Russian State University
for the Humanities

Faculty of Information Security

University of
Duisburg-Essen

Institute for Experimental Mathematics

Andrey Bogdanov

# Efficient and Cryptographically Secure Addition in the Ideal Class Groups of Hyperelliptic Curves

**Diploma Thesis**

Scientific advisor:
Prof. Dr. Vladimir Anashin

Scientific supervisor:
Prof. Dr. Dr. h.c. Gerhard Frey

Essen 2005

# Acknowledgements

First of all I would like to express my deep appreciation to my supervisor Professor Gerhard Frey for his guidance, useful hints, valuable time and constant support throughout my study.

Thanks to Erwin Heß for his initial encouragement, Professor Vladimir Anashin for good advice and support, to Tanja Lange for information and fruitful discussions, to Roberto Avanzi for kindly providing me with the source code of nuMongo and BinGo, to Professor Sergey Gashkov for acquainting me with the foundations of the arithmetic over hyperelliptic curves, to Ilya Kizhvatov for partially proof-reading the manuscript, to Wolfgang Happle for repairing my laptop and to all other people from the Institute for Experimental Mathematics in Essen. I express my gratitude to my parents and to Oxana.

# Abstract

In this thesis asymmetric cryptographical systems based on hyperelliptic curves are discussed. For genus 2 hyperelliptic curves the subgroup $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of the degree 0 part $\mathrm{Pic}^0(C)$ of the Picard group $\mathrm{Pic}(C)$ offers the exponential security level if the parameters were chosen in a proper way, which is also the case for genus 1 hyperelliptic curves. The latter are elliptic curves with the group of $\mathbb{F}_q$-rational points isomorphic to $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$. The arithmetic in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of a genus 2 hyperelliptic curve is very efficient providing performance comparable to that in the group of rational points on an elliptic curve for the same group size. For elliptic curves one has to use the base finite field of double size in comparison with that for hyperelliptic curves to achieve the same security level. That is, in terms of gate area and propagation properties the hardware implementation of the group law in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ for a hyperelliptic curve is more efficient than for an elliptic curve over $\mathbb{F}_{q_1}$ with $q \approx \sqrt{q_1}$. Moreover, some hyperelliptic curves over binary fields (those with $\deg(h) = 1$ over binary fields of odd extension degree) offer a considerable advantage over elliptic curves in terms of performance maintaining a high degree of diversity. There are also fast methods to (de)compress elements of $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ for such curves.

Thus, hyperelliptic curves are especially attractive for constraint applications. These are often mobile and can be subject to side-channel attacks which are aimed at implementations of cryptographical systems and are based on information leakage through measuring physical parameters such as power consumption and execution time or on failure behaviour induced by malicious physical influence. Here one has to solve a problem which is in a certain sense opposite to that of getting the fastest possible arithmetic. The problem consists in finding ways to harden the implementation by some means to significantly reduce or eliminate the possibility of information leakage. The corresponding countermeasures can be of technical or mathematical nature. Here we treat principally the mathematical ones.

The thesis begins with an introduction chapter (Chapter 1) revising the notion of public key cryptography in abstract groups. The main part is divided into 2 chapters. In the first one (Chapter 2) the fast arithmetic in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of genus 2 hyperelliptic and the foundations of the theory of algebraic curves are considered. Here among other things the author uses the opportunity to correct some inaccuracy in the special case of the explicit formulae to perform the group law in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ and gives an improved algorithm to efficiently (de)compress in $\mathrm{Pic}^0_{\mathbb{F}_{2^d}}(C)$ of special genus 2 hyperelliptic curves over binary fields. The second one (Chapter 3) deals with side-channel attacks on the implementations of cryptosystems based on genus 2 hyperelliptic curves. The corresponding countermeasures, including the Montgomery ladder, are regarded to withstand simple side-channel attacks. Here it is shown that some ideas proposed in the mathematical literature seem to be incomplete or do not lead to efficient Montgomery arithmetic in their original form. Moreover, the author tries to take some other approaches to the Montgomery-like arithmetic in $\mathrm{Pic}^0_{\mathbb{F}_{2^d}}(C)$ of genus 2 hyperelliptic curves and to provide a framework to get explicit scalar multiplication formulae resistant to simple side-channel attacks.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Cryptography and Discrete Logarithm Problem

The public key cryptography is a relatively new direction in the evolution of contemporary cryptography. The idea of the public key cryptography, which is also known as asymmetric cryptography, is simple, but it opens a new dimension of applications which were impossible using traditional symmetric cryptography, e.g. public digital signature techniques. The security of asymmetric cryptographic systems is based on the infeasibility of some problems that seem to be practically unsolvable for the time being. It is worth noticing here that there is no proof of unconditional security of asymmetric encryption systems. Their security is based on the lack of information and ideas regarding some computationally difficult problems. But on the other hand there seem to have been no successful (polynomial) attacks[1] on basic public key systems (provided the parameters have been carefully chosen) in spite of the world mathematical community trying to do this for the last 30 years.

There are two basic classes of infeasible problems that are used to build public key schemes such as RSA, ElGamal or DSA. The first one is the factorization of a sufficiently long positive integer number of some specific structure, and the second one is the discrete logarithm problem (DLP) in some cryptographically suitable groups. In this chapter the notion of a public key encryption system is defined, the DLP in its generic form is described, and the computational complexity of the DLP in some example groups is considered.

[1]Note that within this thesis we are speaking of classical computers only. If one takes into account all known computer models, then it turns out that quantum computers suit for solving these problems much better than classical ones (for details and comparisons see [14]). For quantum computers some polynomial algorithms solving the DLP in the subgroups of the multiplicative groups of finite fields [75] and (recently) in the groups of points of elliptic curves over prime [70] and binary [36] fields have been proposed. Moreover, the DLP in the groups of points of elliptic curves proves to be much easier (in terms of qubits needed) to solve than that in the multiplicative group of a field since the group guaranteeing the same security level on classical computers is much larger in the latter case. Though there are no known quantum algorithms for solving the DLP in the Jacobian of a hyperelliptic curve, it is assumed that such algorithms can be easily obtained if enough effort is made. The only problem is that such computers with sufficiently long quantum registers are extremely difficult to build. Some people among the physical research community proceed from the assumption that such computers will not be ever built. But the classical public key cryptography is really very interesting while there is no quantum computer with appropriately long register. Moreover, even if such computer could be built one day, due to some fundamental physical restrictions they are very likely to have to be manufactured anew for every concrete task. Hence, the classical public key cryptography is actually of interest while the creation of a new quantum computer is expensive enough.

## 1.1 Public key cryptography and discrete logarithm in arbitrary groups

### 1.1.1 Asymmetric cryptosystems

Let $\{E_e : e \in K\}$ be a set of encryption mappings, and let $\{D_d : d \in K\}$ be the corresponding set of decryption mappings, where $K$ is a key space with the strict order $<$ defined on its elements. If $X$ and $Y$ are sets of plain and cipher texts respectively, then the mappings defined by $E_e$ and $D_d$ can be described by:

$$\begin{array}{l} E_e : x \times e \mapsto y, e \in K, x \in X, y \in Y \text{ and} \\ D_d : y \times d \mapsto x, d \in K, x \in X, y \in Y. \end{array} \tag{1.1}$$

Moreover, we assume that $E_e$ and $D_d$ are completely given by the corresponding key parameter $e$ or $d$. Now consider the set of pairs of coupled mappings $C = \{(E_e, D_d)\}$ with the following algorithm to construct it:

1. $C \leftarrow \{E_e | e \in K\} \times \{D_d | d \in K\}$;
2. $(\exists e \in K, \exists d \in K : (\exists x \in X : D_d(E_e(x)) \neq x)) \Rightarrow (C \leftarrow C \backslash \{(E_e, D_d)\})$;
3. $(\exists e \in K, \exists e' \in K, e < e', \exists d \in K : (\forall x \in X : D_d(E_e(x)) = D_d(E_{e'}(x)) = x)) \Longrightarrow$ $\Rightarrow (C \leftarrow C \backslash \{(E_{e'}, D_d)\})$;
4. $(\exists e \in K, \exists d \in K, \exists d' \in K, d < d' : (\forall x \in X : D_d(E_e(x)) = D_{d'}(E_e(x)) = x)) \Longrightarrow$ $\Rightarrow (C \leftarrow C \backslash \{(E_e, D_{d'})\})$.

Now we clear up the meaning of each step:

1. In this step we construct $C$ consisting of all the possible combinations of encryption/decryption mapping pairs. At the following steps the number of pairs in $C$ is reduced by applying some conditions.
2. At this step we exclude from $C$ such mapping pairs which are not conjugate and therefore cannot guarantee the unambiguity of decryption.
3. Here we find and exclude from $C$ such mapping pairs that have equivalent encryption components.
4. Analogously, here we exclude from $C$ such mapping pairs that have equivalent decryption components.

Having done 1)-4), one can be sure that $C$ contains all possible coupled mapping pairs and that there are no equivalent mapping pairs in $C$. This is the definition of the set $C$.

Note that while being reduced throughout steps 1)-4) the set $C$ can lose all pairs with some $E_e$ in the first component or some $D_d$ in the second one. For example, for some $e$ there can be no corresponding decryption mapping. Let $K_e = \{e | \exists d \in K : (E_e, D_d) \in C\}$ be the set of all possible encryption keys in $C$, and let $K_d = \{d | \exists e \in K : (E_e, D_d) \in C\}$ be the set of all possible decryption keys. Then one can give the following

**Definition 1.** *The quintuple $(C, K_e, K_d, X, Y)$ of sets described above is called a public key cryptographic system (an asymmetric cryptographic system), if for each pair of mappings $(E_e, D_d)$ from $C$ the key $e \in K_e$ is public (publicly distributed among users), and the key $d \in K_d$ is private (kept in secret by its owner).*

In order for the defined type of cryptographic systems to be *theoretically secure* the following conditions must be satisfied (hereby the theoretical security of an asymmetric crypto system is defined in part following the idea of Shannon[2] [74] for symmetric ciphers):

- For every fixed value $x \in X$, given an arbitrary public key $e \in K_e$, the corresponding ciphertext $y = E_e(x)$, and the mapping $D_d : (E_e, D_d) \in C$ as a "black box" with $d$ unknown, there is no faster way to find $x$ than the exhaustive search in $X$ or determining $d$.

- For every fixed value $d \in K_d$, given an arbitrary public key $e \in K_d : (E_e, D_d) \in C$, and given the mapping $D_d$ as a "black box" with $d$ unknown, there is no faster way to find $d$ than the exhaustive search in $K_d$.

This security condition concerning the possibility to use $D_d$ as a "black box" is quite powerful. For instance, the adversary is allowed to apply "lunch time attack", that is, the attack with chosen plaintexts which is often very mighty. Moreover, Shannon's theoretical security condition does not require that. So the security condition given here is stronger[3] in a certain sense. But such theoretically secure asymmetric cryptographic systems are difficult to be thought of and to make use of, the same being the case for symmetric ciphers. So in the real-world *practically secure* public keys systems are applied. They are based upon difficult computational problems which realize in trapdoor one-way functions ($E_e$ with the secret value $d$ allowing to invert it). If such functions are used to build the asymmetric crypto system it is possible to get some information about $x$ or $d$ from $e$, $y$ and $D_d$ used as a "black box" but it should be computationally infeasible to derive $x$ or $d$ in a system with appropriately selected parameters. So in practical situations there exist attacks on such systems meaning that they require (often considerably) less operations than the exhaustive search. The real-world attacks can be divided into two fundamentally different classes:

- *Mathematical attacks* work with the abstract (mathematical) representation of the structures under consideration and try to make use of the cryptographic imperfection of these structures to determine secret parameters;

- *Non-mathematical attacks* (first of all, physical ones) which use the weaknesses of the implementations of public key methods. These attacks are also known as side-channel attacks (SCA) and are based on information leakage through different timings of operations, diverse power consumption, electromagnetic emission of devices and so on which depend on some secret information. Moreover, sometimes it turns out to be possible to cause faults on the target devices and to get modified output values that can contain more information about the secret internal values than usual ones.

In this thesis the consideration of some specific attacks on the protocols that use asymmetric cryptographical systems is omitted (a good survey of such techniques can be found in [72]) and attacks on the public key algorithms (asymmetric crypto systems) which can be applied as crypto primitives in these protocols are discussed only.

One of these (practically applied) difficult problems is the discrete logarithm one (DLP) which is treated in the next subsection in greater detail.

[2]Note that the definition of Shannon's theoretical security for symmetric ciphers is based on information theory: One gets no additional information about the plaintext $x$ through knowing the corresponding ciphertext $y$, i.e. $p(x) = p(x/y)$. In our setting we could follow this approach literally, but we prefer to define the security properties in terms of operations and complexity.

[3]Though one is permitted to use a key from a non-finite keyspace, there are cases in which the keyspace (together with plaintext and ciphertext spaces) is finite. In such cases there is always an attack: one can find the plaintext or key value with non-zero possibility. But it is worth putting here that in real-world applications all sets are *always* finite.

### 1.1.2 DLP in arbitrary finite groups and its applications

Let $G$ be an additively written finite cyclic abelian group, $\mathrm{ord}(G) = n$. From now on we assume that the order $n$ of $G$ is known *explicitly*. Let $\alpha$ be a generator of $G$. This means that each element $g \in G$ can be represented as a (scalar) multiple of $\alpha$, $g = k\alpha$, for some $k \in \{0, 1, 2, \ldots, n-1\}$. If $G$ is not cyclic, then its cyclic subgroup generated by an element $g$ of the highest order can be used instead. By Lagrange's theorem one can always assume that $\mathrm{ord}(g)|n$.

In 1975 Diffie and Hellman [23] proposed a cryptographic scheme which uses such groups to construct public key cryptographic systems. Based on their proposal the scheme is redefined for an arbitrary cyclic finite group $G$. Strictly speaking, this scheme is not a public key cryptographic system as defined above. But it allows two users to generate a secret value which can be used as a secret key in a symmetric cryptographic system afterwards. Per se it is a key distribution protocol whose security rests upon DLP. The *generalized Diffie-Hellman scheme* can be represented in the following (additive) way:

Generalized Diffie-Hellman key distribution scheme

One-time setup: $G$ which is the above defined cyclic finite group, its order $n = \mathrm{ord}(G)$, and its generator $\alpha$

Input: $G$, $n$, $\alpha$

A chooses $x_A \in \{1, 2, \ldots, n-1\}$ at random, computes $y_A = x_A\alpha$, and sends $y_A$ to B. $x_A$ is kept till the end of the protocol.

B receives $y_A$ from A, chooses $x_B \in \{1, 2, \ldots, n-1\}$ at random, computes $y_B = x_B\alpha$ and sends $y_B$ to A. Then B computes $k_B = x_B y_A$

A receives $y_B$ and computes $k_A = x_A y_B$.

Output: $k = k_A = k_B$ is the generated common secret key.

The protocol is evidently correct since

$$\begin{array}{l} k_A = x_A y_B = x_A(x_B\alpha), \\ k_B = x_B y_A = x_B(x_A\alpha), \text{ and consequently} \\ k = k_A = k_B. \end{array} \tag{1.2}$$

The security of this scheme is based upon the generalized DLP (GDLP) which is introduced through

**Definition 2.** *The generalized discrete logarithm problem is the following: Given $G$, its order $n$, $\alpha$ and $g \in G$, find the integer $k \in \{1, \ldots, n-1\}$ such that $k \cdot \alpha = g$. $k$ is called the discrete logarithm of $g$ to the base $\alpha$.*

It is known that the difficulty of GDLP is independent of the used generator of $G$.

There exist some other asymmetric cryptographic systems and schemes that make use of the fact that DLP is computationally infeasible in some groups. One of the most important applications of public key cryptography generally and DLP-based cryptographical systems specifically is digital signature. Here an additively written generalization of the standard *digital signature algorithm DSA* (*D*igital *S*ignature *A*lgorithm) is given which is the standardized digital signature procedure in the USA. Let $\pi : G \to \mathbb{Z}$ be some deterministic injective integer-valued mapping[4] of the group elements. Actually, since $G$ is finite, one has $|\mathrm{Im}(\pi)| < \infty$. Moreover, let $\mathrm{Im}(\pi) \subset \mathbb{N}$ be a subset of positive integers.

[4] $\pi$ can be usually naturally defined in the real-world applications since implementations make use of internal binary representation of group elements in devices. As a matter of fact it can be a difficult task to find such a representation for an arbitrary group of some nature. But here this problem is not regarded.

Generalized DSA signature scheme

One-time setup: $G$ is the above defined cyclic finite group, its order $n = \mathrm{ord}(G)$ is prime, and its generator is $\alpha$. $A$ generates $d \in \{1, 2, \ldots, n-1\}$ randomly and securely and computes $\beta = d\alpha$. $d$ is the private key of $A$, the quadruple ($\beta$,$G$,$n$,$\alpha$) is the public key of $A$.

Signature generation

Input: $e \in \{1, \ldots, n-1\}$ — message to sign, $G$, $n$, $\alpha$

1. Choose $k \in \{1, \ldots, n-1\}$ which is the ephemeral key securely at random.
2. Compute $k\alpha$.
3. Compute $r = \pi(k\alpha) \bmod n$. If $r = 0$ then go to 1.
4. Compute $k^{-1} \bmod n$.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then go to 1.

Output: The pair $(r, s) \in \{1, \ldots, n-1\}^2$ forms the signature for $e$.

Signature verification

Input: The public key $(\beta, G, n, \alpha)$, the message $e \in \{1, \ldots, n-1\}$, $(r, s) \in \mathbb{Z}^2$ — signature to verify.

1. If $r \notin \{1, \ldots, n-1\}$ or $s \notin \{1, \ldots, n-1\}$ then reject $(r, s)$.
2. Compute $w = s^{-1} \bmod n$.
3. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
4. Compute $\gamma = u_1\alpha + u_2\beta$. If $\gamma = 0$, then reject $(r, s)$.
5. Compute $v = \pi(\gamma) \bmod n$.
6. Accept $(r, s)$ iff $v = r$.

Now we show that $v = r$ if all the parameters are correct:

$$\begin{aligned} &s = k^{-1}(e + dr), \\ &u_1 = es^{-1}, u_2 = rs^{-1}, \\ &\gamma = ew\alpha + rw\beta = es^{-1}\alpha + rs^{-1}\beta = s^{-1}(e\alpha + r\beta) = \\ &= k\tfrac{1}{e+dr}(e\alpha + r\beta) = k\tfrac{1}{e+dr}(e\alpha + rd\alpha) = k\alpha. \end{aligned} \tag{1.3}$$

So we have that $v = \pi(\gamma) = \pi(k\alpha) = r$.

### 1.1.3 Generic mathematical attacks

As above it is assumed that the order $n$ of the group $G$ is given explicitly, the order $k$ of $\beta = k\alpha$, its Hamming weight are unknown to the adversary, and that $k$ is chosen at random from the set $\{1, 2, \ldots, n-1\}$.

There are several algorithms that can be used to compute the discrete logarithm of $\beta$ in arbitrary groups. There exists the lower asymptotical complexity bound for DLP-algorithms in generic groups due to Nechaev [68] and Shoup [76]. Let $G$ be a cyclic finite group of order $n$. Then there are no algorithms that solve the DLP in $G$ using asymptotically fewer than $O(\sqrt{n})$ operations, if $n$ is prime. Moreover, several algorithms with this property are well known, e.g., the baby-step giant-step algorithm and Pollard's rho algorithm.

The baby-step giant-step algorithm is a time-memory trade-off of exhaustive search and requires storage for $O(\sqrt{n})$ group elements, $O(\sqrt{n})$ group operations to construct and $O(\sqrt{n}\log n)$ comparisons to sort the table. But it can give advantage over Pollard's rho algorithm if there

are subgroups in $G$ and their structure is known, since the group elements are referred to by their orders as $(i, \alpha^i)$ where $\alpha$ is the generator of $G$. The method can be parallelized on $t$ computers with a $\sqrt{t}$-fold speed-up only, that is, it requires $O(\sqrt{n})/\sqrt{t}$ clocks on $t$ computers.

Pollard's rho algorithm requires a negligible amount of storage, but is based on the usage of a pseudorandom function which enumerates the group elements independent of their belonging to its subgroups. If $n$ is prime, then the preferable algorithm is Pollard's rho one, since it requires almost no storage and the baby-step giant-step algorithm offers no advantage in the case of prime group order. The method can be well parallelized in practice. It requires $O(\sqrt{n})/t$ clocks when run on $t$ computers and therefore enables a $t$-fold speed-up.

Pohlig-Hellman algorithm is to be applied if $n$ is smooth, i.e. $n = p_1^{s_1} \dots p_r^{s_r}$ has only small prime factors $p_i$, $i = 1, \dots, r$. Then this algorithm requires $O(\sum_{i=1}^{r} s_i(\log n + \sqrt{p_i}))$ group operations only and a negligible amount of storage. The algorithm is based on the Chinese remainder theorem. First the problem in the subgroups of prime orders $p_i$ is solved in correspondingly exponential time and then the solutions are transferred to the corresponding groups of orders $p_i^{s_i}$ using the $p$-adic extension (Hensel's lifting lemma). At last the Chinese remainder theorem is applied to transfer the solutions to the whole group $G$.

The complexity of DLP in $G$ is strongly dependent on construction of $G$. In order for the generic group to offer the *best possible practical security* for the DLP it is advisable to use the group $G$ of a prime order $n$ so as to exclude any non-trivial subgroups of $G$ and to make Pohling-Hellman algorithm not applicable. In this case the best algorithm is Pollard's rho one, Pohling-Hellman algorithm and the baby-step giant-step algorithm having no advantage because of no non-trivial subgroups of $G$. The group $G$, $\operatorname{ord} G = m = c \cdot n$, with some small co-factor $c$ and a large prime main factor $n$ in its order $m$ can be also applied. Here one has to check that the applied generator $\alpha$ of $G$ is of order $n$ at the least. The expected complexity of Pollard's rho algorithm is $s \cdot e^{\frac{1}{2}n}$ with $s$ varying for different implementation approaches. Some of them can be found in [79].

## 1.2 Cryptographically suitable groups

In the following it is discussed what groups can be regarded as cryptographically suitable and consider their properties. Moreover, some example groups are given and it is shown how similar sets of simple mathematical objects can be treated in different ways according to the group nature which results in different cryptographic properties.

A cryptographically suitable group should meet the following requirements:

1. Group elements should have a *compact representation*. It is preferable that each group element can be given through a unique bit string of length about $\log_2 n$, where $n$ is the order of the group $G$ under consideration. This property can be expressed in terms of embedding $G$ into a group with numeration as in [28]. Moreover, the group order $n$ is demanded to be computable and explicitly known.

2. Given a representation for the elements, an *efficient algorithm* should be known to perform the group law. "Efficient" can mean either "polynomial" or (more strongly and concretely formulated) "requiring no more than $t$ computer operations", the set of permitted operations being fixed. Usually, such operations are addition modulo $2^l$ with carry, substraction modulo $2^l$ with carry, multiplication of 2 $l$-bit integer numbers with a $(2 \cdot l)$-bit result, division of a $(2 \cdot l)$-bit integer number through an $l$-bit integer number, bitwise logical binary operations such as XOR, AND, OR, NOT on $l$-bit numbers. On the most contemporary chips $l = 8$, $l = 16$, $l = 32$ or $l = 64$. Some parallelization techniques (such as SIMD-extensions, multi-core processors or parallel cryptographical co-processors) can be taken into account too. But there are platforms which provide

some specialized hardware computational environments with $l$ from 100 to 2000 for integer operations and some additional operations such as addition or multiplication in the binary field $\mathrm{GF}(2^l)$ for $l$ from 100 to 2000.

3. The DLP in the group $G$ should be *computationally infeasible*. The difficulty of the DLP should preferably be *exponential* in $\log n$ as in a generic "black-box" group. But it can be acceptable to use a group realization with the DLP being subexponential, though it is not always reasonable, since to provide the same security one has to use a more expensive representation of group elements and the group law (see Fig. 1.1).

To consider the following examples one needs a precise definition of subexponentiality[5].

**Definition 3.** *An algorithm is called subexponential in* $\log n$ *if it requires*

$$L_n[\gamma, c] = exp\{(c + o(1))(\log n)^{\gamma}(\log\log n)^{1-\gamma}\},$$

*where* $o(1) \to 0$ *if* $n \to \infty$, $c$, $\gamma$ *are constant, and* $\gamma < 1$.

Note that the impact of the group size $n$ is distributed among $\log n$ and $\log\log n$ in the exponent through the value of $\gamma$ which reflects the degree of subexponentiality. Therefore $\gamma$ is the most important parameter of any subexponential algorithm.

The additive group of the quotient ring $\mathbb{Z}/p\mathbb{Z}$, where $p$ is prime, is one of the most simple examples of group structures. The operation is merely addition of two $\lceil \log p \rceil$-bit integers modulo $p$. So the first two requirements for cryptographically suitable groups are met. But the security property is improper: There is an effective algorithm (Euclidean algorithm) with average complexity $\frac{12\ln 2}{\pi^2}\ln p = O(\log p)$ steps each step requiring one integer division which makes this group cryptographically unsuitable. As a matter of fact, there is another algorithm to compute the DLP in this additive group. Given the generator $\alpha \in \{2, \ldots, p-1\}$ and an element $\beta = k\alpha \in \{2, ..., p-1\}$ it suffices to find the multiplicative inverse of $\alpha$ which can be done by exponentiation $\alpha^{p-1} \bmod p$ using any version of the binary exponentiation algorithm with complexity $O(\log p)$ multiplications and squarings modulo $p$. Each multiplication requires $O(\log p)$ additions modulo $p$ which results in $O((\log n)^2)$ group operations (additions) to solve the DLP. Or multiplicatively written: The DLP requires $O(\log p)$ multiplications modulo $p$, in other words it is linear.

The multiplicative group $\mathrm{GF}(p)^*$ of the prime field $\mathrm{GF}(p)$. This is an example of groups which have the subexponential complexity of DLP. Asymptotically the best known algorithm for the computation of DLP in $\mathrm{GF}(p)$ is the number field sieve algorithm (NFS) due to Pollard (see [52] for a thorough history of NFS). It works with complexity $L_p[\frac{1}{3}; c]$. Today the fastest known version of NFS for $\mathrm{GF}(p)^*$ requires the constant $c$ equal to $(92 + 26\sqrt{13})^{\frac{1}{3}}$ [56], [55]. Though the complexity is not polynomial, it is considerably lower as in the case of generic groups. This non-observance of the 3rd preferable requirement for cryptographically suitable groups results in the necessity to use the group of a higher order to achieve the same level of security. The interdependence between $\log_2 n$ for groups with exponential and those with subexponential DLP is represented in Fig. 1.1.

For the time being the infeasible number of operations on computers is thought[6] to be about $2^{80}$. This is approximately the complexity of the exhaustive search task as applied

[5] Exponentiality in $\log n$ can be expressed as $O(L_n[1, c]) = O(exp\{(c + o(1))\log n\}) = O(n)$.

[6] Currently the fastest known computer is BlueGene/L DD2 [82] and provides a peak performance of 70.72 TFlops $\simeq 2^{46}$ Flops. A task with complexity of $2^{80}$ operations could be solved in approximately 545 years of computations using the computer. Although a higher level of performance could be achieved using several such computers or linking a number of computer networks, this machine can serve as a good illustrative example of the computational power available today. The time point to begin the computations plays an important role, since it is reasonable to wait until more powerful resources are available and only then start the attack. By optimally choosing the start point in the latter case, the total time needed for successfully determining the secret parameters significantly decreases.

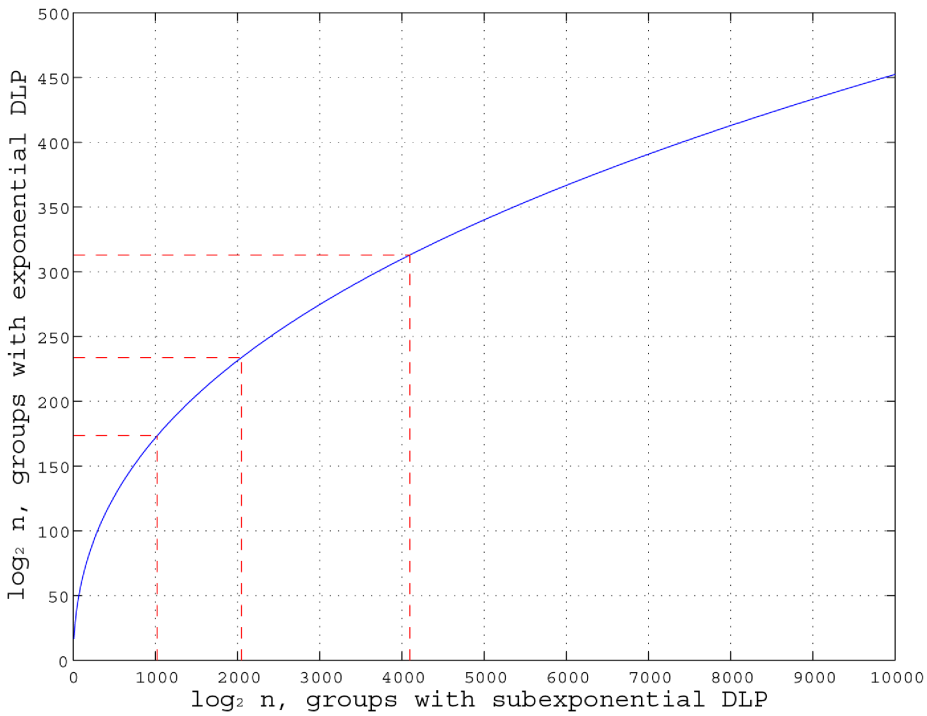


Figure 1.1: Equivalent group orders achieving the same security level for groups with exponential and subexponential DLP. $n = \mathrm{ord}(G)$.

to otherwise secure symmetric ciphers with 80-bit secret keys. To achieve the same security for a cryptographic system based on the exponential DLP one has to use a group of order of about $2^{160}$. For the same level of security in a group with the subexponential DLP (such as $\mathrm{GF}(p)^*$) one has to use the group of order of about $2^{1000}$ which leads to a much longer representation. This gives rise to the search for concrete realizations of groups with the exponential DLP. Such groups turned out to exist and to have very good representation and group law complexity properties. They are tightly connected with the Picard groups of some algebraic curves and will be considered in the next chapter.

# Chapter 2

# Fast arithmetic over Hyperelliptic Curves

## 2.1 Curves and general arithmetic in $\mathbb{J}_C(\mathbb{F}_q)$

### 2.1.1 Elementary algebraic geometry and algebraic curves

In this subsection some basic notions from arithmetic geometry and commutative algebra are given. The consideration results in the definition of the ideal class group and divisor class group of a smooth algebraic curve. The overview is based on [8], [77], [2], [43], [54].

Let $K$ be a finite (and therefore perfect since every extension is separable) field with $q = p^d$ elements, $d \in \mathbb{Z}$, $d \geq 1$ and $d < \infty$. Let $\overline{K}$ be the algebraic closure of $K$ (the field over which every polynomial from the polynomial ring $K[x]$ has no irreducible factors). Let $G_{\overline{K}/K}$ denote the Galois group of the extension $\overline{K}/K$ (by definition $G_{\overline{K}/K} = Aut\ \overline{K}/K$, i.e., it is the set of all automorphisms of $\overline{K}$ which fix $K$).

Affine $n$-space over $K$ is the set of $n$-tuples [77]

$$\mathbb{A}^n = \mathbb{A}^n(\overline{K}) = \{P = (x_1, \ldots, x_n) : \ x_i \in \overline{K}\}. \tag{2.1}$$

The set of $K$-rational points in $\mathbb{A}^n$ is the set

$$\mathbb{A}^n(K) = \{P = (x_1, \ldots, x_n) \in \mathbb{A}^n : \ \forall x_i \in K\}. \tag{2.2}$$

Let $I \subset \overline{K}[X_1, \ldots, X_n]$ be an ideal of a polynomial ring in $n$ variables over $\overline{K}$. Hilbert's basis theorem states that if $K$ is a Noetherian ring, so is the polynomial ring of one variable $K[X]$. Since the field $\overline{K}$ is a Noetherian ring, by multiple application of Hilbert's basis theorem we have that the polynomial ring $\overline{K}[X_1, \ldots, X_n]$ is Noetherian and therefore every ideal $I \subset \overline{K}[X_1, \ldots, X_n]$ is finitely generated. Recall that $I$ is finitely generated if there is a finite set of polynomials $\varphi_1, \ldots, \varphi_l \in I$ such that for every element $\psi \in I$ there exists the corresponding set of coefficients $\gamma_1, \ldots, \gamma_l \in \overline{K}[X_1, \ldots, X_n]$ and $\psi$ is a linear combination of $\varphi_1, \ldots, \varphi_l$ with these coefficients:

$$\psi = \sum_{i=1}^{l} \gamma_i \varphi_i. \tag{2.3}$$

Assuming

$$V_I = \{P \in \mathbb{A}^n : f(P) = 0 \ \forall f \in I\} \subset \mathbb{A}^n \tag{2.4}$$

one can give the following

**Definition 4** (affine algebraic set $V$, ideal of $V$, rational points of $V$)**.** *$V_I$ forms an affine algebraic set $V$,i.e., $V = V_I$ for some ideal $I$. The ideal*

$$I(V) = \{f \in \overline{K}[X_1, \ldots, X_n] : f(P) = 0,\ \forall P \in V\} \tag{2.5}$$

*is called the ideal of the algebraic set $V$. The algebraic set $V$ is said to be defined over $K$ if the ideal $I(V)$ can be (finitely) generated by polynomials[1] from $K[X_1, \ldots, X_n]$. This is denoted by $V/K$. If $V$ is defined over $K$, the set of its $K$-rational points is the set $V(K) = V \cap \mathbb{A}^n(K)$.*

Let $V$ be an algebraic set. The ideal $I(V/K)$ is defined as:

$$I(V/K) = \{f \in K[X_1, \ldots, X_n] :\ f(P) = 0\ \forall P \in V\} = I(V) \cap K[X_1, \ldots, X_n]. \tag{2.6}$$

Note that $V$ is defined over $K$ if and only if $I(V) = I(V/K)\overline{K}[X_1, \ldots, X_n]$.

An ideal $J \subset R$ of some commutative ring $R$ with unity is called *prime* if for every $a, b \in R$ with the property $a \cdot b \in J$ the following holds: $a \in J$ or $b \in J$.

**Definition 5** (affine variety, affine coordinate ring, function field)**.** *An affine algebraic set $V$ is called an affine variety if $I(V)$ is a prime ideal in $\overline{K}[X_1, \ldots, X_n]$. If $V$ is defined over $K$, that is, one deals with $V/K$, The quotient ring $K[V] = K[X_1, \ldots, X_n]/I(V/K)$ is called the affine coordinate ring of the variety $V/K$. The field $K(V)$ of fractions of the ring $K[V]$ is called the affine function field of the variety $V/K$. For a generic affine variety $V$ one has similar definitions: $\overline{K}[V] = \overline{K}[X_1, \ldots, X_n]/I(V)$ is the affine coordinate ring of $V$, and its field of fractions $\overline{K}(V)$ is called the affine function field of $V$.*

Note that $K(V)$ is an integral domain since $I(V/K)$ is prime.

**Definition 6** (dimension of $V$, $\dim V$)**.** *The transcendence degree of the function field $\overline{K}(V)$ over $\overline{K}$ defines the dimension of the affine variety $V$.*

For example the dimension $\dim V$ of hypersurfaces (varieties that are given by a single polynomial in $n$ variables) defined by $f \in K[X_1, \ldots, X_n]$ is $n - 1$.

**Definition 7** (smoothness/non-singularity)**.** *An affine variety $V$ is said to be smooth (or non-singular) at $P = (x_1, \ldots, x_n) \in V$ if the $l \times n$ matrix of partial derivatives evaluated at $P \in V$*

$$M = (\frac{\partial f_i}{\partial X_j}(P)),\ 1 \leq i \leq l, 1 \leq i \leq n \tag{2.7}$$

*has $rank(M) = n - dim(V)$. If $V$ is non-singular at every point, then $V$ is called smooth (or non-singular).*

For a hypersurface $S$ one has that $S$ is singular at $P \in S$ if and only if

$$\frac{\partial f}{\partial X_1}(P) = 0, \ldots, \frac{\partial f}{\partial X_n}(P) = 0 \tag{2.8}$$

simultaneously. In all the other cases the hypersurface $S$ is non-singular at $P \in S$.

Now a slightly generalized notion of projective varieties is defined and all introduced objects are represented in a new form. The $n$-dimensional projective space $\mathbb{P}^n = \mathbb{P}^n(\overline{K})$ over $\overline{K}$ is the set of all lines through the origin in $\mathbb{A}^{n+1}$. In other words the point $[x_0, \ldots, x_n]$ in $\mathbb{P}^n$ is defined through an equivalence class of points in $\mathbb{A}^{n+1}$ in the following way:

$$\mathbb{P}^n(\overline{K}) = \{(x_0, \ldots, x_n) : x_i \in \overline{K} \text{ and at least one } x_i \text{ is nonzero}\}/\sim, \tag{2.9}$$

[1]Note that it is not necessary for every set of the generating polynomials to be defined over $K$. But it is really required that there should be a set of generating polynomials over $K$.

where $\sim$ is the equivalence relation

$$(x_0, \ldots, x_n) \sim (x'_0, \ldots, x'_n) \Longleftrightarrow \exists \lambda \in \overline{K} \ \forall i : x_i = \lambda x'_i. \tag{2.10}$$

The equivalence classes $\{(\lambda x_0, \ldots, \lambda x_n)\} = [x_0, \ldots, x_n]$ are called *projective points*. The coordinates used are called *homogeneous coordinates*. The set of K-rational points of $\mathbb{P}^n(\overline{K})$ is the following subset of $\mathbb{P}^n(\overline{K})$:

$$\mathbb{P}^n(K) = \{[x_0, \ldots, x_n] \in \mathbb{P}^n(\overline{K}) : \exists \lambda \in \overline{K} \ \forall i : \ \lambda x_i \in K\} \subset \mathbb{P}^n(\overline{K}). \tag{2.11}$$

Note that this does not automatically mean that for all $P = [x_0, \ldots, x_n] \in \mathbb{P}^n(K)$ all $x_i \in K$. But this does mean that through choosing some $i$ with $x_i \neq 0$ one has $\forall j : x_j/x_i \in K$. By dividing all coordinates through $x_i$ one gets 1 at the place $i$ and elements from $K$ at the other places. So it can assumed by definition that

$$\mathbb{P}^n(K) = \{[x_0, \ldots, x_n] : \forall i \ x_i \in K\}. \tag{2.12}$$

Moreover, one can prove that $\mathbb{P}^n(K) = \{P \in \mathbb{P}^n(\overline{K}) : \sigma P = P, \forall \sigma \in G_{\overline{K}/K}\}$, there $\sigma P = \sigma[x_0, \ldots, x_n] = [\sigma x_0, \ldots, \sigma x_n]$.

To give the corresponding definition of a projective variety one has to introduce the notion of a homogenous ideal. Recall that a polynomial $f \in \overline{K}[X_0, \ldots, X_n]$ is called *homogenous* if $f(\lambda X_0, \ldots, \lambda X_n) = \lambda^d f(X_0, \ldots, X_n)$, $\forall \lambda \in \overline{K}$. An ideal $I \subset \overline{K}[X_0, \ldots, X_n]$ is *homogenous* if it is generated by homogenous polynomials. Putting

$$V_I = \{P \in \mathbb{P}^n(\overline{K}) : f(P) = 0 \ \forall \text{ homogenous } f \in I\} \tag{2.13}$$

one can proceed with

**Definition 8** (projective algebraic set $V$, homogenous ideal of $V$, rational points of $V$)**.** *The set $V_I$ is a projective algebraic set. The homogenous ideal of $V$ is*

$$I(V) =< f \in \overline{K}[X_0, \ldots, X_n] : f \text{ is homogenous and } f(P) = 0 \ \forall P \in V > . \tag{2.14}$$

*The projective algebraic set $V$ is said to be defined over $K$ if $I(V)$ can be generated by homogenous polynomials from $K[X_0, \ldots, X_n]$. This is denoted by $V/K$. If $V$ is defined over $K$, the set of its $K$-rational points is the set $V(K) = V \cap \mathbb{P}^n(K)$.*

**Definition 9** (projective algebraic variety)**.** *A projective algebraic set $V$ is called a projective variety if its homogenous ideal $I(V)$ is prime in $\overline{K}[X_0, \ldots, X_n]$.*

Note that to some extent one can (pairwise) identify affine and projective spaces, sets and varieties since $\mathbb{P}^n$ contains many copies of $\mathbb{A}^n$. In order to do this one needs the corresponding mapping between $\mathbb{A}^n$ and $\mathbb{P}^n$. Here some rigorous conclusions are omitted and the consideration is limited to mapping between $\mathbb{A}^n$ and $\mathbb{P}^n$ without mentioning the homogenization and dehomogenization of polynomials. The mapping $\phi_i$ from $\mathbb{A}^n$ to $\mathbb{P}^n$ is given by

$$\phi_i : (y_1, \ldots, y_n) \mapsto [y_1, \ldots, y_{i-1}, 1, y_i, \ldots, y_n]. \tag{2.15}$$

The mapping from $\mathbb{P}^n$ to $\mathbb{A}^n$ is given by

$$\phi_i^{-1} : [x_0, \ldots, x_n] \mapsto (x_0/x_i, \ldots, x_{i-1}/x_i, x_{i+1}/x_i, \ldots, x_n/x_i). \tag{2.16}$$

Strictly speaking one should define the reverse mapping $\phi_i^{-1}$ from a subset $U_i$ of points in $\mathbb{P}^n$ which have the nonzero coordinate $x_i$ in order for the mapping to be well-defined. Therefore one can identify $\mathbb{A}^n$ with the set $U_i \subset \mathbb{P}^n$ since $\phi_i$ is bijective. If $V$ is a projective algebraic

set then $V \cap \mathbb{A}^n = \phi_i^{-1}(V \cap U_i)$ is an affine algebraic set for some appropriate $i$. This resulting affine algebraic set[2] $V \cap \mathbb{A}^n$ is called the *affine part* of $V$. For a projective variety (projective) points in $V - (V \cap \mathbb{A}^n)$ are called *points at infinity* on $V$.

Now the properties of projective varieties can be defined through those of affine varieties.

**Definition 10** (dimension of $V$, coordinate ring of $V$, field of fractions of $V$)**.** *Let $V/K$ be a projective variety over $K$. The dimension of $V/K$ is the dimension of its affine part $V/K \cap \mathbb{A}^n$. The coordinate ring $K[V]$ of $V/K$ and the function field $K(V)$ of $V/K$ are those of its affine part $V/K \cap \mathbb{A}^n$.*

**Definition 11** (non-singularity/smoothness)**.** *The projective variety $V$ is smooth (or non-singular) at $P = (x_0, \ldots, x_n) \in V$ if the $V \cap \mathbb{A}^n$ is smooth at $\phi_i^{-1}(P) = (y_1, \ldots, y_n) \in \mathbb{A}^n$ for some appropriate $i$.*

From now on one can study affine algebraic varieties identifying the corresponding projective ones with their affine parts and recalling the projective nature of the varieties under consideration when necessary. An *algebraic curve* is an algebraic variety of dimension 1. A plane algebraic curve over $K$ is a hypersurface given by a polynomial in two variables from $K[X_1, X_2]$. An algebraic curve is smooth (or non-singular) if the corresponding algebraic variety is smooth. In the sequel we concentrate on *non-singular* algebraic varieties $V$ generally and on *non-singular* algebraic curves $C$ specifically.

**Definition 12** (local ring of $C$ at $P$)**.** *Let $C$ be a non-singular algebraic curve and $P \in C$. The local ring $\overline{K}[C]_P$ of $C$ at $P$ is*

$$\overline{K}[C]_P = \{\frac{f}{g} \in \overline{K}(C) : f, g \in \overline{K}[C], g(P) \neq 0\}. \tag{2.17}$$

For every rational function $F$ from $\overline{K}[C]_P$ the evaluation of $F(P)$ at $P$ is well-defined. The functions from $\overline{K}[C]_P$ are said to be *regular* (or *defined*) at $P$. Note that the local ring of $C$ at $P$ coincides with the localization of $\overline{K}[C]$ at its ideal $M_P = \{f \in \overline{K}[C] : f(P) = 0\}$.

For the fraction field $R$ of the local ring $\overline{K}[C]_P$ there exists a mapping $v : R^* \to \mathbb{Z}$ of its multiplicative group to the ring of integers such that:

- $v(xy) = v(x) + v(y)$, i.e., $v$ is the group homomorphism;
- $v(x + y) \geq min(v(x), v(y))$.

Then $v$ is the *discrete valuation* of $R$ with $R = \overline{K}(C)$. Moreover, there exists such $v$ that the following proves to hold:

$$\overline{K}[C]_P = \{x \in R^* : v(x) \geq 0\} \cup \{0\}. \tag{2.18}$$

That is the normalization ring of $v$. Therefore the integrity domain $\overline{K}[C]_P$ is *discrete valuation ring*. By definition $v(0) = \infty$. For smooth algebraic curves under consideration $\overline{K}[C]$ is a Noetherian domain of dimension 1. Together with the property that $\overline{K}[C]_P$ is a discrete valuation ring this leads to the definition of a *Dedekind domain*. That is, the Dedekind property of $\overline{K}[C]$ captures the smoothness and dimension one property of $C$. Now one can define the mapping $v$ over its normalization ring more explicitly:

$$\begin{aligned} &v_P : \overline{K}[C]_P \to \{0, 1, \ldots, \} \cup \{\infty\}, \\ &v_P : f \mapsto \max\{d \in \mathbb{Z} : f \in M_P^d\}. \end{aligned} \tag{2.19}$$

[2]The set $V \cap \mathbb{A}^n$ can be empty. That means that one has to choose the affine space $\mathbb{A}^n$ properly.

The domain of $v_P$ can be extended to the field $R = \overline{K}(C)$ of fractions of $\overline{K}[C]_P$ using $v_P(\frac{f}{g}) = v_P(f) - v_P(g)$ with:

$$v_P : \overline{K}(C) \to \mathbb{Z} \cup \{\infty\}. \tag{2.20}$$

Therefore $v_P$ is a discrete valuation of the fraction field $\overline{K}(C)$ of the curve $C$.

**Definition 13** (fractional ideal, invertible ideal)**.** *For a non-singular curve $C$ $K[C]$-submodule $M \subset K(C)$ is a fractional $K[C]$-ideal, if $\exists f \in K(C)^* : fM$ is an ideal of $K[C]$. The $K[C]$-submodule $M \subset K(C)$ is an invertible ideal, if there exists a $K[C]$-submodule $N \subset K(C)$: $NM = K[C]$.*

The fractional ideal invertibility property of $K[C]$ is exactly captured by the Dedekind property of $K[C]$. That is, an integral domain is a Dedekind domain if and only if every fractional ideal of it is invertible. Hence, the non-zero fractional $K[C]$-ideals form a group with respect to the operation of ideal multiplication. This group is called the *ideal group* of the ring $K[C]$ and is denoted by $I$. $I$ is a free abelian group generated by non-zero prime ideals of $K[C]$. An element $f \in K(C)$ defines a fractional $K[C]$-ideal $(f)$ which is called a *principle fractional ideal*. The set of all principle fractional ideals forms a (multiplicative) subgroup $P$ in $I$, $P \triangleleft I$. The quotient group $I/P$ is called the *ideal class group* which is denoted by $H_{K(C)}$. $H_{K(C)}$ for hyperelliptic curves will be the main object of the current study.

### 2.1.2 Basic arithmetic of hyperelliptic curves

In this subsection the definition of a (imaginary quadratic) hyperelliptic curve is given. The isomorphisms, divisor class group $\mathrm{Pic}^0_K(C)$, explicit representation of elements in the ideal class group $H_{K(C)}$ and the basic algorithm to perform the group law in $H_{K(C)}$ are discussed.

**Definition 14.** *A non-singular algebraic curve $C/K$ is called a hyperelliptic curve if the function field $K(C)$ is a separable extension of degree 2 of the rational function field $K(x)$ for some function $x$. Let $\omega$ denote the nontrivial automorphism of this extension. It induces an involution $\omega_*$ on $C$. Points on $C$ with $P = \omega_*(P)$ are called Weierstraß points. It is assumed[3] here that there is one Weierstraß $K$-rational point on $C/K$ denoted by $P_\infty$ — a point at infinity: $P_\infty \notin \mathbb{A}^2(\overline{K})$, but $P_\infty \in \mathbb{P}^2(K)$. Under this assumption a (imaginary quadratic) hyperelliptic curve $C$ of genus $g$ over $K$ ($g \geq 1$) is a smooth algebraic curve defined by the following equation in two variables:*

$$C : \; y^2 + h(x)y = f(x) \in K[x, y], \tag{2.21}$$

*where $h(x) \in K[x]$ with $deg(h) \leq g$, $f(x) \in K[x]$ is a monic polynomial with $deg(f) = 2g + 1$.*

The smoothness property means that there is no affine point $P = (x, y) \in C$ at which

$$2y + h(x) = 0 \text{ and } h'(x)y - f'(x) = 0 \tag{2.22}$$

simultaneously. Note that the homogenized equation of $C$ has a singularity at $P_\infty$.

Often it is important to find out what equivalent forms of curve equations are applicable. The maps:

$$\begin{aligned} &y \mapsto u^5 y' + ax'^2 + bx' + c \text{ and} \\ &x \mapsto u^2 x' + e, \\ &\text{where } a, b, c, e, u \in K, u \neq 0, \end{aligned} \tag{2.23}$$

[3] By doing this one reduces the class of hyperelliptic curves treated. But for cryptographic purposes in the context of the current consideration it suffices to consider this class of hyperelliptic curves (called imaginary quadratic hyperelliptic curves) only.

act on points of $C$ and are invertible. These changes of variables are the only ones leaving invariant the shape of the defining equation and are the only *admissible isomorphisms* of hyperelliptic curves.

**Definition 15** (divisor group $\mathrm{Div}(C)$, divisor $D$, support of $D$, degree of $D$)**.** *The divisor group $Div(C)$ of a (imaginary quadratic) hyperelliptic curve $C$ is the free abelian group generated by the points of $C$. An element $D \in Div(C)$ is called a divisor. From the definition of a free group it follows that:*

$$D = \sum_{P \in C} n_P P, \tag{2.24}$$

*where $n_P \in \mathbb{Z}$ and only a finite number of the $n_P$ are non-zero. For $D$ the set $supp(D)$ of such points at which $n_P$ are non-zero is called the support of $D$:*

$$supp(D) = \{P \in C : n_P \neq 0\}. \tag{2.25}$$

*The degree $deg(D)$ of $D$ is the sum of $n_P$ for all the points from the support:*

$$deg(D) = \sum_{P \in supp(D)} n_P. \tag{2.26}$$

The addition of two divisors $D_1$ and $D_2$ is defined by:

$$D_1 + D_2 = \sum_{P \in C} n_P P + \sum_{P \in C} m_P P = \sum_{P \in C} (n_P + m_P) P. \tag{2.27}$$

Hence, the divisors of degree 0 form the subgroup $\mathrm{Div}^0(C)$:

$$\mathrm{Div}^0(C) = \{D \in \mathrm{Div}(C) : \deg(D) = 0\} \lhd \mathrm{Div}(C). \tag{2.28}$$

Note that automorphisms from $G_{\overline{K}/K}$ act on $\mathrm{Div}(C)$ by acting on the coordinates of the points in $\mathrm{supp}(D)$. The divisor $D$ is said to be defined over $K$ if $\forall \sigma \in G_{\overline{K}/K}$ it holds that:

$$\sigma D = \sigma(\sum_{P \in C} n_P P) = \sum_{P \in C} n_P \sigma P = D. \tag{2.29}$$

Divisors defined over $K$ form a subgroup $\mathrm{Div}_K(C) \lhd \mathrm{Div}(C)$. The discrete valuation $v_P$ of $\overline{K}(C)$ allows one to define the following mapping:

$$\begin{array}{l} \mathrm{div} : \overline{K}(C)^* \to \mathrm{Div}(C), \\ \mathrm{div} : f \mapsto \sum_{P \in C} v_P(f) P. \end{array} \tag{2.30}$$

This map possesses one crucial property which is very useful in applications:

$$\mathrm{div}(\sigma f) = \sigma \, \mathrm{div}(f), \tag{2.31}$$

and, consequently, if $f \in K(C)$, then $\mathrm{div}(f) \in \mathrm{Div}_K(C)$. That is, the elements of $\mathrm{div}(K(C)^*) \lhd \mathrm{Div}_K(C)$ can be represented by rational functions with coefficients in $K = \mathrm{GF}(q)$. This property is retained by the following constructions which are further specifications of $\mathrm{Div}_K(C)$ and $\mathrm{Div}^0_K(C)$.

**Definition 16** (principle divisor, Picard group)**.** *$D \in Div(C)$ is called a principle divisor if $\exists\ f \in \overline{K}(C)^*$ with $div(f) = D$. Principle divisors form a subgroup $\mathbb{P} \lhd Div(C)$ of the divisor group of $C$. The quotient group $Pic(C) = Div(C)/\mathbb{P}$ is called the Picard group of $C$. $Pic_K(C)$ is the subgroup[4] of $Pic(C)$ which is fixed by all $\sigma \in G_{\overline{K}/K}$.*

[4]Note that generally speaking $\mathrm{Pic}_K(C) \neq \mathrm{Div}_K(C)/\mathbb{P}_K$.

For each $f \in \overline{K}(C)$ $\deg(\mathrm{div}(f)) = 0$. This means that $\mathbb{P} \lhd \mathrm{Div}^0(C)$. Thus, one can build the quotient group of $\mathrm{Div}^0(C)$ by $\mathbb{P}$ which leads to the following very important

**Definition 17** (degree 0 part $\mathrm{Pic}^0(C)$ of $\mathrm{Pic}(C)$, $\mathrm{Pic}^0_K(C)$, equivalent divisors)**.** *The degree 0 part $Pic^0(C)$ of $Pic(C)$ is given by: $Div^0(C)/\mathbb{P}$. $Pic^0_K(C)$ is the subgroup of $Pic^0(C)$ fixed by all $\sigma \in G_{\overline{K}/K}$. For $D_1, D_2 \in Div^0(C)$ we write $D_1 \sim D_2$ if $D_1 - D_2 \in \mathbb{P}$.*

From now on $\mathrm{Pic}^0(C)$, and specifically $\mathrm{Pic}^0_K(C)$, of a hyperelliptic curve $C$ in the imaginary quadratic form is considered and some of its properties (including representation) are stated more explicitly. It is possible to show that for $C$ there is an abelian variety (a projective algebraic group) $\mathbb{J}_C$ which is called the Jacobian variety (Jacobian) of $C$ (here the exact construction of $\mathbb{J}_C$ is omitted, see [8]). The group $\mathbb{J}_C(K)$ of $K$-rational points of $\mathbb{J}_C$ is isomorphic to the divisor class group $\mathrm{Pic}^0_K(C)$: $\mathbb{J}_C(K) \cong \mathrm{Pic}^0_K(C)$. In the following we will use the notions of $\mathbb{J}_C(K)$ and $\mathrm{Pic}^0_K(C)$ as synonyms meaning that we can endow $\mathrm{Pic}^0(C)$ with the structure of an abelian variety and consider its $K$-rational points. (For the rigorous definition of the connection between $\mathrm{Pic}^0(C)$ and $\mathbb{J}_C$ see [54] or [8].) Moreover, for hyperelliptic curves there is another representation of $\mathrm{Pic}^0_K(C)$. This is the ideal class group $H_{K(C)}$. Therefore one has 3 equivalent group representations (for details see [8]):

$$\mathbb{J}_C(K) \cong \mathrm{Pic}^0_K(C) \cong H_{K(C)}. \tag{2.32}$$

The representation through the ideal class group is the most constructive one and gives rise to the Mumford representation.

**Definition 18** (semi-reduced divisor, reduced divisor)**.** *A semi-reduced divisor is a divisor of the form*

$$D = \sum m_i P_i - (\sum m_i) P_\infty, \tag{2.33}$$

*where:*

- *each $m_i \geq 0$,*
- *each $P_i \neq P_\infty$,*
- *$P_i \in supp(D)$ and $P_i$ ordinary $\Rightarrow \widetilde{P}_i \notin supp(D)$,*
- *$P_i \in supp(D)$ and $P_i$ special $\Rightarrow m_i = 1$.*

*A semi-reduced divisor $D$ with $deg(D) \leq g$ is called a reduced divisor.*

Each class of principle divisors in $\mathrm{Pic}^0_K(C)$ has a *unique* reduced divisor. Therefore each element $c \in \mathrm{Pic}^0_K(C)$ can be identified with the corresponding reduced divisor. See [8] for the detailed consideration or [39] for elementary proofs.

Recall that $K = \mathbb{F}_q$ and $\mathbb{J}_C(\mathbb{F}_q) \cong \mathrm{Pic}^0_{\mathbb{F}_q}(C) \cong H_{\mathbb{F}_q(C)}$ for an imaginary quadratic hyperelliptic curve $C$. Then the following theorem (see [67] or [8]) states some very important circumstances that are useful in order to do fast arithmetic in $\mathbb{J}_C(\mathbb{F}_q)$.

**Theorem 1** (Mumford representation)**.** *Each nontrivial ideal class from the ideal class group $H_{\mathbb{F}_q(C)}$ can be represented via the corresponding unique ideal $J \subset \mathbb{F}_q[C]$ generated*[5] *by such polynomials $a(x)$ and $b(x) - y$, $a, b \in \mathbb{F}_q[x]$ that:*

1. *$a$ is monic,*
2. *$deg\, b < deg\, a \leq g$,*

[5] That is, $J =< a(x), y - b(x) >= \mathbb{F}_q[x]a(x) + \mathbb{F}_q[x](b(x) - y)$.

3. $a|b^2 + bh - f = N(b(x) - y)$.

*Moreover, let* $D = (\sum_{i=1}^{r} m_i P_i) - (\sum_{i=1}^{r} m_i)P_\infty$ *with* $P_i \neq \widetilde{P_j}$, $P_i \neq P_\infty$, *for* $i \neq j$ *and with* $(\sum_{i=1}^{r} m_i) = m = deg\, D \leq g$ *be a reduced divisor. Let* $P_i = (x_i, y_i)$, $x_i, y_i \in \overline{\mathbb{F}}_q$, $i \in \{1, \ldots, r\}$ *denote the point from its support. Then:*

1. $a(x) = \sum_{i=1}^{r}(x - x_i)^{m_i}$ *(a is defined by the x-coordinates of the points from supp(D) and their multiplicities),*

2. $\forall P_i \neq P_\infty \in supp(D) :\ b(x_i) = y_i$ *(b interpolates the points in supp(D)).*

The transfer from a rational function $f \in K(C)^*$ to the corresponding (principle) divisor is performed through the discrete valuation $v_P$ of $\overline{K}(C)^*$. Here $v_P$ is defined explicitly for a hyperelliptic curve $C$. At first $v_P$ is explicitly defined for $\overline{K}[C]^*$. Assume $f = a(x) - b(x)y \in \overline{K}[C]^*$ and $P \in C$. Then:

$P = (x_0, y_0) \neq P_\infty$: Let $r$ be the highest power of $(x - x_0)$ which divides both $a(x)$ and $b(x)$, and write $f(x, y) = (x - x_0)^r(a_0(x) - b_0(x)y)$.

- If $a_0(x_0) - b_0(x_0)y_0 \neq 0$, then let $s = 0$.
- Otherwise, let $s$ be the highest power of $(x - x_0)$ which divides $N(a_0(x) - b_0(x)y) = a_0(x)^2 + a_0(x)b_0(x)h(x) - b_0(x)^2 f(x)$. If $P$ is ordinary, then define $v_P(f) = r + s$. If $P$ is special, then define $v_P = 2r + s$.

$P = P_\infty$: Define $v_P(f) = -\max[2\deg(a), 2g + 1 + 2\deg(b)]$.

The discrete valuation of $\frac{f}{g} \in \overline{K}(C)^*$ is defined by $v_P(\frac{f}{g}) = v_P(f) - v_P(g)$.

The transfer from the representation $[a, b]$ of an element in the ideal class group $H_{K(C)}$ to the reduced divisor in the corresponding class $c$ in $\mathrm{Pic}_K^0(C)$ is performed using

**Definition 19** (the greatest common divisor of two divisors)**.** *Let* $D_1 = \sum_{P \in C} m_P P - (\sum_{P \in C} m_P)P_\infty$ *and* $D_2 = \sum_{P \in C} n_P P - (\sum_{P \in C} n_P)P_\infty$ *be two divisors. The greatest common divisor of* $D_1$ *and* $D_2$ *is defined to be*

$$gcd(D_1, D_2) = \sum_{P \in C} min(m_P, n_P)P - (\sum_{P \in C} min(m_P, n_P))P_\infty. \tag{2.34}$$

Then the transfer can be expressed as $[a, b] = \gcd(\mathrm{div}(a(x)), \mathrm{div}(b(x) - y))$.

Below some results about the cardinality of the Jacobian $\mathbb{J}_C(\mathbb{F}_q)$ over a hyperelliptic curve are formulated. To do this one needs the following

**Definition 20** (zeta function)**.** *Let* $C$ *be a hyperelliptic curve defined over* $\mathbb{F}_q$, *and let* $M_r = \#C(\mathbb{F}_{q^r})$, $r \geq 1$, *denote the number of* $\mathbb{F}_{q^r}$*-rational points on the curve* $C$*. The zeta function of* $C$ *is the power series:*

$$Z_C(t) = \exp\left(\sum_{r \geq 1} M_r \frac{t^r}{r}\right).$$

Then some important statements hold. They are put together in the following

**Theorem 2** (properties of the zeta function)**.** *Let* $C$ *be a hyperelliptic curve of genus* $g$ *defined over* $\mathbb{F}_q$, *and let* $Z_C(t)$ *be the zeta-function of* $C$*.*

1. $Z_C(t) \in \mathbb{Z}(t)$*. More precisely,*

$$Z_C(t) = \frac{P(t)}{(1 - t)(1 - qt)},$$

*where* $P(t)$ *is a polynomial of degree* $2g$ *with integer coefficients and has the form:*

$$P(t) = 1 + a_1 t + \cdots + a_{g-1}t^{g-1} + a_g t^g + q a_{g-1} t^{g+1} + q a_{g-2} t^{g+2} + \cdots + q^{g-1} a_1 t^{2g-1} + q^g t^{2g}$$

2. *P(t) factors as:*

$$P(t) = \prod_{i=1}^{g} (1 - \alpha_i t)(1 - \overline{\alpha}_i t),$$

*where each* $\alpha_i \in \mathbb{C}$ *of absolute value* $\sqrt{q}$*, and* $\overline{\alpha}_i$ *denotes the complex conjugate of* $\alpha_i$*.*

3. *The following equation holds:*

$$\#\mathbb{J}_C(\mathbb{F}_{q^l}) = \prod_{i=1}^{g} |1 - \alpha_i^l|^2,$$

*where* $|\ \cdot\ |$ *denotes the usual complex absolute value.*

One of the direct applications of this theorem is the estimation of the bounds on the order of the group of $\mathbb{F}_q$-rational points on the Jacobian of a hyperelliptic curve $C$ which is formulated as

**Corollary 1.** *Let* $C$ *be a hyperelliptic curve of genus* $g$ *defined over* $\mathbb{F}_q$*. Then*

$$(q^{l/2} - 1)^{2g} \leq \#\mathbb{J}_C(\mathbb{F}_{q^l}) \leq (q^{l/2} + 1)^{2g}.$$

*Hence,* $\#\mathbb{J}_C(\mathbb{F}_{q^l}) \approx q^{lg}$*. In particular,*

- *For* $l = 1$ *one has*

$$(q^{1/2} - 1)^{2g} \leq \#\mathbb{J}_C(\mathbb{F}_q) \leq (q^{1/2} + 1)^{2g},$$

*or* $\#\mathbb{J}_C(\mathbb{F}_q) \approx q^g$*.*

- *For* $l = 1$ *and* $g = 1$ *one has an elliptic curve (a hyperelliptic curve of genus 1) and get the theorem of Hasse for the bound on the trace of Frobenius* $t$ *at* $q$*: Under the assumption that*

$$\#\mathbb{J}_C(\mathbb{F}_q) = \#\mathbb{E}_C(\mathbb{F}_q) = q + 1 - t$$

*one gets*

$$|t| \leq 2\sqrt{q}.$$

A straightforward approach to the group operation in $\mathbb{J}_C(\mathbb{F}_q)$ is to use Cantor's algorithm which adds reduced divisors and consists of 2 subalgorithms that perform the combination of two reduced divisors and the reduction of the obtained semi-reduced divisor respectively. This algorithm works with reduced divisors of hyperelliptic curves of every genus. Being based on the Gauß algorithm for composing bilinear quadratic forms, it was proposed by Cantor in [16] and generalized by Koblitz in [38]. Throughout the algorithm the notations $\mathrm{div}(a, b) \in \mathrm{Pic}^0_{\mathbb{F}_q}(C)$ and $[a, b] \in H_{\mathbb{F}_q}$ are used as synonyms meaning that there exists group isomorphism between the ideal class group $H_{\mathbb{F}_q(C)}$ and divisor class group $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$.

Algorithm 1 (composition)

Input: two reduced divisors $D_1 = \mathrm{div}(a_1, b_1)$, $D_2 = \mathrm{div}(a_2, b_2) \in \mathrm{Pic}^0_{\mathbb{F}_q}(C)$ in the Mumford representation

1. $d_1 = \gcd(a_1, a_2) = e_1 a_1 + e_2 a_2$, $d_1, e_1, e_2 \in \mathbb{F}_q[x]$
2. $d = \gcd(d_1, b_1 + b_2 + h) = c_1 d_1 + c_2(b_1 + b_2 + h)$, $d, c_1, c_2 \in \mathbb{F}_q[x]$
3. $s_1 = c_1 e_1, s_2 = c_1 e_2, s_3 = c_2$, so that $d = s_1 a_1 + s_2 a_2 + s_3(b_1 + b_2 + h)$, $s_1, s_2, s_3 \in \mathbb{F}_q[x]$

4. $a = \frac{a_1 a_2}{d^2}$, $b = \frac{s_1 a_1 b_2 + s_2 a_2 b_1 + s_3(b_1 b_2 + f)}{d} \bmod a$, $a, b \in \mathbb{F}_q[x]$

Output: $D = \mathrm{div}(a, b) \sim D_1 + D_2$, semi-reduced divisor.

Algorithm 2 (reduction)

Input: $D = \mathrm{div}(a, b)$, semi-reduced divisors

1. $a' = \frac{f - bh - b^2}{a}$, $b' = (-h - b) \bmod a'$, $a', b' \in \mathbb{F}_q[x]$
2. If $\deg_x a' > g$ then $a = a'$, $b = b'$ and go to step 1.
3. Make $a'$ monic: $a' = c^{-1} a'$, where $c \in \mathbb{F}_q$ is the leading coefficient of $a'$

Output: $D' = \mathrm{div}(a', b') \sim D_1 + D_2$, the (unique) reduced divisor in Mumford representation.

For elementary (but rigorous) proofs of the correctness of these algorithms see [39]. Among other things note that the division at Step 1 in Algorithm 2 is exact by construction.

The group law in the degree zero part of the Picard group is illustrated by the example of a genus 2 curve over the reals $\mathbb{R}$. One is going to add two reduced divisors $D_1 = P_1 + P_2 - 2P_\infty$ and $D_2 = Q_1 + Q_2 - 2P_\infty$ over $C$ (see[6] Fig. 2.1). The four points $P_1, P_2$ and $Q_1, Q_2$ (i.e. the composition of the degree 2 divisors $D_1 = P_1 + P_2 - 2P_\infty$ and $D_2 = Q_1 + Q_2 - 2P_\infty$) define a cubic function (since every $m > 1$ points give rise to a polynomial of degree $m - 1$ by interpolation) which has at most 6 points of intersection with $C$ (of degree 5). The other 2 points (which are uniquely defined by the four points of $D_1$ and $D_2$) of intersection are denoted by $-R_1$ and $-R_2$. Their inflection with respect to the $x$-axis (the reduction procedure) leads to the sum $D$ of $D_1$ and $D_2$: $D = R_1 + R_2 - 2P_\infty \sim D_1 + D_2$.

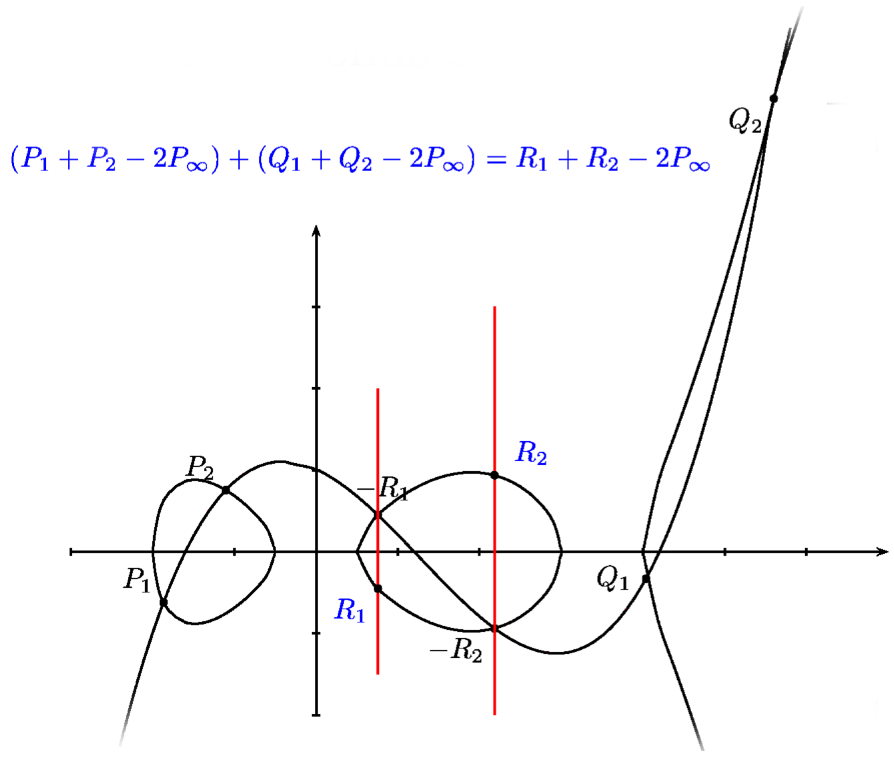


Figure 2.1: Addition in $\mathbb{J}_C(\mathbb{R})$. Genus 2 hyperelliptic curve over the reals.

[6] Courtesy of Tanja Lange and Nicolas Thériault.

One of the research goals pursued here is to perform the group operation explicitly for genus 2 curves over finite fields using the coefficients of the polynomials of the curve equation and divisor representations. This selection of the genus is motivated by security requirements. For further explanations see Section 2.2.

## 2.2 Special mathematical attacks in $\mathbb{J}_C(\mathbb{F}_q)$

In this section the DLP in the Jacobians of hyperelliptic curves is discussed. Assume $|\mathbb{J}_C(\mathbb{F}_q)| = k$ is prime. Then the generic methods to solve the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ give the highest bound of complexity which is equal to $O(\sqrt{k})$. The goal is to describe such hyperelliptic curves for which there are no known methods which bring down the complexity of the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ with respect to $O(\sqrt{k})$. It turns out that if one wants to have an exponential complexity of the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ for some hyperelliptic curve the consideration should be limited to genera 1,2 or 3. Even for $g = 3$ one has to take some countermeasures. Moreover, some other precautions should be taken due to pairings. The motivation of these statements can be found below. In the course of this section it is assumed that $D_1 \in \mathbb{J}_C(\mathbb{F}_q)$ generates $\mathbb{J}_C(\mathbb{F}_q)$. Let $D_2 \in \mathbb{J}_C(\mathbb{F}_q)$ and $D_1 = lD_2$. The adversary wants to find $l$ from $D_2$, $D_1$ and the group structure.

### 2.2.1 Index calculus attack for $\mathbb{J}_C(\mathbb{F}_q)$

The index calculus is the most powerful attack on hyperelliptic crypto systems. The idea behind it rests upon the notion of smoothness with respect to a set of small irreducible elements. A framework for the construction of the index calculus attacks in different class groups can be found in [26]. In case of $\mathbb{J}_C(\mathbb{F}_q) \cong \mathrm{Pic}^0_{\mathbb{F}_q}(C)$ for a hyperelliptic curve $C$ we deal with classes which can be represented by reduced divisors. A reduced divisor $D \in \mathbb{J}_C(\mathbb{F}_q)$ with Mumford representation $D = \mathrm{div}(a, b)$, $a, b \in \mathbb{F}_q[x]$ is called *prime* if the first polynomial $a \in \mathbb{F}_q[x]$ is irreducible over $\mathbb{F}_q$. A divisor $D \in \mathbb{J}_C(\mathbb{F}_q)$ is called *t-smooth* if it can be decomposed so that:

$$D = \mathrm{div}(a, b) = \sum_{i=1}^{l} e_i \,\mathrm{div}(a_i, b_i), \tag{2.35}$$

where $e_i \in \mathbb{Z}$ and $\max\{\deg(a_i)\} \leq t$, $i = 1, \ldots, l$. the polynomials $a, b, a_i, b_i$ and numbers $e_i$ are related with each other in the following way:

$$\begin{array}{l} a = \sum_{i=1}^{l} a_i^{e_i}, \\ b_i = a \mod a_i. \end{array} \tag{2.36}$$

It is clear that to get $a_i$ (and therefore $b_i$ and $e_i$) one has to factor $a$ over $\mathbb{F}_q$.

There are two basic approaches to implement the index calculus for hyperelliptic curves:

1. $|\mathbb{J}_C(\mathbb{F}_q)|$ *is unknown:* First, the *factor base* $S = \{P_1, \ldots, P_n\}$ consisting of prime divisors $P_i = \mathrm{div}(a_i, b_i)$ with $\deg(a_i) \leq t$ for some threshold $t$ is build. Second, $m > n$ $t$-smooth divisors are searched for, each yielding a relation in the form $\sum_j e_j P_j \sim 0$. If the prime divisors in $S$ generate $\mathbb{J}_C(\mathbb{F}_q)$, then the following mapping exists:

$$\begin{array}{l} \phi : (Z^n)^T \to \mathbb{J}_C(\mathbb{F}_q), \\ \phi : (e_1, \ldots, e_n)^T \mapsto \sum_j e_j P_j. \end{array} \tag{2.37}$$

   Each relation yields an element $\vec{e_i}^T = (e_{i1}, \ldots, e_{in}) \in \mathrm{Ker}(\phi)$. If the set of $m$ relations forms a complete generating system of $\mathrm{Ker}(\phi)$, then $\mathbb{J}_C(\mathbb{F}_q) \cong \mathbb{Z}/d_1\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/d_n\mathbb{Z}$.

Moreover, for all $i$'s the group order $d_i$ and generator $X_i$ of $\mathbb{Z}/d_i\mathbb{Z}$ can be found from the matrix $A = (\vec{e_1} \dots \vec{e_m})$ using linear algebra. Then we should find representations of $D_1$ and $D_2$ in $\mathbb{Z}/d_1\mathbb{Z} \oplus \dots \oplus Z/d_n\mathbb{Z}$. If $D_1$ and $D_2$ are generated by $S$, then:

$$\begin{array}{l} D_1 = \sum_{i=1}^{n} \alpha_i P_i, D_2 = \sum_{i=1}^{n} \beta_i P_i, \text{ and} \\ D_1 = \sum_{i=1}^{n} \alpha'_i X_i, D_2 = \sum_{i=1}^{n} \beta'_i X_i \end{array} \tag{2.38}$$

for some integer values $\alpha_i, \alpha'_i, \beta_i, \beta'_i$. At last, $l \in \mathbb{Z}$ can be found by using the Chinese remaider theorem from the following congruence system:

$$\alpha'_i \equiv l\beta'_i \bmod d_i, \ i = 1, \dots, n. \tag{2.39}$$

2. $|\mathbb{J}_C(\mathbb{F}_q)|$ *is known:* First, the factor base $S = \{P_1, \dots, P_n\}$ is constructed. $S$ consists of prime divisors $P_i = \mathrm{div}(a_i, b_i)$ with $\deg(a_i) \leq t$. Second, try to find $t$-smooth divisors of the form:

$$\alpha D_1 + \beta D_2 \sim R_i = \sum e_{ij} P_j. \tag{2.40}$$

over $S$. As soon as (n+1) different relations are found, search for a set of values $\gamma_i$, $i = 1, \dots, n+1$ with $\sum_{i=1}^{n+1} \gamma_i R_i = 0$ by solving the following system of equations:

$$\sum_{i=1}^{n+1} \gamma_i (e_{i1}, \dots, e_{in}) = (0, \dots, 0) \bmod |\mathbb{J}_C(\mathbb{F}_q)|. \tag{2.41}$$

This implies that $\sum_{i=1}^{n+1} \gamma_i(\alpha_i D_1 + \beta_i D_2) = 0$, and:

$$l = -\frac{\sum_{i=1}^{n+1} \gamma_i \alpha_i}{\sum_{i=1}^{n+1} \gamma_i \beta_i} \bmod |\mathbb{J}_C(\mathbb{F}_q)|. \tag{2.42}$$

The first application of this attack for hyperelliptic curves was specified in [1] (so-called ADH algorithm) and uses the first method. The ADH algorithm can be shown to run in expected time $O(L_{q^{2g+1}}[c])$ for some positive real constant $c < 2.313$ (the prove of the bound on $c$ makes use of some unproved assumptions). But this method implies no cardinality of $\mathbb{J}_C(\mathbb{F}_q)$ known. However, for large genera this turns out to be efficient.

In cryptography $|\mathbb{J}_C(\mathbb{F}_q)|$ is, however, almost always known. Since the DLP in the Jacobians for large genus curves is not exponential, it was interesting what complexity the DLP in the Jacobians for small genus curves provides. Gaudry [29] was the first to take the second approach (with known $|\mathbb{J}_C(\mathbb{F}_q)|$) and gave a very efficient algorithm. It uses 1-smooth divisors in the factor base and a special random walk at the second stage. Gaudry showed that his algorithm runs in expected time $O(g^3 q^2 \log^2 q + g^2 g! q \log^2 q)$ for some fixed genus $g$ and the underlying finite field with $q$ elements. The generic DLP algorithms have complexity of $O(q^{g/2})$. That is, the Gaudry's algorithm solves the DLP faster than the generic ones for $g > 4$. Thériault [80] has improved Gaudry's algorithm by refusing to use all 1-smooth divisors in the factor base and restricting himself to a fraction of 1-smooth divisors. This augments the time necessary to build the system of relations, but speeds up the solution of the resulting equation system. Thériault's improvement provides a lower complexity of $O(g^5 q^{2-\frac{2}{g+1}+\epsilon})$. This algorithm is asymptotically faster than Gaudry's one if $q > (g-1)!$. Thériault gives a modification of his algorithm through using the large prime strategy[7]. In this case the runtime is $O(g^5 q^{2-\frac{4}{2g+1}+\epsilon})$. This modified algorithm is asymptotically faster than Gaudry's one if $q > \frac{(g-1)!}{g}$. Thériault's algorithm solves the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ asymptotically faster than generic ones for $g \geq 3$.

[7] That is, he looks for relations generated by the factor base and one additional 'large prime' divisor.

Thus, the genus is a very important security parameter if one wants to apply hyperelliptic curves as the base for cryptographical protocols. For the time being (with group sizes which are currently treated as cryptographically suitable) we can state that the usage of the schemes based on hyperelliptic curves of genus 1 and 2 offers the best possible security (with respect to the DLP algorithms in a generic group). For *elliptic curves* ($g = 1$) there could be some special attacks, but we are aware of no generic algorithms which could solve the DLP in $\mathbb{J}_E(\mathbb{F}_q)$ for some elliptic curve $E$ faster than those for arbitrary groups. The security of the DLP in the Jacobians over genus 3 hyperelliptic curves are somewhat lower - $O(q^{\frac{4}{3}+\epsilon})$ instead of $O(q^{\frac{3}{2}})$ in case the Jacobian were a 'black box' group.

### 2.2.2 Transfer attacks for $\mathbb{J}_C(\mathbb{F}_q)$

Instead of solving the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ one can try to find an appropriate transfer of the problem to some other structures in which it is easier to compute discrete logarithms. Here we mention pairings and Weil descent. These attacks are efficiently applied to some special curves only, but they show that some curves over which the index calculus gives no advantage may nevertheless lead to vulnerabilities. This circumstance excludes the possibility to use certain subclasses of curves for the construction of secure cryptographical systems.

Let $k$ be the order of the group $\mathbb{J}_C(\mathbb{F}_q)$ with $q = p^d$. If $k|q$ then one can build $\mathbb{J}_C(\mathbb{F}_q)$ in polynomial time into the vector space of dimension $g$ over $\mathbb{F}_q$ (i.e. it is possible to efficiently transfer $\mathbb{J}_C(\mathbb{F}_q)$ into a subgroup of this space ). It this group the DLP can be solved using $O((2g-1)\log^c(q)$, where $c$ is a small constant.

If $(q, k) = 1$, then there is another possibility to use pairings destructively. Namely it is known that due to Tate-Lichtenbaum pairing (which is not rigourously defined here) it is possible to efficiently transfer the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ to (a subgroup of) the multiplicative group of the extension field $F_{q^c}$ for some positive integer $c$ under the assumption that $k|(q^c - 1)$. More precisely we have the following

**Theorem 3.** *If $(k, q) = 1$ and $c \in \mathbb{N}$ is minimal with $k|(q^c - 1)$, then there is a map:*

$$T_k : \mathbb{J}_C(\mathbb{F}_q)[k] \times \mathbb{J}_C(\mathbb{F}_{q^c}) \to \mathbb{F}^*_{q^c}/(\mathbb{F}^*_{q^c})^k \tag{2.43}$$

*with the non-degenerate image. For a random element $D \in \mathbb{J}_C(F_{q^c})$ the mapping*

$$\begin{aligned} &T_{k,D} : \mathbb{J}_C(\mathbb{F}_q)[k] \to \mathbb{F}^*_{q^c}[k], \\ &T_{k,D} : P \mapsto (T_k(P, D))^{\frac{q^c-1}{k}} \end{aligned} \tag{2.44}$$

*is an injective homomorphism of groups which is computable using $O(c \log q)$ operations.*

**Corollary 2.** *The DLP in $\mathbb{J}_C(\mathbb{F}_q)$ can be solved with complexity $L_{q^c}[1/2, \sqrt{2}]$.*

That is, the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ is subexponential in $q^c$ with $k|q^c - 1$.

Now assume the extension degree of $\mathbb{F}_q$ over its prime subfield $\mathbb{F}_p$ is $d > 1$ and that there exists $d_0$ with $d_0|d$, $d_0 < d$. Then we can apply Weil descent (which is not defined here). Then the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ proves to be weak if $d/d_0$ is small. If $d$ is prime with $2^t \equiv 1 \bmod d$ for some small $t$, then the DLP in $\mathbb{J}_C(\mathbb{F}_q)$ is weak too. For reasons see [8].

Thus, due to transfer attacks we have to restrict ourselves in the construction of secure cryptographic systems to curves for which the following holds simultaneously:

- $|\mathbb{J}_C(\mathbb{F}_q)|$ prime to $q$,
- $|\mathbb{J}_C(\mathbb{F}_q)|$ prime to $(q^c - 1)$ for every small positive integer $c$,
- either $d = 1$ or $\mathbb{F}_q = \mathrm{GF}(2^d)$ for some prime $d$ with ord(2) large in $\mathbb{F}^*_d$.

## 2.3 Explicit formulae for the group law in $\mathbb{J}_C(\mathbb{F}_q)$ of genus 2 hyperelliptic curves and a point compression technique

Due to the mentioned security restrictions the practical interest is limited to elliptic curves and hyperelliptic curves of genus 2 and 3. Elliptic curves are not considered here, but we refer to [13] or [8] thereupon. The case of $g = 3$ is treated in [8] and is as a rule slower[8] as elliptic or genus 2 curves at the same security level. Here one can find a review of the fast arithmetic of genus 2 hyperelliptic curves over fields of odd and even characteristics. Some minor inaccuracy in the special case of addition formulae for arbitrary characteristic is removed by the author in Subsection 2.3.2. The author's improvement of the point (de)compression technique in $\mathbb{J}_C(\mathbb{F}_q)$ over a subclass of binary curves which can be found in Subsection 2.3.4. Moreover, a thorough comparison between the most efficient cases in terms of *operation count* is provided on the basis of the existing explicit formulae e.g. [8]. For comparisons in terms of *processor time* for a contemporary multi-purpose platform see Section 2.4 where some available implementations due to R. Avanzi [5] and E. Cesena [19] were used.

To do fast arithmetic in the ideal class groups of genus 2 hyperelliptic curves one should concentrate efforts on finding ways to reduce the number of operations in the most frequent cases. For the efficient computation of a scalar product one needs to be able to calculate efficiently the following:

- $D = D_1 + D_2$,
- $D = 2D_1$,

where $D$, $D_1$, $D_2$ are reduced divisors and $D$, $D_1$, $D_2 \in \mathbb{J}_C(\mathbb{F}_q)$. Every reduced divisor in $\mathbb{J}_C(\mathbb{F}_q)$ can be represented through two polynomials (the Mumford representation): $\forall D \in \mathbb{J}_C(\mathbb{F}_q) : \exists[u, v] : D = \gcd(\mathrm{div}(u), \mathrm{div}(y - v))$, where $u, v \in \mathbb{F}_q[x]$, $u$ is monic, $\deg v < \deg u \leq g$, $u \mid v^2 + vh - f = \mathrm{N}(v - y)$. According to the called basic operations in the ideal class group and to the representation of the elements we have two frequent cases:

- *Addition*: $u_1 = x^2 + u_{11}x + u_{10}$, $v_1 = v_{11}x + v_{10}$, and $u_2 = x^2 + u_{21}x + u_{20}$, $v_2 = v_{21}x + v_{20}$ with $\mathrm{res}(u_1, u_2) \neq 0 \Rightarrow \forall P \in \mathrm{supp}(D_1) :$ $P$ and $\widetilde{P}$ are not in $\mathrm{supp}(D_2)$;
- *Doubling*: $u_1 = x^2 + u_{11}x + u_{10}$, $v_1 = v_{11}x + v_{10}$ with $\mathrm{res}(h + 2v_1, a_1) \neq 0 \Rightarrow P_1 \neq \widetilde{P_1}$ and $P_2 \neq \widetilde{P_2}$.

All the formulae below work for these two cases only. The other cases are treated in [47] and are relatively infrequent. If such a case occurs, then one can compute the result by applying Cantor's algorithm directly.

### 2.3.1 Equivalent curve equations

From Cantor's algorithm it is clear that the group law in $\mathbb{J}_C(\mathbb{F}_q)$ and its concrete complexity depend on the curve equation, this being logically the case for explicit formulae too. Assume that $f(x) = x^5 + f_4x^4 + f_3x_3 + f_2x^2 + f_1x + f_0$ and $h(x) = h_2x^2 + h_1x + h_0$, $h_i \in \mathbb{F}_q$, $i = 0, 1, 2$, $f_i \in \mathbb{F}_q$, $i = 0, \ldots, 4$.

Here some equivalent transformations of the curve equation with respect to the admissible changes of variables are given. For different base field characteristics one obtains different equations:

[8] This can be owing to the lack of research in this area.

*char* $\mathbb{F}_q = p$, $p \neq 5$:[44] In this case one can always achieve that $h_2 \in \{0, 1\}$ and $f_4 = 0$ by replacing $x \to x - f_4/5$ and $y \to y$:

$$C: \ y^2 + (h_2x^2 + h_1x + h_0)y = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0. \tag{2.45}$$

*char* $\mathbb{F}_q = p$, *p is odd*, $p \neq 5$: [44] The change of variables $x \to x - f_4/5$, $y \to y - h/2$ leads to $h(x) = 0$, and $f_4 = 0$:

$$C: \ y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0. \tag{2.46}$$

In this case it can be shown that $C$ is non-singular if and only if $f(x)$ has no repeated roots over $\overline{\mathbb{F}}_q$.

*char* $\mathbb{F}_q = 2$: [51] In order for $C$ over a binary field to be smooth $h(x)$ must be non-zero, $h(x) \neq 0$. Moreover, $d$ should be preferably prime and $h(x) \neq const$, since some transfer attacks can be efficient in these cases.

From considerations of group law efficiency one wants to treat different cases of $h(x)$ separately:

- $deg(h) = 2$: $h_2 \neq 0$. The (admissible) change of variables

$$\begin{array}{l} y \to h_2^5 y + f_3h_2x + \frac{f_3(f_3+h_1h_2+f_4h_2^2)+f_2h_2^2}{h_2^3} \\ x \to h_2^2x + f_4 \end{array} \tag{2.47}$$

  and subsequent division by $h_2^{10}$ leads to $h_2 = 1$ and $f_4 = f_3 = f_2 = 0$:

$$C: \ y^2 + (x^2 + h_1x + h_0)y = x^5 + f_1x + f_0. \tag{2.48}$$

  If $\exists b: \ b^2 + bh_1 = h_0$, then there is a change of variables which leads to $h_2 = 1, h_0 = 0$ and $f_3 = f_2 = 0$:

$$C: \ y^2 + (x^2 + h_1x)y = x^5 + f_4x^4 + f_1x + f_0, \tag{2.49}$$

  which allows faster doubling. There are at most $4 \cdot 2^{3d}$ isomoprhism classes of such curves.

- $deg(h) = 1$: Hence, $h_2 = 0$. By replacing:

$$\begin{array}{l} y \to a^5y + a^4\sqrt{f_4 + (h_0/h_1)x^2}, \\ x \to a^2x + (h_0/h_1) \end{array} \tag{2.50}$$

  and dividing through $a^{10}$ one can obtain an isomorphic curve with $f_4 = f_1 = h_0 = 0$:

$$C: \ y^2 + h_1xy = x^5 + f_3x^3 + f_2x^2 + f_0. \tag{2.51}$$

  There are at most $4 \cdot 2^{2d}$ such curves. Note that in case of $d$ being odd (e.g. $d$ is an odd prime) there is no non-trivial cubic roots of unity. That is, an $a$ can be always found with the property $a^3 = h_1$ and, hence, for odd $d$ it can be assumed that $h_1 = 1$ in (2.51):

$$C: \ y^2 + xy = x^5 + f_3x^3 + f_2x^2 + f_0. \tag{2.52}$$

  The number of the isomorphism classes of such curves is at most $2 \cdot 2^{2d}$.

### 2.3.2 Correct addition and doubling in affine coordinates

To get efficient explicit formulae for performing the group law in $\mathbb{J}_C(\mathbb{F}_q)$ some tricks are applied to refine Cantor's algorithm. Among other things Karastuba's trick for multiplication of polynomials, Montgomery's trick for multiple inversions, exact division from the very beginning, Chinese remainder theorem for polynomials are mentioned. For the precise breakdown to the formulae see, for instance, [44]. In this subsection the explicit formulae for the most frequent cases in arbitrary $\operatorname{char}\mathbb{F}_q \neq 5$ (addition in Table 2.1 and doubling in Table 2.2) are given. The author would like to use the opportunity to remove some minor inaccuracy in these (otherwise entirely correct) formulae for addition and doubling which has apparently taken place e.g. [45, 46, 44, 47] merely over an unfortunate misprint. The operation count for the even characteristic in cases of $\deg(h) = 1$ and $\deg(h) = 2$ are given in Table 2.4 for (2.48) , Table 2.3 for (2.49), and Table 2.5 for (2.52). In all the tables the total complexity is provided in operations telling apart inversions (I), multiplications (M) and squarings (S) in the base field $\mathbb{F}_q$.

Table 2.1: Addition: $g = 2$, $\deg(u_1) = \deg(u_2) = 2$ (from [44] with minor modifications by the author)

| |
|---|
| INPUT: $D_1 = [x^2 + u_{11}x + u_{10}, v_{11}x + v_{10}], D_2 = [x^2 + u_{21}x + u_{20}, v_{21}x + v_{20}]$ |
| 1. $z_1 \leftarrow u_{11} - u_{21}, z_2 \leftarrow u_{20} - u_{10}, z_3 \leftarrow u_{11}z_1 + z_2, r \leftarrow z_2z_3 + z_1^2u_{10}$ .<br>2. $inv_1 \leftarrow z_1, inv_0 \leftarrow z_3$.<br>3. $w_0 \leftarrow v_{10} - v_{20}, w_1 \leftarrow v_{11} - v_{21}, w_2 \leftarrow inv_0w_0, w_3 \leftarrow inv_1w_1,$<br>$s_1' \leftarrow (inv_0 + inv_1)(w_0 + w_1) - w_2 - w_3(1 + u_{11}), s_0' \leftarrow w_2 - u_{10}w_3;$<br>if $s_1'$ go to *Special case*.<br>4. $w_1 \leftarrow (rs_1')^{-1}, w_2 \leftarrow rw_1, w_3 \leftarrow (s_1')^2w_1, w_4 \leftarrow rw_2, w_5 \leftarrow w_4^2, s_0'' \leftarrow s_0'w_2$.<br>5. $l_2' \leftarrow u_{21} + s_0'', l_1' \leftarrow u_{21}s_0'' + u_{20}, l_0' \leftarrow u_{20}s_0''$.<br>6. $u_0' \leftarrow (s_0'' - u_{11})(s_0'' - z_1 + h_2w_4) - u_{10},$<br>$u_0' \leftarrow u_0' + l_1' + (h_1 + 2v_{21})w_4 + (2u_{21} + z_1 - f_4)w_5, u_1' \leftarrow 2s_0'' - z_1 + h_2w_4 - w_5$.<br>7. $w_1 \leftarrow l_2' - u_1', w_2 \leftarrow u_1'w_1 + u_0' - l_1', v_1' \leftarrow w_2w_3 - v_{21} - h_1 + h_2u_1',$<br>$w_2 \leftarrow u_0'w_1 - l_0', v_0' \leftarrow w_2w_3 - v_{20} - h_0 + h_2u_0'$. |
| OUTPUT: $D = [x^2 + u_1'x + u_0', v_1'x + v_0'] \sim D_1 + D_2$ |
| *Special case:* Replace lines 4-7 by the following:<br>4. $inv \leftarrow 1/r, s_0 \leftarrow s_0'inv$.<br>5. $u_0' \leftarrow f_4 - u_{21} - u_{11} - s_0^2 - s_0h_2$.<br>6. $w_1 \leftarrow s_0(u_{21} - u_0') + h_1 + v_{21} - h_2u_0', w_2 \leftarrow u_{20}s_0 + v_{20} + h_0, v_0' = u_0'w_1 - w_2$. |
| Total complexity:<br>I+22M+3S (common case)<br>I+12M+2S (special case) |

For the addition of ideal classes with $\deg u_1 = \deg u_2 = 2$ and $\operatorname{res}(u_1, u_2) \neq 0$ for arbitrary

Table 2.2: Doubling: $g = 2$, $\deg(u) = 2$ (from [44] with minor modifications by the author)

| |
|---|
| INPUT: $D = [x^2 + u_1x + u_0, v_1x + v_0]$ |
| 1. $\tilde{v_1} \leftarrow h_1 + 2v_1 - h_2u_1, \tilde{v_0} \leftarrow h_0 + 2v_0 - h_2u_0$.<br>2. $w_0 \leftarrow v_1^2, w_1 \leftarrow \tilde{v_1}^2, w_3 \leftarrow u_1\tilde{v_1}, r \leftarrow u_0w_2 + \tilde{v_0}(\tilde{v_0} - w_3)$.<br>3. $inv'_1 \leftarrow -\tilde{v_1}, inv'_0 \leftarrow v'_0 - w_3$.<br>4. $w_3 \leftarrow f_3 + w_1, w_4 \leftarrow 2u_0, t'_1 \leftarrow 2(w_1 - f_4u_1) + w_3 - w_4 - h_2v_1$,<br>$t'_0 \leftarrow u_1(2w_4 - w_3 + f_4u_1 + h_2v_1) + f_2 - w_0 - 2f_4u_0 - h_1v_1 - h_2v_0$.<br>5. $w_0 \leftarrow t'_0inv'_0, w_1 \leftarrow t'_1inv'_1, s'_1 \leftarrow (inv'_0 + inv'_1)(t'_0 + t'_1) - w_0 - w_1(1 + u_1)$,<br>$s'_0 \leftarrow w_0 - w_1u_0$. If $s'_1 = 0$ go to *Special case*.<br>6. $w_1 \leftarrow 1/(rs'_1), w_2 \leftarrow rw_1, w_3 \leftarrow (s'_1)^2w_1, w_4 \leftarrow rw_2, w_5 \leftarrow w_4^2, s''_0 \leftarrow s'_0w_2$.<br>7. $l'_2 \leftarrow u_1 + s''_0, l'_1 \leftarrow u_1s''_0 + u_0, l'_0 \leftarrow u_0s''_0$.<br>8. $u'_0 \leftarrow (s''_0)^2 + w_4(h_2(s''_0 - u_1) + 2v_1 + h_1) + w_5(2u_1 - f_4), u'_1 \leftarrow 2s''_0 + h_2w_4 - w_5$.<br>9. $w_1 \leftarrow l'_2 - u'_1, w_2 \leftarrow u'_1w_1 + u'_0 - l'_1, v'_1 \leftarrow w_2w_3 - v_1 - h_1 + h_2u'_1$,<br>$w_2 \leftarrow u'_0w_1 - l'_0, v'_0 \leftarrow w_2w_3 - v_0 - h_0 + h_2u'_0$ |
| OUTPUT: $D' = [x^2 + u'_1x + u'_0, v'_1x + v'_0] \sim 2D$ |
| *Special case:* Replace lines 6-9 by the following:<br>6. $w_1 \leftarrow 1/r, s_0 \leftarrow s'_0w_1, w_2 \leftarrow u_0s_0 + v_0 + h_0$,<br>7. $u'_0 \leftarrow f_4 - s_0^2 - s_0h_2 - 2u_1$,<br>8. $w_1 \leftarrow s_0(u_1 - u'_0) - h_2u'_0 + v_1 + h_1, v'_0 \leftarrow u'_0w_1 - w_2$ |
| Total complexity:<br>I+22M+5S (common case)<br>I+13M+3S (special case) |

Table 2.3: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ even, $\deg(h) = 2$, $h_0 = 0$ [8]

| | |
|---|---|
| Addition | (common case) I+22M+3S<br>(special case) I+12M+2S |
| Doubling | (common case) I+17M+5S<br>(special case) I+10M+3S |

Table 2.4: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ even, $\deg(h) = 2$, $h_0 \neq 0$ [8]

| | |
|---|---|
| Addition | (common case) I+22M+3S<br>(special case) I+12M+2S |
| Doubling | (common case) I+21M+6S<br>(special case) I+11M+4S |

Table 2.5: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ even, $d$ odd, $\deg(h) = 1$ [8]

| | |
|---|---|
| Addition | (common case) I+21M+3S<br>(special case) I+12M+2S |
| Doubling | (common case) I+5M+6S<br>(special case) 3M+4S |

genus 2 hyperelliptic curves explicit formulae due to T. Lange can be used (Table 1 in [47] or Table 2.1 here). Though designed to work with affine coordinates (which imply inversions), the formulae are very efficient and can be the optimal ones under some circumstances. Moreover, they serve as basis for all the further refinements. Now some thoughts about the computations in Step 6 in the special case are adduced. This step is the last one in the addition for the special case. Here $v' = -h - (l + v_2) \bmod u'$ is being computed:

$$v' = (-h_2 - s_0)x^2 + (-h_1 - s_0u_{21} - v_{21})x + (-h_0 - s_0u_{20} - v_{20}) \bmod x + u'_0.$$

Then one has $x^2 = (u'_0)^2$, $x = -u'_0$ and, hence,

$$v' = u'_0[s_0(u_{21} - u'_0) + h_1 + v_{21} - u'_0h_2] + [-h_0 - s_0u_{21} - v_{20}].$$

This leads to the final result through $v' = u'_0w_1 - w_2$. Thus, one is supposed to use the following formulae for this step:

- $w_1 = s_0(u_{21} - u'_0) + h_1 + v_{21} - h_2u'_0$,
- $w_2 = u_{20}s_0 + v_{20} + h_0$,
- $v'_0 = u'_0w_1 - w_2$.

Notice that this expression and the original one e.g. [44] differ in two signs and one additional multiplication. Assume $h_2 \in \{0, 1\}$. Then one needs to perform 3 multiplications in compliance with the formulae provided here as opposed to 2 multiplications in the original formulae e.g. [44]. This leads to the following complexity in the special case of addition: I+2S+12M. Though the considered case is relatively infrequent, it may arise in practice and therefore appears in the table for the most frequent case of addition. A similar unfortunate inexactness seems to have taken place in the special case of doubling and is removed in Table 2.2, Step 8 of the special case, but there are no operation count differences in this case under the assumption that $h_2 \in \{0, 1\}$.

Now some computational examples are given to illustrate the correctness of computations in the special case of addition. The hyperelliptic curves used were taken from [39]. At first a genus 2 hyperelliptic curve over $GF(2^5)$ is considered. Let $\mathbb{F}_{2^5} = GF(2^5)$ be defined by $\varphi = u^5 + u^2 + 1$, $\mathbb{F}_{2^5} \cong GF(2)[u]/(u^5 + u^2 + 1)$, $\alpha$ being a root of the primitive (and, hence, irreducible) polynomial $\varphi$. Let C be a hyperelliptic curve defined over $GF(2^5)$ by $y^2 + y(x^2 + x) = x^5 + x^3 + 1$. The formulae for the special case given in Table 2.1 have been implemented in MAGMA and compared with the independent result obtained internally by MAGMA through direct group operations in the Jacobian $\mathbb{J}_C(\mathbb{F}_q)$ of C. In all cases the results agree with each other. One can find an example in Table 2.6. Now we proceed with an example over $\mathbb{F} = GF(7)$. Let the curve be defined by $y^2 + yx = x^5 + 5x^4 + 6x^2 + x + 3$. The result of the corresponding computations can be found in Table 2.7.

Table 2.6: Addition of ideal classes in a special case, char $\mathbb{F}_{2^5} = 2$

| | |
|---|---|
| $D_1$ | $[x^2 + \alpha^5 x + \alpha^{18}, \alpha^{22} x + \alpha^{12}]$ |
| $D_2$ | $[x^2 + \alpha^{15} x + \alpha^8, \alpha^{14} x + \alpha^2]$ |
| $D_1 + D_2$ | $[x + \alpha^5, \alpha^{15}] \in \mathbb{J}_C(\mathbb{F}_q)$ |

Table 2.7: Addition of ideal classes in a special case, char $\mathbb{F}_7 = 7$

| | |
|---|---|
| $D_1$ | $[x^2 + 6x + 6, 6x + 1]$ |
| $D_2$ | $[x^2 + 6, 4x + 1]$ |
| $D_1 + D_2$ | $[x + 2, 3] \in \mathbb{J}_C(\mathbb{F}_q)$ |

### 2.3.3 Addition and doubling in inversion-free coordinates

In real-world applications different operations have different complexity. As a rule, the inversion is the slowest operation on contemporary common purpose processors and special hardware platforms. The second slowest operation is usually multiplication. The fastest one (from the set of the operations that have the most significant impact on the performance of scalar multiplication in $\mathbb{J}_C(\mathbb{F}_q)$) is squaring. It turns out [45], [46], [8] that one can find such sets of coordinates[9], which exclude the necessity to perform inversions while adding or doubling points in $\mathbb{J}_C(\mathbb{F}_q)$ of genus 2 hyperelliptic curves. Recall that one has to perform one inversion for each addition or doubling in affine coordinates.

It should pointed out that one gets different performance for the same operation while computing in finite fields of different characteristics. In addition to the fact that hyperelliptic curves have different curve equation representations for odd and even field characteristics, the squaring in binary finite field can be performed through cyclic bit shifts if a normal basis representation is used and, hence, is practically 'for free'. It means that distinct optimization techniques should be applied in the study of the fast arithmetic over hyperelliptic curves with respect to these two cases. This motivates their separate consideration.

At first, the case of odd characteristic is treated. A (reduced) divisor $D = [x^2 + u_1 x + u_0, v_1 x + v_0] \in \mathbb{J}_C(\mathbb{F}_q)$ in affine coordinates is represented as a quadruple of elements from the base field $\mathbb{F}_q$: $D = [u_1, u_0, v_1, v_0]$. Inversion-free arithmetic can be achieved through adding some further values to these ones. For odd characteristic there are two basic inversion-free coordinate systems: Projective ($\mathcal{P}$) coordinates [45] and new ($\mathcal{N}$) coordinates [46]. For even characteristic there is a further coordinate system called recent ($\mathcal{R}$) which proves to be more efficient in some cases. In projective coordinates an additional coordinate $z \neq 0$ is introduced leading to a slightly modified representation of $D$ through the quintuple $D = [u_1, u_0, v_1, v_0, z]$ of values from $\mathbb{F}_q$. Writing this one means that the rigourous Mumford representation of the divisor is $D = [x^2 + \frac{u_1}{z} x + \frac{u_0}{z}, \frac{v_1}{z} x + \frac{v_0}{z}]$. The operation count in Table 2.8 for addition in the introduced coordinates implies that we distinguish the inputs:

$$\begin{array}{l} D_1 = [u_{11}, u_{10}, v_{11}, v_{10}, z_1] \text{ and} \\ D_2 = [u_{21}, u_{20}, v_{21}, v_{20}, z_2] \end{array} \tag{2.53}$$

with $z_1 \neq 1$, $z_2 \neq 1$ and those with $z_1 = 1$, $z_2 \neq 1$.

In new coordinates it is suggested to use 2 additional coordinates $Z_1 \neq 0$ and $Z_2 \neq 0$ for each argument of the group law procedure with the following correspondence to the

[9]Note that these 'coordinates' which are often called 'projective' are no projective coordinates in the mathematical sense. These are merely sets of variables which are calculated in the course of addition and doubling, contain the necessary information about the inverted values needed to perform these operations and allow relatively low operation count.

Table 2.8: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, projective coordinates, $q$ odd

| | | |
|---|---|---|
| Addition | $z_1 = 1, z_2 \neq 1$ | 47M+4S |
| | $z_1 \neq 1, z_2 \neq 1$ | 40M+4S |
| Doubling | | 38M+6S |

Mumford representation: The sextuple $[u_1, u_0, v_1, v_0, Z_1, Z_2]$ corresponds to the ideal class $D = [x^2 + \frac{u_1}{Z_1^2} + \frac{u_0}{Z_1^2}, \frac{v_1}{Z_1^3 Z_2} x + \frac{v_0}{Z_1^3 Z_2}]$. To make the computations more efficient it is proposed to add two auxiliary coordinates to these set: $z_1 = Z_1^2$ and $z_2 = Z_2^2$ ($z_2$ being used by doubling only) which leads to the following octuple of coordinates: D=$[u_1, u_0, v_1, v_0, Z_1, Z_2, z_1, z_2]$. These coordinates are called new [46].

Assume that

$$\begin{aligned} D_1 &= [u_{11}, u_{10}, v_{11}, v_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}] \text{ and} \\ D_2 &= [u_{21}, u_{20}, v_{21}, v_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]. \end{aligned} \tag{2.54}$$

Table 2.9: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, new coordinates, $q$ odd [8]

| | | |
|---|---|---|
| Addition | $Z_{11} = Z_{12} = z_{11} = z_{12} = 1$ | 37M+6S |
| | otherwise | 47M+7S |
| Doubling | | 34M+7S |

In Table 2.9 we give operation counts for addition and doubling in new coordinates. As for the projective coordinates we distinguish between the cases of both $Z_{11}, Z_{12}, z_{11}, z_{12}$ and $Z_{21}, Z_{22}, z_{21}, z_{22}$ not being equal to 1 simultaneously and of $Z_{11} = Z_{12} = z_{11} = z_{12} = 1$.

From Tables 2.8 and 2.9 it follows that doubling is faster by 4M in new coordinates (at the cost of one additional squaring). In the common case the addition in new coordinates is by 3S slower as in projective ones. In the special case of addition (affine input) new coordinates enable the faster (by 3M at the expense of 2S) computation of the sum. The exact interaction of the computational efficiency in projective and new coordinates depends on the scalar multiplication method used which results in different numbers and types of additions and doublings. This is discussed in Section 2.4.

Now we consider inversion-free coordinates in even characteristic. For even characteristic we have two cases: $\deg(h) = 2$ (curve equation 2.48) and $\deg(h) = 1$ (curve equation 2.52). The case of $\deg(h) = 2$ is dealt with at first. Projective and new coordinates can be efficiently adapted to this case. Projective coordinates are the same as for the case of odd characteristic. The operation count for addition and doubling is given here in Table 2.10.

Table 2.10: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, projective coordinates, $q$ even, $\deg(h) = 2$ [8]

| | | |
|---|---|---|
| Addition | $z_1 = 1, z_2 \neq 1$ | 39M+4S |
| | $z_1 \neq 1, z_2 \neq 1$ | 49M+4S |
| Doubling | | 38M+7S |

New coordinates should be changed if one wants to achieve a better performance. The new coordinates specify a reduced divisor in the following way: $D = [u_1, u_0, v_1, v_0, Z_1, Z_2, z_1, z_2, z_3, z_4]$ meaning that the Mumford representation of $D$ is

$$\begin{aligned} &D = [x^2 + \tfrac{u_1}{Z_1^2} x + \tfrac{u_0}{Z_1^2}, \tfrac{v_1}{Z_1^3 Z_2} x + \tfrac{v_0}{Z_1^3 Z_2}] \text{ with} \\ &z_1 = Z_1^2, z_2 = Z_2^2, z_3 = Z_1 Z_2, z_4 = z_1 z_3, \end{aligned} \tag{2.55}$$

$Z_1$ and $Z_2$ being not used separately any more and, hence, we have the octuple: $D = [u_1, u_0, v_1, v_0, z_1, z_2, z_3, z_4]$. The complexity of the corresponding formulae can be found in Table 2.11.

Table 2.11: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, new coordinates, $q$ even, $\deg(h) = 2$ [8]

| | | |
|---|---|---|
| Addition | $Z_{11} = Z_{12} = Z_{11} = Z_{12} = 1$<br>otherwise | 37M+5S<br>48M+4S |
| Doubling | | 37M+6S |

The addition in new coordinates is faster by 1M in the common case and by 2M (at the cost of 1S which is almost 'for free' in binary fields) in case of the first input being affine. The doubling in new coordinates is faster by 1M and 1S. Though there is no pronounced advantage of new coordinates (which was the case for odd characteristic), they provide a faster computation of the group law.

If $\deg(h) = 1$, then some other coordinate sets can be applied to perform addition and doubling, but projective and new coordinates are still applicable too. The projective coordinates are like in case of odd characteristic. Table 2.12 summarizes the operation counts for the respective formulae.

Table 2.12: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, projective coordinates, $q$ even, $\deg(h) = 1$ [8]

| | | |
|---|---|---|
| Addition | $z_1 = 1, z_2 \neq 1$<br>$z_1 \neq 1, z_2 \neq 1$ | 37M+4S<br>47M+4S |
| Doubling | | 22M+6S |

The new coordinates are omitted here, since there are more efficient coordinates if $\deg(h) = 1$. These are called recent [8]. To represent a reduced divisor in recent coordinates one needs a sextuple: $D = [u_1, u_0, v_1, v_0, Z, z]$ giving rise to the following Mumford representation:

$$D = [x^2 + \tfrac{u_1}{Z}x + \tfrac{u_0}{Z}, \tfrac{v_1}{Z^2}x + \tfrac{v_0}{Z^2}] \text{ with } z = Z^2. \tag{2.56}$$

The coordinate set was chosen to ensure very efficient doubling at the expense of slower additions. The operation counts are represented in Table 2.13. Here it is always assumed that the extension degree of $\mathbb{F}_q$ over GF(2) is odd.

Table 2.13: Addition and doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, recent coordinates, $q$ even, $d$ odd, $\deg(h) = 1$ [8]

| | | |
|---|---|---|
| Addition | $z_1 = 1, z_2 \neq 1$<br>$z_1 \neq 1, z_2 \neq 1$ | 43M+7S<br>49M+8S |
| Doubling | | 20M+8S |

One can see that doubling is extremely cheap in recent coordinates for $\deg(h) = 1$ and odd extension degree $d$.

### 2.3.4 A point (de)compression technique in $\mathbb{J}_C(\mathbb{F}_q)$ over binary fields

Here some thoughts about the ways to (de)compress a point in $\mathbb{J}_C(\mathbb{F}_q)$ of binary genus 2 curves with $\deg(h) = 1$ are given. The approach discussed is based upon [49]. After the idea had kindly been made clear to the author by T. Lange, he found a simple way to reduce the number of operations needed of which the description can be found in this subsection. For

the point compression in $\mathbb{J}_C(\mathbb{F}_q)$ over fields of odd characteristic one is referred to [33] and [78] or to [8] for a survey.

Here the case of binary hyperelliptic curves is discussed. Moreover, curves given by the following equation are considered only:

$$\begin{array}{l} y^2 + h(x)y = f(x), \text{ where} \\ h(x) = h_1 x, \\ f(x) = x^5 + f_3 x^3 + f_2 x^2 + f_0. \end{array} \tag{2.57}$$

The divisor class $[u, v]$ which defines the corresponding element in $\mathbb{J}_C(\mathbb{F}_q)$ is represented by:

$$\left\{ \begin{array}{lcl} u & = & x^2 + u_1 x + u_0, \\ v & = & v_1 x + v_0. \end{array} \right. \tag{2.58}$$

It was suggested to store $u$ instead of allocating memory for $u$ and $v$. This requires twice less memory space to store a reduced divisor and may be of advantage on some very constrained platforms and such devices as cheap smart cards. To be able to decompress the value of $v$ (practically $v_1$ and $v_0$) it is necessary to store some further information as well. For the decompression we make use of the fact that $u | v^2 + hv + f$ for the reduced divisor $[u, v]$ given in the Mumford representation. In the concrete case one has that:

$$x^2 + u_1 x + u_0 | x^5 + f_3 x^3 + (v_1^2 + h_1 v_1 + f_2) x^2 + h_1 v_0 x + (v_0^2 + f_0). \tag{2.59}$$

Dividing out one has:

$$\begin{array}{l} \frac{x^5 + f_3 x^3 + (v_1^2 + h_1 v_1 + f_2) x^2 + h_1 v_0 x + (v_0^2 + f_0)}{x^2 + u_1 x + u_0} = \\ x^3 + u_1 x^2 + (f_3 + u_0 + u_1^2) + c + \frac{(h_1 v_0 + u_0 (f_3 + u_0 + u_1^2) + u_1 c) x + u_0 c + v_0^2 + f_0}{x^2 + u_1 x + u_0}, \end{array}$$

where

$$\left\{ \begin{array}{lcl} h_1 v_0 + u_0 (f_3 + u_0 + u_1^2) + u_1 c & = & 0, \\ u_0 c + v_0^2 + f_0 & = & 0 \end{array} \right. \tag{2.60}$$

with $c = v_1^2 + h_1 v_1 + f_2 + u_1 (f_3 + u_1^2)$.

This (non-linear) system of equations has 2 unknown variables: $v_1$ and $v_0$. The way to solve this system of equations suggested is to get $c$ at first:

$$c = \frac{1}{u_0} (v_0^2 + f_0) \tag{2.61}$$

and to use the first equation to get $v_0$ from the following one:

$$\begin{array}{l} v_0'^2 + v_0' + \beta_1, \text{ where} \\ \beta_1 = \frac{u_1}{h_1^2} (\frac{f_0 u_1}{u_0^2} + c_1), \\ v_0 = \frac{h_1 u_0}{u_1} v_0' \text{ and } c_1 = f_3 + u_0 + u_1^2. \end{array} \tag{2.62}$$

Then the concrete value of $c$ is computed using (2.61). The rest of computations is done by solving the quadric equation:

$$\begin{array}{l} v_1'^2 + v_1' + \beta_2, \text{ where} \\ \beta_2 = \frac{1}{h_1^2} (f_2 + u_1 (f_3 + u_1^2) + c) \text{ and } v_1 = h_1 v_1'. \end{array} \tag{2.63}$$

It is reasonable to require the value $(h_1^{-1})^2$ to have been precomputed or computed by the external control device on-line (the curve equation is not secret in the most security protocols). In the cryptographically interesting case of $d$ being odd we have *no precomputations* since $h_1 = 1$. Note that we have to solve 2 quadratic equations in (2.62) and (2.63) over $\mathbb{F}_q = \mathrm{GF}(2^d)$. This can be easily done [13], since the equations have been brought to the canonical form:

$$x^2 + x + \beta = 0. \tag{2.64}$$

If $x_0$ is a solution of this equation[10], then so is $x_0 + 1$. If $d$ is odd, then $x_0$ can be found as the half-trace of $\beta$:

$$x_0 = \tau(\beta) = \sum_{j=0}^{(d-1)/2} \beta^{2^{2j}}, \tag{2.65}$$

which can be got through computing $(d-1)$ squarings and $\frac{d-1}{2}$ additions in $\mathbb{F}_q$. In a normal basis squarings (bitwise cyclic shifts) and additions (bitwise addition modulo 2, i.e. XOR) are already for free (first of all in hardware). Now the algorithm to decompress a point from $u$ and 2 bits of additional information (Table 2.14) is given. Here Q denotes the complexity of the solution of a quadratic equation in the canonical form over $\mathrm{GF}(2^d)$ with $d$ even. From the table it follows that the decompression is extremely fast for hyperelliptic curves over $\mathrm{GF}(2^d)$ with $d$ odd.

## 2.4 Efficiency of cryptographic systems based on genus 2 hyperelliptic curves

As a result of Section 2.3 one has a framework for building an effective DLP based public key cryptographic system. To choose the best possible combination of coordinates for every case one has to fix some scalar multiplication method[11]. There is no known optimal (in every sense) scalar multiplication method. The designer has to analyse the used protocol and to choose the optimal method for the specific application. Here a scalar multiplication method called 'sliding window' and numbers written in the width-$w$ nonadjacent form ($\mathrm{NAF}_w$) are discussed. This form seems to be optimal in a certain sense. Then scalar multi-multiplication is mentioned. The section provides operation counts and concrete runtimes (on a multi-purpose processor) for the scalar multiplication in $\mathbb{J}_C(\mathbb{F}_q)$ in different sets of coordinates for odd and even char $\mathbb{F}_q$ and for different group sizes. Moreover, some comparisons with the scalar multiplication in the group of points of elliptic curves ($\cong \mathbb{J}_C(\mathbb{F}_q)$ for an elliptic curve $C$) can also be found here. The section ends in some reasoning about the cryptographical suitability of arithmetic over hyperelliptic curves from the point of view of the possibility of efficient implementation.

### 2.4.1 Scalar multiplication methods

Let $G$ be an additive finite cyclic group of order $n$. Assume $\alpha \in G$ and $k \in \{1, 2, \ldots, n\}$. The easiest way to compute $k\alpha$ is binary scalar multiplication (double and add)[12]. The algorithm is given in Table 2.15. Here the binary expansion representation of $k = (k_{l-1} \ldots k_0)_2$, $k_i \in \{0, 1\}$, $i = 0, \ldots, l-1$ (binary radix 2 representation) with $k = \sum_{i=0}^{l-1} k_i 2^i$ is used. The algorithm requires $l$ doublings and on the average $l/2$ additions.

[10] The equation in the form (2.64) has solutions over $\mathbb{F}_q$ if and only if $\mathrm{Tr}_{q|2}(\beta) = 0$.

[11] Note that one could use the notion 'exponentiation' instead if one were working with multiplicatively written groups. But dealing with additively written groups implies the notion 'scalar multiplication'.

[12] It is called square and multiply in multiplicative notation.

Table 2.14: Decompression of $[u, v] \in \mathbb{J}_C(\mathbb{F}_q)$, $q = 2^d$ (made up by the author on the basis of his modification of the decompression technique)

| INPUT: $u$, bits $b_1$ and $b_2$, precomputed $h_1^{-2}$ | |
|---|---|
| $h_1 \neq 1$, $d$ even | $h_1 = 1$, $d$ odd |
| 1. Compute $c_1$:<br>$c_1 \leftarrow f_3 + u_0 + u_1^2$.<br>2. Compute $\gamma_0 = u_0^{-1}$ and $\gamma_1 = u_1^{-1}$:<br>$\gamma_0 \leftarrow u_0 u_1$, $\gamma_0 \leftarrow \gamma_0^{-1}$,<br>$\gamma_1 \leftarrow u_0 \gamma_0 (= u_1^{-1})$,<br>$\gamma_0 \leftarrow \gamma_0 u_1 (= u_0^{-1})$<br>3. Compute $\beta_1$:<br>$\beta_1 \leftarrow u_1 h_1^{-2}(f_0 u_1 \gamma_0^2 + c_1)$<br>4. Solve (2.62) using $b_1$ and get $v_0'$.<br>5. Compute $v_0$ and $c$:<br>$v_0 \leftarrow h_1 u_0 \gamma_1 v_0'$, $c \leftarrow \gamma_0(v_0^2 + f_0)$.<br>6. Compute $\beta_2$:<br>$\beta_2 \leftarrow h_1^{-2}(f_2 + c + u_1(c_1 + u_0))$.<br>7. Solve (2.63) using $b_2$ and get $v_1'$.<br>8. Compute $v_1$:<br>$v_1 \leftarrow h_1 v_1'$. | 1. Compute $c_1$:<br>$c_1 \leftarrow f_3 + u_0 + u_1^2$.<br>2. Compute $\gamma_0 = u_0^{-1}$ and $\gamma_1 = u_1^{-1}$:<br>$\gamma_0 \leftarrow u_0 u_1$, $\gamma_0 \leftarrow \gamma_0^{-1}$,<br>$\gamma_1 \leftarrow u_0 \gamma_0 (= u_1^{-1})$,<br>$\gamma_0 \leftarrow \gamma_0 u_1 (= u_0^{-1})$<br>3. Compute $\beta_1$:<br>$\beta_1 \leftarrow u_1(f_0 u_1 \gamma_0^2 + c_1)$<br>4. Solve (2.62) using $b_1$ and get $v_0'$.<br>5. Compute $v_0$ and $c$:<br>$v_0 \leftarrow u_0 \gamma_1 v_0'$, $c \leftarrow \gamma_0(v_0^2 + f_0)$.<br>6. Compute $\beta_2$:<br>$\beta_2 \leftarrow f_2 + c + u_1(c_1 + u_0)$.<br>7. Solve (2.63) using $b_2$ and get $v_1'$.<br>8. Compute $v_1$:<br>$v_1 \leftarrow v_1'$. |
| I+14M+3S+2Q | I+10M+$(d+2)$S |
| OUTPUT: $D = [u, v]$ | |

Table 2.15: 'Double and add' scalar multiplication method

INPUT: $\alpha \in G$, $k = (k_{l-1} \dots k_0)_2 \in \{1, 2, \dots, n-1\}$

1. $\beta \leftarrow 1$
2. for $i$ from $l-1$ downto 0 do

   $\beta \leftarrow 2\beta$

   if $k_i = 1$ then $\beta \leftarrow \alpha + \beta$

OUTPUT: $\beta = k\alpha$

Note that it requires no additional memory and no precomputations. A further idea is to use other ways to write out the representation of the scalar multiplier $k$. This leads to 2 main groups of scalar multiplication algorithms. The first ones are based on a lager radix $b = 2^w$. The second group uses other (signed, i.e. digits can be negative) ways to write $k$. The combination of the both ways which we will consider by the example of $\mathrm{NAF}_w$ delivers the most effective scalar multiplication methods. The consideration begins with signed representations. Then algorithms for scalar multiplication on radix $b = 2^w$ and the definition of $\mathrm{NAF}_w$ are given.

$k = \sum_{i=0}^{l-1} k_i 2^i$ is called a $\tau$-representation, if $\tau$ is a set of integers and all $k_i \in \tau \cup \{0\}$. If $\tau$ contains negative integers, one speaks of *signed* representation, and if $\tau = \{1, -1\}$, of *signed binary* representations. This signed binary representation is said to be in *non-adjacent form (NAF)*, if $k_i k_{i+1} = 0$, $i = 0, \ldots, l-1$. This is denoted by $(k_{l-1} \ldots k_0)_{NAF}$. In [58] NAF is called a *sparse signed-digit* representation. NAF was introduced in [71] and since then generalized by numerous researchers. It is known that ignoring leading zeros each integer has a unique NAF representation. An algorithm to get the NAF of an integer in binary expansion form is in Table 2.16. The NAF can be computed using simple look-ups from a table of size 8 bit with elements of size 2 bit [58]. Moreover, the NAF representation provides the minimal Hamming weight among all signed digit representations for $k$. Consequently, the NAF representation of the scalar multiplier combined with the binary double-and-add method is the optimal choice if it is possible to efficiently invert in $G$ and no precomputations are allowed. The expected number of non-zero terms in a NAF expansion of length $l$ if all possible scalar multipliers are uniformly distributed is $l/3$. Now one can change Algorithm 2.15 to achieve the best possible performance without precomputations. Assume the input is in NAF. Then at step 1 we should additionally compute $\gamma \leftarrow -\alpha$ (the inversion in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ is almost for free) and at step 2 we should add the following string to each iteration of the cycle:

$$\text{if } k_i = -1 \text{ then } \beta \leftarrow \gamma + \beta. \tag{2.66}$$

That is, if the base element $\alpha$ and the scalar multiplier $k$ are not known in advance and it is impossible to perform computations one has to perform $l$ doublings and $l/3$ additions accompanied by one group inversion. This algorithm can be the one of choice on some very constrained platforms even if such protocols as generalized DSA or Diffie-Hellman are used in which the base group elements are always known beforehand which enables us to tabulate some values.

Assume the base group element is known beforehand only, the scalar multiplier being chosen at random. If precomputations are allowed and some storage is available, one can use the *sliding window* method (see Algorithm 2.17). Let $k$ be in its binary expansion form. To compute $k\alpha$ one precomputes the values $3\alpha, 5\alpha, \ldots, 2^{t-1}\alpha$ first and stores them for future use. Note that one has to compute the values with odd factor only. Then one reads the scalar multiplier $k$ from the left to the right considering $t$ consequent bits at once (a sliding window of length $t$ bits). If the most significant bit in the window under consideration is 0, then shift the window by one bit to the right performing one doubling. Otherwise search for the longest substring of length $\leq t$ in the whole number $k$ with the property that the least significant bit of the substring is 1 and the found substring contains the most significant bit of the current window.

Now one could compute $k_{NAF}$ from $k$ and proceed with the sliding window method, applying it to the signed binary representations by making alterations in Algorithm 2.17 similar to those in case of the binary method. But here another way is taken by computing a generalization of NAF (which would be suitable for sliding window techniques) directly from $k$ [69].

Table 2.16: NAF representation

INPUT: $k = (k_l k_{l-1} \dots k_0)_2$, $k_l = k_{l-1} = 0$

1. $c_0 \leftarrow 0$
2. for $i$ from 0 to $l-1$ do

   $k_{i+1} \leftarrow \lfloor (c_i + k_i + k_{i+1})/2 \rfloor$
   $k'_i \leftarrow c_i + k_i - 2k_{i+1}$

OUTPUT: $(k'_{l-1} \dots k'_0)_{NAF}$

Table 2.17: Sliding window scalar multiplication method

INPUT: $\alpha \in G$, $k = (k_{l-1} \dots k_0)_2$, parameter $t$, precomputed table $3\alpha, 5\alpha, \dots, 2^{t-1}\alpha$

1. $\beta \leftarrow 1$, $i \leftarrow l-1$
2. while $i \geq 0$ do

   if $k_i = 0$ then $\beta \leftarrow 2\beta, i \leftarrow i-1$
   else

      $s \leftarrow \max(i-k+1, 0)$
      while $n_s = 0$ do $s \leftarrow s+1$
      for $h = 1$ to $i-s+1$ do $y \leftarrow 2y$
      $u \leftarrow (k_i \dots k_s)_2$
      $\beta \leftarrow \beta + (u\alpha)$
      $i \leftarrow s-1$

OUTPUT: $\beta = k\alpha$

**Definition 21** (width-$w$ NAF, NAF$_w$). *A signed $\tau$-representation $k' = \sum_{i=0}^{l-1} k'_i 2^i$ is called $NAF_w$ for some integer $w > 1$ if the following three properties hold:*

- *The most significant non-zero bit is positive, i.e. $k'_{l-1} > 0$.*
- *Among any $w$ consecutive digits, at most one is non-zero.*
- *Each non-zero digit $k'_i$ is odd and less than $2^{w-1}$ in absolute value, i.e. $|k'_i| < 2^{w-1}$.*

It is clear that for $w = 2$ one has the classical NAF. The NAF$_w$ $k'$ of an integer $k$ is denoted by $k' = k_{NAF_w}$. There are several known results about NAF$_w$ [66]:

- Every integer has at most one NAF$_w$.
- Every integer has a NAF$_w$.
- An integer's NAF$_w$ is at the most one digit longer that its binary expansion representation, i.e. the length of $k_{NAF_w}$ of $k$ is at most equal to $\lceil \log k \rceil + 1$.
- If $H(k)$ is the Hamming weight of some integer $k$, then $H(k_{NAF_w}) \to \frac{l}{w+1}$, $k \to \infty$.
- [6] NAF$_w$ has minimal weight among all signed radix 2 $\tau$-representations with $\forall a \in \tau : \ |a| \leq 2^{w-1}$.

The NAF$_w$ of $k$ can be computed using Algorithm 2.18. Here mods denotes the smallest residue in absolute value, i.e. $n \bmod 2^w \in \{-2^{w-1}+1, -2^{w-1}, \ldots, 2^{w-1}\}$. Now instead of applying Algorithm 2.17 to $k_{NAF}$ one can use Algorithm 2.15 with $k_{NAF_w}$ replacing the string:

$$\text{if } k_i = 1 \text{ then } \beta \leftarrow \alpha\beta \tag{2.67}$$

with

$$\text{if } k_i \neq 0 \text{ then } \beta \leftarrow (k_i\alpha)\beta, \tag{2.68}$$

where the values of $k_i\alpha$ have been precomputed for all possible $k_i$. This provides the complexity of $l$ additions and asymptotically $\frac{l}{w+1}$ doublings. In fact some tricks [21] lead to such a variant of this algorithm that there exists a real number $\rho > 1$ that this method leads to the following average complexity:

$$\begin{array}{l} D_{l,w} = l - \frac{w(w-1)}{2(w+1)} + O(\rho^{-l}) \text{ doublings and} \\ A_{l,w} = \frac{l}{w+1} - \frac{(w-1)(w+2)}{2(w+1)^2} + O(\rho^{-l}) \text{ additions} \end{array} \tag{2.69}$$

for $l$-bit numbers. Note that one has to precompute the table consisting of:

$$\pm 3\alpha, \pm 5\alpha, \ldots, \pm(2^{w-1}-1)\alpha, \tag{2.70}$$

which is not taken into account in the complexity estimates given above. NAF$_w$ is optimal in the sense of the trade-off between speed and memory for $w > 3$ [13] and can be the method of choice in the majority of applications in which some precomputations are allowed.

Note that there are some cryptographic protocols that require the computation of the sum of scalar products of several group elements. For instance, the signature verification procedure of the generalized DSA implies the computation of $\gamma = u_1\alpha + u_2\beta$, $\alpha, \beta, \gamma \in G$. The generalized version of the ElGamal algorithm [25] requires the computation of the sum of scalar products of three group elements. There exist some methods for the computation of scalar multi-multiplications. The first of them makes use of parallel sliding windows with a set of precomputed group elements in the form $\sum_{i=1}^{d} u_i\alpha_i$: $\alpha_i \in G$, $u_i \in \{1, \ldots, n\}$, $i = 1, \ldots, d$.

Here $d$ denotes the number of group elements of which scalar products are computed at the same time. This method gives advantage [4] over the straightforward computations of the concerned values. Here two basic effective representations are considered: NAF and JSF (Joint Sparse Form) allowing comparable efficiency (the exact relationships being dependent on the bit sizes of $u_i$ and the value of $d$). Another method to take priority of parallelism is the interleaved multi-multiplication for $\mathrm{NAF}_w$. For $d = 3$ this algorithm proves to be the best one. For $d = 2$ the parallel sliding window method with NAF is optimal from the point of view of speed-memory trade-off possibilities [4].

### 2.4.2 Fast scalar multiplication over $\mathbb{J}_C(\mathbb{F}_q)$ in different coordinates and implementation properties of genus 2 hyperelliptic curves

In $\mathbb{J}_C(\mathbb{F}_q)$ the group inversion is almost for free, since $-D = -[u, v] = [u, -h - v \bmod u]$. If $d$ is odd, then one can always assume $-h - v \bmod u = x + v = (v_1 + 1)x + v_0$. If $d$ is even, then $-h - v \bmod u = h + v \bmod u$ for $\deg h = 2$ and $-h - v \bmod u = h + v$ for the most interesting case of $\deg h = 1$). For all the further considerations it is assumed that precomputations are allowed and that the modified binary algorithm is applied to the $\mathrm{NAF}_w$ representation of $k$. All precomputation costs are left out. If no precomputations are allowed, then it is straightforward to derive the operation count from the information provided in this subsection and information about NAF in Subsection 2.4.1. See also [8].

The operation counts (addition and doubling) for different combinations of coordinates in odd characteristic are given in Tables 2.19 and 2.20. From the tables it follows that the fastest scalar multiplication without inversions can be done combining affine and new coordinates for addition and new coordinates for doubling. If inversions are relatively cheap, then it is recommended to use affine coordinates only. The both cases are treated in Table 2.21. The concrete running times taken from [5] on a standard AMD Athlon at 1GHz under SuSe Linux 9.0 for different coordinate systems together with those for elliptic curves which provide the same group size are given in Table 2.22. The $\mathrm{NAF}_w$ representation and the corresponding algorithm seem to be the most effective combination in case of both elliptic and hyperelliptic curves. The results are provided for $\mathrm{NAF}_w$ only together with the optimal value of $w$ which has been chosen experimentally. Table 2.23 provides the ratio of the running times which arise in Table 2.22 in the corresponding coordinates for the same group size. One sees immediately that elliptic curves are faster in odd characteristic in inversion-free coordinate systems with the formulae we have for the time being. In affine coordinates the arithmetic over hyperelliptic curves is faster.

The operations of addition and doubling with different combinations of coordinate systems in even characteristic have complexity values shown in Tables 2.24 and 2.25 respectively. Here the most interesting case of $\deg h = 1$ is considered only and it is assumed that the extension degree of $\mathbb{F}_q$ over GF(2) is odd to get $h$ monic. The operation counts for the scalar multiplication are given in Table 2.26. Everywhere the scalar multiplication is assumed to be performed in accordance with $\mathrm{NAF}_w$. As for the case of odd characteristic we give the running times [19] of the scalar multiplication using $\mathrm{NAF}_w$ for elliptic and hyperelliptic curves in Table 2.27. The measurements were performed on the same hardware and software platform. The implementation results by Cesena are not complete and involve affine coordinates and 2 group sizes only. Since Cesena does not implement any inversion-free arithmetic and, hence, the results are not so representative as those for large prime fields given in Table 2.22 from [5], but provide a notion of the efficiency of the arithmetic over genus 2 hyperelliptic curves in even characteristic.

Thus, genus 2 hyperelliptic curves can provide a framework for high performance implementations of security protocols. The speed of such realizations is very close to the perfor-

Table 2.18: $\text{NAF}_w$ representation

INPUT: integer $k$, parameter $w > 1$

1. $i \leftarrow 0$
2. while $k > 0$ do

   if $k$ is odd then

   $k_i \leftarrow k \bmod 2^w$

   $k \leftarrow k - k_i$

   else $k_i \leftarrow 0$

   $k \leftarrow k/2, i = i + 1$

OUTPUT: $(k_{l-1} \dots k_0)_{NAF_w}$

Table 2.19: Addition in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ odd [8]

| Operation | Costs |
|---|---|
| $\mathcal{N} + \mathcal{N} = \mathcal{P}$ | 51M+7S |
| $\mathcal{N} + \mathcal{P} = \mathcal{P}$ | 51M+4S |
| $\mathcal{N} + \mathcal{N} = \mathcal{N}$ | 47M+7S |
| $\mathcal{N} + \mathcal{P} = \mathcal{N}$ | 48M+4S |
| $\mathcal{P} + \mathcal{P} = \mathcal{P}$ | 47M+4S |
| $\mathcal{P} + \mathcal{P} = \mathcal{N}$ | 44M+4S |
| $\mathcal{A} + \mathcal{N} = \mathcal{P}$ | 40M+5S |
| $\mathcal{A} + \mathcal{P} = \mathcal{P}$ | 40M+3S |
| $\mathcal{A} + \mathcal{N} = \mathcal{N}$ | 36M+5S |
| $\mathcal{A} + \mathcal{P} = \mathcal{N}$ | 37M+3S |
| $\mathcal{A} + \mathcal{A} = \mathcal{A}$ | I+22M+3S |

Table 2.20: Doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ odd [8]

| Operation | Costs |
|---|---|
| $2\mathcal{N} = \mathcal{P}$ | 38M+7S |
| $2\mathcal{P} = \mathcal{P}$ | 38M+6S |
| $2\mathcal{P} = \mathcal{N}$ | 35M+6S |
| $2\mathcal{N} = \mathcal{N}$ | 34M+7S |
| $2\mathcal{A} = \mathcal{A}$ | I+22M+5S |

Table 2.21: Costs of the $\text{NAF}_w$ scalar multiplication method in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ odd (made up by the author partially following [47])

| Case | Costs |
|---|---|
| $2\mathcal{A} = \mathcal{A}, \mathcal{A} + \mathcal{A} = \mathcal{A}$ | $(D_{l,w} + A_{l,w})I + 22(D_{l,w} + A_{l,w})M + (5D_{l,w} + 3A_{l,w})S$ |
| $2\mathcal{N} = \mathcal{N}, \mathcal{A} + \mathcal{N} = \mathcal{N}$ | $(34D_{l,w} + 36A_{l,w})M + (7D_{l,w} + 5A_{l,w})S$ |

Table 2.22: Running times (in msec) of scalar multiplication in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 1$ and $g = 2$, $q = p$ large prime, $\text{NAF}_w$ [5]

| Coordinates | Bitlength of group order | | | | | |
|---|---|---|---|---|---|---|
| | 128 | 144 | 160 | 192 | 224 | 256 |
| $g = 1$ | | | | | | |
| $\mathcal{A}$ | $1.363_{w=4}$ | $2.205_{w=3}$ | $2.489_{w=4}$ | $4.335_{w=4}$ | $6.841_{w=4}$ | $10.099_{w=4}$ |
| $\mathcal{P}$ | $0.551_{w=3}$ | $0.808_{w=3}$ | $0.982_{w=3}$ | $1.591_{w=3}$ | $2.711_{w=4}$ | $3.523_{w=4}$ |
| $\mathcal{T}^m$ | $0.474_{w=3}$ | $0.684_{w=3}$ | $0.838_{w=3}$ | $1.395_{w=3}$ | $2.296_{w=3}$ | $3.048_{w=3}$ |
| $g = 2$ | | | | | | |
| $\mathcal{A}$ | $0.730_{w=4}$ | $1.421_{w=4}$ | $1.558_{w=4}$ | $2.053_{w=4}$ | $3.730_{w=4}$ | $4.464_{w=4}$ |
| $\mathcal{P}$ | $0.703_{w=4}$ | $1.211_{w=4}$ | $1.352_{w=4}$ | $1.742_{w=4}$ | $3.357_{w=4}$ | $4.002_{w=4}$ |
| $\mathcal{N}$ | $0.675_{w=4}$ | $1.14_{w=4}$ | $1.262_{w=4}$ | $1.623_{w=3}$ | $3.020_{w=4}$ | $3.575_{w=4}$ |

Table 2.23: Ratios of the running times of scalar multiplication in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 1$ and $g = 2$, $q$ odd, $\text{NAF}_w$, different group sizes (made up by the author on the basis of Table 2.22 )

| Coordinates | Bitlength of group order | | | | | |
|---|---|---|---|---|---|---|
| | 128 | 144 | 160 | 192 | 224 | 256 |
| $\mathcal{A}_{g=2}/\mathcal{A}_{g=1}$ | 0.53 | 0.64 | 0.63 | 0.47 | 0.55 | 0.44 |
| $\mathcal{P}_{g=2}/\mathcal{P}_{g=1}$ | 1.28 | 0.50 | 1.38 | 1.09 | 1.24 | 1.14 |
| $\mathcal{N}_{g=2}/\mathcal{T}^m_{g=1}$ | 1.42 | 1.67 | 1.51 | 1.16 | 1.31 | 1.17 |

Table 2.24: Addition in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ even, $d$ odd, $\deg(h) = 1$ [8]

| Operation | Costs |
|---|---|
| $\mathcal{R} + \mathcal{R} = \mathcal{R}$ | 49M+8S |
| $\mathcal{A} + \mathcal{R} = \mathcal{R}$ | 42M+7S |
| $\mathcal{A} + \mathcal{A} = \mathcal{A}$ | I+21M+3S |

Table 2.25: Doubling in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ even, $d$ odd, $\deg(h) = 1$ [8]

| Operation | Costs |
|---|---|
| $2\mathcal{P} = \mathcal{P}$ | 22M+6S |
| $2\mathcal{R} = \mathcal{R}$ | 20M+8S |
| $2\mathcal{A} = \mathcal{A}$ | I+5M+6S |

Table 2.26: Costs of the $\text{NAF}_w$ method in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 2$, $q$ even, $d$ odd, $\deg(h) = 1$ (made up by the author partially following [47])

| Case | Costs |
|---|---|
| $2\mathcal{A} = \mathcal{A}, \mathcal{A} + \mathcal{A} = \mathcal{A}$ | $(D_{l,w} + A_{l,w})I + (5D_{l,w} + 21A_{l,w})M + (6D_{l,w} + 3A_{l,w})S$ |
| $2\mathcal{R} = \mathcal{R}, \mathcal{A} + \mathcal{R} = \mathcal{R}$ | $(20D_{l,w} + 42A_{l,w})M + (8D_{l,w} + 7A_{l,w})S$ |

Table 2.27: Running times (in $\mu$sec) of scalar multiplication in $\mathbb{J}_C(\mathbb{F}_q)$, $g = 1$ and $g = 2$, $q$ even, $d$ odd, $\deg(h) = 1$, $\text{NAF}_w$ [19]

| Coordinates | Bitlength of group order | |
|---|---|---|
| | 163 | 191 |
| $g = 1$ | | |
| $\mathcal{A}$ | $4647.129_{w=3}$ | $6064.593_{w=4}$ |
| $g = 2$ | | |
| $\mathcal{A}$ | $2207.223_{w=4}$ | $4581.345_{w=4}$ |

mance characteristics of cryptographical system built on elliptic curves. In odd characteristic hyperelliptic curves cannot provide the same performance level as elliptic curves and are about 30% slower for cryptographically relevant group sizes (with the most efficient formulae we know). This can be due to the fact that not enough research has been invested in the specialization of the general case. It seems to be promising to search for the possibilities to perform the addition/doubling faster in some special cases which are not without fail dependent on a special curve equation. Over binary fields of odd extension degrees genus 2 hyperelliptic curves with $\deg(h) = 1$ overtake elliptic curves due to very efficient doubling in recent coordinates. In this case the arithmetic in $\mathbb{J}_C(\mathbb{F}_q)$ is faster over genus 2 hyperelliptic curves than over the corresponding elliptic curves and our (de)compression technique (see Subsection 2.3.4) offers high performance. All these facts make this type of hyperelliptic curves very attractive for cryptography. This demonstrates that hyperelliptic curves of genus 2 can actually provide a more efficient framework for implementing cryptographic systems than genus 1 curves. Another advantage of genus 2 curves over genus 1 ones is the smaller base field at the same security level. This is of importance in dedicated hardware (for instance, cryptographic accelerators for high-end smart cards). From the hardware point of view every reduction of the digit capacity of operand results in lower propagation properties and brings down the gate count which leads to both better performance and lower costs of the resulting hardware (see for instance [10] for a hardware architecture for genus 2 hyperelliptic curves). Since these solutions are of great importance in restricted environments which are often mobile, they are subject to non-mathematical attacks such as side-channel attacks (SCAs). This kind of attacks can be very dangerous. That is, additional countermeasures should be taken to protect an implementation from SCAs. This has been done for elliptic curves in a number of ways (see for example [53], [15], [63]). For hyperelliptic curves there exists some lack of ideas and concrete effective protected solutions. Some approaches to this problem are discussed in the next chapter after a survey of side-channel attacks and countermeasures.

# Chapter 3

# Side Channel Attacks on Curve Based Crypto Systems and Countermeasures

## 3.1 Threats and vulnerability

In this section the side-channel attacks (SCA) are considered which were introduced in Subsection 1.1.1 of Chapter 1. A number of side-channel attacks are described. They are based on different computational and physical phenomena such as execution time, power consumption, electromagnetic emanation, sound, failure behaviour of an implementation or any combination of them (which has not, however, been carefully studied for the time present). Often SCAs require only restricted time and computational resources to restore sensitive information. The SCAs are partially possible without exact knowledge of implementation. As a rule the SCAs are more efficient than the corresponding mathematical attacks.

### 3.1.1 Timing attack (TA)

TA seems to have been the first side-channel attack in the literature [40]. In TA the execution time till a certain place in the algorithm has been reached serves as a stochastic variable in statistical sense. This aleatory variable is dependent on some secret bits of the key and can be used to successively obtain them. For the determination of unknown bits one observes the mean value or the variance. As a rule this attack is more difficult to perform than power attacks.

### 3.1.2 Simple and differential power analysis (SPA and DPA)

SPA is based on the direct interpretation of the power profile of a device performing an operation of which the concrete run depends on some key bits. That is, the measurement of the power profile is performed once. The aim is to ascertain from this power profile what instruction and is being executed at the moment and what operands this operation is using. For this method the adversary needs the precise knowledge of the implementation and the platform. SPA uses no statistical methods. In Fig. 3.1 SPA is illustrated[1] in case the run of the algorithm is dependent on key bits directly (e.g., conditional branch with consequent key bits as condition) and there is a possibility to distinguish the two operations (A and B in Fig. 3.1) which are performed depending on the current key bit. A is executed if the

[1]The figure is due to Camille Vuillaume [83].

bit is 0, B otherwise. Note that such attacks can be very efficient, if a scalar multiplication method without any countermeasures is employed, e.g. using Algorithm 2.15 (double and add method).

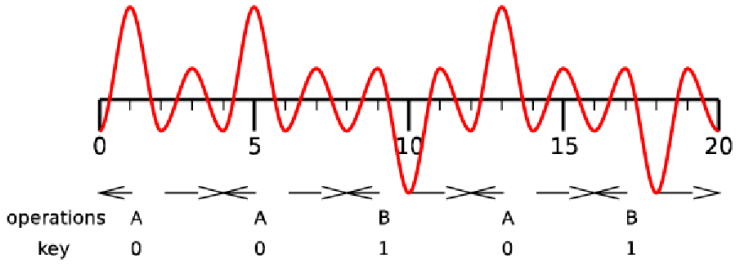


Figure 3.1: Schematic diagram of the simple power analysis method.

If the signal-noise-ratio is very small (that is, the dependence on the secret key induces tiny differences in the power trace, and these are embedded into the noise), then one can use statistical methods (averaging techniques) in order to amplify the signal [41]. DPA is based upon a large number of measurements during which the noise is considerably reduced by applying statistical methods. Note that the exact knowledge of the implementation is not necessary to be able to successfully perform the attack. The basic principle of the attack is to make a hypothesis about the values of so-called target bits at a certain appropriate step of the attacked algorithm. This place is usually at the begin or at the end of the algorithm. The target bits may depend on a few of the key bits only. Then the cryptographic algorithm is repeated with random inputs and the same secret key. During the measurements the corresponding discrete time-dependent curves are drawn and stored. The adversary can divide the power profiles into two subsets on the basis of the hypothesis. In each of these classes a specific attribute (e.g., a spike in the power profile at some place) remains the same. But this attribute is different in the two classes. For classifying a deterministic function of the form $F : (\text{input}, \text{target bits}) \to \{\text{class } 1, \text{class } 2\}$ is used. As the statistic has been gathered, the mean power profile is computed in each class. Random components tend to vanish. The constant attribute remains in the form of spikes. At the end the difference of the mean power profiles is computed (for a simplified illustration see[2] Fig. 3.2). If the assumption about the value of the target bits was incorrect, then the average power traces are the same in the two cases and the computed difference is a relatively straight line. But if the hypothesis was correct, then the average power traces possess a spike at the same place. The orientations of the spikes in the both classes are opposite to each other (downwards and upwards). Hence, the difference is a much more higher (deeper) spike. By successively applying this method it can be very easy to determine the whole secret key.

In a higher order DPA [41], [59] the adversary calculates joint statistical properties of power consumption at multiple sample times within power profiles as opposed to the simple DPA, where the correlation of power consumption at one place of the algorithm is considered only. In other words, a $k$th-order DPA is defined as a DPA that makes use of $k$ different samples in the power consumption signal that correspond to $k$ different intermediate values measured during the execution of the algorithm. The analysis can be especially powerful if the attacked algorithm reuses some variables and their usage is dependent on the secret key.

Address-bit DPA [60], [61] exploits the fact that within an implementation that uses memory operands in a key-dependent manner there is another representation of the data –

[2]The figure is due to Camille Vuillaume [83].

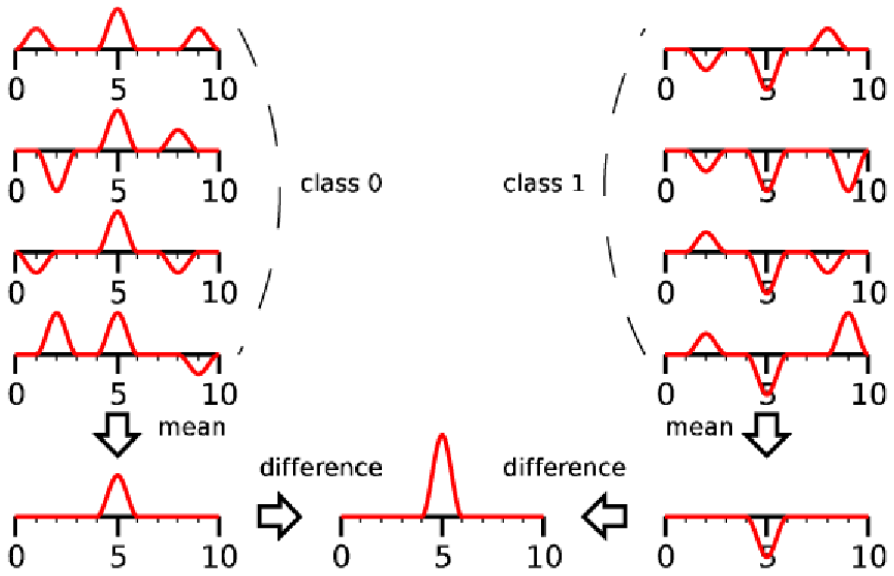


Figure 3.2: Schematic diagram of the differential power analysis method.

memory addresses. If the measurement of the address related values can be performed exactly and the details of the implementation are known to the adversary, one deals with a simple address bit power attack. Otherwise the adversary can try to detect operand reuse using statistical methods. This works obviously better if there are only a few memory locations. The analogous attack using differential electromagnetic emission analysis can be considered too. Against higher order bit-address differential SCAs operand relocation [7] for every run seems the only choice.

### 3.1.3 Differential fault analysis (DFA)

DFA was introduced in [12]. The basic idea of DFA is to cause errors in the course of a cryptographic operation from the outside and herewith to increase the amount of the information about the secret key in the output. The errors can be introduced by glitches (introduction of voltage transients into the power or clock line), x-rays, exposure to electromagnetic radiation in the visible or ultra-violet range and so on. If a device is not protected by any shielding or if the security model allows invasions, the DFA can be very powerful. The application of DFA to RSA can be found in [3]. The attack can be also powerful in case of elliptic and hyperelliptic curves [11].

### 3.1.4 Goubin-type analysis

To withstand differential attacks it is proposed to use different randomization techniques [7]. The Goubin-type analysis [31] is a refinement of the power analysis and allows the adversary to outflank some types of randomization. At first applied to elliptic curves, the attack proved to be a more serious threat for hyperelliptic ones. The Goubin-type analysis is based on the observation that some elements of the group over which a cryptographic system is built are randomized worse (or not randomized at all) by certain randomization techniques. The examples of such elements are reduced divisors from $\mathrm{Pic}_K^0(C)$ with a zero element in their representation over $K$. For genus 2 curves these are divisors of degree 0 and 1 which are given by respectively $[0 \cdot x^2 + 0 \cdot x + u_0, 0 \cdot x + 0]$ and $[0 \cdot x^2 + x + u_0, 0 \cdot x + v_0]$ in the Mumford

representation. For elliptic curves such elements are rational points with a zero coordinate. If the protocol permits the adversary to input the base elements for scalar multiplication using the secret key, this attack can be very efficient. In case one can detect these special elements from the distinct run times of the algorithm, this can be called a Goubin-type timing attack.

## 3.2 Technical and mathematical countermeasures

In this section some countermeasures to resist SCA on public key systems are discussed. The consideration is focused on the curve based asymmetric cryptographic systems. In principle there are two main ways to protect an implementation from SCAs. These are:

- The *homogenization* of the parameters subject to measurement by the adversary according to the security model and
- The *randomization* of these parameters.

All the methods outlined below realize one of these principles in a certain form.

Although differential attacks can be very efficient for the majority of symmetric key ciphers (which are not considered here), this is not the case for public key cryptosystems. In most cases (see generalized DSA and generalized Diffie-Hellman protocol in Subsection 1.1.2 of Chapter 1) the base group element is fixed by the system and the scalar multiplier is ephemeral, varying at each execution. But there are (a minor number of) protocols and cryptosystems where the scalar multiplier is the secret key and is fixed. In such a case one has to face the possibility of an effective differential analysis. For this reason we limit ourselves to the countermeasures against simple SCAs while considering mathematical measures. Among technical methods, however, randomization techniques are also considered.

### 3.2.1 A survey of technical countermeasures against SCA

Although technical countermeasures can be very efficient, there is little information on this topic in the literature. An attractive approach is the design of a hardware device with constant power consumption. This is discussed in [81]. This countermeasure is very expensive to implement, but it protects also against attacks with possible tampering and modification of the device before performing measurements. In [73] it is proposed to detach the internal power supply of a device from the external power consumption by putting additional capacitors or batteries on the power supply path. In practice this approach increases the number of traces required for a successful DPA. The method is easy to implement, but the additional elements can be easily removed by the attacker if the security model allows a direct access to the device. The easiest way to resist EMA is to shield the device. This method is very cheap, but it is useless against an active adversary. Random process interrupts (RPIs) are realized as interleaving the execution of the main computational process and some dummy computations at random. If the processor is capable of multithreading, then one can combine dummy and useful threads at random. On a clocking device it is possible to randomize the clock signal [42]. A processor implementing RPIs on parallel pipelines and random time shifts is described in [57]. The approach combining power-management techniques and randomized clock gating is called randomized power masking and was proposed in [9]. Another idea which the author has not found in the literature is to carry out the power supply of a device directly with noise and so preventing the adversary from measuring the exact power values. The method is very expensive to implement, but it can offer good security even with an active adversary by involving some specific security protocols. One more way to complicate DPA is to limit the number of possible operations with the same secret key. This leads to the restriction of

the available statistical material. This countermeasure is easy and cheap to implement, but is not appropriate at all if the cryptographical operation should be performed very often or the change of the key is connected with baffling organizational complexity. However, all these randomization techniques cannot guarantee the unconditional resistance of an implementation (device) to all SCAs in all security models (passive, active or restrictedly active adversary). Moreover, some of these methods and their combinations are expensive in terms of higher power consumption, increased gate area and additional execution time. For the time being the author sees the best (in terms of the trade-off between the resulting implementation costs and security properties) way of hedging the implementation against SCAs in a reasonable combination of technical and mathematical countermeasures.

### 3.2.2 Mathematical countermeasures against simple SCAs on curve based public key cryptosystems

Now countermeasures against simple SCAs such as SPA are considered. All these methods exploit the idea of the homogenization of the algorithm by some means. Moreover, we confine ourselves to elliptic curves and genus 2 hyperelliptic curves. Within the framework of this target setting we concentrate on dummy arithmetic instructions, indistinguishable or unified addition and doubling formulae and scalar multiplication with fixed sequence of operations.

#### Dummy operations and unified formulae for addition and doubling

Dummy operations can be incorporated into the code to make the run of the scalar multiplication algorithm homogenous. According to [22] the scalar multiplication 'add-and-double' method is modified to the one represented in Table 3.1 ('always-add-and-double' method). It requires $l$(D+A) operations, but protects against simple SCAs. A more sophisticated way to add dummy group operations was proposed in [34] and consists in transforming a NAF into a sequence of the fixed blocks 'double-double-add' by inserting dummy additions and doublings. This saves on average 44% additions and requires 11% extra doublings compared to the 'always-add-and-double' method. No additional storage is needed. However, it may be possible to filter out the dummy operations using DFA.

While the previously mentioned methods homogenize the sequence of group operations, the following methods try to make the flow of underlying field operations homogenous. Thus, another way of using dummy operations is to split addition and doubling in more elementary blocks and then make those look identical in terms of base field operations. If the difference of addition and doubling is not significant (they are almost homogenous), this can be easily achieved by adding several dummy operations and reordering the run of the algorithms. This approach was pursued in [20] for elliptic curves. This method is relatively cheap to implement (the performance penalty can be small). It is of great importance if addition and doubling differ drastically as for genus 2 curves over binary fields of odd extension degree with $\deg h = 1$. The drawback of this approach is that it allows the adversary to obtain the Hamming weight of the scalar multiplier $k$ by merely observing the execution time, that it must be very carefully implemented to really avoid intervals between blocks sequences of different lengths, and that fault attacks may still be used to reveal the dummy operations and thus distinguish the operations.

In [63] the notion of pipelining is introduced and in [64] atomic blocks for addition and doubling over elliptic curves are combined with precomputations according to the comb scalar multiplication method [32] which reduces and makes constant the number of needed doublings as well. This additional countermeasure does not allow the adversary to read the Hamming weight of the scalar multiplier. But the latter approach suffers from the lack of flexibility

being aimed at the single scalar multiplication method, and, hence, applies to the fixed base point scenario only.

**Unified addition and doubling over genus 2 hyperelliptic curves**

For higher genus hyperelliptic curves the problem of finding an effective and generic countermeasure against simple SCAs is still an open question. Recently a method to unify addition and doubling was proposed in [50] for genus 2 hyperelliptic curves over fields of both odd and even characteristics. The unification requires some dummy operations in both doubling and addition formulae. The computations can be parallelized. As a basis for the unification affine coordinates and new coordinates (see Section 2.3) were chosen for curves binary and large prime fields respectively. In Table 3.2 we give the operation count for the prototype (unshielded) addition and doubling operations and for the unified ones. It is assumed that from the point of view of the adversary the operations of multiplication and doubling cannot be distinguished using information from SCA in case $\operatorname{char} \mathbb{F}_q \neq 2$. The approach is relatively straightforward and the corresponding formulae result almost directly from the explicit formulae mentioned in Section 2.3 after properly re-ordering so as to completely unify the runs of the operations. The formulae require practically no additional field operations and can be easily combined with every scalar multiplication method we have considered. Moreover, they can be run in parallel on a limited number of registers (17 registers for $\operatorname{char} \mathbb{F}_q = 2$, 19 registers for $\operatorname{char} \mathbb{F}_q \neq 2$). The only drawback of this approach is the possibility for the adversary to determine the Hamming weight of the scalar multiplier. We see the further evolution in this area in getting atomic blocks for some special cases (such as genus 2 hyperelliptic curves over binary fields with $\deg h = 1$), minimizing the number of required registers, getting similar unified representations for other coordinate systems including inversion-free coordinates for hyperelliptic curves over binary fields.

**Montgomery ladder**

All the above discussed countermeasures against simple SCAs have (at least) one of the following drawbacks:

- It is possible for the adversary to get the Hamming weight of the secret scalar multiplier;
- One can try to filter out dummy operations by applying DFA;
- Some precomputations are needed (targeted at a fixed base point cryptosystem).

In order to be able to avoid these security and architectural flaws one needs a representation of scalar multiplication with fixed group operations which would be computationally more efficient than the always-double-and-add method. It turned out that there is such a method. It had been initially proposed by Montgomery in [65] for elliptic curves of a special (Montgomery) form in the context of speeding up the elliptic curve factorization method.

Here a generalization of the Montgomery scalar multiplication method is given partially following [35]. Let $G$ be a commutative group, $g \in G$ be one of its elements, $k \in \{1, \ldots, \operatorname{ord}(g)\}$ be a $l$-bit positive integer. Let

$$k = \sum_{i=0}^{l-1} k_i 2^i \tag{3.1}$$

be the binary expansion of the scalar multiplier $k$. Now we define:

$$\begin{array}{l} L_j = \sum_{i=j}^{l-1} k_i 2^{i-j} \text{ and} \\ H_j = L_j + 1. \end{array} \tag{3.2}$$

Table 3.1: Always double and add method

INPUT: $\alpha \in G$, $k = (k_{l-1} \dots k_0)_2 \in \{1, 2, \dots, n-1\}$

1. $\beta_0 \leftarrow \alpha$
2. for $i$ from $l-1$ downto 0 do

   $\beta_0 \leftarrow 2\beta_0$
   $\beta_1 \leftarrow \beta_0 + \alpha$
   $\beta_0 \leftarrow \beta_{k_i}$

OUTPUT: $\beta_0$

Table 3.2: Efficiency of the unified group operation in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$, g=2, $q$ odd and $q$ even [50]

| char $\mathbb{F}_q$ | Type | Operation | I | M | S | Dummy |
|---|---|---|---|---|---|---|
| $= 2$ | not unified, affine | ADD | 1 | 22 | 3 | - |
| | | DBL | 1 | 23 | 5 | - |
| | unified | ADD | 1 | 23 | 5 | 2M+2S+5A |
| | | DBL | 1 | 23 | 5 | 4A |
| $\neq 2$ | not unified, new | ADD | - | 41 | - | - |
| | | DBL | - | 41 | - | - |
| | unified | ADD | - | 41 | - | additions and negations |
| | | DBL | - | 41 | - | additions and negations |

As a matter of fact $L_j$ is $k$ shifted (with zero fill) towards the least significant bits by $j$ binary positions. In this notation one can prove the following simple

**Lemma 1** (Properties of $L_j$ and $H_j$)**.**

$$\begin{array}{l} \textit{(1)}\ L_j = 2L_{j+1} + k_j, \\ \textit{(2)}\ L_j = L_{j+1} + H_{j+1} + k_j - 1, \\ \textit{(3)}\ L_j = 2H_{j+1} + k_j - 2. \end{array} \tag{3.3}$$

*Proof.* These simple properties can be easily proved:

1. $2L_{j+1} + k_j = 2 \cdot \sum_{i=j+1}^{t-1} k_i 2^{i-j-1} + k_j = \sum_{i=j+1}^{t-1} k_i 2^{i-j} + k_j 2^0 = \sum_{i=j}^{t-1} k_i 2^{i-j} = L_j$.
2. From $H_j = L_j + 1$, $L_j = H_j - 1$, and (1) it follows that $L_j = L_{j+1} + L_{j+1} + k_j = L_{j+1} + H_{j+1} + k_j - 1$.
3. Follows directly from (2) by substituting $L_{j+1}$ for $H_{j+1} - 1$.

□

The following constructive statement follows from Lemma 1:

**Lemma 2** (Montgomery ladder)**.**

$$(L_j, H_j) = \begin{cases} (2L_{j+1}, & L_{j+1} + H_{j+1}) & \textit{if } k_j = 0, \\ (L_{j+1} + H_{j+1}, & 2H_{j+1}) & \textit{if } k_j = 1, \end{cases} \tag{3.4}$$

*and, hence:*

$$(L_j g, H_j g) = \begin{cases} ((2L_{j+1})g, & (L_{j+1} + H_{j+1})g) & \textit{if } k_j = 0, \\ ((L_{j+1} + H_{j+1})g, & (2H_{j+1})g) & \textit{if } k_j = 1. \end{cases} \tag{3.5}$$

The observation that $L_0 = k$ and Lemma 2 give rise to an elegant algorithm for evaluating $k \cdot g$ which uses only 2 registers $\beta_0$ and $\beta_1$ holding group elements, runs homogenously with respect to different values of $k$ of the same length, requires no precomputations and often offers significant performance speedup as compared to the 'always-add-and-double' Algorithm 3.1. Here two representations of the Montgomery ladder are given: The graphic (Fig. 3.3) and algorithmic (Algorithm 3.3) ones.

This representation of scalar multiplication requires in general $t$ group additions and $t$ group doublings. But in some cases the Montgomery ladder can reveal itself more efficient by observing that:

- $\beta_1 - \beta_0 = \alpha = const$ throughout the algorithm;
- At each iteration the operations (D and A) are independent and can be performed in parallel;
- At each iteration, the operations (D and A) share a common operand which can be of advantage too.

For the realization some of these techniques over elliptic curves see [53], [15] and [27]. The Montgomery ladder method is free from the usual flaws mentioned at the beginning of this subsection while protecting against simple SCAs. If the applied group allows efficient Montgomery arithmetic, then the Montgomery ladder seems to be the scalar multiplication method of choice in general. But for some cryptographically interesting groups such as $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of a genus 2 hyperelliptic curve this representation in an efficient form is relatively difficult to get. In the next section we define the properties of an efficient Montgomery representation more formally and try to take some approaches to get it for the group mentioned.

Table 3.3: Montgomery ladder in arbitrary groups

| |
|---|
| INPUT: $\alpha \in G$, $k = (k_{l-1} \dots k_0)_2 \in \{1, 2, \dots, n-1\}$ |
| 1. $\beta_0 \leftarrow 1$, $\beta_1 \leftarrow \alpha$<br>2. for $j$ from $l-1$ downto 0 do<br>if $k_j = 0$ then $\beta_1 \leftarrow \beta_1 + \beta_0$, $\beta_0 \leftarrow 2\beta_0$<br>else [if $k_j = 1$] $\beta_0 \leftarrow \beta_1 + \beta_0$, $\beta_1 \leftarrow 2\beta_1$ |
| OUTPUT: $\beta_0 = k\alpha$ |

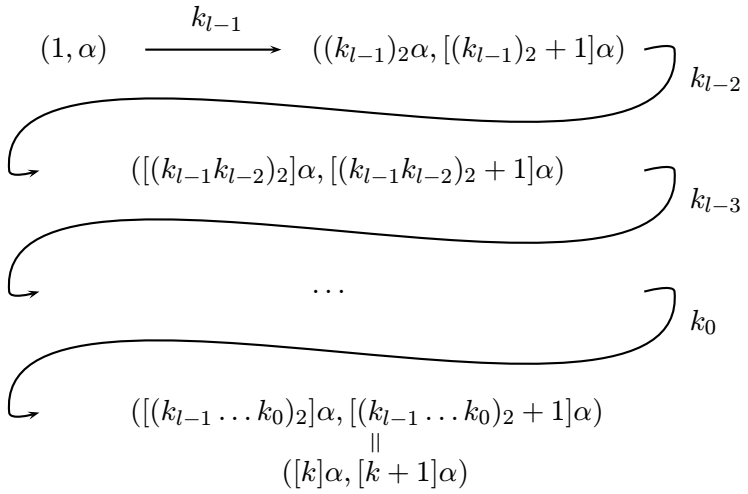


Figure 3.3: Montgomery ladder in arbitrary additive groups. Graphic representation.

## 3.3 Approaches to the Montgomery arithmetic over genus 2 hyperelliptic curves

As one can see from Subsection 3.2.2, the main idea behind the Montgomery arithmetic in any concrete group is to find such a new representation of the elements of the group (we call this the *Montgomery representation*) that the additional information contained in the difference of two elements can speed up the computation of the sum of these two elements in this representation. The necessary precondition is the possibility to efficiently double a group element in the new Montgomery representation. Another requirement to the Montgomery representation is the possibility to efficiently convert between the ordinary representation and the Montgomery representation using the additional information in the difference of the two elements [3].

Now these requirements are formulated more formally as applied to the degree 0 part $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of $\mathrm{Pic}_{\mathbb{F}_q}(C)$ for a genus 2 hyperelliptic curve. Let $D_1, D_2 \in \mathrm{Pic}^0_{\mathbb{F}_q}(C)$ be two reduced divisors $D_1 = [u_1, v_1] \in (\mathbb{F}_q)^4$ and $D_2 = [u_2, v_2] \in (\mathbb{F}_q)^4$ (each of such divisors can be represented through 4 base field elements, see Subsection 2.1.2). One is looking for:

- A Montgomery representation space $M$;
- An invertible mapping $\varphi : \mathrm{Pic}^0_C(\mathbb{F}_q) \to M$;
- A mapping $\psi_A : M \times M \to M$ representing addition;
- A mapping $\psi_D : M \to M$ representing doubling;

with a further property which can be formulated as follows. Let $L_k$:

$$\begin{array}{l} L_k : M \to M, \\ L_k : m \mapsto k \cdot m, \\ \text{where } m \in M, k \in \{1, \ldots, n\}, \end{array} \tag{3.6}$$

be a Montgomery ladder representation using $\psi_A$ and $\psi_D$ in $M$. Then it is required that $M$,$\varphi$, $\psi_A$ and $\psi_D$ should be selected with the following property fulfilled:

$$\begin{array}{l} \forall\ D \in \mathrm{Pic}^0_C(\mathbb{F}_q) \text{ and } \forall\ k \in \{1, \ldots, n\} : \\ \varphi^{-1}(L_k(\varphi(D))) = kD. \end{array} \tag{3.7}$$

If this requirement is fulfilled, then one can speak about the well-defined Montgomery representation $(M, \psi_A, \psi_D, \varphi)$.

### 3.3.1 Imitation of the Montgomery ladder for elliptic curves

For the elliptic curve arithmetic such a Montgomery representation can be easily obtained by refusing of the usage of the $y$-coordinate of a rational point. This can be easily done over both large prime fields [15] and binary fields of arbitrary extension degree [53]. Some parallel techniques for the Infineon cryptographical co-processor Crypto 2000 are introduced in [27]. For elliptic curves this technique can be combined with inversion-free coordinates resulting in an additional randomization of the base point which can hedge an implementation against differential attacks.

In [48] an approach to the Montgomery arithmetic over genus 2 hyperelliptic curves defined over finite fields of odd characteristic is given. There the proof is provided that in order to

[3] One could, however, build a crypto system directly in the Montgomery representation space. In this case there can be no need to define the map to the Montgomery representation space.

compute the $u$-coordinate of $D_1 + D_2$ (each of degree 2) one does not need the $v$-coordinates of $D_1$ and $D_2$ if the difference $D_1 - D_2$ is known. The efficiency of the explicit formulae could not be determined exactly. This approach to addition is generic in the sense that it does not require any specific curve equation and can be applied to all genus 2 curves over fields of odd characteristic. But it seems to be impossible to directly apply the approach in [48]. The doubling of an element is not discussed there in detail and the consideration is limited to the note that the doubling without $v$-coordinate can be in principle done using Cantor's analogue of division polynomials for hyperelliptic curves. Here the author shows that some information about the $v$-coordinate should nevertheless be computed at every step in order for the group doubling to be possible at all.

Let $D = [u, v]$ be a reduced divisor of degree 2 ($\deg(u) = 2$) from $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$. The goal is to compute $u'$ uniquely as a function of the coefficients of the $u$-coordinate $u(x)$ of $D$ and the curve equation, where $2D = [u', v']$. The reference to Cantor's division polynomials delivers no proof of the existence of such a function $\phi$. In [17] Cantor notes that his work provides the division polynomials for the case of $\deg(u) = 1$ only. [17], page 93: "We shall limit ourselves to the case where $D = (x, y) - \infty$, or, equivalently, ... to computing $r \cdot (x, y)$ ... However, if $D = \sum_i m_i(P_i - \infty)$, then we can form, using the formulas of this paper, the reduced element of $\mathbb{J}$ representing $rm_i(P_i - \infty)$ for each $i$, and then add these results using the methods in Cantor [7]."[4] Thus, for the application of these polynomials to a degree 2 reduced divisor one is referred to the Cantor algorithm to perform the group law in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ (see Subsection 2.1.2). But in the algorithm the $u$-coordinate of the output is made dependent on the $v$-coordinate of the input because of the reduction step (step 1 of Algorithm 2). Moreover, the author is not aware of any concrete applications of Cantor's division polynomials of genus $g$ curves to the even characteristic of the base field. Hence, one cannot refer to Cantor's division polynomials to show that doubling in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ requires no $v$-coordinate.

To illustrate the dependence of $u'$ on $v$ it suffices to find such a pair of divisors $D_1 = [u_1, v_1]$ and $D_2 = [u_2, v_2]$ with $u_1 = u_2 = u$ and $v_1 \neq v_2$ for a hyperelliptic curve $C$ over $\mathbb{F}_q$ that $u'_1 \neq u'_2$, where $D'_1 = 2D_1 = [u'_1, v'_1]$ and $D'_2 = 2D_2 = [u'_2, v'_2]$. Using the computer algebra system MAGMA the author found a number of curves over which this inequality holds for some degree 2 reduced divisors in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$. Several negative examples of this kind are given in Table 3.4.

Thus, if one wants to pursue this approach, it is necessary to compute some additional values in order to be able to double a group element.

### 3.3.2 Special choice degree 2 divisors in $\mathbf{Pic}^0_{\mathbb{F}_q}(C)$

By testing different curves over different fields of odd and even characteristics is was found out that there do exist genus 2 hyperelliptic curves of which $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ contains no other degree 2 reduced divisors except for those that pass the test described in Subsection 3.3.1. In Table 3.5 one can find some examples of such curves defined over small finite fields. As a matter of fact it turned out that this phenomenon is of purely combinatorial nature. The following consideration is limited to the case of genus 2 hyperelliptic curves over binary fields of odd expansion degree with $\deg(h) = 1$ and a method to choose such a group $\mathrm{Pic}^0_{\mathbb{F}_{2^d}}(C)$ that one does not have to know $v(D)$ to be able to compute $u(2D)$ in it. The positive examples found are all of this type.

Consider a genus 2 hyperelliptic curve of the following form:

$$y^2 + h_1 xy = x^5 + f_3 x^3 + f_2 x^2 + f_1 x \text{ over } \mathrm{GF}(2^n). \tag{3.8}$$

[4]The reference '[7]' in paper [17] cited here is paper [16] in our notation.

In accordance with the explicit formulae for this case the only dependence on the $v$-coordinate is the computation of

$$w_1 = \frac{u_0}{f_0 + v_0^2} = \frac{u_0^2}{v_0^2}. \tag{3.9}$$

To pass the test all the degree 2 elements of $\mathrm{Pic}^0_{\mathbb{F}_{2^n}}(C)$ must be divided into classes as shown in Table 3.6.

The degree 2 reduced divisors are divided into $t$ classes according to the $u$-part. For each $u$-coordinate there are $k_i$, $i = 1, \ldots, t$ corresponding degree 1 $v$-polynomials. We have the following property of this division:

$$\begin{array}{l} \forall D,\ \deg(D) = 2 :\ D \in M_1 \cup \cdots \cup M_t, \\ M_i \cap M_j = \emptyset,\ i \neq j. \end{array} \tag{3.10}$$

For the test to be passed one needs:

$$\forall D \in M_i :\ u_0^{(i)} = c^{(i)} v^{(i,j)},\ i = 1, \ldots, t,\ j = 1, \ldots, k_i. \tag{3.11}$$

That is, the the value of $w_1$ in the explicit formula for addition is constant in each class of reduced divisors. For the doubling operation to be completely independent of the value of the $v$-coordinate (at least from the computational point of view for a particular hyperelliptic curve of form 3.8) we should demand that:

$$\forall D \in M_i :\ u_0^{(i)} = c v^{(i,j)},\ i = 1, \ldots, t,\ j = 1, \ldots, k_i,\ c = \mathrm{const}, \tag{3.12}$$

thus, making the value of $w_1$ constant on the set of degree 2 reduced divisors. In fact such requirements (of purely combinatorial art) can be imposed for every genus 2 hyperelliptic curve. Such curves are relatively easy to find over small finite fields (and there are such curves over small fields which has been proved through numerous experiment using the computer algebra system MAGMA). However, the author conjectures that such curves over large fields of cryptographical interest are extremely difficult to find (due to the uselessness of the total test method in this case) if there are any. Note that it seems to be possible to add two divisors of degree 2 on a generic genus 2 hyperelliptic curve following the strategy outlined in [48] without exact knowledge of the $v$-coordinate (this gives rise to the function $\psi_A$). Performing doubling with the method outlined here it is theoretically possible to get rid of the $v$-coordinate while computing in the Montgomery representation space $M$ (this defines the function $\psi_D$). The function $\varphi$ mapping from $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ to the Montgomery representation space $M$ can be easily defined by discarding the $v$-coordinate, however, it is not clear whether $\varphi$ is invertible in all cases. Thus, in the case of a special form of a genus 2 hyperelliptic curve one could find a full Montgomery representation $(M, \psi_A, \psi_D, \varphi)$, but this approach which uses a special choice of $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ seems to be quite unpractical.

### 3.3.3 Map to the Kummer surface

In [24] the Montgomery method for scalar multiplication is generalized to the Jacobian of genus 2 hyperelliptic curves using a constructive approach to building the Kummer surface given in [18]. It is proposed to identify a point on the Jacobian and its opposite. This possibility is provided by the corresponding map to the Kummer surface where such identification is natural. The map to the Kummer surface has been constructively considered for genus 2 hyperelliptic curves over $\mathbb{F}_q$ with $\mathrm{char}(\mathbb{F}_q) > 5$ only. Because of this the consideration in [24] is limited to prime fields $\mathbb{F}_p$ with $p \geq 7$.

Table 3.4: Doubling in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ with $v$-coordinate only, negative examples

| Curve equation | Ground fields |
|---|---|
| $y^2 + xy = x^5 + x^3 + x^2 + x$ | $\mathrm{GF}(2^m)$, $m = 1, 2, \ldots$ |
| $y^2 + (x^2 + x)y = x^5 + x^3 + x^2 + x$ | $\mathrm{GF}(2^m)$, $m = 1, 2, \ldots$ |
| $y^2 = x^5 + x^2 + x$ | $\mathrm{GF}(13)$, $\mathrm{GF}(17)$ |

Table 3.5: Doubling in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ with $v$-coordinate only, positive examples

| Curve equation | Ground fields |
|---|---|
| $y^2 + xy = x^5 + x^2 + x$ | $\mathrm{GF}(2^3)$ |
| $y^2 = x^5 + x^2 + x$ | $\mathrm{GF}(3)$, $\mathrm{GF}(5)$, $\mathrm{GF}(7)$ |
| $y^2 = x^5 + x^2 + 2x$ | $\mathrm{GF}(5)$ |
| $y^2 = x^5 + 5x^4 + x^2 + x$ | $\mathrm{GF}(5)$, $\mathrm{GF}(3)$ |
| $y^2 = x^5 + 5x^4 + 6x^3 + x^2 + x$ | $\mathrm{GF}(3)$, $\mathrm{GF}(7)$ |

Table 3.6: Classes of reduced degree 2 divisors in $\mathrm{Pic}^0_{\mathbb{F}_q}$ of a genus 2 hyperelliptic curve

| $M_1$ | $M_2$ | $\ldots$ | $M_t$ |
|---|---|---|---|
| $x^2 + u_1^{(1)}x + u_0^{(1)}$: | $x^2 + u_1^{(2)}x + u_0^{(2)}$: | $\ldots$ | $x^2 + u_1^{(t)}x + u_0^{(t)}$: |
| $v_1^{(1,1)}x + v_0^{(1,1)}$, | $v_1^{(2,1)}x + v_0^{(2,1)}$, | $\ldots$ | $v_1^{(t,1)}x + v_0^{(t,1)}$, |
| $v_1^{(1,2)}x + v_0^{(1,2)}$, | $v_1^{(2,2)}x + v_0^{(2,2)}$, | $\ldots$ | $v_1^{(t,2)}x + v_0^{(t,2)}$, |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| $v_1^{(1,k_1)}x + v_0^{(1,k_1)}$. | $v_1^{(2,k_2)}x + v_0^{(2,k_2)}$. | $\ldots$ | $v_1^{(t,k_t)}x + v_0^{(t,k_t)}$. |

In [24] the curves of so-called Montgomery form are considered only. A curve over $\mathbb{F}_p$ is transformable into Montgomery-like form if it is isomorphic to a curve given by an equation of the type:

$$By^2 = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + x. \tag{3.13}$$

A curve is transformable into the Montgomery-like form iff:

- $f(x)$ has at least one root $\alpha$ in the base field $\mathbb{F}_p$;
- the number $f'(\alpha)$ is a 4th power in the field $\mathbb{F}_p$.

The Kummer surface is a quartic surface in $\mathbb{P}^3$. Let $\mathbb{S}$ be the subset of $\mathbb{J}_C(\mathbb{F}_p) \cong \mathrm{Pic}^0_{\mathbb{F}_p}(C)$ consisting of reduced divisors with exactly 2 different points in its support. For a Montgomery-like hyperelliptic curve the Kummer surface is the image of the map

$$\begin{array}{l}
\varphi : \mathbb{S} \to \mathbb{P}^3(\mathbb{F}_p), \\
\varphi : D = P_1 + P_2 - 2P_\infty \mapsto (1, x_1 + x_2, x_1x_2, \frac{F_0(x_1,x_2)-2By_1y_2}{(x_1-x_2)^2}), \\
\text{where} \\
P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in \overline{\mathbb{F}}_p, \\
F_0(x_1, x_2) = (x_1 + x_2) + 2f_2x_1x_2 + f_3(x_1 + x_2)x_1x_2 + 2f_4x_1^2x_2^2 + (x_1 + x_2)x_1^2x_2^2.
\end{array} \tag{3.14}$$

Thus, one could refer to the image of a reduced divisor $D \in \mathbb{J}_C(\mathbb{F}_p)$ on the Kummer surface as:

$$\varphi(D) = (k_1(D), k_2(D), k_3(D), k_4(D)). \tag{3.15}$$

For the exact definition of the Kummer surface for the Jacobian of a genus 2 hyperelliptic curve see [18], where the defining homogenous quartic polynomial $\phi$ of degree four in the first three variables and of degree two in the last one is given. That is, the Kummer surface is the projective locus given by the equation $\phi = 0$.

Although the Kummer surface loses the group structure, it turned out to be possible to double there directly and to compute $\varphi(D_1+D_2)$ on the basis of $\varphi(D_1)$, $\varphi(D_2)$ and $\varphi(D_1-D_2)$, $D_1, D_2 \in \mathbb{J}_C(\mathbb{F}_p)$. S. Duquesne [24] provides a precise breakdown to the explicit formulae to perform the group law in this Montgomery representation using the Kummer surface for the Jacobian of a Montgomery-like genus 2 hyperelliptic curve over $\mathbb{F}_p$. The operation count taken from [24] is provided in Table 3.7. Note that the arithmetic is naturally inversion-free. The complexity of the corresponding Montgomery ladder scalar multiplication is considerably faster than the 'always-add-and-double' method. Moreover, some the diversity-speed trade-off can drastically increase the performance of the Montgomery ladder in this case.

The only drawback of this method is that it is limited to $\mathbb{F}_p$ and a special form of curve due to efficiency considerations. Theoretically, however, there is no restrictions and the method can be applied to all genus 2 hyperelliptic curves, but the constructive basis for characteristic 2 fields has not been elaborated precisely yet. This is a very promising research area, since genus 2 hyperelliptic curves over finite fields of even characteristic offer performance sometimes exceeding that for elliptic curves, and it would be interesting to see whether it could be also the case for the Montgomery arithmetic through the map to the Kummer surface.

Table 3.7: Efficiency of the arithmetic in the Kummer surface, $D_1, D_2 \in \mathbb{J}_C(\mathbb{F}_p)$, $D_1 - D_2$ known, Montgomery-like genus 2 hyperelliptic curve over $\mathbb{F}_p$, field multiplications (M) and squarings (S), [24]

| Operation | Precomputations | Complexity |
|---|---|---|
| $\varphi(D_1 + D_2)$ | 16M | 31M + 2S |
| $\varphi(2D_1)$ | 6M+4S | 31M + 5S |

# Conclusion

Having been proposed almost simultaneously with elliptic curves [62], [37], the divisor class groups $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of hyperelliptic curves [38] have not been widely applied in the real-world security solutions yet, although offering the best possible (for groups) exponential security level and providing approximately the same or sometimes even higher performance on contemporary computer platforms than elliptic curves.

In this thesis an introduction to the arithmetic over hyperelliptic curves was given and the ideal class group (isomorphic to $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$) of a hyperelliptic curve was defined. Then a number of currently known ways to efficiently perform the group law in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ for genus 2 hyperelliptic curves were considered and some minor inexactness in the explicit formulae for addition and doubling was removed by the author. Moreover, the author improved the complexity of a point (de)compression method for a special subclass of genus 2 hyperelliptic curves over $\mathrm{GF}(2^n)$.

Although the task of finding efficient ways to implement arithmetic in $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of a genus 2 curve seems to be solved (some improvements for special forms of curves being expected), the problem of efficiently securing the arithmetic against side-channel attacks is still open. Some ways of doing it were proposed in the literature. The main drawback of the existing techniques in this area is the possibility to get the Hamming weight of the scalar multiplier through simple SCAs or to filter out dummy operations through the application of fault-type attacks. A way which is free from these flaws is the usage of the Montgomery ladder. But here one has to search for some computational speed-up compared to the 'always-add-and-double' method. Few publications on this topic are known only. The approaches proposed are either incomplete [24] (although delivering good diversity and performance properties) or cannot be directly applied in their original form [48] (however, giving rise to the generic Montgomery arithmetic). This was shown in the last chapter of the thesis. One of the most efficient approaches to the Montgomery ladder for genus 2 hyperelliptic curves the author is currently aware of is that using the map from $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ to the Kummer surface [18], [24] or, alternatively, to perform the arithmetic directly on the Kummer surface. But it is connected with some computational difficulties which the author hopes to be able to overcome in his future work.

# Bibliography


[1] L. Adleman, J. DeMarrais, and M. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In *Algorithmic Number Theory*, volume 877 of *LNCS*, pages 28–40. Springer-Verlag, 1994.

[2] M. Atiyah and I. Macdonald. *Introduction to Commutative Algebra*. Addison-Wesley, 1969.

[3] C. Aumueller, P. Brier, W. Fischer, P. Hofreiter, and J.-P-Seifert. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In *CHES 2002*, LNCS, pages 261–276. Springer-Verlag, 2002.

[4] R. Avanzi. On multi-exponentiation in cryptography. Cryptology ePrint Archive, Report 2002/154, 2002.

[5] R. Avanzi. Aspects of hyperelliptic curves over large prime fields in software implementations. Cryptology ePrint Archive, Report 2003/253, December 2003.

[6] R. Avanzi. A note on the signed sliding window integer recording and a left-to-right analogue. In H. Handschuh and M.A. Hasan, editors, *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada*, LNCS. Springer-Verlag, 2004.

[7] R. Avanzi. Side channel attacks on implementations of curve-based cryptographic primitives. Cryptology ePrint Archive, Report 2005/017, January 2005.

[8] R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC, 2005. to appear.

[9] L. Benini, A. Macii, E. Macii, E. Omerbergovic, M. Poncino, and F. Pro. Energy-aware design techniques for differential power analysis protection. In *DAC-40 - ACM/IEEE Design Automation Conference*, pages 36–41. ACM, 2003.

[10] G. Bertoni, L. Breveglieri, T. Wollinger, and C. Paar. Finding optimum parallel coprocessor design for genus 2 hyperelliptic curve cryptosystems. Cryptology ePrint Archive, Report 2004/029, 2004.

[11] I. Biehl, B. Meyer, and V. Mueller. Differential fault attacks on elliptic curve cryptosystems. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 131–146. Springer-Verlag, 2000.

[12] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Crypto'97*, volume 1294 of *LNCS*, pages 513–525. Springer-Verlag, 1998.

[13] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[14] A. Bogdanov and I. Kizhvatov. Quantum algorithms and their impact on the security of classical cryptographical systems. In Russian. Available from `www.cryptography.ru:8200/pubd/2005/05/12/0001169858/bogdanov_kizhvatov.pdf`, February 2005.

[15] E. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In D. Naccache and P. Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 335–345. Springer-Verlag, 2002.

[16] D. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of Computation*, 48:95–101, 1987.

[17] D. Cantor. On the analogue of the division polynomials for hyperelliptic curves. *Journal fuer reine und angewandte Mathematik*, 447:91–145, 1994.

[18] J. W. S. Cassels and E. V. Flynn. *Prolegomena to a middlebrow arithmetic of curves of genus 2.* Number 230 in London Mathematical Society Lecture Notes Series. Cambridge University Press, 1996.

[19] E. Cesena. Varieta a traccia zero su campi binari applicazioni crittografiche. Master's thesis, Universita' Degli Studi di Milano, 2004.

[20] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. Cryptology ePrint Archive, Report 2003/237, 2003.

[21] H. Cohen. Analysis of the flexible window powering algorithm. Available from `http://www.math.u-bordeaux.fr/~cohen/window.dvi`.

[22] J.-S. Coron. Resistance against power analysis for elliptic curve cryptosystems. In *CHES'99*, volume 1717 of *LNCS*, pages 292–302. Springer-Verlag, 1999.

[23] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transiations on Information Theory*, 22:644–654, 1976.

[24] S. Duquesne. Montgomery scalar multiplication for genus 2 curves. In D.A. Buell, editor, *ANTS 2004*, volume 3076 of *LNCS*, pages 153–168. Springer-Verlag, 2004.

[25] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[26] A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, 102:83–103, 2002.

[27] W. Fischer, C. Giraud, E. Knudsen, and J. Seifert. Parallel scalar multiplication on general ellipitic curves over $\mathbb{F}_p$ hedged against non-differential side-channel attacks. Cryptology ePrint Archive, Report 2002/007, 2002.

[28] G. Frey. Applications of arithmetical geometry to cryptographic constructions. In *Finite fields and applications (Augsburg 1999)*, pages 128–161. Springer-Verlag, 2001.

[29] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In *Andvances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 19–37. Springer-Verlag, 2000.

[30] D. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27(1):129–146, 1998.

[31] L. Goubin. A refined power analysis attack on elliptic curve cryptosystems. In *Public Key Cryptography - PKC 2003*, volume 2567 of *LNCS*, pages 199–210. Springer-Verlag, 2003.

[32] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag, 2004.

[33] F. Hess, G. Seroussi, and N. Smart. Two topics in hyperelliptic cryptography. In *Selected Areas in Cryptography - SAC 2001*, volume 2259 of *LNCS*, pages 181–189. Springer-Verlag, 2001.

[34] Y. Hitchcock and P. Montague. A new elliptic curve scalar multiplication algorithm to resist simple power analysis. In *Information Security and Privacy*, volume 2384 of *LNCS*, pages 214–225. Springer-Verlag, 2002.

[35] M. Joye and S.-M. Yen. The Mongomery powering ladder. In B.S. Kaliski Jr. et al. (Eds.), editor, *CHES 2002, LNCS 2523*, pages 291–302. Springer-Verlag, 2003.

[36] P. Kaye and C. Zalka. Optimized quantum implementation of elliptic curve arithmetic over binary fields. Available from `arXiv.org/quant-ph/0407095`, July 2004.

[37] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[38] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.

[39] N. Koblitz, A. Menezes, Y.-H. Wu, and R. Zuccherato. *Algebraic Aspects of Cryptography*, chapter An Elementary Introduction to Hyperelliptic Curves. Springer, 1999.

[40] P. Kocher. Timings attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In *Advances in Cryptology - Crypto 1996*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.

[41] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology - Crypto 1999*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.

[42] O. Koemmerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the Usenix Workshop on Smartcard Technology, Chicago, 10-11 May*, pages 9–20, 1999.

[43] E. Kunz. *Introduction to Commutative Algebra and Algebraic Geometry*. Birkhaeuser, 1985.

[44] T. Lange. Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. Cryptology ePrint Archive, Report 2002/121, 2002.

[45] T. Lange. Inversion-free coordinates on genus 2 hypereliptic curves. Cryptology ePrint Archive, Report 2002/223, 2002.

[46] T. Lange. Weighted coordinates on genus 2 hyperelliptic curves. Cryptology ePrint Archive, Report 2002/153, 2002.

[47] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Available from* `www2.mat.dtu.dk/people/T.Lange/preprints.html`, May 2004.

[48] T. Lange. Montgomery addition for genus two curves. In D.A. Buell, editor, *ANTS 2004*, volume 3076 of *LNCS*, pages 309–317. Springer-Verlag, 2004.

[49] T. Lange. Private communication. Ruhr University Bochum, April 2005.

[50] T. Lange and P. Mishra. SCA resistant parallel explicit formula for addition and doubling of divisors in the Jacobian of hyperellipic curves of genus 2. To appear, 2005.

[51] T. Lange and M. Stevens. Efficient doubling for genus two curves over binary fields. In H. Handschuh and M.A. Hasan, editors, *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada*, LNCS, pages 189–202. Springer-Verlag, 2004.

[52] A. Lenstra and Jr. (Eds.) H. Lenstra. *The developement of the number field sieve*. Springer-Verlag, 1993.

[53] J. Lopez and R. Dahab. Fast multiplication on elliptic curves over $GF(2^n)$ without precomputation. In *CHES'99*, volume 1717 of *LNCS*, pages 316–327. Springer-Verlag, 1999.

[54] D. Lorenzini. *An Invitation to Arithmetic Geometry*. American Mathematical Society, 1996.

[55] D. V. Matuhin. About the asymptotical complexity of the discrete logarithm in the field gf(p). *Discrete mathematics*, 15(1):28–49, 2002. In Russian.

[56] D. V. Matuhin and N. N. Murashov. A modification of the number field sieve method for the discrete logarithm in the field gf(p). *Survey of applied and industrial mathematics*, 7(2):387–389, 2000. In Russian.

[57] D. May, H. Muller, and N. Smart. Non-deterministic processors. In *Information Security and Privacy, 6th Australasian Conference - ACISP 2001*, volume 2119 of *LNCS*, pages 115–129. Springer-Verlag, 2001.

[58] A. Menezes, P. van Oorshot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[59] T. Messerges. Using second order power analysis to attak DPA resistant software. In *CHES 2000*, volume 1965 of *LNCS*, pages 238–251. Springer-Verlag, 2000.

[60] T. Messerges, E. Dabbish, and R. Sloan. Inverstigations of power analysis attacks on smart cards. In *CHES'99*, volume 1717 of *LNCS*. Springer-Verlag, 1999.

[61] T. Messerges, Dabbish E, and R. Sloan. Power analysis attacks of modular exponentiation in smart cards. In *Usenix*, 1999.

[62] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology –Crypto '85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.

[63] P. Mishra. Pipelined computation of scalar multiplication in elliptic curve cryptosystems. In *CHES 2004*, LNCS. Springer-Verlag, 2004.

[64] P. Mishra. Scalar multiplication in elliptic curve cryptosystems: Pipelining with Precomputations. Cryptology ePrint Archive, Report 2004/191, 2004.

[65] P. Montgomery. Speeding up the pollard and elliptic curve methods of factorization. *Math. Comp.*, 48(177):243–264, 1987.

[66] J. Muir and D. Stinson. Minimality and other properties of the width-$w$ nonadjacent form. Technical report, Centre for Applied Cryptographic Research (CACR), the University of Waterloo, August 2004. Avaliable at `www.cacr.math.uwaterloo.ca/techreports/2004/corr2004-08.pdf`.

[67] D. Mumford. *Tata Lectures on Theta II.* Birkhauser, 1984.

[68] V. Nechaev. Complexity of a determinate algorithm for the discrete logarithm problem. *Mathematical Notes*, 55:165–172, 1994.

[69] K. Okeya, K. Schmidt-Simon, C. Spahn, and T. Takagi. Signed binary representations revised. Cryptology ePrint Archive, Report 2004/195, 2004.

[70] J. Proos and C. Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. Available from `arXiv.org/quant-ph/0301141`, 2004.

[71] G. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

[72] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols.* Addison Wesley Professional, 2001.

[73] A. Shamir. Protecting smart cards from passive power analysis with detached power supplies. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2000*, volume 1965 of *LNCS*, pages 71–77. Springer-Verlag, 2000.

[74] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28-4:656–715, 1949.

[75] P. Shor. Algorithms for quantum computer: Discrete logarithms and factoring. In *35th IEEE Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994.

[76] V. Shoup. Lower bounds for discrete logarithm and related problems. In *Advances in Cryptology - EUROCRYPT '97*, volume 1233, pages 256–266. Springer-Verlag, 1997.

[77] J. Silverman. *The Arithmetic of Elliptic Curves.* Springer-Verlag, 1986.

[78] C. Stahlke. Point compression on Jacobians of hyperelliptic curves over $\mathbb{F}_q^*$. Cryptology ePrint Archive, Report 2004/030, 2004.

[79] E. Teske. Square-root algorithms for the discrete logarithm problem (A Survey). Technical report, University of Waterloo, 2001.

[80] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In *ASIACRYPT 2003*, 2003.

[81] K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *28th European Solid-State Circuits Conference (ESSCIRC 2002)*, 2002.

[82] TOP500. Supercomputer sites. TOP500 List. Available from `http://www.top500.org/lists/2004/11/`, November 2004.

[83] C. Vuillaume. Side channel attacks on elliptic curve cryptosystems. Master's thesis, Technische Universitaet Darmstadt, Fachgebiet Informatik, Fachbereich Kryptographische Protokolle, 2004.