Chapter 23

Algebraic Realizations of DL Systems

Gerhard Frey and Tanja Lange

Contents in Brief

We want to design a public-key cryptosystem that enables us to exchange keys, sign and authenticate documents and encrypt and decrypt (small) messages. The system should rely on simple protocols that are based on secure cryptographic primitives with a well understood mathematical background. The implementation rules should be clear and easy to understand.

By using the results obtained in the previous chapters, we hope to convince the reader that these criterions can be realized quite satisfyingly by systems based on the discrete logarithm in finite groups G of prime order ℓ.

The purpose of this chapter is to serve as a digest of the other chapters. To this aim we briefly state the main results and provide many references to the much more detailed descriptions in the book.

23.1 Candidates for secure DL systems

The protocols based on discrete logarithms are described and discussed in Chapter 1. It is obvious that the complexity of computing discrete logarithms in the chosen group is a key ingredient for the security of the system. For actual use in practice, the DLP is used as a *cryptographic primitive*

in protocols. In this section we shall introduce examples for groups that are expected to be good candidates for this.

23.1.1 Groups with numeration and the DLP

For the convenience of the reader we shall repeat the essential notions from Chapter 1. Let $(G, \oplus)$ be a group of order ℓ, where ℓ is a prime number. Let $P, Q \in G$ be two elements. The discrete logarithm in G of Q with respect to P is a number $n = \log_P(Q)$ with

$$Q = [n]P = \underbrace{P \oplus P \oplus \cdots \oplus P}_{n \text{ times}}.$$

The number n is determined modulo ℓ. To compute n (for randomly given (P, Q)) is the *discrete logarithm problem* in G (the DLP in G).

To enable this computation for a computer, we have to assume that G is given in a very concrete way. So, we shall assume that the elements of G are given as bit-strings of length $O\big(\lg(\ell)\big)$. The assumption used here is that G is a group with *numeration*. For the exact definition of this notion we refer to [FRLA 2003].

Furthermore, for estimating the hardness of the discrete logarithm problem, the instantiation of G is very important. The fact that the representation plays a key role for the complexity of the computations of discrete logarithms is demonstrated by two examples.

Up to isomorphisms there is only one group with ℓ elements.

As a first representation, we choose $(G, \oplus) = (\mathbb{Z}/\ell\mathbb{Z}, +)$ with natural numeration

$$\begin{aligned} f : \mathbb{Z}/\ell\mathbb{Z} &\to \{1, \ldots, \ell\} \\ m + \ell\mathbb{Z} &\mapsto r_m \text{ such that } r_m \equiv m \pmod{\ell}. \end{aligned}$$

The discrete logarithm of $m_1 + \ell\mathbb{Z}$ with respect to $m_2 + \ell\mathbb{Z}$ is computed in $O(\lg \ell)$ bit operations (cf. Chapter 10).

In Example 1.13 we choose another representation for such a group. We find G embedded in the multiplicative group of the finite field $\mathbb{F}_p$ as the group of roots of unity of order ℓ, where p is a prime such that ℓ divides $p - 1$. In this case the complexity of the DLP is subexponential in p (cf. Chapter 20).

We take this opportunity to recall our measure for the complexity of algorithms introduced in Chapter 1 and used many times.

Let N be a natural number. Define

$$L_N(\alpha, c) := \exp\big((c + o(1))(\ln N)^{\alpha}(\ln \ln N)^{1-\alpha}\big)$$

with $0 \leqslant \alpha \leqslant 1$ and $c > 0$. $L_N(\alpha, c)$ interpolates between polynomial complexity for $\alpha = 0$ and exponential complexity for $\alpha = 1$. For $\alpha < 1$ the complexity is said to be *subexponential*.

So the second numeration makes G to a group in which the DLP is much harder than in the first example but not optimal, i.e., not exponential in ℓ. More details on computing discrete logarithms using the index calculus method can be found in Chapter 20. To make related cryptosystems secure one has to take p rather large (the bit-size of p should be at least 1024).

23.1.2 Ideal class groups and divisor class groups

All DL-based cryptosystems applied today use as groups the ideal classes of convenient (commutative) rings $\mathcal{O}$ with unit element and without zero divisors. Using a class group requires that in the scalar multiplication some notion of reduction of ideals is available and that the groups are chosen in a manner to guarantee that ideal classes can be (almost) uniquely represented by reduced ideals.

23.1.2.a Mathematical background

Let $\mathcal{O}$ be a commutative ring with unit element and without zero divisors. The quotient field K of $\mathcal{O}$ consists of all fractions f/g with $f, g \in \mathcal{O}$ with addition and multiplication defined by rules one is used to follow in $\mathbb{Q}$.

A fractional ideal is a set $A \subseteq K$ such that there exists an element $f \in \mathcal{O}$, so that fA is an ideal in $\mathcal{O}$. The multiplication of fractional ideals is defined and associative. A first criterion to choose $\mathcal{O}$ is that the fractional ideals form a group called the *ideal group* $\mathrm{I}(\mathcal{O})$ of $\mathcal{O}$.

The group $\mathrm{I}(\mathcal{O})$ will have no elements of finite order. This changes if we introduce the following equivalence relation: two fractional ideals A, A' are equivalent if there exists an element $f \in K$ such that $A = fA'$.

The resulting group is denoted by $\mathrm{Cl}(\mathcal{O})$, the *ideal class group* of $\mathcal{O}$. The neutral element in the class group consists of the group of *principal ideals* $\mathrm{Princ}(\mathcal{O})$ and so $\mathrm{Cl}(\mathcal{O}) = \mathrm{I}(\mathcal{O})/\mathrm{Princ}(\mathcal{O})$. The next criterion to choose $\mathcal{O}$ is that $\mathrm{Cl}(\mathcal{O})$ should have many elements of finite order. In fact, in all existing systems $\mathrm{Cl}(\mathcal{O})$ is a finite group.

This advantage has a price. As usual one computes in quotient structures like $\mathrm{Cl}(\mathcal{O})$ by using representatives (in our case, ideals) of the classes, composes these representatives, and then forms the class of the result. Since there are infinitely many elements in an ideal class, this does not lead to an algorithm if one does not have more information. There are two ways out of this difficulty.

1. It is possible to find a distinguished element in each ideal class (respectively a finite [small] subset of such elements).
2. It is possible to define "coordinates" and addition formulas directly for elements of $\mathrm{Cl}(\mathcal{O})$.

The first possibility can be used if we have efficient "reduction algorithms" that compute the distinguished element in ideal classes, and the second possibility can be realized if there is a geometric background of $\mathrm{Cl}(\mathcal{O})$.

Most interesting cases are those for which both methods can be used!

23.1.2.b Realization in number fields

Having in mind the requirements stated above, it is no wonder that the first suggestions for systems of discrete logarithms based on ideal classes came from number theory [BUWI 1988]. The highly developed "computational number theory" based on Minkowski's geometry of numbers made it possible to compute efficiently with ideal class groups of the ring of integers $\mathcal{O}_K$ of number fields K that are finite algebraic extensions of $\mathbb{Q}$. An important special case is that K is an imaginary quadratic field. In this case already Gauß developed a fast algorithm for computing with ideal classes. It relies on the identification of these classes with classes of binary quadratic forms. The distinguished ideals correspond to the uniquely determined reduced quadratic forms. A given form is transformed into a reduced one by an explicit algorithm using Euclid's algorithm which runs in polynomial time.

The disadvantage of these systems is that the *index calculus attack* is very effective, i.e., the algorithm based on the principles explained in Chapter 20 has only subexponential complexity. One uses prime ideals of $\mathcal{O}_K$ with small norm to build up a factor base for $\mathrm{Cl}(\mathcal{O})$.

23.1.2.c Realization in function fields

This motivates us to look for rings $\mathcal{O}$ having a similar simple structure as $\mathcal{O}_K$ but being more resistant against index calculus attacks. Since the beginning of arithmetic geometry in the last century, it is well-known and has been often exploited that the ring $\mathcal{O}_{C_a}$ of regular functions on a nonsingular irreducible affine curve C_a defined over a finite field $\mathbb{F}_q$ is a Dedekind domain with

finite ideal class group $\mathrm{Cl}(\mathcal{O}_{C_a})$ and that the arithmetic is analogous to the arithmetic of the ring of integers in number fields. The quotient field of $\mathcal{O}_{C_a}$ is the function field (of meromorphic functions) of C_a and is denoted by $K(C_a)$.

These rings and their ideal class groups are explained in Section 4.4. As size of ideals one uses their degree, and the theorem of Minkowski about points in lattices is replaced by the Riemann–Roch theorem, which yields (amongst other results) that in every ideal class in $\mathrm{Cl}(\mathcal{O}_{C_a})$ there are ideals contained in $\mathcal{O}_{C_a}$ with small degree. (For a precise formulation see Section 4.4.6 and especially Theorem 4.143.) The reader familiar with the geometry of numbers will remark that the logarithm of the absolute value of number fields is replaced by the *genus g of the curve C_a*, respectively of its function field $K(C_a)$ (cf. Definition 4.107). Already at this stage it follows that the group $\mathrm{Cl}(\mathcal{O}_{C_a})$ is at least as good as a candidate for groups, in which the group operation can be executed effectively, as the rings of integers in number fields.

But we can go further because of the geometrical background of $\mathcal{O}_{C_a}$. The prime ideals in $\mathcal{O}_{C_a}$ are closely related to points on C_a. To see this one takes the points P on C_a with coordinates in the algebraic closure $\overline{\mathbb{F}}_q$ of $\mathbb{F}_q$ *together* with the operation of the Galois group $G_{\mathbb{F}_q}$ of $\mathbb{F}_q$ (cf. Section 4.4.4). The resulting Galois orbits $G_{\mathbb{F}_q} \cdot P$ of points correspond one-to-one to the prime ideals (always taken differently from $\{0\}$) of $\mathcal{O}_{C_a}$ consisting of functions $f \in \mathcal{O}_{C_a}$ vanishing in P. By taking the order of vanishing of f at P we define a valuation v_P on $\mathcal{O}_{C_a}$, and hence on $K(C_a)$, whose valuation ring contains $\mathcal{O}_{C_a}$. Its equivalence class is called a *prime divisor* $\mathfrak{p}$ of $K(C_a)$ corresponding to $G_{\mathbb{F}_q} \cdot P$.

One of the major advantages of the geometric theory of curves over finite fields is that it is very easy to compactify the affine curve C_a by going to its projective closure. In principle this is done by homogenizing the equations defining C_a. The procedure is explained in Section 4.22. We find a projective curve C containing the affine curve C_a and the difference set of points consists of finitely many "points at infinity." The Galois orbits of these points define, as above, equivalence classes of valuations of $K(C_a)$ and hence divisors $\mathfrak{p}$. They correspond one-to-one to the equivalence classes of valuations (and hence to prime divisors) of $K(C_a)$ which do not contain $\mathcal{O}_{C_a}$.

Because of the compactness of C the only functions that are regular at all points of C are constants. To study the arithmetic of C one has to introduce the divisor group of C (see Section 4.4.2) replacing the ideal group of $\mathcal{O}_{C_a}$. Divisors can be identified with formal sums of points on C with integers as coefficients. The role of principal ideals is taken by principal divisors (cf. Definition 4.102) and the resulting quotient group is the *divisor class group of degree* 0 of C denoted by Pic_C^0. By construction, it is closely related to $\mathrm{Cl}(\mathcal{O}_{C_a})$ For instance, if there is only one point at infinity of C then Pic_C^0 is isomorphic to $\mathrm{Cl}(\mathcal{O}_{C_a})$ by Proposition 4.140. We shall assume from now on that this is satisfied. We denote by P_∞ this *unique* point at infinity.

The group Pic_C^0 is (in a canonical way) isomorphic to the group of rational points of the Jacobian variety J_C of C which is an abelian variety (cf. Definition 4.134) defined over $\mathbb{F}_q$ and intrinsically attached to C. Together with the abstract theory comes a very concrete way to construct J_C (up to birational equivalence): a consequence of the theorem of Riemann–Roch is that in every divisor class of degree 0 of C, there is a divisor of the form $\sum_{i=1}^{r} P_i - rP\infty$ with $r \leqslant g$. Details are found in Section 4.4.4.

23.1.2.d Conclusion

Beginning with an affine curve C_a and its ring of regular functions $\mathcal{O}_{C_a}$ we construct the associated projective curve C. Under the assumption that there is only a single point at infinity we can interpret Pic_C^0, its divisor class group of degree 0, as both — as class group of ideals of the ring $\mathcal{O}_{C_a}$, for which we have an efficient reduction theory, and as an abelian variety. Hence, we have a compact way to represent its elements and we can introduce coordinates for ideal classes and expect algebraic formulas describing the group composition in these coordinates.

Since we have related the ideal class group of $\mathcal{O}_{C_a}$ to rational points of abelian varieties we can use their rich structure in general as described in Section 4.3 and especially over finite fields (see Section 5.2).

For the choice of a ring $\mathcal{O}_{C_a}$ or equivalently, an affine or projective curve C we have a lot of freedom compared with the case that the chosen ring is a ring of integers in a number field. The parameters are

1. the characteristic p of the base field $\mathbb{F}_p$,
2. the degree d of the ground field $\mathbb{F}_q$ over $\mathbb{F}_p$,
3. the genus $g_C = g$ of the curve C (resp. the function field $K(C)$).

The number of isomorphism classes of curves of genus $g = 1$ over $\mathbb{F}_{p^d}$ is about p^d and for genus $g \geqslant 2$ it is of size $p^{d(3g-3)}$.

Even more important is that in the geometric case we have a much stronger relation between these parameters and the expected order of the divisor class group. By Theorem 5.76 of Hasse–Weil we get

$$|\mathrm{Pic}^0_C| \ \sim \ p^{dg}.$$

So, if ℓ is the desired size of the group of prime order we want to embed into Pic^0_C, we have to take $d \lg p$ slightly larger than $\lg(\ell)/g$.

The explicit construction of the Jacobian variety represented as divisors of degree zero containing at most g affine points yields that the number of bits needed to represent group elements is $O(dg \lg p) = O(\lg \ell)$.

23.1.3 Examples: elliptic and hyperelliptic curves

In this section we shall apply our general theory to two special families of curves called elliptic and hyperelliptic curves. The assumption we have made in Section 23.1.2.c that there is only a single point at infinity implies that the hyperelliptic curves have a rational Weierstraß point. In Section 4.4.2.b and Chapter 14 we have studied these curves in great detail. We shall repeat their definition and crucial properties for the convenience of the reader.

23.1.3.a Elliptic curves

An elliptic curve E defined over $\mathbb{F}_q$ is a projective absolute irreducible curve of genus 1 with a rational point P_∞. It can be given by an affine Weierstraß equation

$$E_a : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

If the characteristic is prime to 6 this equation can be transformed to

$$E_a : y^2 = x^3 + a_4x + a_6, \ \text{ with } \ a_4, a_6 \in \mathbb{F}_q.$$

For normal forms and invariants of the curve we refer to Table 4.1.

The ring $\mathcal{O}_{E_a}$ is $\mathbb{F}_q[x,y]/(y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6)$. So it is a polynomial order in $\mathbb{F}_q(x,y)$ which has rank 2 over $\mathbb{F}_q[x]$.

By Riemann–Roch, we find in each ideal class of $\mathcal{O}_{E_a}$ a uniquely determined prime ideal M_P of degree 1. It is generated by the two functions $x - x_1, y - y_1$ with $x_1, y_1 \in \mathbb{F}_q$ corresponding to the point $P = (x_1, y_1)$ of $E(\mathbb{F}_q)$. The Jacobian of E is isomorphic to E with zero element P_∞. The point P corresponds to the divisor class of $P - P_\infty$. Hence the $\mathbb{F}_q$-rational points of E form in a natural way an abelian group. The addition law can be expressed either by polynomials in the

homogeneous coordinates of points on E or by rational functions (with the appropriate interpretation if the point P_∞ is involved) in the affine coordinates (x_1, y_1).

The explicit formulas can be found in great detail in Section 4.4.5 and Chapter 13.

23.1.3.b Hyperelliptic curves

A generalization of elliptic curves are hyperelliptic curves C. These are projective curves of genus $g > 1$. We assume that they have a rational Weierstraß point P_∞. Then they are determined by affine equations

$$C_a : y^2 + h(x)y = f(x), \text{ with } h(x), f(x) \in \mathbb{F}_q[x], \tag{23.1}$$

where $\deg(h) \leqslant g$ and $\deg(f) \leqslant 2g + 1$. Hence $\mathcal{O}_{C_a} = \mathbb{F}_q[x, y]/\big(y^2 + h(x)y - f(x)\big)$ is a polynomial order of rank 2 in $\mathbb{F}_q(x, y)$.

By Riemann–Roch, we find in each ideal class of $\mathcal{O}_{C_a}$ an ideal lying in $\mathcal{O}_{C_a}$ of degree less than or equal to g. Again we get that $\mathrm{Cl}(\mathcal{O}_{C_a})$ is isomorphic to Pic^0_C, the divisor class group of degree 0 of C. In the language of divisors we get that in each divisor class of degree 0 there is a divisor $D = \sum_{i=1}^{r} P_i - rP_\infty$ where P_i are points on C_a, which are now not necessarily rational over $\mathbb{F}_q$. If we assume that D is reduced (cf. Theorem 4.143) then D is uniquely determined. For computations the representation by ideals is most convenient. It leads to the Mumford representation discussed already in Theorem 4.145, and which we repeat here to give a flavor of how to introduce "coordinates" for compact presentations of the classes.

Theorem 23.1 (Mumford representation)
Let C be a genus g hyperelliptic curve given as in (23.1). Each nontrivial group element $\overline{D} \in \mathrm{Pic}^0_C$ can be represented via a unique pair of polynomials $u(x)$ and $v(x)$, $u, v \in \mathbb{F}_q[x]$, where

(i) u is monic,
(ii) $\deg v < \deg u \leqslant g$,
(iii) $u \mid v^2 + vh - f$.

Let $D \in \overline{D}$ be the unique reduced divisor, i.e., $D = \sum_{i=1}^{r} P_i - rP_\infty$, where $P_i \neq P_\infty, P_i \neq -P_j$ for $i \neq j$ and $r \leqslant g$. Put $P_i = (x_i, y_i)$. Then the divisor class of D is represented by

$$u(x) = \prod_{i=1}^{r} (x - x_i)$$

and the property that if P_i occurs n_i times then

$$\left(\frac{d}{dx}\right)^j \big[v(x)^2 + v(x)h(x) - f(x)\big]_{|x=x_i} = 0, \text{ for } 0 \leqslant j \leqslant n_i - 1.$$

So the polynomials $[u, v]$ can be taken as coordinates of elements in Pic^0_C. The group operations can be computed using this representation as given below in Cantor's algorithm. Moreover, it is explained in Section 14.1 how to express the group law in terms of the coefficients of u and v.

Algorithm 23.2 Cantor's algorithm

INPUT: Two divisor classes $\overline{D}_1 = [u_1, v_1]$ and $\overline{D}_2 = [u_2, v_2]$ on the curve $C : y^2 + h(x)y = f(x)$.
OUTPUT: The unique reduced divisor D such that $\overline{D} = \overline{D}_1 \oplus \overline{D}_2$.

1. $d_1 \leftarrow \gcd(u_1, u_2)$ $\qquad [d_1 = e_1u_1 + e_2u_2]$
2. $d \leftarrow \gcd(d_1, v_1 + v_2 + h)$ $\qquad [d = c_1d_1 + c_2(v_1 + v_2 + h)]$

3. $s_1 \leftarrow c_1 e_1$, $s_2 \leftarrow c_1 e_2$ and $s_3 \leftarrow c_2$
4. $u \leftarrow \dfrac{u_1 u_2}{d^2}$ and $v \leftarrow \dfrac{s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + f)}{d} \bmod u$
5. **repeat**
6. $\quad u' \leftarrow \dfrac{f - vh - v^2}{u}$ and $v' \leftarrow (-h - v) \bmod u'$
7. $\quad u \leftarrow u'$ and $v \leftarrow v'$
8. **until** $\deg u \leqslant g$
9. make u monic
10. **return** $[u, v]$

23.1.4 Conclusion

To design DL-based cryptosystems we have the following results:

- In ideal class groups $\mathrm{Cl}(\mathcal{O})$ of orders $\mathcal{O}$ in number fields or function fields over finite fields one can perform the group operation in polynomial time.
- Let $\mathcal{O}$ be the ring of regular functions on an affine nonsingular curve C_a of genus g over a finite field $\mathbb{F}_q$. There is a close connection between $\mathrm{Cl}(\mathcal{O})$ and Pic^0_C, the divisor class group of the projective curve associated to C_a. Moreover Pic^0_C is equal to the set of rational points of the Jacobian variety of C. So its order is of size g^g.
- If C is an elliptic curve E we have that

$$\mathrm{Pic}^0_E \simeq \mathrm{Cl}\big(\mathbb{F}_q[x,y]/(y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6)\big).$$

  The curve E is isomorphic to its Jacobian variety with P_∞ as zero element. The elements in Pic^0_E correspond one-to-one to the rational points $E(\mathbb{F}_q)$ and hence can be represented by bit strings of size $O(\lg|E(\mathbb{F}_q)|) = O(\lg q)$. The addition law described by explicit formulas can be found in Chapter 13 and is done in $O(\lg q)$ bit operations.
- If C is a hyperelliptic curve of genus g with an $\mathbb{F}_q$-rational Weierstraß point P_∞ we have that

$$J_C(\mathbb{F}_q) \simeq \mathrm{Pic}^0_C \simeq \mathrm{Cl}\big(\mathbb{F}_q[x,y]/\big(y^2 + h(x)y - f(x)\big)\big),$$

  with $h, f \in \mathbb{F}_q[x]$ and $\deg(h) \leqslant g$, $\deg(f) = 2g + 1$.
  Hence, the points in Pic^0_C can be given by ideals in Mumford representation and the addition is done by Cantor's algorithm. So Pic^0_C is a group in which elements can be represented by bit strings of size $O(\lg|\mathrm{Pic}^0_C|)$ and addition is done in $O(\lg|\mathrm{Pic}^0_C|)$ bit operations.

Remark 23.3 Obviously the arithmetical properties of elliptic and of hyperelliptic curves are completely analogous. For this reason we often interpret elliptic curves as "hyperelliptic curves of genus 1" without mentioning it.

23.2 Security of systems based on Pic_C^0

In the last section we have seen that ideal class groups attached to elliptic and hyperelliptic curves lead to groups in which elements have a compact representation and in which the composition law is computable in polynomial time.

In generic black-box groups of order N, the results by Shoup show that the discrete logarithm problem cannot be solved in less than $O(\sqrt{N})$. Hence, if only the generic algorithms (cf. Chapter 19) can be used, we can consider the discrete logarithm problem to be hard as they run in time $O(\sqrt{N})$.

There are also some results showing that the discrete logarithm problem and the Diffie–Hellman problem on elliptic curves have at least a certain complexity when attacked by means like Boolean functions or polynomials. However, the results given in [LAWI 2002, LAWI 2003] are much weaker than expected and can only give an indication on the hardness — only the opposite result would have catastrophic implications.

We now deal with security issues showing under which conditions there exist attacks that are stronger than the generic ones.

23.2.1 Security under index calculus attacks

We have stressed the similarity with the case of orders in imaginary quadratic fields and, hence, the disadvantage: namely, the possibility of computing the discrete logarithm by index calculus algorithms must be discussed.

In fact, one has seen in Chapter 21 that this type of algorithms works in certain ranges. In the sequel we briefly present these results.

We recall the result of Gaudry, Enge, and Stein [ENG 2002, ENGA 2002, ENST 2002] which is strong for large genus g.

Theorem 23.4 For $g/\ln(q) > t$ the discrete logarithm in the divisor class group of a hyperelliptic curve of genus g defined over $\mathbb{F}_q$ can be computed with complexity bounded by

$$L_{q^g}\left(\tfrac{1}{2}, \sqrt{2}\Big(\big(1+\tfrac{1}{2t}\big)^{1/2} + \big(\tfrac{1}{2t}\big)^{1/2}\Big)\right).$$

The results of Gaudry [GAU 2000b] and more recently of Thériault [THÉ 2003a] and Gaudry, Thériault, and Thomé [GATH$^+$ 2004] are serious for hyperelliptic curves of relatively small genus (in practice: $g \leqslant 9$).

There is an index calculus attack of complexity

$$O\big(g^5 q^{2-\frac{2}{g}+\epsilon}\big)$$

with "reasonably small" constants and even for $g = 3$ and 4 the security is reduced.

The explicit result for $g = 4$ is: for hyperelliptic curves C of genus 4 defined over $\mathbb{F}_q$ there is an index calculus algorithm that computes the discrete logarithm in Pic_C^0 with complexity

$$O\big(q^{3/2+\epsilon}\big) = O\big(|\mathrm{Pic}_C^0|^{0.375}\big).$$

This means that the discrete logarithm is considerably weaker than generically expected.

The explicit result for $g = 3$ is: for hyperelliptic curves C of genus 3 defined over $\mathbb{F}_q$, there is an index calculus algorithm that computes the discrete logarithm in Pic_C^0 with complexity

$$O\big(q^{4/3+\epsilon}\big) = O\big(|\mathrm{Pic}_C^0|^{0.44}\big).$$

This shows that asymptotically genus 3 curves are weaker than elliptic curves and even for group sizes encountered in practice the security is reduced. However, the main application of genus three curves is over fields of 64 bits using one word on a 64-bit processor or two words on a 32-bit processor. This setting can easily offer group sizes of 192 bits. So even with some percent less security, one is above the usual security threshold of working in groups of size 2^{160}.

We stress that these results hold for *all* curves of that genus, not only for hyperelliptic curves. The difference appears only in the constants that are smaller for hyperelliptic curves.

23.2.1.a Conclusion

We can summarize our results.

- For curves C of genus $g \geqslant 4$, a direct application of index calculus algorithms to $\mathrm{Cl}(\mathcal{O}_{C_a})$ gives a complexity of the DLP that is smaller than the generic one. Hence, orders related to curves of genus $g \geqslant 4$ or closely related abelian varieties should not be used as crypto primitives for public-key systems, or, if one has very good reasons for using them, one has to enlarge the group size considerably.
- The state of the art is: we have only three types of rings $\mathcal{O}$ which avoid serious index calculus attacks and for which addition in $\mathrm{Cl}(\mathcal{O})$ is fast enough. These are the *maximal orders belonging to curves of genus* 1, 2 *and* 3. Even for $g = 3$ one needs to take into account the group size to compare the complexities of the generic attacks and Thériault's large prime variant of the index calculus attack and the more recent double large prime variants (cf. Section 21.3).

23.2.2 Transfers by Galois theory

In the last section we have studied a "generic" attack to compute discrete logarithms based on ideal class groups and as a consequence we have to exclude all (not only hyperelliptic) curves of genus $g \geqslant 4$ from the candidate list. Now we want to show that special curves of genus $g \leqslant 3$ over special fields can deliver weak DL systems though no direct application of an index calculus algorithm can be used. The method is to transfer the discrete logarithm problem in the original group in polynomial time to a group in which index calculus algorithms are efficient. The transfer maps known today are treated in Chapter 22.

In the following sections we shall always work with a projective absolutely irreducible nonsingular curve C defined over the finite field $\mathbb{F}_q$.

23.2.2.a Pairings

The first method uses the duality theory of Jacobian varieties.

First look at the special case that ℓ divides q. Then we can transfer Pic_C^0 in polynomial time into a subgroup of a vector space of dimension g over $\mathbb{F}_q$, and in this group the discrete logarithm has complexity $O\big((2g-1)\lg(q)^k\big)$, where k is a small constant. Hence, one has to avoid this case in all circumstances.

So assume now that ℓ is prime to q. In Chapter 6 one finds the definition and the mathematical background of the Tate pairing in the Lichtenbaum version. In Chapter 16 one finds algorithms to implement the pairing efficiently. The result to be kept in mind is Theorem 6.15. Together with its corollary it states: let $\mathbb{F}_q$ be the field with q elements, ℓ a prime number prime to q and $k \in \mathbb{N}$ be minimal with $\ell \mid (q^k - 1)$.

There is a bilinear map

$$T_\ell : \mathrm{Pic}_C^0[\ell] \times \mathrm{Pic}_{C\cdot\mathbb{F}_{q^k}}^0 \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell,$$

which is nondegenerate on the right side, i.e., for a random element in $\overline{D} \in \mathrm{Pic}^0_{C \cdot \mathbb{F}_{q^k}}$ the map

$$\begin{aligned} T_{\ell,\overline{D}} : \mathrm{Pic}^0_C[\ell] &\to \mathbb{F}^*_{q^k}[\ell] \\ P &\mapsto T_{\ell,\overline{D}}(P) := \big(T_\ell(P,\overline{D})\big)^{\frac{q^k-1}{\ell}} \end{aligned}$$

is an injective homomorphism of groups, which has complexity $O(k \lg q)$.

As a consequence we get from the results in Chapter 20 the following proposition.

Proposition 23.5 The discrete logarithm in Pic^0_C has complexity $L_{q^k}\big(\frac{1}{2}, \sqrt{2}\big)$ and so is subexponential in q^k.

Example 23.6 Assume that $k \leqslant \ln q$. Then the complexity to compute the discrete logarithm in Pic^0_C is $L_q\big(\frac{1}{2}, \sqrt{2k}\big)$.

It is rather easy to avoid curves C of genus $\leqslant 3$ for which a large prime ℓ divides $|\mathrm{Pic}^0_C|$ and for which at the same time the corresponding k is relatively small. For elliptic curves we have proved in Section 6.4.2 a necessary and sufficient criterion for this situation. It turns out that for *supersingular elliptic curves* (cf. Definition 4.74) k is always small, namely less than or equal to 6, while for ordinary random elliptic curves k can be expected to be large. Analogous results hold for curves of genus $g \leqslant 3$. So, for a cryptographic system with discrete logarithms as crypto primitive, the divisor class groups Pic^0_C of supersingular curves are to be avoided — or, if there is a strong reason for using them, one has to choose the parameters in such a way that for given k the complexity $L_q\big(\frac{1}{2}, \sqrt{2k}\big)$ is large enough. One of these strong reasons is motivated by the constructive aspects of the Tate pairing related to the bilinear structure on Pic^0_C. A detailed discussion is found in Chapter 24.

23.2.2.b Weil descent

In Section 23.1.2.d we have given a list for parameters of DL systems. By index calculus methods we have seen that we should restrict the genus to be less than or equal to 3. Because of duality we have to make sure that for the pair $(\mathbb{F}_q, \ell)$ the minimal number k with $\ell \mid q^k - 1$ has to be large enough.

Now we come to the parameter d with $q = p^d$. Assume that $d > 1$ and that $d_0 \mid d$ with $d_0 < d$. So we have a nontrivial operation of the Frobenius $\phi_{p^{d_0}}$ and we can use Weil descent (see Chapters 7 and 22). The results there give strong indications for the weakness of the discrete logarithm problem if either d/d_0 is small or if d is a prime with the additional property that there is a small number t with $2^t \equiv 1 \pmod{d}$. For instance d is not allowed to be a Mersenne prime number. These assertions follow from the GHS algorithm (cf. Section 22.3.2) and its generalizations. But the reader should be aware that the algorithms described in Chapter 22 are only examples of possible attacks and, contrary to the attack by pairings, we do not have a clean criterion for a curve defined over nonprime fields with composite degree to be resistant against Weil descent attack.

The new ideas of Gaudry and Diem make the situation even worse. For instance the result of Gaudry in Section 22.3.5.a can be applied to *all* elliptic curves defined over extension fields of degree 4, and the result of Diem stated in Section 22.3.5.b at least gives a strong hint that for elliptic curves d should be a large prime.

Taking this into account the choice of the ground field $\mathbb{F}_q$ for a DL system realized in Pic^0_C should be:

- a prime field, namely $\mathbb{F}_q = \mathbb{F}_p$, or
- an extension field $\mathbb{F}_q = \mathbb{F}_{p^d}$ with p very small (usually $p = 2$) and d a prime such that the multiplicative order of 2 modulo p is large.

There should be a very good reason if one does not follow this rule, and then a careful discussion of the security has to be performed. We shall give one example for this.

Example 23.7 Trace zero varieties introduced in Section 15.3 are constructed from curves defined over $\mathbb{F}_p$ which are considered over $\mathbb{F}_{p^d}$ for $d = 3$ or $d = 5$. The latter is only proposed for elliptic curves while the former construction could be used for elliptic and genus 2 curves.

As DL system one uses the group of divisor classes that have trace zero (hence the name), i.e.,

$$G = \left\{ \overline{D} \in \mathrm{Pic}^0_C(\mathbb{F}_{p^d}) \;\middle|\; \overline{D} \oplus \phi_p(\overline{D}) \oplus \cdots \oplus \phi_p^{d-1}(\overline{D}) = 0 \right\}.$$

Geometrically this group is isomorphic to an abelian variety $\mathcal{G}$ of dimension $g(d-1)$ in the Weil descent of C, which shows $|G| \sim p^{g(d-1)}$. The advantages of this group come from efficient implementation and will be made clear in the following section. From the security considerations the choice of $(g, d) = (1, 3)$ has no known weakness. For good choices of parameters the other two possibilities that lead to groups of size p^4 offer better security than genus 4 curves or elliptic curves over fields $\mathbb{F}_{p^4}$, which would have the same group order. For $(g, d) = (2, 3)$ this comes from the fact that $\mathcal{G}$ can always be embedded in the Jacobian of a genus 6 curve over $\mathbb{F}_p$ and that there the index calculus algorithm runs in $O\big(q^{\frac{22}{13}+\epsilon}\big) = O\big(|G|^{\frac{11}{16}+\epsilon}\big)$. In this rough bound we did not include the constants. A more thorough study (cf. Section 22.3.4.b, [DISC 2003]) reveals that for low security applications, e.g., group sizes of 128 bits, the security is close to the generic one whereas for larger bit sizes one loses asymptotically about 17% of security.

23.2.2.c Conclusion

The results of Section 23.2.2 show that one has to be careful with the choice of the *pair* $(C, \mathbb{F}_q)$ if one wants to have instances in which the complexity of algorithms computing the discrete logarithm is $O(\ell^{1/2})$.

1. One has to choose ℓ so that it does not divide q and so that the field $\mathbb{F}_q(\zeta_\ell)$ is an extension of $\mathbb{F}_q$ of sufficiently large degree k. For instance $k \geqslant 1000$ should be ensured. (Note that this does not mean that one needs to actually compute k but that one checks for all $k' \leqslant k$ that $\ell \nmid q^{k'} - 1$.) This excludes especially the case that C is supersingular.
2. One should take $\mathbb{F}_q$ either as prime field or as an extension of a field of small characteristic p (e.g., $p = 2$), which has prime degree d over $\mathbb{F}_p$. Moreover the number 2 should have large order modulo d, e.g., d must not be a Mersenne or a Fermat prime number.

If one has strong reasons not to follow these directions, one has to make a careful analysis of the situation.

Example 23.8 Assume the situation that one would like to find a group of order equivalent to q^4 as, e.g., the arithmetic in $\mathbb{F}_q$ is particularly suited to the hardware. In this case, one should not take curves of genus 4 over $\mathbb{F}_q$ or elliptic curves defined over $\mathbb{F}_{q^4}$, as due to the attacks described in Section 22.3.5.a the size of q needs to be chosen much larger. One can do better with trace zero varieties of curves of genus 2 defined over $\mathbb{F}_q$ with respect to extensions of degree 3 (see Example 23.7).

23.3 Efficient systems

In the previous section we have shown that ideal class groups of function fields over finite fields allow us to obtain groups in which the discrete logarithm is supposed to be hard to compute, given

that the genus is less than or equal to 3. The parameters p, d, and g may then be chosen to accommodate fast computation of scalar multiples in the group provided that the group size $O(p^{dg})$ is of the correct size and none of the weaknesses mentioned before is introduced.

23.3.1 Choice of the finite field

From the point of view of finite field arithmetic, prime fields $\mathbb{F}_p$ and binary fields $\mathbb{F}_{2^d}$ are the most common choices, but for some implementations optimal extension fields and their generalizations might offer advantages, as they are ideally suited to the word size. For full details on arithmetic on finite fields we refer to Chapter 11, and the mathematical background can be found in Chapter 2. For special considerations for hardware implementation one should consult Chapter 26.

23.3.1.a Prime fields

In prime fields $\mathbb{F}_p$, addition and subtraction are performed as in the integers and the result is reduced modulo the prime p. In general, much of the considerations for integer arithmetic (cf. Chapter 10) can be applied for the arithmetic modulo p.

For multiplication one uses the schoolbook method or Karatsuba's trick. Fast multiplication methods like FFT do not apply for the small size of p we are considering here. Squaring has about the same complexity as multiplication and can either be implemented separately (which can lead to a speedup in trade-off for more code) or by reusing the multiplication routine. In general one should consider the multiplication separately from the reduction and take into account the effect that for computing $ab + cd$, for field elements a, b, c, d, one only needs one reduction instead of two.

Inversions and divisions are computed using the (binary) extended gcd.

To have fast arithmetic in $\mathbb{F}_p$ it is advisable to choose p of low Hamming weight, ideally of the form $2^{wn} + c$, where w is the word size and c is small. This allows us to compute the modular reduction more efficiently.

Montgomery representation of elements in $\mathbb{F}_p$ speeds up the computations even further. The element x is represented by xR, where R is the smallest power of 2^w larger than p. This representation behaves well with respect to addition, multiplication, and inversion and the reductions are particularly simple. For full details we refer to Algorithms 11.3 and 11.9.

23.3.1.b Extension fields

To represent extension fields one has two general methods, either using the multiplicative or the additive structure of $\mathbb{F}_{p^d}$. In the first case one uses a generator g and represents each element as a power of g. This way, multiplications are very efficient but additions are problematic. For small fields one can use a lookup table but this gets inefficient very quickly.

So, for implementations in cryptographically relevant ranges, representations using the additive structure of $\mathbb{F}_{p^d}$ as d-dimensional vector space over $\mathbb{F}_p$ are favored. In this representation addition is done coefficient-wise.

There are two main trends for choosing the basis: either find an irreducible polynomial $m(X) \in \mathbb{F}_p[X]$ of degree d and use $\left(1, \theta, \ldots, \theta^{d-1}\right)$ as basis, where θ is a root of $m(X)$ over $\mathbb{F}_{p^d}$, or choose a basis of the form $\left(\alpha, \alpha^p, \ldots, \alpha^{p^{d-1}}\right)$. The latter is called a *normal basis*; it has the advantage that the operation of the Frobenius automorphism ϕ_p is computed by a cyclic shift of the coefficients. As a drawback, multiplications are more complicated than in *polynomial basis* representation, where they are computed by a multiplication of polynomials followed by a reduction modulo $m(X)$.

We now give some details for binary fields $\mathbb{F}_{2^d}$ and optimal extension fields $\mathbb{F}_{p^d}$. Some applications discussed in Chapter 24 use the field $\mathbb{F}_{3^d}$. It shares many properties with $\mathbb{F}_{2^d}$. Hence, the mathematical background is covered in the following paragraph. Note that the basic arithmetic in

$\mathbb{F}_3$ is slower than in $\mathbb{F}_2$ as two bits are needed to represent a field element.

Binary fields

The security considerations impose that d is a large prime. So, in all applications d is odd implying that $\mathrm{Tr}_{\mathbb{F}_{2^d}/\mathbb{F}_2}(1) = 1$. This means that some transformations on the curve equations, discussed later, are always possible. On the other hand this means that there is no type I optimal normal basis (cf. Section 11.2.2.b), i.e., if one represents the multiplication using a matrix, then it is not possible to get the most sparse type of matrix.

The use of normal basis representation can be useful if many more squarings than multiplications are needed or if one needs to compute square roots. For many of the coordinate systems on elliptic and hyperelliptic curves and ways of scalar multiplication discussed below, normal basis representation leads to slower implementations than cleverly chosen irreducible polynomials with a polynomial basis.

To be able to multiply in normal basis representation one is either given an explicit multiplication matrix or uses Gauß periods over extensions. Inversion is either done by transforming to a polynomial representation and then performing it there as described in the sequel or by using Lagrange's theorem. In the latter case one should apply an addition chain involving the Frobenius automorphism, as this map is particularly fast in normal basis representation.

We now turn our attention to polynomial basis representations. As for primes it is very useful if the irreducible polynomial is sparse, i.e., it has only few nonzero coefficients. Over $\mathbb{F}_2$ each binomial has either 0 or 1 as its root and hence it cannot be irreducible. For implementations irreducible trinomials are the best choice if they exist. Otherwise Section 11.2.1.b proposes to use redundant trinomials to obtain sparse polynomials for reduction.

Multiplication is performed as a multiplication followed by a reduction modulo $m(X)$. For the fields $\mathbb{F}_{2^d}$ a Montgomery representation of the fields exists as well. Also, for inversions the same tricks can be applied and usually they are computed as an extended gcd. For restricted devices inversions are usually too time and space-consuming. However, one should note that the ratio between multiplication and inversion is not as bad as in prime fields.

Even though in polynomial basis representation a squaring is more complicated than a cyclic shift, it is still a fast operation compared to multiplication as no mixed terms occur. A rough estimate is $\mathrm{S} = 0.1\mathrm{M}$, where S abbreviates a squaring and M stands for a multiplication.

Optimal extension fields

For some platforms or applications other choices of p and d are more advantageous. For security reasons one chooses d to be prime. The idea behind optimal extension fields and their generalization as *processor adapted finite fields* is to use a ground field $\mathbb{F}_p$ such that the elements of $\mathbb{F}_p$ fit within one word, with the consequence that they can be particularly efficiently handled. For the choice of p the same considerations as in the prime field case apply.

To construct the field extension one tries to use an irreducible binomial $m(X) = X^d - a$, where optimally a is small such that multiplications involving a can be performed by a few modular additions. Hence, reduction modulo $m(X)$ can be efficiently computed.

Addition is trivial. To multiply, the reductions are simplified by the choices of p and $m(X)$ and for small extension degrees d like those needed for trace zero varieties, one can save a few multiplications over $\mathbb{F}_p$ by making a detailed look at the code. For $d = 3$ and 5 these formulas are included in Section 11.3.6.

For inversion one uses $\alpha^{p^d-1} = 1$ and expresses

$$\alpha^{-1} = \prod_{i=1}^{d-1} \alpha^{p^i} \Big/ \prod_{j=0}^{d-1} \alpha^{p^j}.$$

Note that some of the powers can be rearranged to save even more operations. For $d = 5$ this reads

$$\alpha^{-1} = \frac{(\alpha^p \alpha^{p^2} (\alpha^p \alpha^{p^2})^{p^2}}{\alpha \big(\alpha^p \alpha^{p^2} (\alpha^p \alpha^{p^2})^{p^2}\big)}.$$

For very small $d \leqslant 4$ an approach based on linear algebra is more efficient. We refer to Section 11.3.4.

23.3.2 Choice of genus and curve equation

In this section we assume that the scalar multiplications appearing in the DL-based protocols are carried out using doublings and additions, and, depending on the storage capacities, by using some precomputed points. The latter can be applied in windowing methods as explained in Chapter 9 to reduce the number of additions needed in a scalar multiplication. As the number of doublings remains unchanged such systems need especially cheap repeated doublings.

Clearly, Cantor's algorithm can be used to perform arithmetic in the ideal class group of arbitrary hyperelliptic curves. As soon as one fixes the genus of the curve one can derive explicit formulas from the general group operations, which are usually much faster in implementations. As the group size behaves like p^{dg} a larger genus allows smaller finite fields. By the security considerations we are, however, limited to $g = 1, 2$, and 3.

23.3.2.a Elliptic curves

Elliptic curves are curves of genus 1. On them the ideal and divisor class group are isomorphic to the group of points and thus one can define addition and doubling in terms of the coordinates of points. The general addition formula is given in Section 13.1.1.

Over a binary field, each nonsupersingular curve can be given (after an isomorphic transformation) by an affine equation

$$y^2 + xy = x^2 + a_2 x^2 + a_6, \text{ where } a_2 \in \mathbb{F}_{2^d} \text{ and } a_6 \in \mathbb{F}_{2^d}^*.$$

As d should be odd, we can even choose $a_2 \in \mathbb{F}_2$, cf. Remark 13.40.

In affine coordinates the addition formulas read as follows, where $-P = (x_1, x_1 + y_1)$.

Addition

Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ such that $P \neq \pm Q$ then $P \oplus Q = (x_3, y_3)$ is given by

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a_2, \qquad y_3 = \lambda(x_1 + x_3) + x_3 + y_1, \qquad \lambda = \frac{y_1 + y_2}{x_1 + x_2}.$$

Doubling

Let $P = (x_1, y_1)$ then $[2]P = (x_3, y_3)$, where

$$x_3 = \lambda^2 + \lambda + a_2, \qquad y_3 = \lambda(x_1 + x_3) + x_3 + y_1, \qquad \lambda = x_1 + \frac{y_1}{x_1}.$$

Thus an addition and a doubling require exactly the same number of operations that is $\mathrm{I} + 2\mathrm{M} + \mathrm{S}$. Here, we use the abbreviations I for inversion, M for multiplication, and S for squaring. We use M_2 to denote a multiplication by a_2. This is counted separately as for odd d there always exists an isomorphic curve with $a_2 \in \mathbb{F}_2$ such that the multiplication is computed as an addition.

To avoid inversions one can use projective and different weighted projective coordinates. The following Table 23.1 lists the number of operations needed to double or add in different coordinate systems ($\mathcal{A}$ denotes affine coordinates, $\mathcal{P}$ projective coordinates, and $\mathcal{J}$ and $\mathcal{LD}$ stand for Jacobian and López–Dahab coordinates, respectively). For details on the arithmetic we refer to Section 13.3.

Table 23.1 *Operations required for addition and doubling*.

Doubling		Addition	
Operation	Costs	Operation	Costs
$2\mathcal{P}$	$7\mathrm{M}+4\mathrm{S}+\mathrm{M}_2$	$\mathcal{J}+\mathcal{J}$	$15\mathrm{M}+3\mathrm{S}+\mathrm{M}_2$
$2\mathcal{J}$	$5\mathrm{M}+5\mathrm{S}$	$\mathcal{P}+\mathcal{P}$	$15\mathrm{M}+2\mathrm{S}+\mathrm{M}_2$
$2\mathcal{LD}$	$4\mathrm{M}+4\mathrm{S}+\mathrm{M}_2$	$\mathcal{LD}+\mathcal{LD}$	$13\mathrm{M}+4\mathrm{S}$
$2\mathcal{A}=\mathcal{P}$	$5\mathrm{M}+2\mathrm{S}+\mathrm{M}_2$	$\mathcal{P}+\mathcal{A}=\mathcal{P}$	$11\mathrm{M}+2\mathrm{S}+\mathrm{M}_2$
$2\mathcal{A}=\mathcal{LD}$	$2\mathrm{M}+3\mathrm{S}+\mathrm{M}_2$	$\mathcal{J}+\mathcal{A}=\mathcal{J}$	$10\mathrm{M}+3\mathrm{S}+\mathrm{M}_2$
$2\mathcal{A}=\mathcal{J}$	$\mathrm{M}+2\mathrm{S}+\mathrm{M}_2$	$\mathcal{LD}+\mathcal{A}=\mathcal{LD}$	$8\mathrm{M}+5\mathrm{S}+\mathrm{M}_2$
—	—	$\mathcal{A}+\mathcal{A}=\mathcal{LD}$	$5\mathrm{M}+2\mathrm{S}+\mathrm{M}_2$
$2\mathcal{A}$	$\mathrm{I}+2\mathrm{M}+\mathrm{S}$	$\mathcal{A}+\mathcal{A}=\mathcal{J}$	$4\mathrm{M}+\mathrm{S}+\mathrm{M}_2$
$2\mathcal{A}'$	$\mathrm{I}+\mathrm{M}+\mathrm{S}$	$\mathcal{A}+\mathcal{A}=\mathcal{A}'$	$2\mathrm{I}+3\mathrm{M}+\mathrm{S}$
$2\mathcal{A}'=\mathcal{A}$	$\mathrm{M}+2\mathrm{S}$	$\mathcal{A}+\mathcal{A}$	$\mathrm{I}+2\mathrm{M}+\mathrm{S}$

If inversions are affordable, affine coordinates are preferred as the total number of field operations is lowest. Otherwise, one should use López–Dahab coordinates, as for them doublings are fastest and additions, especially mixed addition $\mathcal{A}+\mathcal{LD}=\mathcal{LD}$, are cheap as well. For the binary elliptic curves proposed in the standards a_6 is chosen to be small such that multiplications by this constant are fast. Then the standard doubling formulas should be applied, while for random curves, and thus large and changing a_6, formulas (13.9) are preferred.

In odd characteristic for $p>3$ one can make an isomorphic transform to get each elliptic curve represented by an affine equation of the form

$$y^2=x^3+a_4x+a_6, \quad \text{with } a_4,a_6\in\mathbb{F}_q,$$

such that $x^3+a_4x+a_6$ has only simple roots over $\overline{\mathbb{F}}_q$. The negative of $P=(x_1,y_1)$ is given by $-P=(x_1,-y_1)$.

Addition

Let $P=(x_1,y_1)$, $Q=(x_2,y_2)$ such that $P\neq\pm Q$ and $P\oplus Q=(x_3,y_3)$. In this case, addition is given by

$$x_3=\lambda^2-x_1-x_2, \qquad y_3=\lambda(x_1-x_3)-y_1, \qquad \lambda=\frac{y_1-y_2}{x_1-x_2}.$$

Doubling

Let $[2]P=(x_3,y_3)$. Then

$$x_3=\lambda^2-2x_1, \qquad y_3=\lambda(x_1-x_3)-y_1, \qquad \lambda=\frac{3x_1^2+a_4}{2y_1}.$$

An addition needs $\mathrm{I} + 2\mathrm{M} + \mathrm{S}$ and to compute a doubling $\mathrm{I} + 2\mathrm{M} + 2\mathrm{S}$ are used.

As stated in the previous section, in odd characteristic inversions are very time-consuming compared to multiplications. Hence, it is even more important to consider inversion-free coordinate systems. The following Table 23.2 gives an overview of the number of field operations per group operation. In addition to the above abbreviations we use $\mathcal{J}^m$ for modified Jacobian coordinates and $\mathcal{J}^c$ for Chudnovsky Jacobian coordinates. For full details on the formulas, we refer to Section 13.2.

Table 23.2 *Operations required for addition and doubling.*

Doubling		Addition	
Operation	Costs	Operation	Costs
$2\mathcal{P}$	$7\mathrm{M} + 5\mathrm{S}$	$\mathcal{J}^m + \mathcal{J}^m$	$13\mathrm{M} + 6\mathrm{S}$
$2\mathcal{J}^c$	$5\mathrm{M} + 6\mathrm{S}$	$\mathcal{J}^m + \mathcal{J}^c = \mathcal{J}^m$	$12\mathrm{M} + 5\mathrm{S}$
$2\mathcal{J}$	$4\mathrm{M} + 6\mathrm{S}$	$\mathcal{J} + \mathcal{J}^c = \mathcal{J}^m$	$12\mathrm{M} + 5\mathrm{S}$
$2\mathcal{J}^m = \mathcal{J}^c$	$4\mathrm{M} + 5\mathrm{S}$	$\mathcal{J} + \mathcal{J}$	$12\mathrm{M} + 4\mathrm{S}$
$2\mathcal{J}^m$	$4\mathrm{M} + 4\mathrm{S}$	$\mathcal{P} + \mathcal{P}$	$12\mathrm{M} + 2\mathrm{S}$
$2\mathcal{A} = \mathcal{J}^c$	$3\mathrm{M} + 5\mathrm{S}$	$\mathcal{J}^c + \mathcal{J}^c = \mathcal{J}^m$	$11\mathrm{M} + 4\mathrm{S}$
$2\mathcal{J}^m = \mathcal{J}$	$3\mathrm{M} + 4\mathrm{S}$	$\mathcal{J}^c + \mathcal{J}^c$	$11\mathrm{M} + 3\mathrm{S}$
$2\mathcal{A} = \mathcal{J}^m$	$3\mathrm{M} + 4\mathrm{S}$	$\mathcal{J}^c + \mathcal{J} = \mathcal{J}$	$11\mathrm{M} + 3\mathrm{S}$
$2\mathcal{A} = \mathcal{J}$	$2\mathrm{M} + 4\mathrm{S}$	$\mathcal{J}^c + \mathcal{J}^c = \mathcal{J}$	$10\mathrm{M} + 2\mathrm{S}$
—	—	$\mathcal{J} + \mathcal{A} = \mathcal{J}^m$	$9\mathrm{M} + 5\mathrm{S}$
—	—	$\mathcal{J}^m + \mathcal{A} = \mathcal{J}^m$	$9\mathrm{M} + 5\mathrm{S}$
—	—	$\mathcal{J}^c + \mathcal{A} = \mathcal{J}^m$	$8\mathrm{M} + 4\mathrm{S}$
—	—	$\mathcal{J}^c + \mathcal{A} = \mathcal{J}^c$	$8\mathrm{M} + 3\mathrm{S}$
—	—	$\mathcal{J} + \mathcal{A} = \mathcal{J}$	$8\mathrm{M} + 3\mathrm{S}$
—	—	$\mathcal{J}^m + \mathcal{A} = \mathcal{J}$	$8\mathrm{M} + 3\mathrm{S}$
—	—	$\mathcal{A} + \mathcal{A} = \mathcal{J}^m$	$5\mathrm{M} + 4\mathrm{S}$
—	—	$\mathcal{A} + \mathcal{A} = \mathcal{J}^c$	$5\mathrm{M} + 3\mathrm{S}$
$2\mathcal{A}$	$\mathrm{I} + 2\mathrm{M} + 2\mathrm{S}$	$\mathcal{A} + \mathcal{A}$	$\mathrm{I} + 2\mathrm{M} + \mathrm{S}$

23.3.2.b Curves of genus 2

For these curves the explicit formulas are more involved compared to elliptic curves. On the other hand the field one works in is of half the size for equal security as the group order is given by $O(p^{2d})$. In affine coordinates an addition needs $\mathrm{I} + 22\mathrm{M} + 3\mathrm{S}$ and a doubling needs $2\mathrm{S}$ more. We refer to Chapter 14 for the details on the group operation but mention that as for elliptic curves one has the choice between different coordinate systems including inversion-free systems. The choice of genus 2 curves leads to similar speed for scalar multiplication as elliptic curves. This can be seen from the implementation results listed in Table 14.13, p. 353. For each environment one needs to check which fields are faster to implement and also whether the longer code needed for the group operations for genus 2 curves is a problem. It is only possible to state the advantages or disadvantages of elliptic curves over genus 2 curves based on implementations. From a theoretical

point of view their performance is similar.

Special choices of curves, like the family of binary curves

$$y^2 + xy = x^5 + f_3x^3 + \varepsilon x^2 + f_0, \text{ with } f_3, f_0 \in \mathbb{F}_{2^d} \text{ and } \varepsilon \in \mathbb{F}_2,$$

over fields with odd d allow us to obtain especially fast doubling formulas that outperform elliptic curves. There is no similar family of elliptic curves, and, hence, the richer structure of genus 2 curves pays off.

23.3.2.c Curves of genus 3

So far explicit formulas for arithmetic on genus 3 curves were obtained in affine and projective coordinates. For general curves it seems that the number of operations is so large that the performance is worse than for elliptic and genus 2 curves. Furthermore, one needs to take into account the reduction of security due to index calculus attacks, such that the group order needs to be made even larger.

For special choices of curves like the binary curves

$$y^2 + y = f(x), \text{ with } f(x) \in \mathbb{F}_{2^d}[x]$$

particularly fast doublings can be designed without weakening security.

23.3.2.d Conclusion

Over prime fields elliptic curves and genus 2 curves offer similar efficiency, while genus 3 curves are less efficient with the current explicit formulas. For completely general curves over binary fields the same observations hold.

If one is willing to trade off generality for higher speed, there exist genus 2 and 3 curves over $\mathbb{F}_{2^d}$ offering fast arithmetic. Comparable choices cannot be made for elliptic curves. We mention that these curves constitute families and thus they are not considered special curves.

23.3.3 Special choices of curves and scalar multiplication

Some curves have extra properties that can be used to compute scalar multiplication faster. This does not mean that the single group operations, i.e., additions and doublings, are sped up, but one uses a different procedure to compute the scalar multiples.

In Example 23.7 we introduced trace zero varieties. In general, subfield curves defined over $\mathbb{F}_p$ and considered over $\mathbb{F}_{p^d}$ offer the advantage that the Frobenius endomorphism ϕ_p operates on the points of the curve by raising each coordinate to the power of p. This can be used to obtain faster methods for scalar multiplication. In Chapter 15 we showed how to compute $[n]\,\overline{D}$ by using ϕ_p. Instead of using a *double and add* algorithm, one applies a *Frobenius and add* algorithm, which leads to faster computations as ϕ_p can be computed efficiently.

It is also possible to use other endomorphisms of curves. A general curve will not have an efficiently computable endomorphism, but on specially chosen curves it can be used to speed up the scalar multiplication. The methods are described in Section 15.2.

In both cases one needs to be aware that the endomorphisms also speed up the attacks. The effects are discussed in Section 19.5.5. The use of special curves is particularly interesting for either large systems, where a central server has to perform a huge amount of encryptions and has no problems with a slightly larger group size (as long as the protocols run faster), or in lightweight cryptography applications, where the security needs are reduced but the devices cannot consume too much power and have limited storage. In this scenario curves with endomorphisms make the operations faster.

23.4 Construction of systems

Depending on the security needs one chooses the size N of the group in which the discrete logarithm is to be used. The list of candidates for secure DL systems in Section 23.1.4 gives possible choices of parameters for the system. By considering the given environment and by looking at the results on efficiency in Section 23.3 one chooses candidate pairs $(\mathbb{F}_q, C)$ such that the expected group order of Pic_C^0 is of size N and that the other needs are met. The final step is to look for primes $\ell \sim N$ for which $\mathbb{Z}/\ell\mathbb{Z}$ can be embedded into Pic_C^0 for a candidate curve C in a bit-efficient way. Usually one has to do this by randomly choosing curves C in a certain family $\mathcal{F}$ and then computing Pic_C^0.

So, one needs two ingredients: there have to be sufficiently many instances leading to an almost prime group order and one has to be able to compute the group order efficiently.

23.4.1 Heuristics of class group orders

First there has to be (at least) a heuristic prediction stating that with large probability the order of Pic_C^0 for $C \in \mathcal{F}$ is almost a prime, i.e., there is a number $c \leqslant B$ with $|\mathrm{Pic}_C^0| = c\ell$ with ℓ a prime number. The number c is called the cofactor, usually one sets the bound to $B \leqslant 1000$. A large choice of B will raise the probability of finding a suitable curve C but it will imply that we have to take the size of the ground field $\mathbb{F}_q$ (and hence the key length) larger, since we have

$$g \lg q \sim \lg c + \lg \ell.$$

The main sources for such heuristics are analytic number theory and analogous techniques over finite fields. Key words are Cohen–Lenstra heuristics about the behavior of class groups of global fields and Lang–Trotter conjectures about the distribution of traces of Frobenius automorphisms acting on torsion points of abelian varieties over number fields. Combined with sieving techniques one finds a positive probability for the existence of class numbers with only a few prime divisors. It is outside the scope of this book to give more details. For applications and statistics in the cases we are interested in we refer readers to [BAKO 1998] and [WEN 2001b]. For our purposes it is enough to state that in all families $\mathcal{F}$ that can be used in practice it will not take much time to find curves that can be used for DL systems.

In many cases one can impose $c = 1$. But one has to be cautious in special cases. There can be families $\mathcal{F}$ for which there is a number c_0 such that for each member, C, one has that c_0 divides $|\mathrm{Pic}_C^0|$ and so c has to be a multiple of c_0. For example take $\mathcal{F}$ as the set of ordinary binary elliptic curves. They are all of the form $y^2 + xy = x^3 + a_2x^2 + a_6$ with $a_6 \neq 0$. Therefore, 2 always divides the cofactor as $(0, \sqrt{a_6})$ is a point of order 2 defined over the ground field. Another example is the family $\mathcal{F}$ consisting of curves C of genus 3, which have an automorphism of order 4. Their class number is always even (cf. Section 18.3).

In other families it can occur that the class numbers are always divisible by different numbers bounded by c_0. For instance take $\mathcal{F}$ as a family of curves defined over a field $\mathbb{F}_{q_0}$ with $q_0 \mid q$. Then the class number of C over $\mathbb{F}_q$ is divided by the class number of C regarded as curve over $\mathbb{F}_{q_0}$ and so the cofactor c has to be $\geqslant (\sqrt{q_0} - 1)^{2g}$. Examples of families $\mathcal{F}$ for which $c = 1$ is possible and for which one can (efficiently) determine the group order are random elliptic curves or hyperelliptic curves of genus 2 over fields $\mathbb{F}_q$ of odd characteristic, as well as elliptic curves over prime fields constructed by the method of complex multiplication (cf. Section 18), and not isogenous to a curve with j-invariant equal to 0 or 12^3.

23.4.2 Finding groups of suitable size

In the following we shall give examples for families of curves $\mathcal{F}$ for which the (heuristic) results of the Section 23.4.1 predict that with a reasonable probability the number of elements in Pic_C^0 for $C \in \mathcal{F}$ is divisible by a large prime ℓ and for which it is possible to determine $|\mathrm{Pic}_C^0|$ efficiently. To find a curve C usable for DL systems one chooses a random curve in $\mathcal{F}$ and computes the order of Pic_C^0. If it is not of the right shape one chooses another random element in $\mathcal{F}$ and repeats the computation until finally a usable curve is found.

Once a suitable curve is found, one has to choose a base point in Pic_C^0 of order ℓ.

23.4.2.a Finding a curve

As said already the key ingredients for finding curves with divisor class group usable for DL systems are point counting algorithms, which are described in Chapter 17. Their common principle is that they compute the characteristic polynomial $\chi(\phi_q)_{\overline{C}}(T)$ of the Frobenius endomorphism ϕ_{q_C} acting on J_C (cf. Section 4.3.6) and use the fact that $|\mathrm{Pic}_C^0| = \chi(\phi_q)_C(1)$ (cf. Corollary 5.70).

Curves defined over subfields

Take as $\mathcal{F}$ the family of curves C of genus g over fields $\mathbb{F}_q$, which are defined over subfields $\mathbb{F}_{q_0}$. If q_0 is small, elementary methods like counting by enumeration (cf. Section 17.1.1) or square root algorithms (cf. Chapter 19) can be used. This is reasonable e.g., for $q_0^g \leqslant 10^{16}$.

For larger q_0 it may be necessary to replace these counting algorithms by more refined algorithms explained in the next paragraphs. Nevertheless this counting will be very fast compared to counting methods for curves defined over $\mathbb{F}_q$.

So, it is possible to compute the characteristic polynomial $\chi(\phi_{q_0})(T)$ of the Frobenius endomorphism ϕ_{q_0} over $\mathbb{F}_{q_0}$. As explained in Section 17.1.2 this can be used to compute the characteristic polynomial of ϕ_q and hence the order of Pic_C^0.

Remark 23.9 This method is especially fast if q_0 is small. Moreover, the operation of the Frobenius endomorphism ϕ_{q_0} can be used to accelerate the computation of scalar multiples in Pic_C^0 considerably (cf. Section 15.1). The key word is "Koblitz curves."

Example 23.10 The most famous Koblitz curves are

$$E_{a_2} : y^2 + xy = x^3 + a_2x^2 + 1, \text{ with } a_2 = 0 \text{ or } 1$$

seen as curves over $\mathbb{F}_2$. The characteristic polynomial is given by

$$\chi_{a_2}(T) = T^2 - (-1)^{1-a_2}T + 2$$

and the number of points of $E(\mathbb{F}_{2^d})$ is given by (15.9).

The disadvantages are that if q_0 is very small there will be only a few curves in the cryptographically interesting range. At the same time the cofactors will be divisible by $\chi(\phi_{q_0})_C(1)$, which is of size equivalent to q_0^g. Moreover the degree of $\mathbb{F}_q$ over $\mathbb{F}_{q_0}$ should be a prime number because of the transfers related to Weil descent (cf. Section 22.3). This attack has to be considered for small extensions, too.

Remark 23.11 If the degree of $\mathbb{F}_q$ over $\mathbb{F}_{q_0}$ is small ($\leqslant 5$) one is led to systems based on trace zero varieties (cf. Section 7.4.2), which can be interesting alternatives in special situations for fast arithmetic; see Section 15.3.

Random elliptic curves

One chooses the size of the allowed cofactor c and of the prime ℓ, which should divide the order of the group of rational points of an elliptic curve E. Let q be a power of a prime p of size $c\ell$. As family $\mathcal{F}$ we take the set of elliptic curves defined over $\mathbb{F}_q$.

Let E be a randomly chosen elliptic curve defined over $\mathbb{F}_q$. We can apply the SEA Algorithm 17.25 to count the elements of $E(\mathbb{F}_q)$ in $O\big(\lg(q)^{2\mu+2}\big)$ bit operations, where μ is a constant such that two B-bit integers can be multiplied in time B^μ. The algorithm is fast enough to find cryptographically usable elliptic curves even with cofactor 1 in a short time.

Random curves of genus 2

As above, the desired size of the group order is denoted by N. One chooses q to be of size $N^{1/2}$. The family $\mathcal{F}$ consists of all curves of genus 2 defined over $\mathbb{F}_q$. In [GASC 2004a] one finds an algorithm combining p-adic, ℓ-adic, and generic methods to count points on the Jacobian variety of random curves of genus 2 defined over $\mathbb{F}_q$, and to find such curves that can be used for DL systems. At the moment this is rather time-consuming — for a curve over a $\mathbb{F}_p$ with $p = 5 \times 10^{24} + 8503491$, the time reported in [GASC 2004a] is one week per curve — but it is to be expected that refinements will accelerate the algorithm in the near future.

Curves over fields of small characteristic

Let p be a small prime number, in practice $p = 2$ or $p = 3$. So, to reach cryptographic group sizes we need $q = p^d$ with d large. (For security reasons d should be a prime such that the multiplicative order of 2 modulo d is large (cf. Section 22.3)). The family $\mathcal{F}$ consists of random curves C of genus g (with $1 \leqslant g \leqslant 3$) defined over $\mathbb{F}_q$. Hence $d \sim \log_g N/g$. To count points on random curves C in $\mathcal{F}$ one uses p-adic methods as described in Section 17.3. For $p = 2$ the AGM method (cf. Section 17.3.2) generalizes Satoh's method for elliptic curves and constructs canonical liftings of Jacobian varieties of curves of genus 1, 2, and 3 in a most efficient way — provided that this curve is ordinary. For random curves this condition will be satisfied.

The AGM algorithm is very easy to implement, especially for elliptic curves. As an example, we state the algorithm for elliptic curves over $\mathbb{F}_2$. The idea is to use a recurrence to compute the trace t of the Frobenius endomorphism.

Algorithm 23.12 Elliptic curve AGM

INPUT: An elliptic curve $E : y^2 + xy = x^3 + a_6$ over $\mathbb{F}_{2^d}$ with $j(E) \neq 0$.
OUTPUT: The number of points on $E(\mathbb{F}_{2^d})$.

1. $L \leftarrow \lceil \frac{d}{2} \rceil + 3$ [L is the precision to be used]
2. $a \leftarrow 1$ and $b \leftarrow (1 + 8e) \bmod 2^4$ [e arbitrary lift of a_6]
3. **for** $i = 5$ **to** L **do**
4. $\quad (a, b) \leftarrow \big((a + b)/2, \sqrt{ab}\big) \bmod 2^i$
5. $a_0 \leftarrow a$
6. **for** $i = 0$ **to** $d - 1$ **do**
7. $\quad (a, b) \leftarrow \big((a + b)/2, \sqrt{ab}\big) \bmod 2^L$
8. $t \leftarrow \dfrac{a_0}{a} \bmod 2^{L-1}$
9. **if** $t^2 > 2^{d+2}$ **then** $t \leftarrow t - 2^{L-1}$
10. **return** $2^d + 1 - t$

In Section 17.3.2.c one finds the algorithms for $g \geqslant 2$. The theoretical background is given in Section 17.3.

A universal method of computing the characteristic polynomial of the Frobenius endomorphism ϕ_q for arbitrary (small) p and arbitrary hyperelliptic curves was developed for the first time by Kedlaya [KED 2001]. He uses the Monsky–Washnitzer cohomology, which computes the de Rham cohomology of a formal lifting of the curve. Though the background is rather involved, the algorithm is easily implemented. It is given in Section 17.3.3 as Algorithm 17.80.

If the AGM method can be used it is faster than algorithms based on Kedlaya's idea. The big advantage of Kedlaya's approach is that it is a universally applicable algorithm (which can be generalized to practically all curves). In any case, one can state that point counting for random hyperelliptic curves of genus 1, 2, and 3 over fields of characteristic 2, 3, and 5 in cryptographically relevant ranges can be done in a satisfying way.

The CM method

The methods described until now leave one gap open: it is difficult to compute the characteristic polynomial of the Frobenius endomorphism acting on the Jacobian variety of curves of genus 2 and 3 defined over fields $\mathbb{F}_q$ of large characteristic. To fill this gap, especially in the case that q is a prime number p, one can use the theory of complex multiplication relying on the results of Taniyama and Shimura. Even for elliptic curves this method remains interesting, especially if one wants to find many curves to check heuristics or to construct an elliptic curve with prescribed group order [BRST 2004], e.g., if one wants to obtain a group order with low Hamming weight.

For the background of this theory, we refer to Section 5.1. In this method one begins with the choice of a ring of endomorphisms End_C of an abelian variety that has complex multiplication and that is the Jacobian variety of a hyperelliptic curve C of genus g. This ring End_C has to be an order in a CM-field of degree $2g$. One characterizes the curve by its invariants, and to obtain the equation of the curve one computes the minimal polynomials over $\mathbb{Q}$ of the invariants, the *class polynomials*. For details on the computation of the class polynomial we refer to Section 18.1.3.

In practice both finding the ring End_C and computing the class polynomials are to be regarded as part of a precomputation. For elliptic curves they can be taken from published lists (cf. [WENG]).

The family $\mathcal{F}$ consists of the curves C_p, which are obtained from C by reduction modulo primes p. So, our space of parameters is now the set of prime numbers. Arithmetic geometry tells us that this can be regarded as the analogue of a curve, and so the richness of the family $\mathcal{F}$ is analogous to that of the family consisting of all curves of genus g over a fixed finite field $\mathbb{F}_q$.

By class field theory it is possible to determine the order of $\mathrm{Pic}^0_{C_p}$ by solving a norm equation in End_C that can be done quickly. Only after having found a "good" prime p one constructs the curve C_p in an explicit way.

The algorithm for elliptic curves E is found in Section 18.1.5. We repeat it here for this case.

Algorithm 23.13 Construction of elliptic curves via CM

INPUT: A squarefree integer $d \neq 1, 3$, parameters ε and δ, Hilbert class polynomial $H_d(X)$, desired size of p and ℓ.

OUTPUT: A prime p of the desired size, an elliptic curve $E/\mathbb{F}_p$ whose group order $|E(\mathbb{F}_p)|$ has a large prime factor ℓ.

1. **repeat**
2. **repeat** choose p prime of desired size
3. **until** $\varepsilon p = x^2 + dy^2$ with $x, y \in \mathbb{Z}$
4. $n_1 \leftarrow p + 1 - 2x/\delta$ and $n_2 \leftarrow p + 1 + 2x/\delta$
5. **until** n_1 **or** n_2 has a large prime factor ℓ

6. compute a root j of $H_d(X) \pmod p$
7. compute $E_j/\mathbb{F}_p$ from (18.1) and its twist $\widetilde{E}_j/\mathbb{F}_p$
8. **while true do**
9. take $P \in_R E_j(\mathbb{F}_p)$ and compute $Q \leftarrow [n_1]P$
10. **if** $Q = P_\infty$ **and** $[n_2]P \neq P_\infty$ **then return** p and E_j
11. **else if** $Q \neq P_\infty$ **then return** p and $\widetilde{E}_j$

The situation is more complicated for $g = 2$ but nevertheless the corresponding algorithm works very well. The details can be found in Section 18.2. For $g = 3$ one has to assume that End_C contains $\sqrt{-1}$ and hence C has an automorphism of order 4, which implies that the cofactor is divisible by 2. So we get only special curves, and the situation is not totally satisfying. Nevertheless we can construct many hyperelliptic curves of genus 3, which can be used for DL systems.

23.4.2.b State of the art of point counting

We have seen that the computation of the order of Pic_C^0 for curves C of genus g defined over fields $\mathbb{F}_q$ can be performed efficiently by not too complicated algorithms if

- The curve C is already defined over a small subfield $\mathbb{F}_{q_0}$ of $\mathbb{F}_q$, or
- The genus g is equal to 1, or
- The characteristic of $\mathbb{F}_q$ is small, or
- The genus of C is 1 or 2, the field $\mathbb{F}_q$ is a prime field, and the curve C is the reduction modulo q of a curve $\widetilde{C}$ with complex multiplication over a given order End_C in a CM-field.

For random curves C of genus 2 there is hope that one will find an efficient algorithm for point counting. The results of Gaudry and Schost [GASC 2004a] are very encouraging.

For genus $g = 3$ and $\mathbb{F}_q$ a prime field, we have to restrict ourselves, at least at the moment, to hyperelliptic curves that have an automorphism of order 4.

23.4.2.c Finding a base point

We assume now that C is a hyperelliptic curve defined over $\mathbb{F}_q$ such that

$$|\mathrm{Pic}_C^0| = c\ell$$

and that C is given by an affine equation

$$C_a : y^2 + h(x)y = f(x).$$

One chooses a random element $x_1 \in \mathbb{F}_q$ and tests whether the polynomial $y^2 + h(x_1)y - f(x_1)$ has a solution y_1 in $\mathbb{F}_q$. This will be the case with probability $1/2$. If the polynomial has no solution in $\mathbb{F}_q$ one chooses another x_1. After a few trials one has found a point $P = (x_1, y_1) \in C(\mathbb{F}_q)$. One uses the embedding of C in its Jacobian. With probability $1 - 1/\ell$ the divisor class of $\overline{P - P_\infty}$ has order divisible by ℓ and so $[c]\overline{(P - P_\infty)}$ has order ℓ. To check this one verifies that $[c]\overline{(P - P_\infty)}$ is not the neutral element.

If this is the case we can take $\overline{D} = [c]\overline{(P - P_\infty)}$ as the base point of our DL system and embed $\mathbb{Z}/\ell\mathbb{Z}$ into Pic_C^0 by mapping $n \mapsto [n]\overline{D}$ for $0 \leqslant n < \ell$.

If the trial multiplication fails, i.e., if $[c]\overline{(P - P_\infty)} = 0$, one repeats the procedure by choosing a new element x_1.

The expected time to find a base point by this method is $O(\lg \ell)$. For details concerning hyperelliptic curves implemented with explicit formulas we refer to Section 14.1.2.

23.5 Protocols

So far we have presented groups in which the discrete logarithm problem is assumed to be hard. In Chapter 1 we stated some protocols in Section 1.6 using only the structure of a DL system. We now briefly cover two issues: how to actually instantiate these protocols with groups based on Pic_C^0 of curves over finite fields; and which pitfalls one needs to be aware of.

We need to make a general remark. As of the time of writing this, only elliptic curves are in the standards. In this section we state system parameters and protocols for hyperelliptic curves by giving the generalizations from the elliptic curve setting.

23.5.1 System parameters

We now assume that the system parameters consisting of the ground field, the genus of the curve, and the curve equation are chosen in a manner that the DL problem should be hard to compute.

To use these parameters in a cryptosystem they need to be published. For the finite field this means that $q = p^d$ needs to be given but also the way the field elements are represented, i.e., for prime fields one needs to state whether the following parameters of curve and points refer to Montgomery or standard representation. In extension fields the type of basis and how to multiply in it need to be stated, e.g., for polynomial basis one gives the irreducible polynomial $m(X)$ of degree d over $\mathbb{F}_p$.

Once the field elements can be represented, the equation of the curve, the base point, and also public keys can be stated. This information is sufficient for most applications — if all users behave well. In the following we sometimes assume that an attacker could get signatures on innocent-looking messages. In practice this is not too far fetched — there are service providers that sign all packages transmitted by them so that one could easily get signatures implying arbitrary group elements unless the following checks are implemented.

Standards additionally require that the group order $N = |\mathrm{Pic}_C^0|$ is given together with the cofactor c such that $N/c = \ell$ is prime, and also ask for a small cofactor. In signature schemes like Algorithm 1.18 one needs ℓ, as the signature is an integer modulo ℓ, and an inversion modulo ℓ is required when signing. We point out that there are inversion-free signature schemes that would not require knowledge of the group order but usually they are less efficient. However, what is worse is that a malicious user could ask for a signature on an element $\overline{D}$ that comes from a subgroup of Pic_C^0 of order dividing c. If c has enough prime factors the attacker could determine the secret scalar modulo all these primes and recover a large part of the secret by using Chinese remaindering. This attack is called the *small subgroup attack*.

To avoid this attack one should check whether $\overline{D}$ has order ℓ. This can be done by either actually checking $[\ell]\overline{D} = 0$ or by computing $[h]\overline{D}$ for $h = c/p_i$ for all prime divisors p_i of c and checking that the result is not 0. Both methods require that ℓ is prime, which could also be checked.

Browsing the algorithms for addition and doubling in Pic_C^0 one notices that not all curve coefficients are needed in the arithmetic. Hence, the same set of formulas could be used for different curves. The *invalid curve attack* works by requesting a signature on a group element of $\mathrm{Pic}_{\overline{C}}^0$, where $\overline{C}$ has the same coefficients as C on those positions appearing in the group operations and is different otherwise. If it is possible to find a curve $\overline{C}$ that offers less security, this could be used to find

the secret key by attacking the DL problem in the easier group $\mathrm{Pic}_{\overline{C}}^{0}$. In fact, it is very likely that there are curves $\overline{C}$ such that the group order contains many factors of moderate size, such that the DLP could be solved by using Chinese remaindering.

Hence, the protocols require to check whether $\overline{D}$ is actually a group element.

To sum up: the system parameters consist of the five entries

$$(\mathbb{F}_{p^d}, C, \overline{D}, N, c),$$

where we assume that $\mathbb{F}_{q^d}$ contains information on the field representation and $\overline{D}$ is the base point of the system. The system parameters are accepted if

1. all field elements are correctly represented,
2. $\overline{D} \neq 0 \in \mathrm{Pic}_C^0$, and
3. $\overline{D}$ has order $\ell = N/c$.

User A has public key $\overline{D}_{\mathrm{A}} = [a_{\mathrm{A}}]\overline{D}$, which is included in her system parameters. To verify a signature issued by her or before sending her an encrypted message, B checks whether $\overline{D}_{\mathrm{A}} \in \mathrm{Pic}_C^0$.

Before signing a message that involves computation of $[a_{\mathrm{A}}]\overline{E}$ for some $\overline{E}$, A needs to check whether $\overline{E} \in \mathrm{Pic}_C^0$ and if so whether $\overline{E}$ has order ℓ.

23.5.2 Protocols on Pic_C^0

For using elliptic curves, some special protocols were designed that make use of the representation of group elements by points. Applying compression techniques (cf. Sections 13.2.5 and 13.3.7) a point $P = (x_1, y_1)$ can be uniquely stored using its x-coordinate and a bit of y. As for each x_1 there are at most two y_1's such that (x_1, y_1) is on the curve and the points are the negatives of one another, one can also give up the uniqueness and only use x_1 to represent both P and $-P$.

Hence, some reduction in the length of the key and signatures is possible such as is not possible in a generic group. In the following, we detail a signature algorithm to show the differences to the general purpose algorithm presented in Chapter 1. For other protocols similar observations hold.

The elliptic curve digital signature algorithm (ECDSA) [ANSI X9.62] is a signature scheme of ElGamal type as given in Section 1.6.3. We state the generalized version applicable to hyperelliptic curves in the following. For easier reference we call it *HECDSA*. We assume Mumford representation for elements in Pic_C^0 and assume that the finite field elements are ordered such that $0 \leqslant L(\alpha) < q$ is an integer uniquely assigned to $\alpha \in \mathbb{F}_q$ in an invertible way.

In Germany and Korea slightly different versions of the signature scheme are applied. The interested reader is referred to the standards and [HAME+ 2003].

Algorithm 23.14 HECDSA – Signature generation

INPUT: The system parameters $(\mathbb{F}_{p^d}, C, \overline{D}, N, c)$, the private key a_{A}, a hash function h, and a message m.

OUTPUT: The signature (U, s) on m.

1. **repeat**
2. **repeat**
3. $r \in_R [0, \ell - 1]$
4. $\overline{E} \leftarrow [r]\overline{D}$ $\quad \left[\overline{E} = [u_E, v_E] \text{ with } u_E = x^{\nu} + \sum_{i=0}^{\nu-1} u_i \text{ for some } \nu \leqslant g\right]$
5. $U \leftarrow \sum_{i=0}^{\nu-1} L(u_i) q^i \bmod \ell$

6. **until** $U \neq 0$ **and** 1
7. $s \leftarrow \big(r^{-1}\big(h(m) - [a_{\mathrm{A}}]U\big)\big) \bmod \ell$
8. **until** $s \neq 0$
9. **return** (U, s)

Algorithm 23.15 HECDSA – Signature verification

INPUT: The system parameters $(\mathbb{F}_{p^d}, C, \overline{D}, N, c)$, the public key $\overline{D}_{\mathrm{A}}$, a hash function h, the message m, and the possible signature (U, s) on m.
OUTPUT: Acceptance or rejection of signature.

1. **if** U **or** $s \notin [1, \ell - 1]$ **then return** "reject"
2. **else** $w \leftarrow s^{-1} \bmod \ell$
3. $u_1 \leftarrow \big(h(m)w\big) \bmod \ell$ and $u_2 \leftarrow (Uw) \bmod \ell$
4. $\overline{F} \leftarrow [u_1]\overline{D} \oplus [u_2]\overline{D}_{\mathrm{A}}$ $\qquad [\overline{F} = [u_F, v_F]]$
5. **if** $\overline{F} = 0$ **then return** "reject"
6. **else** $U_1 \leftarrow \big(\sum_{i=0}^{\nu-1} L(u_{F,i})q^i\big) \bmod \ell$ $\big[u_F = x^\nu + \sum_{i=0}^{\nu-1} u_{F,i}$ for some $\nu \leqslant g\big]$
7. **if** $U = U_F$ **then return** "accept" **else return** "reject"

The signature scheme works as specified as it is a special instance of Algorithm 1.18. Note, however, that the storage requirements are reduced as the first entry of the signature is not a group element but an integer modulo ℓ.

As only the first part of the representation of $\overline{E}$ is used in the signature, the scheme is *malleable*, i.e., it is possible to obtain a valid signature on a different message by the observation that $\overline{E}$ and $-\overline{E}$ result in the same signature. This property was pointed out in [STPO$^+$ 2002] together with an attack that uses this property to choose the private key in a manner that two *a priori* chosen messages will result in the same signature.

An easy way to avoid this drawback is to apply a compression, (cf. Section 14.2) to $\overline{E}$, which means that U depends uniquely on $\overline{E}$.

There are also encryption and key agreement protocols that take into account the special properties of elliptic curves. Usually it is easy to obtain hyperelliptic curve analogies. We do not include them in this book but refer the reader to the vast literature. A useful overview with many references is given in [HAME$^+$ 2003].

23.6 Summary

In Chapter 1 we have explained which purposes public-key cryptography can be used for as part of systems that provide data security. In Section 1.5 we have defined discrete logarithms in an abstract way and in Section 1.6 one finds protocols that use discrete logarithms as crypto primitives.

The purpose of the present chapter is to help to decide how to *realize* such systems in the most efficient way.

Remark 23.16 In Chapter 1 we have discussed bilinear structures as additional structures of certain DL systems, as well as their use in protocols. Their realization is discussed in the next chapter.

To use DL systems in practice one first has to analyze which grade of security is needed. Then one inspects the offered types of groups and the hardness in the DLP in these groups. It could be that for specific practical reasons one is satisfied with subexponential complexity of DLP and chooses as the group a subgroup of prime order in the multiplicative group of a sufficiently large finite field $\mathbb{F}_q$. The system XTR is an example of an efficient realization of such a system (see Example 1.13). But in most cases one will want to have key sizes as small as possible with an easy scalable security, and so groups for which there are good reasons to believe in the exponential hardness of the DLP are the appropriate choice. In Section 23.2 we have discussed such groups; the results are stated in Sections 23.2.1.a and 23.2.2.c. The short answer is that the groups Pic_C^0 for curves of genus $1, 2, 3$ over prime fields, or over fields of order 2^d with d a prime such that the multiplicative order of 2 modulo ℓ is large, are good candidates. A more subtle answer can be found by following the references in Section 23.2 leading to detailed discussions of the mathematical background and implementational details of attacks.

Roughly spoken, the proposed groups have the same level of security, require the same space for keys, and need the same time for scalar multiplication. But in concrete realizations, in particular the efficiency will depend heavily on the specific implementation and the computational environment. So, the discussion in Section 23.3 becomes relevant. One of the basic decisions is the choice of the ground field $\mathbb{F}_q$ and its arithmetic. Hints for this are to be found in Section 23.3.1 and references cited there. Since the group size ℓ is determined by the security level the choice of the bit size of the ground field is related to the genus g of the curve C by $\lg \ell \sim g \lg q$. So, the choice of q determines g and hence one has to check whether one finds a family of curves of genus g over $\mathbb{F}_q$ that fits into the special situation one has.

The next criterion for the choice of C is the efficiency of the group law in Pic_C^0. This is discussed in Sections 23.3.2 and 23.3.3. In many cases it will be sufficient to take random curves and standard versions of the scalar multiplication. But in special cases "the" best implementation will depend on special properties of the processors used, and adding additional structures like endomorphisms can result in significant accelerations. But then implementations (and choices) have to be done more carefully and the relevant background chapters have to be consulted.

Having gone through all choices one needs a pair $(C, \mathbb{F}_q)$ satisfying the properties one wants to have. At this stage one can either look for standard curves (especially for $g = 1$) or use the results described in Section 23.4. This part is one of the most attractive aspects of cryptography from the mathematical point of view and the algorithms related to point counting are relatively involved. On the other side, for all practical needs they are well documented and can be implemented without knowing the whole theoretical background. So in praxis it will be no problem to find suitable instances $(C, \mathbb{F}_q)$.

Now the last but not the least step has to be done: the embedding of the crypto primitive in the chosen protocol. In Section 23.5 it is explained how the generic protocols from Chapter 1 have to be transformed into protocols using Pic_C^0 as crypto primitive. This was done for signatures in Section 23.5.2, and it is not difficult to treat other protocols in a similar way.