for a scalar multiplication in Section 4.2.

Then Section 4.3 discusses the problem if a special choice of curve parameters can significantly decrease the number of bits to represent elliptic curve points. We close this chapter with a discussion of special parameters for smart card implementations in Section 4.4.

4.1 Arithmetic in Finite Fields

In this section we review how to efficiently implement the arithmetic in a finite field $\mathbb{F}_q$. First, in Section 4.1.1 we show how to efficiently implement the arithmetic in $\mathbb{F}_p$. In addition, we present primes of a special form as proposed in [NIST]. Next, in Section 4.1.2 we discuss the two common ways to represent a finite field of characteristic 2.

4.1.1 Arithmetic in Finite Prime Fields

In this section we show how to speed up the multiplication operation in a finite prime field. As usual we denote by p the cardinality of the prime field. We remark that the generation process of the elliptic curve may be performed more efficiently, too, when using the methods of this section.

In Section 4.2.4 we will see that the multiplication of elements in $\mathbb{F}_p$ is by far the most important field operation to implement arithmetic in elliptic curve groups. As usual we assume that elements in $\mathbb{F}_p$ are represented as non-negative integers less than p. If we have to compute a product of elements e_1 and e_2 in $\mathbb{F}_p$, a naive approach would be to first compute the integer $e_1 \cdot e_2$ in $\mathbb{Z}$. Thus in this intermediate state we get an integer of order of magnitude p^2. The result of the multiplication is the unique integer in $[0, \ldots, p-1]$ which is congruent $e_1 \cdot e_2$ modulo p. Thus we would have to perform a reduction operation modulo p.

However, there are representations of the field elements of $\mathbb{F}_p$ to speed up this elementary method. The most famous and in general most efficient one is due to Montgomery ([Mon85]). We refer to his paper or to [BSS99] for details.

In addition, we mention primes of a special form as proposed in [NIST]. The fundamental idea is to use prime numbers p of a special form, that is their binary expansion is very sparse. The primes are called *generalized Mersenne numbers*. The main speed up is due to the fact that the reduction of $e_1 \cdot e_2$ modulo p may be performed by means of integers of order of magnitude less than p. Furthermore, if we choose the non-vanishing 2-powers in the binary expansion of p with care, the representation of integers may be adapted to the hardware in use.

For instance, assume that the processor uses words of 64 bits. The prime $p = 2^{192} - 2^{64} - 1$ is very attractive for this platform, as we now see. The product $e_1 \cdot e_2$ in $\mathbb{Z}$ may be written in the form

$$A_5 \cdot 2^{320} + A_4 \cdot 2^{256} + A_3 \cdot 2^{192} + A_2 \cdot 2^{128} + A_1 \cdot 2^{64} + A_0 \,, \tag{4.1}$$

where each A_i is of bitlength 64 and hence fits in a word.

Furthermore, the sparse binary expansion of p shows $2^{192} \equiv 2^{64} + 1 \bmod p$. This obviously yields

$$e_1 \cdot e_2 \equiv (A_5 + A_4 + A_2) \cdot 2^{128} + (A_5 + A_4 + A_3 + A_1) \cdot 2^{64} + A_5 + A_3 + A_0 \,. \tag{4.2}$$

It is possible that $(A_5 + A_4 + A_2) \cdot 2^{128}$ has to be reduced modulo p. However, the computation in Equation (4.2) mostly consists of additions of integers which fit in word.

More generalized Mersenne numbers for fields of different bitlengths may be found in [NIST]. In addition, a further discussion of moduli of a special form may be found in [MOV97].

4.1.2 Arithmetic in Finite Fields of Characteristic 2

In this section we shortly review the common representations of fields with 2^n elements. First, we discuss the representation of $\mathbb{F}_{2^n}$ with respect to a polynomial basis. Then we turn to the notion of a normal basis. We remark that the implementation of the efficient point counting algorithm described in Section 3.3 uses a polynomial basis to represent the field $\mathbb{F}_{2^n}$. However, as efficient algorithms to change the basis are known (see for instance Appendix D.2.3 of [X9.62]), this constitutes no restriction for use of these parameters with respect to a normal basis.

Polynomial Basis Representation

Let f be an irreducible polynomial with coefficients in $\mathbb{F}_2$ of degree n. It is well known that $\mathbb{F}_{2^n} = \mathbb{F}_2[X]/(f)$, where we write (f) for the principal ideal in the ring $\mathbb{F}_2[X]$ generated by

the polynomial f. Such a representation of $\mathbb{F}_{2^n}$ is called a *polynomial representation*. The elements $1, X, \ldots, X^{n-1}$ form a basis, which we call a polynomial basis. A polynomial basis is mostly used to implement the arithmetic in $\mathbb{F}_{2^n}$ in software.

In order to decrease the computational complexity, f should be as sparse as possible. It is easy to see that at least 3 coefficients of f are non-zero. If f actually is of the form $X^n + X^\kappa + 1$, $1 \leq \kappa \leq n-1$, f is called a *trinomial*. The polynomial basis with respect to f is said to be a trinomial polynomial basis, which commonly is abbreviated by TPB. If a TPB exists, the smallest possible value κ should be used for interoperability reasons, as proposed in [X9.62] or [P1363]. A table of fields $\mathbb{F}_{2^n}$, $160 \leq n \leq 2000$, for which a TPB exists may be found in Annex C.2 of [X9.62].

However, if a TPB does not exist for the field $\mathbb{F}_{2^n}$, a polynomial f of the form $X^n + X^{\kappa_3} + X^{\kappa_2} + X^{\kappa_1} + 1$, $1 \leq \kappa_1 < \kappa_2 < \kappa_3 \leq n-1$, may be chosen (it is obvious that a polynomial $X^n + X^{\kappa_2} + X^{\kappa_1} + 1$, $1 \leq \kappa_1 < \kappa_2 \leq n-1$ is not irreducible in $\mathbb{F}_2[X]$). The polynomial f is called a *pentanomial* in this case, and the corresponding basis is said to be a pentanomial polynomial basis, which commonly is abbreviated by PPB. If $n \geq 4$, the existence of a PPB is known. Again, for interoperability reasons, a PPB should be used where κ_1 is as small as possible, κ_2 is as small as possible for this particular κ_1, and finally κ_3 is as small as possible for these particular chosen κ_1 and κ_2. A table of fields $\mathbb{F}_{2^n}$, $160 \leq n \leq 2000$, for which a PPB, but no TPB exists, may be found in Annex C.3 of [X9.62].

Normal Basis Representation

In this section we discuss normal basis representations. A *normal basis* is a basis of the form $\alpha, \alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^{n-1}}$, where $\alpha \in \mathbb{F}_{2^n}$. A normal basis is attractive for implementing arithmetic in $\mathbb{F}_{2^n}$ in hardware, as squaring an element in $\mathbb{F}_{2^n}$ is simply a cyclic shift. However, multiplying elements of $\mathbb{F}_{2^n}$ with respect to a normal basis is in general a non-trivial and cumbersome task. It is therefore common to use a *Gaussian normal basis*, abbreviated by GNB. It is well known that if $8 \nmid n$, a GNB exists. However, as our requirement (E3) of Section 2.2 assumes n to be prime, this is no restriction for cryptographic purposes.

The complexity of arithmetic with respect to a GNB is measured in terms of the type of the GNB in use. The type is a positive integer, and as in [X9.62] we denote the type by T. Roughly speaking, the smaller the type T is, the more efficiently the arithmetic in $\mathbb{F}_{2^n}$ may be implemented. A necessary condition for a positive integer T' to be the type of a GNB is that $T'n + 1$ is prime. Thus in this report a GNB of type 1 is not possible.

Recommended Fields of Characteristic 2

In this section we list finite fields of characteristic 2, which we propose for use in cryptography. The results may be found in Table 4.1. The table bases on Annex C of [X9.62]. In addition, for each chosen n we show if a TPB exists or not. In Table 4.1 we plot a star in the column of n if a TPB for the field $\mathbb{F}_{2^n}$ exists. Otherwise, we set a star in the corresponding row of PPB. Finally, we give the type T of the GNB. We remark that in Table C-1.a of [X9.62], no value T is given for $n = 179$, although a GNB of type 2 exists for this field. We therefore checked the relevant data of Annex C.1 in [X9.62] using the computer algebra system LiDIA ([LiDIA]). Besides $n = 179$ we got the same results as presented in [X9.62]. Furthermore, our table is in conformance with the data of Annex A.8 in [P1363].

n	163	167	173	179	181	191	193	197	199	211	223	227	229	233
TPB		*				*	*		*		*			*
PPB	*		*	*	*			*		*		*	*	
T	4	14	2	2	6	2	4	18	4	10	12	24	12	2
n	239	241	251	257	263	269	271	277	281	283	293	307	311	313
TPB	*	*		*	*		*		*					*
PPB			*			*		*		*	*	*	*	
T	2	6	2	6	6	8	6	4	2	6	2	4	6	6
n	317	331	337	347	349	353	359	367	373	379	383	389	397	401
TPB			*			*	*	*			*			*
PPB	*	*		*	*				*	*		*	*	
T	26	6	10	6	10	14	2	6	4	12	12	24	6	8
n	409	419	421	431	433	439	443	449	457	461	463	467	479	487
TPB	*			*	*	*		*	*		*		*	*
PPB		*	*				*			*		*		
T	4	2	10	2	4	10	2	8	30	6	12	6	8	4
n	491	499												
TPB														
PPB	*	*												
T	2	4												

Table 4.1: Recommended finite fields $\mathbb{F}_{2^n}$, $160 \leq n \leq 500$, for use in elliptic curve cryptography. The star in the corresponding row of TPB and PPB indicates, which polynomial representation should be used. In addition, we list the type T of the GNB to choose.

4.2 Scalar Multiplication

In this section we review methods for efficiently performing a scalar multiplication. An important issue in this context is the addition of two rational points. We address this subject in Section 4.2.1 and Section 4.2.2 for finite prime fields and finite fields of characteristic 2, respectively.

We then present in Section 4.2.3 a method, which is only applicable for Koblitz curves. Finally, in Section 4.2.4 we turn to general methods for efficiently performing the scalar multiplication.

4.2.1 Point Addition for Elliptic Curves over $\mathbb{F}_p$

We describe a method for efficiently adding two rational points in a group $E(\mathbb{F}_p)$. We follow the discussion in [P1363] and [BSS99]. We remark that this method is independent of the prime field $\mathbb{F}_p$.

As of today it is common to implement arithmetic in a group $E(\mathbb{F}_p)$ with respect to *weighted projective coordinates* as proposed in [CC87] or [CMO98]. The curve equation is then of the form $Y^2 = X^3 + aXZ^4 + bZ^6$, and if (X, Y, Z) is a point on the curve, its affine coordinates are $(X/Z^2, Y/Z^3)$, provided that $Z \neq 0$. A point with $Z = 0$ corresponds to the point at infinity O.

The main advantage when using weighted projective coordinates is that we do not have to do a field inversion in $\mathbb{F}_p$ when adding points. However, the number of multiplications in $\mathbb{F}_p$

increases. Often the use of projective coordinates is then superior to an implementation with respect to affine coordinates.

More precisely, let $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ be points in $E(\mathbb{F}_p)$. We skip the trivial cases $P_1 = O$, $P_2 = O$, and $P_1 = \pm P_2$. Then according to [P1363] and [BSS99] a general addition of P_1 and P_2 requires 16 multiplications in $\mathbb{F}_p$. If in addition one point is given in affine coordinates, that is $Z_1 = 1$ or $Z_2 = 1$ the number of multiplications decreases to 11. This case is denoted by mixed coordinates. If we add two points represented in affine coordinates we have to do 1 inversion and 3 multiplications in $\mathbb{F}_p$. Thus in the general case the use of weighted coordinates is superior to an implementation with respect to affine coordinates, if an inversion costs more than 13 multiplications. If one of the Z-coordinates is equal to 1, this number decreases to 8.

If we double the point (X_1, Y_1, Z_1), the number of multiplications is in general 10. If we have $a = -3$, this number decreases to 8. The affine doubling requires 1 inversion and 4 multiplications in $\mathbb{F}_p$. Thus weighted projective coordinates are faster in practice, if an inversion in $\mathbb{F}_p$ is slower than 6 or 4 multiplications, respectively. We remark that choosing elliptic curves verifiably at random with $a = -3$ is proposed as an extension to the general algorithm in [P1363] (see Annex A.12.4). Furthermore, the implementation of the CM-method as described in [Bai02a] yields elliptic curves with $a = -3$, too.

Table 4.2 summarizes the above discussion.

Operation	Affine Coord.	Mixed Coord.	Weighted Proj. Coord.
General Addition	$1I + 3M$	$11M$	$16M$
General Doubling	$1I + 4M$	n/a	$10M$
Doubling ($a = -3$)	$1I + 4M$	n/a	$8M$

Table 4.2: Cost of a point addition in a group $E(\mathbb{F}_p)$. A field inversion is abbreviated by I, a multiplication in $\mathbb{F}_p$ by M.

4.2.2 Point Addition for Elliptic Curves over $\mathbb{F}_{2^n}$

The method presented in this section is very similar to the ideas of the previous section. We therefore summarize the results. Again we follow the discussion in [P1363] and [BSS99].

An elliptic curve over $\mathbb{F}_{2^n}$ in weighted projective coordinates is given by the equation $Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$. Let $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ be points in $E(\mathbb{F}_{2^n})$. As above we leave out the trivial cases $P_1 = O$, $P_2 = O$, and $P_1 = \pm P_2$. Then according to [P1363] and [BSS99] a general addition of P_1 and P_2 requires 15 multiplications and 5 squarings in $\mathbb{F}_{2^n}$. As the complexity of a squaring depends on the representation of the field $\mathbb{F}_{2^n}$, it is common to enumerate it separately. If in addition one point is given in affine coordinates the number of multiplications and squarings decrease to 11 and 4, respectively.

We remark that it is attractive to use $a = 0$ from an implementation point of view of the point addition, as the number of multiplications and squarings is less than above. However, on the other hand we have $k = 4$ in this case, yielding an additional bit compared to an elliptic curve group with the same security level, but $k = 2$.

The corresponding complexity assertions for a point doubling and a comparison with an implementation with respect to affine coordinates may be seen from Table 4.3.

Operation	Affine Coord.	Mixed Coord.	Weighted Proj. Coord.
General Addition	$1I + 2M + 1S$	$11M + 4S$	$15M + 5S$
Addition ($a = 0$)	$1I + 2M + 1S$	$10M + 3S$	$14M + 4S$
Doubling	$1I + 2M + 1S$	n/a	$5M + 5S$

Table 4.3: Cost of a point addition in a group $E(\mathbb{F}_{2^n})$. A field inversion is abbreviated by I, a multiplication in $\mathbb{F}_{2^n}$ by M, and a squaring in $\mathbb{F}_{2^n}$ by S.

4.2.3 Scalar Multiplication on Koblitz Curves

In this section we describe a method for performing the scalar multiplication in the groups $K_1(\mathbb{F}_{2^n})$ and $K_2(\mathbb{F}_{2^n})$ as introduced in Section 3.3.3. The most significant improvement is due to Solinas ([Sol97], [Sol00]). Solinas proposes to represent the integer s with respect to the Frobenius map. According to his paper [Sol97] his method yields a speed up by a factor 2 compared to the previous best known algorithms for scalar multiplications on Koblitz curves. Thus Koblitz curves are very attractive if fast arithmetic is important (e.g. in smart cards).

We remark that the method of Solinas was extended to a larger class of elliptic curves by Gallant, Lambert, and Vanstone ([GLV01]). Their method even comprises elliptic curves over finite prime fields.

4.2.4 Scalar Multiplication on a General Elliptic Curve

This section deals with methods for computing a scalar multiple of an elliptic curve point. We focus our discussion with respect to the following two requisites. First, we present methods for efficiently performing the scalar multiplication if the point P is not known in advance. Nevertheless in cryptographic schemes we may assume that P is in the subgroup generated by G. Second, we turn to the case that P is a previously known, fixed point. An application of such a method is, for instance, the scalar multiplication for the cryptographic base point G. The discussion in this section is valid for both a field $\mathbb{F}_p$ and $\mathbb{F}_{2^n}$.

We first assume that P is not fixed. A fundamental algorithm to determine the point sP is to implement fast exponentiation. Its underlying idea is to write s as its binary expansion and compute sP by the double and add algorithms of the previous sections (see for instance Algorithm IV.1 in [BSS99]). However, most of the standards (e.g. [P1363], [X9.62]) propose to implement a variant which uses a signed representation of the scalar s. Often this expansion is referred to as a NAF, where NAF stands for *non-adjacent form* of the scalar s. A signed representation of s works fine, as inversion in the group $E(\mathbb{F}_q)$ is for free. We present a variant of the algorithms in [P1363] or [X9.62] as our algorithm $\texttt{NAF}(s, P)$. The algorithm requires an integer s and a point P. It returns the point sP. We make use of the fact that we know the order of P.

We turn to the second case. We assume that we have to compute a scalar multiplication for a fixed point G. Then it is often advantageous to initially do some precomputations and then

Algorithm 4.1: NAF(s, P)

Input: An integer s and a point P of order r.
Output: The point sP.

1: $s \leftarrow s \bmod r$; //$s \in \{0, \ldots, r-1\}$
2: **if** $s = 0$ OR $P = O$ **then**
3: return(O);
4: **if** $s > r/2$ **then**
5: $s \leftarrow r - s$; $P \leftarrow -P$;
6: $3s = \sum_{i=0}^{l} h_i 2^i$; $s = \sum_{i=0}^{l} s_i 2^i$; //find binary expansions of $3s$ and s with $h_l = 1$
7: $R \leftarrow P$; $i \leftarrow l - 1$; //initialize the result point and counting variable
8: **while** $i \geq 1$ **do**
9: $R \leftarrow 2R$;
10: **if** $h_i = 1$ AND $s_i = 0$ **then**
11: $R \leftarrow R + P$;
12: **if** $h_i = 0$ AND $s_i = 1$ **then**
13: $R \leftarrow R - P$;
14: $i \leftarrow i - 1$;
15: return(R);

determine the point sG. The initial step has only to be performed once. The precomputed points have to be stored.

The most common method is the sliding window method (see for instance [BSS99], Algorithm IV.4). Additionally we point to a method of Lim and Lee [LL94]. Their method may be very fast, if a lot of precomputations are performed. However, as we then have to store quite a lot of points, this may cause memory problems.

A survey of the cost of the different methods to compute a scalar multiple may be found in [BSS99], Table IV.3 .

4.2.5 Scalar Multiplication on Special Elliptic Curves

In this section we mention two representations of elliptic curves for speeding up the scalar multiplication. The first one is called the *Hesse form*, the second one is called the *Montgomery form*.

We first discuss the Hesse form and follow Smart ([Sma01]. He shows how to speed up the scalar multiplication by parallelizing computational steps if the elliptic curve is given in Hesse form. Let E denote this curve. We assume that E is given by an equation as defined in Chapter 2. In his paper, Smart shows that if $q \equiv 2 \bmod 3$ and $3 \mid |E(\mathbb{F}_q)|$, then the defining equation of E may be transformed to a representation in Hesse form. This representation allows efficient implementation of the group law in $E(\mathbb{F}_q)$. We refer to Smart's paper for details.

The requirement $3 \mid |E(\mathbb{F}_q)|$ implies in our context, that we can apply Smart's method if and only if $q = p$ (if $q = 2^n$, the group order is divisible by 6 in this case). Thus let $q = p$. We then search for an elliptic curve group $E(\mathbb{F}_p)$ of order $3r$. We remark that both generation methods of Section 3.2 seem to be appropriate in this case. As far as the CM-method is concerned this is obvious from the fact that we can verify this requirement once we know

the group orders of Equation (3.2). If the random approach is used, the early-abort-strategy shows at a very early stage of the SEA-algorithm, if the group order is divisible by 3. Thus we expect that the assertions of Section 3.2.4 are valid in this case, too.

We now discuss the Montgomery form. Initially it was proposed by Montgomery to accellerate the elliptic curve factoring method ([Mon87]). Montgomery shows how to reduce the number of multiplications in $\mathbb{F}_q$ to compute the x-coordinate of a scalar multiple of a rational point, if an elliptic curve in a special representation is used.

However, as in the case of the Hesse form, we have to impose a restriction on the group order of the elliptic curve group. If as above $q = p$ is the cardinality of the prime field, Izu shows that if $p \equiv 1 \bmod 4$ then the defining equation of E may be transformed to a Montgomery representation if and only if $4 \mid |E(\mathbb{F}_q)|$ ([Izu99]). If, however, $p \equiv 3 \bmod 4$, then only if $8 \mid |E(\mathbb{F}_q)|$ we know that a Montgomery representation of E exists. Thus we propose to use primes $p \equiv 1 \bmod 4$ if a Montgomery representation shall be used. Again as above both generation methods seem to be appropriate to find such elliptic curves.

4.3 Point Compression

In this section we address the problem of point compression. We answer the question if the generation of special curve parameters decreases significantly the number of bits to represent a point. We show that the answer actually is 'no'.

In some situations it is desirable to represent a point with as few bits as possible. Such a situation occurs if storage or bandwidth are at a premium. We refer to such a notation as *point compression*.

Most of our discussion is independent of the characteristic of the field. Thus let E be an elliptic curve defined over some finite field $\mathbb{F}_q$. We denote the bitlength of q by n. In the framework of elliptic curve cryptography a non-trivial point P is represented in (affine) coordinates by (x, y), where both x and y are elements of $\mathbb{F}_q$. Thus we need at most $2n$ bits to represent such a point. However, the coordinates of P satisfy a quadratic equation in y. Thus it is easy to see how to recover y, once x and some additional bit related to y are given. Hence the transmission of $n + 1$ bits is sufficient to uniquely identify P (the methods to compress the representation may be found for example in [P1363] or [X9.62]).

The boundary condition in our context is as follows. Let $E(\mathbb{F}_q)$ be a cryptographically strong elliptic curve group, and let P be a point of order r. Then we have

$$r \geq \frac{|E(\mathbb{F}_q)|}{4} \geq \frac{q + 1 - 2\sqrt{q}}{4} \geq \frac{2^{n-1} + 1 - 2\sqrt{2^n}}{4} > 2^{n-4} \,. \tag{4.3}$$

Thus $n - 3$ is a lower bound of the bitlength of r. We therefore need at least $n - 3$ bits to represent an elliptic curve point in our context. The saving of at most 4 bits seems to us not to justify the search for a curve with special parameters.

We finally mention a minor improvement. If $q = 2^n$ and the order of P is odd, we only need n bits to represent P (see [Ser98], [BSS99]).

4.4 Implementation on Smart Cards

In this closing section we discuss elliptic curve parameters for use in a constraint environment such as a smart card. There is no general answer to this problem. Nevertheless, we point to some characteristics of a smart card implementation.

As of today most of the smart cards for cryptographic use come with a cryptographic coprocessor. Thus we only discuss this type of smart cards. The first important point from a performance point of view is the information sharing between the main processor and the cryptographic coprocessor. In general the bandwidth is at a premium. Thus there should be as few transmissions as possible. For instance, if an elliptic curve over $\mathbb{F}_p$ should be used, the developer of such a system could skip the transmission of the elliptic curve parameter a by always setting $a = -3$. As mentioned in Section 4.2.1 this yields a performance speed up, too.

In addition, the use of NIST primes as explained in Section 4.1.1 seem to be a good choice. Again we can decrease the number of bits to be transferred, if only the non-trivial bit positions of the binary expansion of p are exchanged. Furthermore, the implementation of the arithmetic in $\mathbb{F}_p$ is very fast in this case.

Bibliography

[AKS02] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. available via WWW from http://www.cse.iitk.ac.in/primality.pdf, August 2002.

[AM93] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of Computation*, 61:29–67, 1993.

[Bai02a] H. Baier. *Efficient Algorithms for Generating Elliptic Curves over Finite Fields Suitable for Use in Cryptography.* PhD thesis, Darmstadt University of Technology, 2002.

[Bai02b] H. Baier. How to find Elliptic Curve Groups of Prime Order. Technical Report, Darmstadt University of Technology, 2002. Technical Report TI-6/02.

[BK98] R. Balasubramanian and N. Koblitz. The Improbability that an Elliptic Curve has Subexponential Discrete Log Problem under the Menezes Okamoto Vanstone Algorithm. *Journal of Cryptology*, 11:141–145, 1998.

[BSS99] I. Blake, G. Seroussi, and N. Smart. Elliptic Curves in Cryptography. Cambridge University Press, 1999.

[CC87] D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Adv. in Appl. Math.*, 7:385–434, 1987.

[CMO98] H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve Exponentiation using mixed coordinates. In *Proceedings of ASIACRYPT '98*, LNCS 1514, pages 51–65, Berlin, 1998. Springer-Verlag.

[Coh95] H. Cohen. A Course in Computational Algebraic Number Theory. Springer-Verlag, 1995.

[EM02] A. Enge and F. Morain. Comparing Invariants for Class Fields of Imaginary Quadratic Fields. In *Proceedings of ANTS-V*, LNCS 2369, pages 252–266, Berlin, 2002. Springer-Verlag.

[FGH00] M. Fouquet, P. Gaudry, and R. Harley. An extension of Satoh's algorithm and its implementation. *J. Ramanujan Math. Soc.*, 15:281–318, 2000.

[FGH01] M. Fouquet, P. Gaudry, and R. Harley. Finding Secure Curves with the Satoh-FGH Algorithm and an Early-Abort Strategy. In *Proceedings of Eurocrypt 2001*, LNCS 2045, pages 14–29, Berlin, 2001. Springer-Verlag.

[FIPS186] FIPS186 Digital Signature Standard. Federal Information Processing Standards Publication 186, 1994.

[FR94] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.

[GHS02a] S. Galbraith, F. Hess, and N.P. Smart. Extending the GHS Weil Descent Attack. In *Proceedings of Eurocrypt 2002*, LNCS 2332, pages 29–44, Berlin, 2002. Springer-Verlag.

[GHS02b] P. Gaudry, F. Hess, and N.P. Smart. Constructive and Descructive Facets of Weil Descent on Elliptic Curves. *Journal of Cryptology*, 15:19–46, 2002.

[GIS01] Geeignete Kryptoalgorithmen, In Erfüllung der Anforderungen nach §17(1) SigG vom 16. Mai 2001 in Verbindung mit §17(2) SigV vom 22. Oktober 1997, July 2001. Bundesanzeiger Nr. 158 - Seite 18 562 vom 24. August 2001.

[GLV00] R. Gallant, R. Lambert, and S. Vanstone. Improving the Parallelized Pollard Lambda Search on Anomalous Binary Curves. *Mathematics of Computation*, 69(232):1699–1705, 2000.

[GLV01] R. Gallant, R. Lambert, and S. Vanstone. Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In *Proceedings of CRYPTO 2001*, LNCS 2139, pages 190–200, Berlin, 2001. Springer-Verlag.

[Gor98] D. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, pages 129–146, 1998.

[Har02] R. Harley. Elliptic Curve Point Counting: 32003 bits. Posted at the Number Theory Web, available via WWW from http://listserv.nodak.edu/archives/nmbrthry.html, August 2002.

[Izu99] T. Izu. On the Computation of Elliptic Curve Cryptosystems. In *SCIS'99*, W4-1-1, 1999.

[Kob92] N. Koblitz. CM-Curves with Good Cryptographic Properties. In *Advances in Cryptology-CRYPTO '91*, LNCS 576, pages 279–287, 1992.

[LiDIA] LiDIA. A library for computational number theory. Darmstadt University of Technology. URL: http://www.informatik.tu-darmstadt.de/TI/LiDIA/Welcome.html.

[LL94] C. Lim and P. Lee. More Flexible Exponentiation with Precomputation. In *Advances in Cryptology - CRYPTO'94*, LNCS 839, pages 95–107, Berlin, 1994. Springer-Verlag.

[LV01] A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. *Journal of Cryptology*, 14:255–293, 2001.

[LZ94] G.-J. Lay and H.G. Zimmer. Constructing elliptic curves with given group order over large finite fields. In *Proceedings of ANTS I*, LNCS 877, pages 250–263, 1994.

[Mül95] V. Müller. *Ein Algorithmus zur Bestimmung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik größer drei*. PhD thesis, University of Saarbrücken, 1995.

[Mon85] P.L. Montgomery. Modular multiplication without trial division. *Math. of. Comp.*, 44:519–521, 1985.

[Mon87] P.L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Math. of. Comp.*, 48:243–264, 1987.

[MOV91] A. Menezes, T. Okamoto, and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 80–89, 1991.

[MOV97] A. Menezes, P.v. Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.

[NIST] NIST Recommended Elliptic Curves for Federal Government Use. National Institute of Standards and Technology, 1999.

[P1363] P1363 Standard Specifications for Public Key Cryptography. IEEE, 2000.

[SA98] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comm. Math. Univ. Sancti Pauli*, 47:81–92, 1998.

[Sat99] T. Satoh. The Canonical Lift of an Ordinary Elliptic Curve over a Finite Field and its Point Counting. http://www.rimath.saitama-u.ac.jp/lab.en/TkkzSatoh/, 1999.

[Sem98] I. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Mathematics of Computation*, 67:353–356, 1998.

[Ser98] G. Seroussi. Compact representation of elliptic curve points over $\mathbb{F}_{2^n}$. Technical Report, Hewlett Packard Laboratories Technical Report No. HPL-98-94R1, September 1998.

[Sma99] N.P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, 12/3:193–196, 1999.

[Sma01] N.P. Smart. The Hessian form of an elliptic curve. In *Proceedings of CHES 2001*, LNCS 2162, pages 118–128, Berlin, 2001. Springer-Verlag.

[Sol97] J. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In *Advances in Cryptology - CRYPTO '97*, LNCS 1294, pages 357–371, Berlin, 1997. Springer-Verlag.

[Sol00] J. Solinas. Efficient arithmetic on Koblitz curves. *Designs, Codes and Cryptography*, 19:195–249, 2000.

[SEC1] SEC1 Standards for Efficient Cryptography: Elliptic Curve Cryptography. Version 1.0, 2000.

[vOW99] P.C. van Oorschot and M.J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12/1:1–28, 1999.

[X9.62] X9.62 Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). ANSI, 1998.

[X9.63] X9.63 Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography. ANSI, 2002.

Index