MASTER'S THESIS IN MATHEMATICS

The Use of Elliptic Curves in Cryptography

Tibor Juhas

May, 2007

FACULTY OF SCIENCE
Department of Mathematics
University of Tromsø

Contents

1 Introduction to cryptography

1.1 The objectives of cryptography

Suppose that someone wants to send a message either by letter or electronically to a receiver, and wants to be sure that no-one else can read the message. However, there is the possibility that someone else opens the letter or reads the electronic communication. The solution to this problem is *cryptography*. Cryptography enables us to store sensitive information or transmit it across insecure networks, like the Internet, so that it cannot be read by anyone else except the intended recipient. Cryptography is the science of using mathematics to encrypt and decrypt messages. In cryptographic terminology, the original, undisguised message is called *plain text* or *cleartext*. Encoding the contents of the message in such a way that it hides its contents from outsiders is called *encryption*. The encrypted message is called *ciphertext*. The process of retrieving the plaintext from the ciphertext is called *decryption*. Modern cryptography, as applied in the commercial world, is concerned with a number of problems. The most important of these are:

1) *Confidentiality*, which is the process of keeping information private and secret so that only the intended recipient is able to understand it.

2) *Authentication*, which is the process of providing proof of identity of the sender to the recipient, so that the recipient can be assured that the person sending the information is who or what he or she claims to be.

3) *Integrity*, which is the method to ensure that information is not tempered with during its transit or its storage on the network.

4) *Non-repudiation*, which is the method to ensure that information cannot be disowned. Once the non-repudiation process is in place, the sender cannot deny being the originator of the information.

1.2 Symmetric key algorithms

Both encryption and decryption make use of a *key* and are parts of a cryptographic *algorithm* or *system*. There are two classes of key based algorithms, *symmetric* or *secret-key* and *asymmetric* or *public key* algorithms. The difference is that symmetric algorithms use the same key for both encryption and decryption (or the decryption key is easily derived from the encryption key), whereas the asymmetric algorithms use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key. Symmetric algorithms can be divided into *stream ciphers* and *block ciphers*. Stream ciphers encrypt a single bit of plaintext at a time, whereas block ciphers take a number of bits (typically 64 bits in modern ciphers) and encrypt them as a single unit. The most studied and probably the most widely spread symmetric cipher is DES or Data Encryption Standard. Because of the increase in the computing power of computers, the basic version of DES cannot be considered sufficiently safe anymore. Therefore a new, more powerful cipher called AES or Advanced Encryption Standard was standardized in 2001. Other popular and respected algorithms include Twofish, Serpent, CAST5, RC4, TDES and IDEA.

The main problem with symmetric key algorithms is that since the sender and the receiver have to agree on a common key, a secure channel is required between them in order to exchange the key. Transferring the key over the Internet either in an e-mail message or through simple IRC services is insecure. Verbally communicating the key over a phone line runs the risk of eavesdropping. Similarly, snail mail runs the risk of possible interception. The security risks that are involved in secret key cryptography have been, to a large extent, overcome in public key cryptography.

1.3 Public key algorithms

Public key cryptography uses a *key pair* instead of just one secret key. Of this key pair, one key, known as the *private* key, is always kept secret by the key holder and is used for decryption. The private key is not transferred to anyone and is stored securely by the holder. The key used for encryption is the *public* key and is freely distributable, for instance it can be placed on one of the many public key repositories on the Internet. Over the past 30 years, public key cryptography has become a mainstay for secure communications over the Internet and throughout many other forms of communications. It provides the foundation for both digital signatures and key management. For digital signatures, public key cryptography is used to authenticate the origin of data and protect the integrity of that data. In key management, public key cryptography is used to distribute the secret keys used in other cryptographic algorithms (e.g. DES). The technique is to use a public key algorithm to encrypt a randomly generated encryption key, and the random key is used to encrypt the actual message using a symmetric algorithm. This combined technique is used widely. It is used for Secure Shell (SSH), which is used to secure communication between a client and a server and PGP (Pretty Good Privacy) for sending messages. Above all, it is the heart of SSL (Secure Socket Layer), which is the most widely deployed and used security protocol on the Internet today. The protocol has withstood years of scrutiny by the security community and is now trusted to secure virtually all sensitive web-based applications ranging from on-line banking and stock trading to e-commerce. SSL offers encryption, source authentication and integrity protection for data exchanged over insecure, public networks. It operates above a reliable transport service like TCP and has the flexibility to accommodate different cryptographic algorithms for key agreement, encryption and hashing. However, the specification does recommend particular combinations of these algorithms, called cipher suites, which have wellunderstood security

properties. For example, a cipher suite such as RSA-RC4-SHA would indicate RSA as the key exchange mechanism, RC4 for bulk encryption, and SHA as the hashing function. Here we note that hashing function are very fast cryptographic functions that take a message of arbitrary length and produce a message digest of specified size. The two main components of SSL are the Handshake protocol and the Record Layer protocol. The Handshake protocol allows an SSL client and server to negotiate a common cipher suite, authenticate each other, and establish a shared master secret using public key cryptographic algorithms. The Record Layer derives symmetric-keys from the master secret and uses them with faster symmetric-key algorithms for bulk encryption and authentication of application data. Public key cryptographic operations are the most computationally expensive portion of SSL processing. SSL allows the re-use of a previously established master secret, resulting in an abbreviated handshake that does not involve any public key cryptography, and requires fewer and shorter messages. However, a client and server must perform a full handshake on their first interaction. Moreover, practical issues such as server load, limited session cache and naive Client authentication are optional. Only the server is typically authenticated at the SSL layer and client authentication is achieved at the application layer, e.g. through the use of passwords sent over an SSL-protected channel.

The two most important first generation public key algorithms used to secure the Internet today are known as RSA and Diffie-Hellman (DH). The security of the first is based on the difficulty of factoring the product of two large primes. The second is related to a problem known as the discrete logarithm problem for finite groups. Both are based on the use of elementary number theory. The majority of public key systems in use today use 1024-bit parameters for RSA and Diffie-Hellman. The US National Institute for Standards and Technology has recommended that these 1024-bit systems are sufficient for use until 2010. After that, NIST recommends that they be up-

graded to something providing more security. The question is what should these systems be changed to? One option is to simply increase the public key parameter size to a level appropriate for another decade of use. Another option is to take advantage of the past 30 years of public key research and analysis and move from first generation public key algorithms and on to elliptic curves. The length of a key, in bits, for a conventional encryption algorithm is a common measure of security. The following table taken from [63] gives the key sizes recommended by the National Institute of Standards and Technology to protect keys used in conventional encryption algorithms like the (DES) and (AES) together with the key sizes for RSA, Diffie-Hellman and elliptic curves that are needed to provide equivalent security:

Table 1: Comparison of key sizes

Symmetric key size (bits)	RSA and Diffie-Hellman key size (bits)	Elliptic curve key size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

To use RSA or Diffie-Hellman to protect 128-bit AES keys one should use 3072-bit parameters: three times the size in use throughout the Internet today. The equivalent key size for elliptic curves is only 256 bits. We can see that as symmetric key sizes increase the required key sizes for RSA and Diffie-Hellman increase at a much faster rate than the required key sizes for elliptic curve cryptosystems. Hence, elliptic curve systems offer more security per bit increase in key size than either RSA or Diffie-Hellman public key systems.

The mathematical problems that RSA and Diffie-Hellman owe their security are the problem of integer factorization and the discrete logarithm problem, respectively. The reason why such a large keys are necessary in

the implementation of RSA and Diffie-Hellman cryptosystems is that there are well known sub-exponential time attacks on the mathematical problems which these systems are based upon.

The mathematical problem that elliptic curve cryptosystems rely on is the discrete logarithm problem over elliptic curves or ECDLP. The reason why such short key lengths may be used in the implementation of cryptosystems based on elliptic curves is that there is no known sub-exponential time attack on the underlying mathematical problem when it is applied over a generic elliptic curve. The objective of this thesis is to try and prove this last statement.

The reminder of this thesis is organized as follows. Section 2 presents a short introduction to the parts of the theory of elliptic curves that are relevant for our work. Section 3 present an overview of attacks that are applicable for general elliptic curves. The focus will be on the in depth presentation and analysis of attacks proposed by Pollard. These are the best 'known' attacks on ECDLP over general elliptic curves. Section 4 present purpose built attacks that exploit weaknesses in special type of elliptic curves. Section 5 present an introduction to the theory of hyperelliptic curves and their application in attacking the ECDLP.

2 Introduction to elliptic curves

We will now introduce some basic facts about the elliptic curves. This introduction will describe those parts of the theory of elliptic curves which are relevant for cryptography and the definitions will be given from a cryptographic point of view. A profound treatment of the general theory of elliptic curves is given in [52] and [53].

Let $\mathbf{k}$ be a field, $\overline{\mathbf{k}}$ its algebraic closure and $\mathbf{k}^*$ its multiplicative group. The projective plane $\mathbb{P}^2(\mathbf{k})$ over $\mathbf{k}$ is the set of equivalence classes of the relation $\sim$ acting on $\mathbf{k}^3\backslash\{(0,0,0)\}$, where $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ if and only if there exists $u \in \mathbf{k}^*$ such that $x_1 = ux_2$, $y_1 = uy_2$, and $z_1 = uz_2$.

Definition 1 *An elliptic curve over* $\mathbf{k}$ *is defined as the set of solutions in the projective plane* $\mathbb{P}^2(\overline{\mathbf{k}})$ *of a homogeneous Weirstrass equation of the form*

$$E : Y^2Z+a_1XYZ+a_3YZ^2 = X^3+a_2X^2Z+a_4XZ^2+a_6Z^3$$

with $a_1, a_2, a_3, a_4, a_6 \in \mathbf{k}$. *This equation is referred to as the long Weierstrass form.* ♦

Such a curve should be *non-singular* in the sense that, if the equation is written in the form $F(X, Y, Z) = 0$, then the partial derivatives of the curve equation $\partial F/\partial X$, $\partial F/\partial Y$ and $\partial F/\partial Z$ should not vanish simultaneously at any point on the curve. If all three partial derivatives vanish at some point P, then P is called a *singular point* and the equation is said to be singular. The curve has exactly one point with Z - coordinate equal to 0, namely $(0, 1, 0)$. This point is called the point at *infinity* and is denoted by $\mathcal{O}$. For convenience reasons it is usual to write the Weierstrass equation using affine coordinates $x = X/Z$, $y = Y/Z$,

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

An elliptic curve E is then the set of solutions to this equation in the affine

plane $A^2(\overline{\mathbf{k}}) = \overline{\mathbf{k}} \times \overline{\mathbf{k}}$, together with the extra point at the infinity $\mathcal{O}$. If the coefficients of the equation are in $\mathbf{k}$, then E is said to be *defined* over $\mathbf{k}$, and this is denoted as $E/\mathbf{k}$. If E is defined over $\mathbf{k}$, then the set of $\mathbf{k}$ - *rational* points of E, denoted $E(\mathbf{k})$, is the set of points whose both coordinates lie in $\mathbf{k}$, together with the point $\mathcal{O}$.

Let E be a curve given by the affine Weierstrass equation. We define the quantities

$$
\begin{aligned}
d_2 &= a_1^2 + 4a_2 \\
d_4 &= 2a_4 + a_1a_3 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\
c_4 &= d_2^2 - 24d_4 \\
\Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\
j(E) &= c_4^3/\Delta.
\end{aligned}
$$

Definition 2 *Discriminant and $j-$invariant*

The quantity Δ is called the discriminant *of the Weierstrass equation, while $j(E)$ is called the* j-invariant *of E if $\Delta \neq 0$.* ♦

The Weierstrass equation is non-singular if and only if $\Delta \neq 0$. The $j-$invariant is closely related to the notion of elliptic curve isomorphism. Two elliptic curves, $E_1/\mathbf{k}$ and $E_2/\mathbf{k}$ that are isomorphic over $\mathbf{k}$ have the same $j-$invariant, i.e. $j(E_1) = j(E_2)$. Conversely, two curves with the same $j-$invariant are isomorphic over $\overline{\mathbf{k}}$.

Definition 3 *Point addition*

The points on an elliptic curve form an abelian group under a certain addition. The addition operation of two points $P, Q \in E(\mathbf{k})$ is defined as follows:

1. *draw a line through P and Q which intersects the curve at a point T.*

2. *draw a vertical line through T which intersects the curve at a point R and define $P + Q = R$.* ♦

If $P = Q \neq \mathcal{O}$ then the line in step 1 is the tangent line of the curve through P. Adding P to $\mathcal{O}$ means that the line drawn in step 1 is the vertical line passing through P, because $\mathcal{O}$ is infinitely far and the vertical line in step 2 is the same as the line in step 1, which intersects the curve at the same point P. This means that $P+\mathcal{O} = P$ and $\mathcal{O}+P = P$ and $\mathcal{O}$ is the identity element. The inverse of P, denoted $-P$, requires $P + (-P) = \mathcal{O}$. According to the addition rule we can find that $-P = (x_1, y_1-a_1x_1-a_3)$. The formal definition of addition in $E(\mathbf{k})$ is as follows. Suppose $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points on E other than $\mathcal{O}$. If $x_1 = x_2$ and $y_1 + y_2 + a_1x_2 + a_3 = 0$, then $P + Q = \mathcal{O}$. Otherwise $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$$

$$y_3 = -(\lambda+a_1)x_3-\nu-a_3$$

and

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q. \\ \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, & \text{if } P = Q. \end{cases}$$

$$\nu = \begin{cases} \dfrac{y_1x_2 - y_2x_1}{x_2 - x_1}, & \text{if } P \neq Q. \\ \dfrac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}, & \text{if } P = Q. \end{cases}$$

2.1 Elliptic curves over finite fields

Over a finite field $\mathbb{F}_q$, $E(\mathbb{F}_q)$ is an abelian group of rank 1 or 2. The type of the group is (n_1, n_2), i.e., $E(\mathbb{F}_q) \widetilde{=} Z_{n_1} \oplus Z_{n_2}$, where $n_2 \mid n_1$, and furthermore $n_2 \mid (q-1)$.
The number of rational points on a curve is finite, and it will be denoted by $\#E(\mathbb{F}_q)$.

Definition 4 *Trace of the Frobenius*

The quantity t, defined by

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

is called the trace *of Frobenius at q. The trace of Frobenius satisfies $\mid t \mid \leq 2q$. The Frobenius endomorphism is a map ϕ which sends (x, y) to (x^q, y^q) and fixes $\mathcal{O}$.* ♦

The problem of determining the order of the group of points on an elliptic curve over a finite field is of critical importance in cryptographic applications. This is because the best method for generating elliptic curves suitable for cryptography depends on the ability of solving this problem. There are several approaches, but the best known algorithm is due to Schoof [47]. Although the original algorithm has polynomial running time, it is inefficient in practice. It was further developed thanks to the ideas and improvements of Elkies [10] and Atkin [2].

Practical implementations of elliptic curve cryptosystems are usually based on either the field $\mathbb{F}_p$, where p is a large prime number, or $\mathbb{F}_{2^n}$, fields with characteristic 2.

2.2 Curves in fields of characteristic $p > 3$

Assume that $\mathbf{k} = \mathbb{F}_q$, where $q = p^n$ for a prime $p > 3$ and an integer $n \geq 1$. The curve equation over $\mathbf{k}$ in this case can be simplified to the short Weierstrass form

$$E : y^2 = x^3 + ax + b.$$

The discriminant of the curve then reduces to $\Delta = -16(4a^3 + 27b^2)$, and its $j-$invariant to $j(E) = -1728\ (4a)^3/\Delta$. The inverse of the point $P = (x_1, y_1)$ is now $-P = (x_1, -y_1)$. The addition rules are as follows: for the points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ the coordinates of the point $P + Q = (x_3, y_3)$, $Q \neq -P$, are given as

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = (x_1 - x_3)\lambda - y_1$$

where

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q. \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{if } P = Q. \end{cases}$$

2.3 Curves in fields of characteristic 2

We assume now that $\mathbf{k} = \mathbb{F}_q$, where $q = 2^n$, for an integer $n \geq 1$. In this case, the expression for the $j-$invariant reduces to $j(E) = a_1^{12}/\Delta$. In characteristic 2 we can differentiate between two cases, $j(E) = 0$, i.e. $a_1 = 0$ and $j(E) \neq 0$. The condition $j(E) = 0$ is equivalent to the curve being supersingular. This is a type of curves avoided in cryptography for reasons to be explained later. We will even though describe this case for reasons of completeness.

If $j(E) \neq 0$ then the curve equation over $\mathbf{k}$ reduces to

$$E : y^2 + xy = x^3 + a_2x^2 + a_6.$$

The discriminant of the curve then reduces to $\Delta = a_6$, and its $j-$invariant to $j(E) = 1/a_6$. The inverse of the point $P = (x_1, y_1)$ is given as $-P = (x_1, y_1 + x_1)$. The coordinates of the sum $P + Q = (x_3, y_3)$ of P and $Q = (x_2, y_2)$, $Q \neq -P$, are given as

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a_2, & \text{if } P \neq Q. \\ \\ x_1^2 + \dfrac{a_6}{x_1^2}, & \text{if } P = Q. \end{cases}$$

and

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1, & \text{if } P \neq Q. \\ \\ x_1^2 + \left(x_1 + \dfrac{y_1}{x_1}\right) x_3 + x_3, & \text{if } P = Q. \end{cases}$$

If $j(E) = 0$ then the curve equation over $\mathbf{k}$ reduces to

$$E : y^2 + a_3 y = x^3 + a_4 x + a_6.$$

The discriminant of the curve then reduces to $\Delta = a_3^4$. The inverse of the point $P = (x_1, y_1)$ is given as $-P = (x_1, y_1 + a_3)$. The coordinates of the sum $P + Q = (x_3, y_3)$ of P and $Q = (x_2, y_2)$, $Q \neq -P$ are given as

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + x_1 + x_2, & \text{if } P \neq Q. \\ \\ \dfrac{x_1^4 + a_4^2}{a_3^2}, & \text{if } P = Q. \end{cases}$$

and

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + y_1 + a_3, & \text{if } P \neq Q. \\ \\ \left(\dfrac{x_1^2 + a_4}{a_3}\right)(x_1 + x_3) + y_1 + a_3, & \text{if } P = Q. \end{cases}$$

3 General attacks on the ECDLP

The attack in this section are general in the sense that they can be applied to attack the ECDLP over any elliptic curve, they do not exploit any possible weaknesses on the curve. The main focus of the section is to give an in depth analysis of the best known attacks, namely the parallelized Pollard ρ and λ methods.

3.1 The Pohlig-Hellman and BSGS attack

We start by examining the algorithms that work for any cyclic finite abelian groups. But, first of all we have to define the elliptic curve discrete logarithm problem: Let $E(\mathbf{k})$ be an elliptic curve defined over the ground field $\mathbf{k} = \mathbb{F}_q$ and P a point of order n from the curve. Given another point $Q \in E(\mathbf{k})$ we have to find λ such that $Q = \lambda P$, $0 \leq \lambda \leq n-1$, if such an integer exists.

The most obvious method of solving the ECDLP is exhaustive search. One computes $R = [\mu]P$ for $\mu = 1, 2, 3, ...$, and checks whether $R = Q$. When equality is reached we conclude $\mu = \lambda$. The algorithm has no storage requirements, but has a running time of $O(N)$, where N is the order of the group, in both the average and worst case.

Pohlig and Hellman have observed that the DLP in a group G is only as hard as the discrete logarithm problem in the largest prime subgroup of G. A very important consequence of this is that for elliptic curve cryptography we select elliptic curves such that $\#E(\mathbf{k}) = N = h \cdot l$, where l is a large prime an the cofactor h is very small, usually $h = 1, 2$ or 4. The details of the algorithm can be found in [5].

As a consequence of the Pohlig-Hellman simplification we can concentrate on solving the DLP in groups of prime order. One way to do this is to use Shanks' the Baby-step/Giant-step, BSGS, algorithm.

For the start we have a group $G = \langle P \rangle$, which we now assume to have prime order l. As before we are given $Q \in G$, and we want to find $\lambda (\text{mod } l)$

such that $Q = [\lambda]P$.
We define λ as:

$$\lambda = \lambda_0 + \lambda_1 \lceil \sqrt{l} \rceil$$

Since $\lambda \leq l$, we know that $0 \leq \lambda_0, \lambda_1 < \lceil \sqrt{l} \rceil$. We compute the Baby-steps as

$$P_i = [i]P \qquad \text{for } 0 \leq i \leq \lceil \sqrt{l} \rceil.$$

The pairs (P_i, i) are stored in a table so that one can easily search for items indexed by the first entry in the pair. One way to do this is to use a hash table. A hash table is a database accessed by one or more hash functions. The computation of Baby-steps takes $O(\lceil \sqrt{l} \rceil)$ time, but there is a similar amount of storage requirement. We start the computation of Giant-steps by writing $P' = [\lceil \sqrt{l} \rceil]P$, followed by the computation of

$$Q_j = Q - [j]P' \qquad \text{for } 0 \leq j \leq \lceil \sqrt{l} \rceil.$$

The time required to compute the Giant-steps is at most $O(\lceil \sqrt{l} \rceil)$. The next step is to try to find a match for Q_i in the table of Baby-steps. If we find a value of P_i such that $P_i = Q_j$, then $\lambda_0 = i$ and $\lambda_1 = j$, since

$$[i]P = Q - j[\lceil \sqrt{l} \rceil]P$$

and

$$[i + j \lceil \sqrt{l} \rceil]P = Q.$$

The running time of the algorithm is $O(\sqrt{l})$, in both the average and worst case. The main problem with the algorithm is the requirement of $O(\sqrt{l})$ storage space. For this reason the algorithm is infeasible in practice.

It has been shown that the BSGS algorithm is the fastest possible method for solving the DLP in a 'black box group' [48]. Black box groups are a theoretical tool which allow the analysis of algorithms in idealized setting. A black box group is modelled in such a way that the representations of field elements provide no structure.

3.2 Pollard's ρ algorithm

Pollard based his algorithm on the birthday paradox. That is, if we choose elements at random from a set of S numbered elements, we only need to choose $\sqrt{S}$ elements in order to get a repetition, also called a *collision*. Just as with other methods that are based on a collision search, the goal is to take a given function f and find two different inputs that produce the same output. The best attack known on the general ECDLP is the parallel collision search based on Pollard's ρ method. But we start with the simple processor case. We are given a finite cyclic group G of order N, which we as a result of Pohlig-Hellman assume to be of prime order, and a function $f : G \rightarrow G$, which we call the *iterating* function. We select a starting value $Z_0 \in G$ and then generate successive terms by the rule $Z_{k+1} = f(Z_k)$, for $k = 0, 1, 2....$ Since G is finite, this sequence, also called a '*walk*', eventually begins to cycle. Since the sequence is a walk, each application of the iterating function is called a '*step*'. One simple approach to detecting a collision with Pollard's ρ method is to use Brent's algorithm [8]. That is, there exist two uniquely determined smallest integers $\mu \geq 0$ and $\varepsilon \geq 1$ such that $Z_k = Z_{k+\varepsilon}$ for all $k > \mu$. We call μ the *preperiod* or '*tail*' and ε the *period* or '*cycle*'. For performance reasons we wish the function f to be a random mapping, meaning the function f should be equally probable among all functions in form $G \rightarrow G$. The probability that no collision is found after selecting k inputs is $(1 - \frac{1}{N})(1 - \frac{2}{N})...(1 - \frac{k-1}{N}) \approx e^{-k^2/2N}$ for large N and $k = O(\sqrt{N})$. Let $E(\mu + \varepsilon)$ denote the expected value of the sum of the tail and cycle of the sequence (Z_i), i.e. the expected number of steps taken on the pseudo-random walk before a collision occurs. Then, under the assumption that f is a random mapping, the value of $E(\mu + \varepsilon) = \sqrt{\frac{\pi|G|}{2}} \approx 1,253\sqrt{|G|}$ [61]. Using Brent's algorithm the collision is found after an expected number of $\approx 1,97\sqrt{|G|}$ iterations [59].

The idea behind the iterating function used by Pollard is the following:

we partition the group G into 3 distinct subsets of roughly equal size, S_1, S_2 and S_3 based on some easily testable property. Pollard's original method was developed to solve the DLP and implemented for finite fields of the type $\mathbb{F}_p$. In the ECDLP, we are dealing with a cyclic subgroup of points $\langle P \rangle$ of order l, with generator P and group element Q. When adapted for elliptic curves the original iterating function becomes the following:

$$f(Z) = \left\{ \begin{array}{ll} Z + P, & \text{if } Z \in S_1. \\ 2Z, & \text{if } Z \in S_2. \\ Z + Q, & \text{if } Z \in S_3. \end{array} \right\}$$

The resulting terms are expressed as $Z_k = a_k P + b_k Q$, where the scalars a_k, $b_k \in \{0, ..., l-1\}$ are computed as:

$a_0 = 1$, $b_0 = 0$

$a_{k+1} = a_k + 1$, $a_{k+1} \equiv 2a_k (\text{mod } l)$, $a_{k+1} = a_k \quad$ for $k = 0, 1, 2...$

$b_{k+1} = b_k$, $b_{k+1} \equiv 2b_k (\text{mod } l)$, $b_{k+1} = b_k + 1 \quad$ for $k = 0, 1, 2...$

according to the three cases above.

Because the number of points in the group is finite, the sequence of points must begin to repeat. Upon detection of a collision, that is $Z_i = Z_j$, we have

$$a_i P + b_i Q = a_j P + b_j Q$$

Since $Q = \lambda P$, we have

$$a_i P + \lambda b_i P = a_j P + \lambda b_j P$$

Using modular arithmetic, we get

$$a_i + \lambda b_i \equiv (a_j + \lambda b_j) (\text{mod } l)$$

and

$$\lambda \equiv \left(\frac{a_i - a_j}{b_j - b_i} \right) (\text{mod } l)$$

unless we are very unlucky and $b_i \equiv b_j (\text{mod } l)$. So the method is a 'Monte Carlo' method, since there is no guarantee of success. Since l has no other factors others than 1 and itself, the only time $gcd(b_j - b_i, l) > 1$ holds is if $b_j - b_i$ is a multiple of l. Given that the size of l in practice is greater than 2^{160}, this is extremely unlikely.

3.3 Better random walks

We have said in the beginning that we wish the iterating function f to be a random mapping. The original ρ method uses an iterating function with 3 clauses. In 2 clauses we perform point addition, $Z + P$ and $Z + Q$ and are thus taking small steps. Under the third clause, we are performing point doubling, $2Z$, so we are taking a good size step. Unless Q is a small scalar multiple of P it will take considerable time to walk through the tail and the cycle and find a match. On the other hand, always taking large steps could lead to skipping over several terms in the cycle and not obtain a match right away, which is our objective. It was shown [61] that the value of $E(\mu + \varepsilon)$ using Pollard's original walk is approximately $1,596\sqrt{l}$, which is considerably slower than the expected value of $1,253\sqrt{l}$. In the following we are going to look at the work done by Teske on improving this result.

3.3.1 Linear and combined walk

The original Pollard ρ algorithm does not achieve the performance of a random walk. Teske [61] investigated the effect of changing the number of subgroup partitions and therefore function clauses on the performance of the ρ method. Two types of better random walks were suggested: *linear walk* and *combined walk*. Linear walks use an iterating function that contains a fixed number r of clauses, each of which defines a point addition operation unique to its partition. The question is how should the parameter r be chosen? In her work Teske experimented with elliptic curve subgroups of prime order up to 13 digits. The experiments showed that $r = 20$ is a good choice. It was also established [61] that taking $r = 20$ is suitable for simulating random walks for any size of group orders. Thus, when performing the linear walk we first partition the group G into 20 sets, S_1,...S_{20}. The next step is to define a set of multipliers M_i, these are produced by generating random integers $s_i, t_i \in [1, ..., l-1]$ and then computing

$$M_i = [s_i]P + [t_i]Q \qquad i = 1, ..., 20.$$

The iterating function is defined as

$$f(Z) = Z + M_i \qquad \text{for } Z \in S_i.$$

As before the resulting terms are expressed as $Z_k = a_k P + b_k Q$, where the scalars a_k, $b_k \in \{0, ..., l-1\}$ are computed as:

$$a_{k+1} \equiv (a_k + s_i) (\mathrm{mod}\, l) \quad \text{and} \quad b_{k+1} \equiv (b_k + t_i) (\mathrm{mod}\, l).$$

Through the experiments Teske found that when using linear walk the running time of the algorithm is $E(\mu + \varepsilon) \approx 1,292\sqrt{l}$, which is very close to the expected value of $1,253\sqrt{l}$.

Similar to linear walks, combined walks use a fixed number of partitions, $r + q$. The iterating function contains r rules that specify point addition operations and q rules that specify point doubling operations, making a total of $r + q$ rules. The experimental findings [62] indicate that the best results is obtained if the ratio of doublings and addings is between 1/4 and 1/2, while the performance gets worse if the ratio gets much larger than 1. To explain how to perform this type of walk we choose the values of $r = 16$ and $q = 4$. This means that the group G is again partitioned into 20 sets, S_1,...S_{20}. We choose 4 pairwise distinct numbers $u_1, ..., u_4$ between 1 and 20 and again define a set of multipliers M_i,

$$M_i = [s_i]P + [t_i]Q \qquad \text{where } i = \{1, ..., 20\} \setminus \{u_1, ..., u_4\}.$$

The iterating function is defined as

$$f(Z) = \left\{ \begin{array}{ll} Z + M_i, & \text{if } i \notin \{u_1, ..., u_4\} \text{ and } Z \in S_i. \\ 2Z, & \text{otherwise.} \end{array} \right\}.$$

The scalars a_k and b_k are computed as:

$$a_{k+1} \equiv (a_k + s_i) (\mathrm{mod}\ l) \quad \text{or} \quad a_{k+1} = 2a_k (\mathrm{mod}\ l).$$

$$b_{k+1} \equiv (b_k + t_i) (\mathrm{mod}\, l) \quad \text{or} \quad b_{k+1} = 2b_k (\mathrm{mod}\, l).$$

The reason behind including point doubling is to take bigger steps in our walk and thus move faster through the tail and cycle to obtain a solution. Through the experiments it was found that although the combined walk is slightly faster than the linear walk for small values of p, the latter is to prefer as p grows. In Teske' experiments [61] the expected number of steps to be taken with this walk is approximately $1,3\sqrt{l}$.

So far we have not defined how the partition is done. In practice it's usual to map an input point $Z \in \langle P \rangle$ to a partition number between 1 and r with a hash function of the form $h : \langle P \rangle \rightarrow \{1, ..., r\}$. This hash function uses an arithmetic operation that is very fast. This ensures the efficiency of the iterating function at every evaluation of a new term. The hash function used is:

$$h(Z) = \left\{ \begin{array}{ll} 1, & \text{if } 0 < x < k. \\ 2, & \text{if } k < x < 2k. \\ ..... & \\ r, & \text{if } (r-1)k < x < rk. \end{array} \right\}$$

where $k = 2^m/r$.

We can base on the hash function on either coordinate without effecting the performance of the algorithm. Here our hash function is based on the x coordinate when treated as a binary value. The boundary value $k = 2^m/r$ is used to slice the space of binary strings of fixed length into r subsets of equal size. Now the k smallest binary values are mapped to the first partition, the k next largest to the second, and so on until the k largest values, which are mapped to the last partition r. For the linear walk this means that we should still compute the set of multipliers M_i as before, but the iterating function is given by

$$f(Z) = Z + M_{h(P_i)} \qquad \text{for } Z \in S_i$$

and the scalars are given as:

$$a_{k+1} \equiv (a_k + s_{h(P_i)}) (\text{mod } l)$$

$$b_{k+1} \equiv (b_k + t_{h(P_i)}) (\text{mod } l).$$

3.4 Parallel collision search

The parallelized version of Pollard's ρ method is the method of choice when solving the ECDLP in practice. The algorithm is though inherently serial in nature and cannot be directly parallelized over several processors efficiently. One must wait for a given application of the function f to complete before the next can start. One way to parallelize the algorithm is to start each processor with a different starting value Z_0 and wait until one of them finds a collision. If m processors run the algorithm in this way, the speed-up we get is only about $\sqrt{m}$. It was Wiener and Van Oorschot [42] who presented an efficient way of parallelization which was based on '*distinguished points*'. A distinguished point is a group element with an easily testable property. An often used distinguishing property is whether a point's binary representation has a certain number of leading zeros. Several processors each create their own starting point Z_0 and iterate until a distinguished points Z_d is reached. When $Z_k = a_kP + b_kQ$ is a distinguished point, the triple (Z_k, a_k, b_k) is sent and stored in a central list common to all processors. As soon as a point occurs in two iterations, the remainder of those two iteration trails will be the same and thus lead to the same distinguished point. Therefore, by performing the iterations, all processors calculate random group elements and as soon as the same element has been calculated twice, we are going to get the same distinguished point twice, as well. If the two representations of the point, where the trails collided, are different, the representation of the distinguished point are different too, and therefore we are able to calculate λ. If we denote the number of processors involved in the search by m and suppose that each processor will send a distinguished point to the central list every $1/\theta$ group operations on average, where θ denotes the proportion of the points that constitute the distinguished points, the expected running time of the parallel Pollard ρ method is $E(\mu + \lambda) = \sqrt{\frac{\pi l}{2}}/m + 1/\theta$ [42].

The great advantage of the parallelized ρ method is that storage require-

ment is negligible. The reason for this is that it is only the distinguished points that are stored rather than all points encountered in the search. The expected space needed in the central list is $E(S) = m\theta E(\mu + \lambda) = \theta\sqrt{\frac{\pi l}{2}} + m$ distinguished points. We see that for the memory requirements to be as small as possible, we have to chose θ as small as possible. But, as θ gets smaller, the running time of the algorithm gets bigger. We see that there is a time-space trade-off. In practice the space requirement is chosen in such a way that the central server has enough memory, and that single processors can produce distinguished points at an convenient rate, for example one or two distinguished points each day.

3.5 Improving the algorithm

One way of speeding up the algorithm is to reduce the size of the space that is being searched by a factor of 2. This can be done by replacing Z_i by $\pm Z_i$ at each step, here $-Z_i$ being the negative of Z_i. We can do this by choosing the point which has a smallest y coordinate when it is interpreted as an integer. When performing the search Z_i, a_i and b_i should be computed as normal, but this time we compute $-Z_i$ as well. The point with the smallest y coordinate is taken to be Z_i. If it is Z_i, then we have the usual triple (Z_i, a_i, b_i). Should $-Z_i$ be used our triple becomes $(-Z_i, -a_i, -b_i)$, i.e. a_i is replaced by $-a_i$ and b_i is replaced by $-b_i$. Doing this we restrict our search to the points that have a smaller y coordinate than their negative. Since it yields exactly half of the points, $\neq \mathcal{O}$, we reduce the search space by a factor of 2. We have to remember that computing which of Z_i and $-Z_i$ to use also takes some time, so the running time of the algorithm is reduced by $\sqrt{2}$.

A problem that we might encounter is the appearance of trivial $2-cycles$. Suppose that Z_i and $-Z_i$ both belong to the same S_j and that in both cases after f is applied, the negative of the resulting point is used. This is when $Z_{i+1} = -(Z_i + c_jP + d_jQ)$ and $Z_{i+2} = -(Z_{i+1} + c_jP + d_jQ) = Z_i$. The

occurrence of these $2-cycles$ is reduced by using the linear walk. To reduce their occurrence we can usee the *look-ahead technique* which proceeds as follows. We define $f_w(Z) \equiv Z + c_w + d_w Q$ and suppose that $Z_i \in S_j$. Then $f(Z_i) = f_j(Z_i)$. We begin by computing $R = \pm f_j(Z_i)$, a candidate for Z_{i+1}. If $R \notin S_j$ then $Z_{i+1} = R$. If $R \in S_j$, then we treat Z_i as though it were in S_{j+1} (where $j+1$ is reduced modulo 20) and compute a new candidate $R = \pm f_{j+1}(Z_i)$. If $R \notin S_{j+1}$, then $Z_{i+1} = R$, otherwise we continue trying $j+2$, $j+3$, ... If all 20 choices fail, which is highly unlikely to happen, then we just use $Z_{i+1} = \pm f_j(Z_i)$. The idea is to reduce the probability that two successive points will belong to the same set. We also note that Z_{i+1} depends solely on Z_i, which is a requirement for parallel collision search to work.

The method for speeding up the parallel collision search described above can be applied to elliptic curves over any field. Further improvements are possible if we use special classes of elliptic curves.

3.6 Anomalous binary curves

We say that we are using a *subfield curve* when the elliptic curve we are going to use is defined over the field $\mathbb{F}_{q^n}$, $n > 1$, but the coefficients of the curve are in $\mathbb{F}_q$. The value of n should be chosen either to be a prime or a product of a small factor and a large prime to allow for a large enough prime divisor of $\#E(\mathbb{F}_{q^n})$. This because if n factors non-trivially as $n = n_1 n_1$, then both $\#E(\mathbb{F}_{q^{n_1}})$ and $\#E(\mathbb{F}_{q^{n_2}})$ divide $\#E(\mathbb{F}_{q^n})$, limiting the range of the largest prime divisor of $\#E(\mathbb{F}_{q^n})$. In practical implementations it is most usual to use a class of elliptic curves over $\mathbb{F}_{2^n}$ whose defining equations have coefficients in $\mathbb{F}_2$. Since it is required that $a_6 \neq 0$, they must be defined by either the equation

$$y^2 + xy = x^3 + 1$$

or the equation

$$y^2 + xy = x^3 + x^2 + 1.$$

These curves are called *anomalous binary curves* or *Koblitz curves*, although lately, the term 'Koblitz curve' is used for any elliptic curve which has a special endomorphism structure which enables efficient implementations. (It is very important not to confuse these curves with anomalous curves over prime fields). The reason for the extended use of these curves are:

1) It is easy to compute the group order $\#E(\mathbb{F}_{2^n})$.

2) The arithmetic can be made faster by using the Frobenius endomorphism. The Weil theorem enables us to compute the number of points on an elliptic curve over an extension field, $\#E(\mathbb{F}_{q^n})$ for $n \geq 2$, from $\#E(\mathbb{F}_q)$ as follows:

Theorem 5 *Let E be an elliptic curve defined over $\mathbb{F}_q$ and the trace of the Frobenius endomorphism $t = q + 1 - \#E(\mathbb{F}_q)$. Then $\#E(\mathbb{F}_{q^n}) = q^n + 1 - \alpha^n - \beta^n$, where α and β are complex numbers, $\mid \alpha \mid = \mid \beta \mid = \sqrt{q}$, determined from the factorization of $1 - tT + qT^2 = 0$.*

Proof. See [52]. ■

An alternative formulation, also leading to an efficient computation is as follows:

Let $\#E(\mathbb{F}_{q^n}) = q^n + 1 - a_n$, $n \geq 1$. Then the coefficients a_i are given by $a_0 = 2$, $a_1 = 1$, $a_{i+1} = a_i - qa_{i-1}$.

For the second part, from section 2 we know that the Frobenius endomorphism ρ acts as $\rho : (x, y) \rightarrow (x^q, y^q)$ on the curve E. In our case $\rho : (x, y) \rightarrow (x^2, y^2)$, since the ground field is $\mathbb{F}_2$. On points $Z = (x, y) \in E(\mathbb{F}_{2^n})$ we have $\rho^n(Z) = (x^{2^n}, y^{2^n}) = Z$. Actually, there is an integer σ, $0 \leq \sigma \leq l - 1$ such that $\rho(Z) = [\sigma]Z$ for every point $Z = (x, y) \in E(\mathbb{F}_{2^n})$. This integer is called the *eigenvalue* of the Frobenius endomorphism.

Now we let $(\alpha, \alpha^2, \alpha^{2^2}, ..., \alpha^{2^{n-1}})$ be a *normal basis* of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, for some $\alpha \in \mathbb{F}_{2^n}$. Such a basis always exists for all $n \geq 1$. Using a normal basis is

very efficient, because squaring a field element can than be accomplished by a cyclic shift of the coefficients of each point coordinate. As it was explained in [27] this leads to considerable improvements for point multiplication.

This improvement leads to very efficient implementations in both hardware and software and is the reason behind the popularity of anomalous binary curves. One important note is that the Koblitz curves are resistent to all the known special attacks which are to be discussed later.
Now we can look at the improvements in Pollard's algorithm offered by the use of Koblitz curves.

3.7 Further improvements and practice

The principle behind the improvement is to use equivalence classes, that is, if there is a convenient equivalence relation on the set, then we can consider a random walk on the set of equivalence classes rather than the whole set. This principle can be applied on subfield curves as well, but it is on the anomalous binary curves that we get the best improvement.

We will use a parallel collision search and compute Z_k, a_k, b_k as usual. Since we know that $\rho^n(Z) = (x^{2^n}, y^{2^n}) = Z$, we can also compute the $2n$ different points $\pm\rho^j(Z_k)$, for $0 \leq j \leq n-1$. We would now like to choose a representative element from this set. We will first consider the n points $\rho^j(Z_k)$ and use the one whose x is minimal subject to an ordering condition, we can for instance choose x such that its binary representation has smallest value when interpreted as an integer. We can then either choose that point or its negative, applying the same ordering condition used on x to its y coordinate. This point will replace Z_i. If we have chosen $\pm\rho^j(Z_k)$ to replace Z_k, we must then replace a_k with $\pm\rho^j a_k$ and also replace b_k with $\pm\rho^j b_k$ to maintain the relationship $Z_k = a_k P + b_k Q$. The powers of ρ^j can be precomputed to obtain further efficiency. The iteration function must be chosen carefully to avoid the appearance of trivial cycles. Experimentation carried out by Wiener and Zuccherato shows that if the parallel walk is used, the occurrence of these

trivial cycles is reduced sufficiently for practical purposes. By using the method just described we reduce the search space by a factor of $2n$, which reduces the expected running time by a factor of $\sqrt{2n}$, meaning that the running time of the ρ algorithm is now $\sqrt{\pi l/4n}$.

The SEC standard [56] gives 20 predefined curves in characteristic 2 and six of these are Koblitz curves, meaning that they have a convenient endomorphism which can be used to speed up the group law. The curves are labelled $sect163k1$, $sect233k1$, $sect239k1$, $sect283k1$, $sect409k1$ and $sect571k1$. The existence of the technic above means that these curves are not as secure as general curves over the same field. For example one would expect to need approximately 2^{81} operations to break a general elliptic curve over $\mathbb{F}_{163}$ while the Koblitz curve requires 2^{77} operations. This improvement although modest, means that we should choose another curve if a security level of 2^{80} is wanted. In a table taken from [56], we can see the difference between the security of a general curve and a Koblitz curve for the field sizes in the mentioned standard. It is assumed that the cofactor is two for the general curves, as this is the most common case in practice.

Table 2: Information on Koblitz vs general curve security

Curves	Field size	Cofactor	General curve security	Koblitz curve security
$sect163k1$	2^{163}	2	2^{81}	2^{77}
$sect233k1$	2^{233}	4	2^{116}	2^{111}
$sect239k1$	2^{239}	4	2^{119}	2^{114}
$sect283k1$	2^{283}	4	2^{141}	2^{136}
$sect409k1$	2^{409}	4	2^{204}	2^{198}
$sect571k1$	2^{571}	4	2^{285}	2^{279}

In 1997 Certicom [9] announced a series of elliptic curve challenges. The last break came in April 2004, when the ECC2-109 was solved. This problem, as well as all the solved problems before that, was done by using the Pollard algorithm with distinguished points and the ideas of Teske. The problem was

distributed over the internet and there was approximately 2600 users who contributed in the work. The distinguished points were chosen so that approximately one of every 2^{30} points on the curve would be distinguished. The team that won began their work in November 2002, which means that the solution was found some 17 months later. This also means that the field sizes that are used in practice offer long time security. Actually, we can draw a table, taken from [28], that shows the MIPS estimates for the ECDLP over fields considered for practical use:

Table 3: MIPS years to solve a generic ECDLP using the parallel Pollard method

q	$\sqrt{\pi q/2}$	$MIPS\ years$
160	2^{80}	$8,5 \times 10^{11}$
186	2^{93}	$7,0 \times 10^{15}$
234	2^{117}	$1,2 \times 10^{23}$
354	2^{177}	$1,3 \times 10^{41}$
426	2^{213}	$9,2 \times 10^{51}$

3.8 Pollard's λ algorithm

Pollard developed this method in order to solve the so called '*interval-[a,b]-discrete logarithm problem*'. The problem at hand is the following: Let $G = \langle P \rangle$ be a cyclic group of order g and Q a point from the group such that $Q = P^x$. We have to find the exponent x, for which we know that $x \in [a, b] \subset [0, g]$. In practice it is usual to choose $a = 0$ and $b \ll g$, so that a certain number of high-order bits of x are known to be zero. The reason why to choose x of such a form is that the exponentiation P^x is faster than for randomly chosen x. But, as we shall see later, this leads to reduced security and one should thus be careful when choosing x from such intervals.

Originally, the algorithm was called as 'the method for catching kangaroos', as it is described through two kangaroos, a tame and a wild one. It is also called the λ method because if the terms of the sequences of both kangaroos are drawn on a piece of paper, then the figure obtains the shape of

the Greek letter *lambda*. In the following we will use the latter. We shall now describe the algorithm in both its original form and in the version given by Wiener and van Oorschot. First we give the setup for the methods as it is the same for them both:

We define a set of jump '*distances*' $S = \{s_1, ..., s_r\}$ with $s_i > 0$ and a set of *jumps* $J = \{P^{s_1}, ..., P^{s_r}\}$ and let the mean of the values from S be α. Just as in the ρ method, we find a hash function $G \to \{1, ..., r\}$ that divides G into r disjoint sets $M_1, ..., M_r$, which give the rules of the kangaroos' jumps. We can, for example, use the same hash function as before. We denote the tame kangaroo with T and the wild kangaroo with W, T's position with t_k and W's position with w_k. The travel of the kangaroos consists of jumps, where each jump is a multiplication of the kangaroos current position by some $P^{s_i} \in J$. The sequences $\langle t_k \rangle$ and $\langle w_k \rangle$ for $k = 0, 1, 2...$ are given as

$$t_{k+1} = t_k * P^{s_i} \qquad \text{when } t_k \in M_i$$

$$w_{k+1} = w_k * P^{s_j} \quad \text{when } w_k \in M_j.$$

We denote the distances travelled by the kangaroos with $d_{k,tame}$ and $d_{k,wild}$. With starting distances $d_{k,tame} = d_{k,wild} = 0$, we define the sequences as

$$d_{k+1,tame} = d_{k,tame} + s_i \ \text{ when } t_k \in M_i$$

$$d_{k+1,wild} = d_{k,wild} + s_j \quad \text{when } w_k \in M_j.$$

Now to the algorithms: in Pollard's version , T is set off from the position at the end of the interval, $t_0 = P^b$ and we let it make $C\alpha$ jumps, where C is a constant. We mark the final spot of T with (t_N, d_N). This position is our '*trap*'. In terms of the exponents of P, at each time we know the position of T. Then W is set off from $w_0 = Q$. Since we do not know x, we do not know the exact location of W, that is why it is called wild. If the path of W meets that of T, he continues down the same path and falls into the trap. If we denote W's position at the trap with w_M, then $t_N = w_M$, that is

$P^b * P^{d_{N,tame}} = P^x * P^{d_{M,wild}}$. From here we can compute x as

$$x \equiv (b + d_{N,tame} - d_{M,wild}) (\mathrm{mod}\ g).$$

If the method fails, we set off another wild kangaroo with a starting point $w_0 = Q * P^z$, for some small known z. Now we determine the running time of this method. After passing P^b, the wild kangaroo makes approximately $C\alpha$ jumps before catching up with the tame kangaroo. The probability of landing on the tame kangaroo's trail is $1/\alpha$ for each jump. The probability of success after $C\alpha$ jumps is approximately $1-(1-1/\alpha)^{C\alpha} \approx 1-e^{-C}$. The trap T made is at a distance of about $b+C\alpha^2$ form P^0. W jumps an approximate $(b-a)/\alpha + C\alpha$ times to come this far and has it not yet landed on the trail of T it can be stopped, because it must have passed T, without landing on its trail. Because the expected starting point for W is $P^{(a+b)/2}$, when the algorithm succeeds, the expected number of jumps is $(b-a)/2\alpha + C\alpha$. Now we know that in the algorithm T is sent on its way once and makes $C\alpha$ jumps, while W succeeds once and fails $1/(1-e^{-C}) - 1$ times. The total running time of the algorithm is thus $C\alpha - (b-a)/2\alpha + (C\alpha + (b-a)/\alpha)/(1-e^{-C})$. Wiener and van Oorschot calculated [42] that this is minimized when

$$\alpha = \sqrt{b-a}\sqrt{\frac{1+e^{-C}}{2C(2-e^{-C})}}.$$

Evaluating this expression they find that the running time of the algorithm is minimal when $C \approx 1,39$ and $\alpha \approx 0,51\sqrt{b-a}$, and is approximately $3,3\sqrt{b-a}$ group iterations. All we have to store in this version of the algorithm is the set of jumps and the current position of the two kangaroos. If we allow more storage, we can use the alternative approach by Wiener and van Oorschot. They again used the distinguished point technique, but this time even for the single processor case. We can use the same distinguishing property as before: a point from the group is distinguished if the point's binary representation has a certain number of leading zeros. Now T is set

off from $t_0 = P^{\frac{a+b}{2}}$ and $w_0 = Q$. After each jump of the kangaroos we check whether the current terms are distinguished points. If this is the case, they are stored in a hash table. We can check whether a collision has occurred each time we store a distinguished point. If we come across a distinguished point such that $t_N = w_M$, with $N \neq M$, then $P^{\frac{a+b}{2}} * P^{d_{N,tame}} = P^x * P^{d_{M,wild}}$ and

$$x \equiv (\tfrac{a+b}{2} + d_{N,tame} - d_{M,wild}) (\mathrm{mod}\, g).$$

For the estimation of the running time we let θ denote the proportion of group elements that are distinguished and assume that $\alpha = \frac{\sqrt{b-a}}{2}$. It is expected that the kangaroos trail will collide after $2\sqrt{b-a}$ jumps and it will take additional $2/\theta$ iterations to find the distinguished point. In general, the time until a collision occurs is between $\sqrt{b-a}$, when the solution is near the middle of the interval, and $3\sqrt{b-a}$, when it is near the ends. Thus, the running time is $2\sqrt{b-a} + 2/\theta$ group iterations. The expected storage requirement is $2\theta\sqrt{b-a}$.

Just as the ρ algorithm, the λ algorithm can also be parallelized so that we get linear speed up. In fact, there are two different ways of paralellization, one by Wiener and van Oorschot and another by Pollard.

3.9 The Wiener-van Oorschot parallelization of the λ algorithm

For a start we assume that we have m processors, with m even. The single processor case given by Wiener and van Oorschot is actually just a special case of their parallelization of the algorithm. It corresponds to $m = 2$, with two processors simulated on one machine. If m is odd or indefinite, we can simulate $m' = 2m$ virtual processors by having one pair of wild and tame kangaroos on each processor and letting them jump alternately.

Instead of one tame and one wild kangaroo, we will now work with a herd of $m/2$ tame kangaroos and a herd of $m/2$ wild kangaroos, with one

kangaroo on each processor. We use the same setup as before and assume that the mean value α of the jump lenghts is $m\sqrt{b-a}/4$. We will count the running time in terms of iterations, where one iteration comprises one kangaroo jump on each processor. There is a couple of important variables we have to discuss before we start our kangaroos. One of them is the choice of jump distances s_i. There are two good choices: the first is to choose s_i to be powers of two starting with $s_i = 1$ up to $s_r = 2^{r-1}$, where r is such that the mean value of the s_i is close to the optimal value of $\alpha = m\sqrt{b-a}/4$. Because α varies with the number of processors and the length of the interval $[a, b]$, so does r as well. The second good choice consists of k integers $\{q_1, ..., q_k\}$ randomly chosen from the interval $[1, 2\alpha]$. The values of q_i must be pairwise distinct and $\gcd\{q_1, ..., q_k\} = 1$. Based on the experience from the ρ method, we may choose $k = 20$ in order to get sufficiently random kangaroo paths.

The other important variable to consider is the distance δ between members of the same herd. It is not desirable to choose δ either too small or too big. If the distance is too small, it could easily cause collisions between members of the same herd. The colliding kangaroos would follow the same path and the herd would effectively be reduced by one member for each such collision. On the other hand, if the distance was too big, the gap between the members of the herd in the front and those at the back would eventually get so big that it would not be possible to view the herd as a group, rather it would be a collection of kangaroos travelling individually. Experiments in [60] show that δ should be chosen so that $\delta \approx 2\alpha/m$. The $m/2$ tame kangaroos, $T_1, ..., T_{m/2}$, are set off from

$$t_0(T_0) = P^{\frac{a+b}{2}+(i-1)\delta}$$

the $m/2$ wild kangaroos , $W_1, ..., W_{m/2}$, from

$$w_0(W_i) = Q * P^{(i-1)\delta}$$

where $i = 1, ..., m/2$, on each processor.

The initial travel distances are $d_{0,tame}(T_i) = d_{0,wild}(W_i) = (i-1)\delta$, and each kangaroo gets a tag which indicates whether it is a tame or wild. As before, after each jump of the kangaroos it is checked whether some of the new spots are distinguished points. If we find some distinguished point, we send it to the central list, together with the corresponding travel distance and the tame/wild tag. Here it is checked whether there is a reoccurrence of a distinguished point, and if it is the case the solution is found in a manner already described.

An unwanted occurrence in this version of parallelization are collisions between members of the same herd. Such collisions are called *useless*. There are $\frac{m}{2}(\frac{m}{2}-1)$ possible pairs for useless collisions among the each of the herds. It is expected that there will be at most two useless collisions. This expected value is confirmed through experiments in [60]. The impact of useless collisions on the running time of the algorithm can be divided into two cases: $m = 4$ and $m > 4$. In general, there are $(m/2)^2$ possible pairs for useful collisions. The first useless collision reduces this number to $\frac{m}{2}(\frac{m}{2}-1)$. The second useless collision reduces further this number to either $\frac{m}{2}(\frac{m}{2}-2)$ if it happened in the same herd as the first one, or to $(\frac{m}{2}-1)^2$ if it happened in different herds. This means that the running time is decreased by a factor of at most $\frac{m}{2}(\frac{m}{2}-2)$. For $m > 4$ and specially $m \gg 4$ the effect of useless collisions is only marginal. For the case $m = 4$ however, there is an increase of the running time due to useless collisions. This is because after the first useless collision the number of useful ones decreases from 4 to 2. Experimental results in [60] show that the running time in this case is 'noticeably' larger. Therefore, if we only have a network of 4 processors, it is desirable to work with more than one kangaroo on each of them. It should also be mentioned that for some choices of the sets of jumps and distances δ, the occurrence of useless collisions is higher then expected. See [60] for further comments on this subject.

To calculate the expected running time of the algorithm, we can divide the movement of the herds into three parts: the time while they travel in separate regions, the time while they travel in a common region and a useful collision occurs and the time until this collision is detected. Since we do not know which of the herds lies further to the right on the interval $[a, b]$, in this analysis we will simply talk about a leading and a following herd, rather than herds of tame and wild kangaroos.

The initial distance between the herds is between 0 and $\frac{a+b}{2}$. The expected separation when the algorithm succeeds is $\frac{a+b}{4}$. This means that it takes about $\frac{b-a}{4\alpha}$ jumps for the trailing herd to catch up with the one in the front. After this has happened, the trailing herd enter a region where the herd of leading kangaroos already landed on $m/2\alpha$ spots. On each step, the probability that one of the $m/2$ trailing kangaroos lands on one of these spots is $m^2/4\alpha$. The expected number of jumps for each kangaroo before this happens is $4\alpha/m^2$. Thus, the expected running time until a useful collision occurs is $\frac{b-a}{4\alpha} + 4\alpha/m^2$ iterations. This value is at its minimum of $2\sqrt{b-a}/m$ for $\alpha = \frac{m\sqrt{b-a}}{4}$, which is in correspondence with the assumption from the start. In general this part of the running time will be somewhere between $\sqrt{b-a}/m$ iterations, when the solution is near the middle of the interval, and $3\sqrt{b-a}/m$ iterations, when it is near the ends. By adding $1/\theta$, the time needed to reach the next distinguished point after a useful collision occurred, to this, we get the expected overall running time of $T = 2\sqrt{b-a}/m + 1/\theta$ iterations.

3.10 Pollard's parallelization of the λ algorithm

Again, we use the same setup as before and let θ denote the proportion of distinguished points. The main difference between the two parallelization methods is that in this one there is no possibility of useless collisions. We will work with u tame and v wild kangaroos, where u and v are coprime. If

the number of processors involved is m, we will choose u and v such that $u \approx v \approx m/2$, as this gives the best running time, and $u+v \leq m$. We choose the r jump distances as $s_i = q_i uv$. Again, we can choose q_i to be either powers of two starting from $q_1 = 1$ up to $q_r = 2^{r-1}$, or random integers from the interval $[1, 2\alpha]$. The mean value of the jump distances should be close to $\alpha = \sqrt{\frac{b-a}{uv}}/2$. We set off the tame kangaroos from

$$t_0 = P^{\frac{a+b}{2}+iv} \quad \text{where } i = 0, ..., u-1$$

and the wild kangaroos from

$$w_0 = Q * P^{ju} \quad \text{where } j = 0, ..., v-1.$$

This implies that any two tame or any two wild kangaroos travel with travel distances that are in pairwise distinct residue classes modulo uv. Since the equation

$$(\tfrac{a+b}{2} + iv \equiv x + ju) (\mathrm{mod}\, uv)$$

has a unique solution in i and j, there is just *one* pair of tame and wild kangaroos that travel in the same residue class modulo uv. This means that this is the only pair that can collide.

As before, we keep track of the distinguished points and when some has been stored twice, we can find the solution we are looking for. The analysis of the running time is similar to that from the previous parallelization method, the only adjustment that has to be done is to replace the interval $[a, b]$ with $[a, b]/uv$ and to put $m = 2$, as we only have to consider the expected number of jumps of the two kangaroos that are destined to collide, and they travel in a fixed residue class modulo uv. The expected overall running time is $T = \sqrt{(b-a)/uv} + 1/\theta$ iterations on each processor. This is approximately the same running time as for the Wiener-van Oorschot parallelization, provided that we stick too our assumption that $u \approx v \approx m/2$. This version of parallelization is easier to handle because we do not have to deal with use-

less collisions and a proper choice of spacing. However, it only works if the number of processors is known in advance and all processors take part in the computation until the end. Since from the beginning of the computation it is determined which pair of kangaroos is the one to collide, a failure of one of the two corresponding processors would lead to the computation not being finished.

At the end, we should note that we might use the λ method to solve the general discrete logarithm problem. But, it is approximately $1,6$ times slower than the ρ method if $a = 0$ and $b = ord(G)$. It becomes faster than the ρ method when $b - a < \pi/8 \cdot ord(G)$ [60].

4 Special methods for solving the ECDLP

The attacks that will be analyzed in this section are special in the sense that they exploit weaknesses in special types of curves.

4.1 Pairing based attacks on ECDLP

Menezes, Okamoto and Vanstone [36] showed how the Weil pairing can be used to efficiently reduce the ECDLP in $E(\mathbb{F}_q)$ to the discrete logarithm problem in the multiplicative group of an extension field $\mathbb{F}_{q^n}$, where sub-exponential running time index calculus methods are known. We refer to their attack as the MOV-attack. Frey and Ruck [14] proposed a similar method, but based on the Tate pairing. In the following we will both analyze the methods and describe an important part of the theory of elliptic curves, namely the divisor theory.

4.1.1 Divisor theory

For the remainder of this section we let $\mathbf{k} = \mathbb{F}_q$, where q is a power of a prime p and let E be an elliptic curve defined over $\mathbf{k}$.

Definition 6 *Divisors*

The divisor group of the curve E, denoted by $\mathbf{D}(E)$, is the free abelian group generated by the points on E. Thus a divisor $D \in \mathbf{D}(E)$ is a formal sum $D = \sum_{P \in E} n_P(P)$, where $n_P \in \mathbb{Z}$ are 0 for all but finitely many P. ♦

The quantity n_P specifies the zero/pole property of a point P and its respective order. Inequality $n_P > 0$ indicates that a point P is a zero, and $n_P < 0$ indicates that P is a pole.

Definition 7 *Group operation, degree , order and support of a divisor*

The group operation on the divisor group is given by $D_1 + D_2 = \sum_{P \in E} n_P(P) + \sum_{P \in E} m_P(P) = \sum_{P \in E} (n_P + m_P)(P)$, *where* D_1, $D_2 \in \mathbf{D}(E)$. *The degree of* D *is defined by* $\deg\ D = \sum_{P \in E} n_P$. *The divisors of degree* 0 *form a subgroup of* $\mathbf{D}(E)$, *which we denote by* $\mathbf{D}^0(E) = \{D \in \mathbf{D}(E) \mid \deg\ D = 0\}$. *The order of* D *at* P *is* n_P, $ord_P(D) = n_P$. *The support of a divisor* D, *denoted supp*(D), *is the set of points* $\{P \in E \mid n_P \neq 0\}$. ♦

If E is defined by the Weierstrass equation $r(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0$, where $r \in \mathbf{k}[x, y]$, then the *coordinate ring* of E over $\mathbf{k}$, denoted $\mathbf{k}[E]$, is the integral domain $\mathbf{k}[E] = \mathbf{k}[x, y]/(r)$, where (r) denotes the ideal generated by r. The *function field* $\mathbf{k}(E)$ of E over $\mathbf{k}$ is the field of fractions of $\mathbf{k}[E]$. Now let $\overline{\mathbf{k}} = \cup_{n \geq 1} \mathbb{F}_{q^n}$ be the algebraic closure of $\mathbf{k}$. Then $\overline{\mathbf{k}}[E] = \overline{\mathbf{k}}[x, y]/(r)$ and $\overline{\mathbf{k}}(E)$, the function field of E over $\overline{\mathbf{k}}$, is the field of fractions of $\overline{\mathbf{k}}[E]$. The elements of $\overline{\mathbf{k}}(E)$ are called *rational functions*. Let now $f \in \overline{\mathbf{k}}(E)^*$ be a non-zero rational function and $P \in E \backslash \{\mathcal{O}\}$. For each point $P \in E$ there exists a rational function $u \in \overline{\mathbf{k}}(E)$, $u(P) = 0$ such that if $f \in \overline{\mathbf{k}}(E)^*$, then we can write $f = u^d s$, where $s \in \overline{\mathbf{k}}(E)$, $s(P) \neq 0, \infty$. The integer d does not depend on the choice of u. The function u is called a *uniformizing parameter* for P. The order of f at P is defined to be d, and we write $ord_P(f) = d$. The point P is a zero of f if and only if $ord_P(f) > 0$ and P is a pole if and only if $ord_P(f) < 0$.

We can define $div(f)$, the divisor of f, as $div(f) = \sum_{P \in E} ord_P(f)(P)$. If $f \in \overline{\mathbf{k}}(E)^*$, then $div(f) \in \mathbf{D}^0$ and $div(f) = 0$ if and only if $f \in \overline{\mathbf{k}}^*$. For two rational functions f_1 and f_2, we have $div(f_1) + div(f_2) = div(f_1 f_2)$ and $div(f_1) - div(f_2) = div(f_1/f_2)$.

Definition 8 *Principal divisors*

A divisor $D \in \mathbf{D}^0(E)$ is defined to be principal if $D = div(f)$ for some $f \in \overline{\mathbf{k}}(E)^*$. Another way of defining principality is to say that a divisor $D = \sum_{P \in E} n_P(P)$ of degree 0 is principal if and only if $\sum_{P \in E} [n_P](P) = \mathcal{O}$. ♦

The evaluation of f on a divisor $D = \sum_{P \in E} n_P(P)$ is defined by $f(D) = \prod_{P \in \sup p(D)} f(P)^{n_P}$. Two divisors are equivalent, denoted $D_1 \sim D_2$, if $D_1 - D_2$ is principal. The set $\mathbf{D}_{princ} = \{div(f) \mid f \in \overline{\mathbf{k}}(E)^*\}$ of all principal divisors form a subgroup of $\mathbf{D}^0$. The *degree 0 part divisor class group* or *Picard group* of E, denoted $Pic^0(E)$, is the quotient of $\mathbf{D}^0(E)$ by the subgroup of principal divisors. Further, $Pic^0_{\mathbf{k}}(E)$ is the subgroup of $Pic^0(E)$ fixed by $Gal_{\overline{\mathbf{k}}/\mathbf{k}}$. Similarly, $Pic(E)$ is the quotient of $\mathbf{D}(E)$ by the subgroup of principal divisors and $Pic_{\mathbf{k}}(E)$ is the subgroup of $Pic(E)$ fixed by $Gal_{\overline{\mathbf{k}}/\mathbf{k}}$. For each $D \in \mathbf{D}^0(E)$, there exists a unique point $Q \in E$ such that $D \sim (Q) - (\mathcal{O})$. Another way to represent a degree 0 divisor D is in its *canonical* form, $D = (Q) - (\mathcal{O}) + div(f)$, for a unique point $Q \in E$ and some $f \in \overline{\mathbf{k}}(E)$. The function f is determined up to multiplication by a non-zero element of $\overline{\mathbf{k}}$. In order to compute the Weil pairing we must be able to perform two important computations: firstly, we must know how to add two divisors written in the canonical form and express the result in canonical form, and secondly, given a principal divisor $D = \sum_{P \in E} n_P(P)$, we must be able to find $f \in \overline{\mathbf{k}}(E)^*$ such that $D = div(f)$.

4.1.2 The Weil pairing

Now we introduce a formula for adding two divisors in canonical form, such that the result is still in canonical form. This formula provides a method of finding a rational function f such that $div(f) = D$ for a given divisor D, and is critical for computing the Weil pairing. Let D_1, $D_2 \in \mathbf{D}^0(E)$ be given

by $D_1 = (P_1) - (\mathcal{O}) + div(f_1)$ and $D_2 = (P_2) - (\mathcal{O}) + div(f_2)$. Assume that $P_1 + P_2 = P_3$. Let $h_{P_1,P_2}(x, y) = ay + bx + c$ be the equation of the straight line passing through P_1 and P_2, and $h_{P_3}(x, y) = x + d$ be the equation of vertical line passing through P_3. (Note that if $P_1 = P_2$, $h_{P_1,P_2}(x, y)$ is the line tangent to P_1, and if $P_3 = \mathcal{O}$, we have $h_{P_3}(x, y) = 1$, a constant equation). Then we have $div(h_{P_1,P_2}) = (P_1)+(P_2)+(-P_3)-3(\mathcal{O})$, where P_1, P_2, and $-P_3$ are zeros because they are on line h_{P_1,P_2}, and $div(h_{P_3}) = (P_3) + (-P_3) - 2(\mathcal{O})$ where P_3, $-P_3$ are zeros because they are on line h_{P_3}. From the above discussion, the sum of divisors $D_1 + D_2$ is written as:

$$\begin{aligned} D_1 + D_2 &= (P_1) + (P_2) - 2(\mathcal{O}) + div(f_1 f_2) \\ &= (P_3) - (\mathcal{O}) + div(f_1 f_2) + div(h_{P_1,P_2}) - div(h_{P_3}) \\ &= (P_3) - (\mathcal{O}) + div(f_1 f_2 h_{P_1,P_2}/h_{P_3}). \end{aligned}$$

Before we can discuss the Weil pairing, we need to define the group of n-torsion points.

Definition 9 *Torsion point and torsion subgroup*

An n-torsion point P is a point satisfying $n(P) = \mathcal{O}$, $n \in \mathbb{Z}$. The set of $n-$torsion points forms a subgroup of E, denoted by $E[n]$. ♦

Let $E(\mathbf{k})[n]$ denote the subgroup of n-torsion points in $E(\mathbf{k})$, where $n \neq 0$. From now on we will write $E[n]$ for $E(\overline{\mathbf{k}})[n]$, where $\overline{\mathbf{k}}$ denotes the algebraic closure of $\mathbf{k}$. If $\gcd(n, q) = 1$, then $E[n] \simeq \mathbb{Z}_n \oplus \mathbb{Z}_n$. If $n = p^e$, then either $E[p^e] \simeq \{\mathcal{O}\}$ if E is supersingular or $E[p^e] \simeq \mathbb{Z}_{p^e}$ if E is non-supersingular. The notion of supersingularity will be explained in the following subsections. The following result provides necessary and sufficient conditions for $E(\mathbf{k})$ to contain all of the n-torsion points in $E(\overline{\mathbf{k}})$.

Theorem 10 *If $gcd(n, q) = 1$, then the following is equivalent*

i) $E[n] \subset E(\mathbf{k})$

ii) $n^2 \mid q+1-t$, $n \mid q-1$ *and either* $\phi \in \mathbb{Z}$ *or* $\vartheta\left(\frac{t^2-4q}{n^2}\right) \subset End_{\mathbf{k}}(E)$.

Here t is the trace of the Frobenius endomorphism ϕ, ϑ is the order of the discriminant $\left(\frac{t^2-4q}{n^2}\right)$ and $End_{\mathbf{k}}(E)$ the ring of $\mathbf{k}$-endomorphisms of E.

Proof. The proof and the explanation how the quantity $\vartheta\left(\frac{t^2-4q}{n^2}\right)$ is deduced can be found in [46]. ■

Now we turn our attention to the definition of the Weil-pairing. Let n be a positive integer coprime to p and $\mu_n \subset \overline{\mathbf{k}}^*$ be the group of n-th roots of unity, $\mu_n = \{u \in \overline{\mathbf{k}}^* \mid u^n = 1\}$. Given $P, Q \in E[n]$, there exist divisors D_P, $D_Q \in \mathbf{D}^0(E)$ such that $D_P \sim (P) - (O)$ and $D_Q \sim (Q) - (O)$. As $n(P) = n(Q) = \mathcal{O}$, divisors nD_P and nD_Q are principal and there exist rational functions f_P, f_Q such that $div(f_P) = nD_P$ and $div(f_Q) = nD_Q$. Suppose that D_P and D_Q have disjoint supports, i.e., $supp(D_P) \cap supp(D_Q) = \emptyset$.

Definition 11 *Weil pairing*

The Weil pairing, denoted e_n, is a function $e_n : E[n] \times E[n] \to \mu_n$ and is defined as

$$e_n(P, Q) = \frac{f_P(D_Q)}{f_Q(D_P)}.$$

The value of $e_n(P, Q)$ is independent of the choice of D_P, D_q, f_P and f_Q.

The Weil pairing has the following properties:

1) *Identity: For all $P \in E[n]$, $e_n(P, P) = 1$.*

2) *Alternation: $P, Q \in E[n]$, $e_n(P, Q) = e_n(Q, P)^{-1}$.*

3) *Bilinearity: $P, Q, R \in E[n]$, $e_n(P + Q, R) = e_n(P, R)e_n(Q, R)$, and*
$$e_n(P, Q + R) = e_n(P, Q)e_n(P, R).$$

4) *Non-degeneracy: If* $P \in E[n]$ *then* $e_n(P, \mathcal{O}) = 1$. *Moreover, if* $e_n(P, Q) = 1$ *for all* $Q \in E[n]$, *then* $P = \mathcal{O}$.

5) *Compatible: If* $P \in E[n]$ *and* $Q \in E[nn']$, *then* $e_{nn'}(P, Q) = e_n(P, n'Q)$.

6) *If* $E[n] \subseteq E(\mathbf{k})$, *then* $e_n(P, Q) \in \mathbf{k}$ *for all* $P, Q \in E[n]$. ♦

In order to compute the Weil pairing we will proceed using the following three steps:

1. Pick points $T, U \in E$ such that $P+T \neq U, Q+U$ and $T \neq U, Q+U$. Let $D_P = (P+T)-(T)$ and $D_Q = (Q+U)-(U)$. Then $D_P \sim (P)-(\mathcal{O})$ and $D_Q \sim (Q)-(\mathcal{O})$.

Step 2. Use an evaluation algorithm to compute $f_P(Q+U)$, $f_P(U)$, $f_Q(P+T)$ and $f_q(T)$ with $div(f_P) = nD_P$ and $div(f_Q) = nD_Q$.

Step 3. Compute

$$e_n(P, Q) = \frac{f_P(D_Q)}{f_Q(D_P)} = \frac{f_P((Q+U)-(U))}{f_Q((P+T)-(T))} = \frac{f_P(Q+U)f_Q(T)}{f_Q(P+T)f_P(U)}.$$

A crucial part in the evaluation algorithm in Step 2. For each integer m, there exists a rational function f_m such that $div(f_m) = m(P+T)-m(T)-(mP)+(\mathcal{O})$.

If $m = n$, then $div(f_n) = n(P+T)-n(T)-(nP)+(\mathcal{O})$, and $f_P = f_n$. For any points R, S, let $h_{R,S}$ and h_R be linear functions, where $h_{R,S}(x, y) = 0$ is the straight line passing through R, S, and $h_R(x, y) = 0$ is the vertical line passing through R.

Notice that

$$div(h_{m_1P,m_2P}) = (m_1P)+(m_2P)+(-(m_1+m_2)P)-3(\mathcal{O})$$

and

$$div(h_{(m_1+m_2)P}) = ((m_1+m_2)P)+(-(m_1+m_2)P)-2(\mathcal{O}).$$

Then we have

$div(f_{m_1+m_2}) = (m_1+m_2)(P+T)-(m_1+m_2)(T)-((m_1+m_2)P)+(\mathcal{O}) = m_1(P+T)-m_1(T)-(m_1P)+(\mathcal{O})+m_2(P+T)-m_2(T)-(m_2P)+(\mathcal{O})+(m_1P)+(m_2P)+(-(m_1+m_2)P)-3(\mathcal{O})-[((m_1+m_2)P)+(-(m_1+m_2)P)-2(\mathcal{O})] = div(f_{m_1}) + div(f_{m_2}) + div(h_{m_1P,m_2P}) - div(h_{(m_1+m_2)P})$ and hence

$$f_{m_1+m_2} = \frac{f_{m_1} f_{m_2} h_{m_1P,m_2P}}{h_{(m_1+m_2)P}}.$$

The last equation is recursive with initial conditions $f_0 = 1$ and $f_1 = \frac{h_{P+T}}{h_{P,T}}$, since $div(f_1) = (P+T)-(T)-(P)+(\mathcal{O}) = (P+T)+(-(P+T))-2(\mathcal{O})-[(P)+(T)+(-(P+T))-3(\mathcal{O})] = div(h_{P+T}) - div(h_{P,T})$. The following, more formal description of Miller's algorithm is given in [7]:

Algorithm 12 *Miller's algorithm*

Input: Integer $n = \sum_{i=0}^{t} b_i 2^i$ *with* $b_i \in \{0,1\}$ *and* $b_t = 1$, *and a point* $S \in E$.

Output: $f = f_n(S)$.

$f \leftarrow f_1; Z \leftarrow P;$

For $j \leftarrow t-1, t-2, ..., 1, 0$ *do*

$f \leftarrow f^2 \frac{h_{Z,Z}(S)}{h_{2Z}(S)}$; $Z \leftarrow 2Z;$

if $b_j = 1$ *then*

$f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$; $Z \leftarrow Z + P;$

Endif

Endfor

Return f ♦

In the same article, three refinements to the algorithm are presented and the interested reader is invited to study them closely. With all the necessary computational prerequisites in place we are now ready to take a look at the MOV-reduction.

4.1.3 The MOV reduction

Before looking at the reduction itself however, we need to further explore the theoretical background:

Theorem 13 *If $P \in E(\mathbf{k})$ is a point of order n, then there exists $Q \in E[n]$, such that $e_n(P,Q)$ is a primitive n-th root of unity.*

Proof. Let $Q \in E[n]$. From the Weil pairing we have that $e_n(P,Q)^n = e_n(P,[n]Q) = e_n(P,\mathcal{O}) = 1$. Thus $e_n(P,Q) \in \mu_n$, where μ_n the subgroup of the n-th roots of unity in $\mathbb{F}_{q^l}$. Now there are n cosets of the subgroup generated by P, and by the above lemma, as Q varies among the representatives of these n cosets, $e_n(P,Q)$ varies among the elements of μ_n. ■

Thus if we let $Q \in E[n]$ such that $e_n(P,Q)$ is a primitive $n-$th root of unity we get the following map and theorem:

Theorem 14 *The map*

$$f : \langle P \rangle \to \mu_n$$
$$R \to e_n(R,Q)$$

is a group isomorphism.

Proof. Clearly f is a homomorphism due to the properties of the Weil pairing. Suppose that $e_n(R,Q) = e_n(R',Q)$, then $e_n(R,Q)e_n(R',Q)^{-1} = 1 \Longrightarrow e_n(R,Q)e_n(-R',Q) = 1 \Longrightarrow e_n(R-R',Q) = 1 \Longrightarrow R - R' = \mathcal{O} \Longrightarrow R = R'$, thus f is injective. Now since both $\langle P \rangle$ and μ_n are finite of order n, this implies that f is surjective and hence bijective. Therefore $\langle P \rangle \simeq \mu_n$ as required. ■

Now to the reduction: let $P \in E(\mathbf{k})$ be a point of order n, n is an odd prime number, $gcd(n,q) = 1$, such that $\#E(k) = nv$ and $n \sim \#E(\mathbf{k})$, and $Q \in E(\mathbf{k})$. As usual we want to find λ, $0 \leq \lambda \leq n-1$, such that $Q = \lambda P$. It is easy to check whether a solution exists: $Q \in \langle P \rangle$ if and only if $n(Q) = \mathcal{O}$ and $e_n(P,Q) = 1$. We can now describe the method for reducing the ECDLP to the DLP in a finite field in four steps:

Algorithm 15 *The MOV reduction*

Input: An element $P \in E(\mathbf{k})$ of order n and $Q \in \langle P \rangle$.

Output: An integer λ such that $Q = \lambda P$.

1) Determine the smallest integer l such that $E[n] \subseteq E(\mathbb{F}_{q^l})$.

2) Find $R \in E[n]$ such that $\alpha = e_n(P, R)$ has order n.

3) Compute $\beta = e_n(Q, R)$.

4) Compute λ, the discrete logarithm of β to the base α in $\mathbb{F}_{q^l}$. ♦

The output of the algorithm is correct since $\beta = e_n(Q, R) = e_n(\lambda P, R) = e_n(P, R)^\lambda = \alpha^\lambda$. The discrete logarithm problem in a finite field may then be solved the subexponential running time index calculus method [58].
There are two major issues we have to deal with in order to be able to apply the algorithm:
1) the problem of explicitly determining the minimum positive integer l such that $E[n] \subseteq E(\mathbb{F}_{q^l})$.
2) the problem of efficiently finding n-torsion point R such that $e_n(P, R)$ has order n.

4.1.4 The modified MOV algorithm

The authors of the algorithm have presented successful solutions for both problems for the class of supersingular elliptic curves. But before looking at the modified algorithm, we have to define supersingularity:

Definition 16 *Supersingularity*

An elliptic curve $E(\mathbf{k})$ is supersingular if p divides t.

Here p is a characteristic of the field and t is a trace of Frobenius endomorphism. ♦

Equivalently, it can be shown that a curve over $\mathbf{k}$ with characteristic p is supersingular if and only if *(i)* $p = 2$, 3 and $j(E) = 0$ or *(ii)* $p \geq 5$ and $t = 0$, [5]. Supersingularity imposes limitations on the different group structures $E(\mathbf{k})$ can assume. It turns out supersingular curves have corresponding groups that are either cyclic of order q or isomorphic to either $\mathbb{Z}_{\sqrt{q}+1} \oplus \mathbb{Z}_{\sqrt{q}+1}$, $\mathbb{Z}_{\sqrt{q}-1} \oplus \mathbb{Z}_{\sqrt{q}-1}$or $\mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$. This means that supersingular curves can be divided into 6 categories. For each category we can precompute l such that $E[n] \subseteq E(\mathbb{F}_{q^l})$. Table 1 of [36] summarizes all the relevant information on supersingular curves. This takes care finding l. To find R we again take advantage of the limited group structures. From section 2 we recall that elliptic curve groups are, in general of the form $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$. The extensions of each category of supersingular curve will be of the form $\mathbb{Z}_{cn_1} \oplus \mathbb{Z}_{cn_1}$ for appropriate c. This will help limit our choices for R. The modified algorithm is as follows:

Algorithm 17 *The MOV reduction for supersingular curves*

Input: An element $P \in E(\mathbf{k})$ of order n and $Q \in \langle P \rangle$.
Output: An integer λ such that $Q = \lambda P$.

1) Determine the smallest integer l such that $E[n] \subseteq E(\mathbb{F}_{q^l})$ and the appropriate value of c by using the table from [36].
2) Pick a random point $R' \in E(\mathbb{F}_{q^l})$ and set $R = (cn_1/n)R'$.
3) Compute $\alpha = e_n(P, R)$ and $\beta = e_n(Q, R)$.
4) Compute the discrete logarithm λ' of β to the base α in $\mathbb{F}_{q^l}$.
5) Check whether $\lambda' P = R$. If this is so, then $\lambda = \lambda'$ and we are done. Otherwise, the order of α must be less than n, so go to 2). ♦

For supersingular curves the reduction runs in probabilistic polynomial time in *log* q. The detailed calculation of the running time can be found in [37].

4.1.5 Application of the algorithm to general elliptic curves

In the previous subsection we saw how the algorithm was completed for the class of supersingular elliptic curves. Finding solutions to steps 1) and 2) of the algorithm is however significantly more difficult when working with general, non-supersingular curves. For the first problem, the answer can be found in an article by Balasubramanian and Koblitz [3]. They proved that if $n \nmid (q-1)$, then $E[n] \subseteq E(\mathbb{F}_{q^l})$ if and only if l is the minimum integer such that $n \mid q^l - 1$. Hence by verifying that n does not divide $q^l - 1$ for all integers $l \in [1, c]$, where c is chosen so that the DLP problem in $\mathbf{k}^c$ is deemed to be intractable, the Weil pairing attack can be circumvented. In the same paper, they also suggested that we need $l = n$ if $n \mid (q-1)$ and $E[n] \nsubseteq E(\mathbf{k})$. Thus, when n is much larger than $\log q$, we may give up applying the MOV reduction since the extension degree in this case is too large in order for the reduced DLP in $\mathbb{F}_{q^l}$ to be solved in subexponential time in $log\ q$.

Possible solutions for the problem of finding adequate torsion points have been suggested in [50]. The authors have proposed three different methods. The first one is the simple 'brute force' approach of repeatedly choosing points from the curve, until a 'good' point is found. The second one is a method using the multiplication by constant maps. Both of these methods require exponential time in $l\ log\ q$. There is however a third method which, under the assumptions that $n \nmid q$ and $n \nmid q - 1$, actually is completed in probabilistic polynomial time in $l \log q$. The method is too detailed to be presented here, but can be found in its entirety in [50].

4.1.6 The Tate-Lichtenbaum pairing

The Frey-Ruck attack is quite similar in nature to the MOV attack, but uses the Tate-Lichtenbaum pairing instead of the Weil pairing. Just like the MOV attack, the Frey-Ruck attack attempts to reduce the ECDLP to the DLP in a suitable extension field over which the elliptic curve in question is defined,

where the DLP can be solved with subexponential algorithms.

In the construction of the Tate-Lichtenbaum pairing we will need the following theorem:

Theorem 18 *Weil Reciprocity:*

Let f and g be non-zero constant functions defined on a curve E over $\mathbf{k}$*, with $div(f)$ and $div(g)$ having disjoint support. Then*

$$f((g)) = g((f)).$$

Proof. See [5]. ■

Let E be an elliptic curve defined over $\mathbf{k}$, n a positive integer which is coprime to q and points P, $Q \in E(\mathbf{k})$. Let l be a positive integer such that the field $\mathbb{F}_{q^l}$ contains the $n-$th roots of unity, i.e. $n \mid q^l - 1$. From now on we let $\mathbb{F}_{q^l} = K$. Let $E[n]$ denote the subgroup of n torsion points in $E(K)$ and $nE(K) = \{n(P) \mid E(K)\}$. Notice that $nE(K)$ is a subgroup of $E(K)$, and hence we can look at the quotient group $E(K)/nE(K)$. We are now going to define a pairing on $E[n] \times E(K)/nE(K)$, but we need a place to map to. If we define the following set, $(K^*)^n = \{u^n \mid u \in K^*\}$, we can form the quotient $K^*/(K^*)^n$, which is a group of exponent n and is isomorphic to μ_n.

Now let $P \in E[n]$ and $Q \in E(K)/nE(K)$. Here we notice that technically we should be writing Q as a coset in the second group, instead we will simply think of Q are representative of an equivalence class. Now since $n(P) = \mathcal{O}$, we can find a function f such that $div(f) = n(P) - n(\mathcal{O})$. Take D to be a degree zero divisor equivalent to $(Q) - (\mathcal{O})$, and such that D is defined over $\mathbb{F}_{q^l}$ with disjoint support from $div(f)$. To do this we can simply choose a random $S \in E(K)$ and define $D = (Q + S) - (S)$. Since both $div(f)$ and D are defined over K, the value $f(D) \in K$. Since $div(f)$ and D were constructed to have disjoint support, $f(D) \neq 0$, thus $f(D) \in K$. We now define the Tate-Lichtenbaum pairing:

Definition 19 *The Tate-Lichtenbaum pairing*

Let E be an elliptic curve defined over $\mathbf{k}$. *Let n be a positive integer with* $\gcd(n,q)=1$ *and* $D=\sum_{P\in E} n_P(P)$. *The map*

$$\langle\cdot,\cdot\rangle : E[n]\times E(K)/nE(K)\rightarrow K^*/(K^*)^n$$

$$\langle P,Q\rangle \quad \rightarrow f(D)=\prod_P f(P)^{n_P}$$

is called the Tate-Lichtenbaum pairing and satisfies the following properties:

1) *Well defined:* $\langle\mathcal{O},Q\rangle\in(K^*)^n$ *for all* $Q\in E(K)$, $\langle P,Q\rangle\in(K^*)^n$ *for all* $P\in E[n]$ *and all* $Q\in nE(K)$.

2) *Non degeneracy: For each point* $P\in E[n]-\{0\}$*there is some point* $Q\in E(K)$ *such that* $\langle P,Q\rangle\notin(K^*)^n$.

3) *Bilinearity: For any integer* t, $\langle[t]P,Q\rangle\equiv\langle P,[t]Q\rangle\equiv\langle P,Q\rangle^t$ *modulo n-th powers.* ♦

In general there is no relationship between the Tate and Weil pairing, however when E is an elliptic curve such that $n^2 \mid \#E(K)$ and P, Q are independent points in $E[n]$ then we have $e_n(P,Q)=\langle P,Q\rangle/\langle Q,P\rangle$.

4.1.7 The Frey-Ruck attack

For the pourpose of the attack we will use what is referred to as a modified Tate-Lichtenbaum pairing. We note that the group $K^*/(K^*)^n$ is isomorphic to the group of roots of unity μ_n and thus an instance of the ECDLP on $E(K)$ can be mapped to an instance of the DLP in μ_n. Now we can define τ_n to be the following bilinear map:

$$\tau_n(\cdot,\cdot) : E[n]\times E(K)/nE(K)\rightarrow \mu_n$$

$$\tau_n(P,Q) \quad =\langle P,Q\rangle^{(q-1)/n}.$$

Although the setting is exactly the same, the second setup is more desirable since it will yield a definite answer instead of a coset in K^* modulo $n-$th powers. Again, since we are mapping into the group of $n-$th roots of unity, we are mapping into a suitable extension field K such that $\mu_n \subseteq K$. Now we describe the Frey-Ruck algorithm as given in [22]:

Algorithm 20 *The Frey-Ruck algorithm*

Input: An element $P \in E(\mathbf{k})$ of order n and $Q \in \langle P \rangle$.
Output: An integer λ such that $Q = \lambda P$.

1) Determine the smallest integer l such that $n \mid q^l - 1$ and set $K = \mathbb{F}_{q^l}$.
2) Pick $S; T \in E(K)$ randomly.
3) Compute the element $f \in K(E)^*$ such that $div(f) = n((P) - (\mathcal{O}))$ and compute $\alpha = f(S)/f(T)$.
4) Compute the element $\gamma = \alpha^{(q^l-1)/n}$. If $\gamma = 1$, then go to 2).
5) Compute the element $g \in K(E)^*$ such that $div(g) = n((Q) - (\mathcal{O}))$ and compute $\beta = g(S)/g(T)$, and $\delta = \beta^{(q^l-1)/n}$.
6) Solve the DLP $\delta = \gamma^\lambda$ in K^*, i.e. the logarithm of δ to the base γ in K^*. ♦

4.1.8 Comparing the pairing attacks

From the above considerations, we can measure the effectiveness of the Frey-Ruck algorithm by the extension degree l, which is the smallest integer such that $n \mid (q^l - 1)$, while we can measure that of MOV algorithm by l, which is the smallest integer such that $E(\overline{\mathbf{k}})[n] \subseteq E(\mathbb{F}_{q^l})$. Although, the conditions of the extension degree for the Frey-Ruck algorithm is usually weaker than that for the MOV algorithm, the work of Balasubramanian and Koblitz shows that the condition $n \mid (q^l - 1)$ is equivalent to the condition $E(\overline{\mathbf{k}})[n] \subseteq E(\mathbb{F}_{q^l})$ if $n \nmid (q - 1)$, i.e. the effectiveness of the MOV algorithm is the same as that for the Frey-Ruck algorithm if $n \nmid (q - 1)$. It was shown in [26] that elliptic curves of trace 2 are the only case for which this is not true. For these curves the MOV algorithm is exponential, while the Frey-Ruck algorithm is subexponential.

4.2 The Smart attack against anomalous elliptic curves

In order to avoid the MOV-attack Miyaji [41] proposed the use of anomalous elliptic curves over $\mathbb{F}_p$ which are such that $\#E(\mathbb{F}_p) = p$. However, such curves are themselves weak. Methods to attack the anomalous curves have independently been proposed by Smart [55], Satoh and Araki, and Semaev [49]. We will outline the method proposed by N. Smart. It uses the theory of elliptic curves defined over the field of $p-$adic numbers $\mathbb{Q}_p$. Details from the theory that are important for the attack will be given here.

4.2.1 Introduction to the p-adic numbers

We introduce here the $p-$adic numbers and their basic properties. Let p be a prime number and a a rational number. The number a can be expressed as $a = p^r \frac{m}{n}$, where $r \in \mathbb{N}$ and m, $n \in \mathbb{Z}$ are not divisible by p. We then define:

Definition 21 *The norm:*

$$ord_p(a) = r \text{ and } |a|_p = \begin{cases} p^{-r}, & \text{if } a \neq 0. \\ 0, & \text{if } a = 0. \end{cases}.$$

The function $|.|_p : \mathbb{Q} \to [0, \infty)$ *is a norm on* $\mathbb{Q}$ *, i.e.*

i) $|a|_p = 0 \Longleftrightarrow a = 0.$

ii) $|ab|_p = |a|_p |b|_p.$

iii) $|a + b|_p \leq |a|_p + |b|_p.$ ♦

This norm induces a metric $d_p(.,.)$ on $\mathbb{Q}$ defined by $d_p(a, b) = |a - b|_p$. The field $\mathbb{Q}_p$ of $p - adic\ numbers$ is the completion of $\mathbb{Q}$ for the metric d_p, i.e. $a \in \mathbb{Q}_p$ if and only if there exists a sequence (a_n), $n \in \mathbb{N}$, such that $|a_n - a|_p \to 0$ as $n \to \infty$. The natural representation of $p-$adic numbers is by an infinite series of the form $c_{-n}p^{-n} + ... + c_0 + ... + c_m p^m + ...$, where the c_i's are integers such that $0 \leq c_i \leq p - 1$. An element $a \in \mathbb{Q}_p$ is called a $p - adic\ integer$, if $ord_p(a) \geq 0$. The set of $p-$adic integers is denoted as $\mathbb{Z}_p$. (The latter must not be confused with $\mathbb{Z}/p\mathbb{Z}$). For more details on $p-$adic numbers, see [29].

4.2.2 Theoretical tools necessary for the attack

We consider an elliptic curve E defined over $\mathbf{k}$, where $\mathbf{k} = \mathbb{F}_q$. We would like to represent the points of E with one parameter in $\mathbf{k}$. In order to do this, we make the change of variables:

$$z = -\frac{x}{y} \text{ and } w = -\frac{1}{y} \left(\text{so } x = \frac{z}{w} \text{ and } y = -\frac{1}{w} \right)$$

The coordinate z has no connection with the projective coordinate Z. The point $\mathcal{O}$ is now represented as the pair $(0, 0)$ in the (z, w)-plane. The usual Weierstrass equation for E becomes

$$w = z^3 + a_1 zw + a_2 z^2 w + a_3 w^2 + a_4 zw^2 + a_6 w^3 \ (= f(z, w)).$$

As the next step, we substitute the equation into itself recursively and obtain ω as a power series in z:

$$\begin{aligned}
w(z) &= z^3 + (a_1 z + a_2 z^2)w + (a_3 + a_4 z)w^2 + a_6 w^3 \\
&= z^3 + (a_1 z + a_2 z^2)[z^3 + (a_1 z + a_2 z^2)w + (a_3 + a_4 z)w^2 + a_6 w^3] \\
&+ (a_3 + a_4 z)[z^3 + (a_1 z + a_2 z^2)w + (a_3 + a_4 z)w^2 + a_6 w^3]^2 \\
&+ a_6[z^3 + (a_1 z + a_2 z^2)w + (a_3 + a_4 z)w^2 + a_6 w^3]^3 + ... \\
&= z^3 + a_1 z^4 + (a_1^2 + a_2)z^5 + (a_1^3 + 2a_1 a_2 + a_3)z^6 \\
&+ (a_1^4 + 3a_1^2 a_2 + 3a_1 a_3 + a_2^2 + a_4)z^7 + ... \\
&= z^3(1 + A_1 z + A_2 z^2 + ...)
\end{aligned}$$

where $A_n \in \mathbb{Z}[a_1, ..., a_6]$ is a polynomial in the coefficients of E. In [52] it is shown that this recursion converges to a power series. Using the power series $w(z)$, we find the Laurent series for x and y.

$$x(z) = \frac{z}{w(z)} = \frac{1}{z^2} - \frac{a_1}{z} - a_2 - a_3 z - (a_4 + a_1 a_3)z^2 - ...$$

$$y(z) = -\frac{1}{w(z)} = -\frac{1}{z^3} + \frac{a_1}{z^2} + \frac{a_2}{z} + a_3 + (a_4 + a_1 a_3)z + ...$$

Thus, we see that the pair $\Big(x(z),y(z)\Big)$ yields a solution in the sense of formal power series, i.e. if we substitute the formal power series $x(z)$, $y(z)$ into the equation for E, we get the same formal power series on each side. Then, if we want to produce some points on $E(\mathbf{k})$ using the $z-$coordinate, we have to verify that the series $x(z)$, $y(z)$ converge in the field $\mathbf{k}$. In the field $\mathbb{Q}_p$, it is the case if $ord_p(z) \geq 1$, i.e. $z \in p\mathbb{Z}_p$ and the coefficients a_1, a_2, a_3, a_4 and a_6 lie in $\mathbb{Z}_p$. This gives an injection $p\mathbb{Z}_p \to E(\mathbb{Q}_p)$.

There is an addition law on the formal power series that corresponds to the addition law on $E(\mathbf{k})$. Let (z_1, w_1), (z_2, w_2) two points of E in the (z, w)-plane, then the $z-$coordinate of the sum of these points z_3 is obtained as a power series in z_1 and z_2

$$z_3 = F(z_1, z_2) = z_1 + z_2 - a_1 z_1 z_2 - a_2(z_1^2 z_2 + z_1 z_2^2) - (2a_3 z_1^3 z_2 - (a_1 a_2 - 3a_3) z_1^2 z_2^2 + 2a_3 z_1 z_2^3) + ... \in Z\left[a_1, ..., a_6\right]\left[z_1, z_2\right]$$

The development used to find F is explained in [52].

From now on, we assume that E is defined over $\mathbb{Q}_p$. In the following we will define various groups and isomorphisms.

The first group to be defined is $\widehat{E}(p\mathbb{Z}_p)$; it is essentially the set $p\mathbb{Z}_p$ with the addition law $x \oplus y = F(x, y)$ for all $x, y \in p\mathbb{Z}_p$, where F is the formal power series defined before. Similarly, $\widehat{E}(p^n\mathbb{Z}_p)$ is the set $p^n\mathbb{Z}_p$ with this addition law for all $x, y \in p^n\mathbb{Z}_p$.

Let now π be a function that reduces $p-$adic integers modulo p, i.e.

$$\begin{aligned} \pi : \quad \mathbb{Z}_p \quad &\to \mathbb{F}_p \\ a_0 + a_1 p + ... \ &\to a_0 \end{aligned}$$

The reduction of E modulo p is the elliptic curve $\widetilde{E}/\mathbb{F}_p$ obtained after reducing the coefficients of E modulo p. A point $P \in E$ can be represented as (x_1, y_1, z_1) with $x_1, y_1, z_1 \in \mathbb{Z}_p$ and at least one of x_1, y_1, z_1 in $\mathbb{Z}_p/p\mathbb{Z}_p$. The reduced point $\widetilde{P}$ of P is obtained by reducing every projective co-

ordinate of P modulo p, namely $\widetilde{P} = \Big(\pi(x_1), \pi(y_1), \pi(z_1)\Big) = (\widetilde{x}_1, \widetilde{y}_1, \widetilde{z}_1)$. The $n-$th subgroup of E is defined as $E_n(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid ord_p(P_x) \leq -2n\} \cup \{\mathcal{O}\}$, where P_x denotes the $x-$coordinate of the point P.
The three subsets of E important for us are:

i) The set $E_0(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid \widetilde{P} \in \widetilde{E}(\mathbb{F}_p)\}$, contains the points which reduce modulo p to an element of $E(\mathbb{F}_p)$.

ii) The set $E_1(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid \widetilde{P} = \widetilde{\mathcal{O}}\}$, contains the points which reduce modulo p to the identity element.

iii) The set $E_2(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid ord_p(P_x) \leq -4\} \cup \{\mathcal{O}\}$.

There are two exact sequences defined by these subgroups:

$$0 \to E_1(\mathbb{Q}_p) \to E_0(\mathbb{Q}_p) \to E(\mathbb{F}_p) \to 0$$

which means that multiplying an element of $E_0(\mathbb{Q}_p)$ by a multiple of p will produce a result which lies in $E_1(\mathbb{Q}_p)$.

$$0 \to E_2(\mathbb{Q}_p) \to E_1(\mathbb{Q}_p) \to \mathbb{F}_p^+ \to 0$$

where $\mathbb{F}_p^+$ denotes the additive group of $\mathbb{F}_p$. This sequence tells us that if we multiply an element in $E_1(\mathbb{Q}_p)$ by a multiple of p we will obtain an element of $E_2(\mathbb{Q}_p)$.
We will now define three important isomorphism:

Definition 22 *Three isomorphisms:*

Isomorphism 1: $\vartheta_p : \widehat{E}(p\mathbb{Z}_p) \to E_1(\mathbb{Q}_p)$.

$$z \quad \to \left(\frac{z}{w(z)}, -\frac{1}{w(z)}\right).$$

In general, $\widehat{E}(p^n\mathbb{Z}_p) \simeq E_n(\mathbb{Q}_p)$.

Isomorphism 2: The formal logarithm $\log_{\mathcal{F}}$ *induces an isomorphism between* $\widehat{E}(p\mathbb{Z}_p)$ *and* $p\mathbb{Z}_p : \log_{\mathcal{F}} : \widehat{E}(p\mathbb{Z}_p) \to p\mathbb{Z}_p$.

$$\log_{\mathcal{F}}(z) = \int \omega(z) = z + \frac{c_1}{2}z^2 + \frac{c_2}{3}z^3 + ...$$

where $\omega(z) = (1 + c_1 z + c_2 z^2 + c_3 z^3 + ...)dz$ *is the invariant differential on* $\widehat{E}(p\mathbb{Z}_p)$.

In general, $\widehat{E}(p^n\mathbb{Z}_p) \simeq p^n\mathbb{Z}_p$ *through* $\log_{\mathcal{F}}$.

Isomorphism 3: $\psi_p : E_1(\mathbb{Q}_p) \to p\mathbb{Z}_p$.

$$P \quad \to \log_{\mathcal{F}} \circ \vartheta_p^{-1}(P).$$

In general, $E_n(\mathbb{Q}_p) \simeq p^n\mathbb{Z}_p$. ♦

4.2.3 The reduction

We let $\widetilde{E}$ be a curve of trace one defined over a finite field $\mathbb{F}_p$ with p prime, i.e. $\#\widetilde{E}(\mathbb{F}_p) = p$. Since p is a prime, $\widetilde{E}(\mathbb{F}_p)$ is cyclic group and therefore $\widetilde{E}(\mathbb{F}_p) \simeq \mathbb{F}_p^+$. As usual we are given $\widetilde{P}, \widetilde{Q} \in \widetilde{E}(\mathbb{F}_p)$ and we want to find λ, such that $\widetilde{Q} = \lambda\widetilde{P}$.

Before looking at the reduction itself, we present two isomorphisms that are of crucial importance for the method:

Theorem 23 *Two isomorphisms:*

$E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p) \simeq \widetilde{E}(\mathbb{F}_p)$ *and* $E_0(\mathbb{Q}_p)/E_1(\mathbb{Q}_p) \simeq E_1(\mathbb{Q}_p)/E_2(\mathbb{Q}_p) \simeq \mathbb{F}_p^+$.

Proof. In order to prove the first one it suffices to consider the reduction map modulo p, $\pi : E(\mathbb{Q}_p) \to \widetilde{E}(\mathbb{F}_p)$ and its kernel $E_1(\mathbb{Q}_p)$. The isomorphism is given by applying the first isomorphism theorem of the group theory on π. The second one is a consequence of the *isomorphisms 1*, *2* and *3* given earlier.

■

The first step is to compute the lifts $P, Q \in E(\mathbb{Q}_p)$ of the points $\widetilde{P}$ and $\widetilde{Q}$. A point $P \in E(\mathbb{Q}_p)$ is said to be a lift of $\widetilde{P}$ if it reduces to $\widetilde{P}$ modulo p. A method for computing a lift is given in [32]. It follows that

$Q - \lambda P = R \in E_1(\mathbb{Q}_p)$

The next step is to multiply both sides of the last expression by p. This gives

$$[p]Q - \lambda([p]P) = [p]R \in E_2(\mathbb{Q}_p)$$

Since $[p]P$ and $[p]Q$ lie in $E_1(\mathbb{Q}_p)$, we can apply *isomorphism 3* from Definition 22. Then we get

$$\psi_p([p]Q) - \lambda\psi_p([p]P) \in p^2\mathbb{Z}_p.$$

So, this expression can be written in the form

$$c_1p + c_2p^2 + ... - \lambda(d_1p + d_2p^2 + ...) = b_2p^2 + b_3p^3 + ...$$

where the c_1's are the coefficients of the $p-$adic expansion of $\psi_p([p]Q)$ and the d_1's are the coefficients of the $p-$adic expansion of $\psi_p([p]P)$. Finally, we obtain

$$\lambda = \frac{\psi_p([p]Q)}{\psi_p([p]P)} \bmod p = \frac{c_1}{d_1} \bmod p.$$

Now it suffices to show how $\psi_p(P)$ can be computed for a point $P \in E_1(\mathbb{Q}_p)$. In order to find λ, we only have to compute this modulo p^2. According to the definition of ϑ_p, we have $\vartheta_p^{-1}(P) = -\frac{P_x}{P_y} \in \widehat{E}(p\mathbb{Z}_p)$, where P_x, P_y denote the $x-$, $y-$coordinate of P. Hence, using the definitions of the formal logarithm and of ψ_p, we get

$$\psi_p(P) \equiv -\frac{P_x}{P_y} (\bmod\, p^2).$$

The algorithm requires $O(log\ p)$ group operations on $E(\mathbb{Q}_p)$ [5]. With probability $1/p$ the above method will fail to find the required discrete logarithm as we will obtain $\psi_p([p]P) \equiv 0$. However, a different curve $E(\mathbb{Q}_p)$ can then be chosen which reduces to $\widetilde{E}(\mathbb{F}_p)$ modulo p and the method repeated.

5 The use of hyperelliptic curves in attacking the ECDLP

5.1 Basic definitions and properties

Hyperelliptic curves are a special class of algebraic curves and can be viewed as generalizations of elliptic curves. There are hyperelliptic curves of every genus $g \geq 1$. A hyperelliptic curve of genus $g = 1$ is an elliptic curve. We start by giving a formal definition of hyperelliptic curves:

Definition 24 *Hyperelliptic curve*

Let $\mathbf{k}$ *be a field and* $\overline{\mathbf{k}}$ *its algebraic closure. A hyperelliptic curve* C *of genus* g *over* $\mathbf{k}$ $(g \geq 1)$ *is an equation of the form*

$$C : v^2 + h(u)v = f(u) \text{ in } \mathbf{k}[u, v] \qquad (1)$$

where $h(u) \in \mathbf{k}[u]$ *is a polynomial of degree at most* g, $f(u) \in \mathbf{k}[u]$ *is a monic polynomial of degree* $2g+1$ *and there are no solutions* $(x, y) \in \overline{\mathbf{k}} \times \overline{\mathbf{k}}$ *which simultaneously satisfy the equation* $y^2+h(x)y = f(x)$ *and the partial derivative equations* $2y + h(x) = 0$ *and* $h'(x)y - f'(x) = 0$. ♦

A singular point on C is a solution $(x, y) \in \overline{\mathbf{k}} \times \overline{\mathbf{k}}$ which simultaneously satisfies the equation $y^2 + h(x)y = f(x)$ and the partial derivative equations $2y + h(x) = 0$ and $h'(x)y - f'(x) = 0$. This means that hyperelliptic curves are by definition non-singular.

Lemma 25 *Let* C *be a hyperelliptic curve over* $\mathbf{k}$ *defined by equation (1).*

1) If $h(u) = 0$, *then* $char(\mathbf{k}) \neq 2$.

2) If $char(\mathbf{k}) \neq 2$, *then the change of variables* $u \rightarrow u$, $v \rightarrow (v - h(u)/2)$ *transforms* C *to the form* $v^2 = f(u)$, *where* $\deg_u = 2g + 1$.

3) Let C *be an equation of the form (1) with* $h(u) = 0$ *and* $char(\mathbf{k}) \neq 2$. *Then* C *is a hyperelliptic curve if and only if* $f(u)$ *has no repeated roots in* $\overline{\mathbf{k}}$.

Proof. 1) Suppose that $h(u) = 0$ and $char(\mathbf{k}) = 2$. Then the partial derivative equations reduce to $f'(u) = 0$. Note that $\deg_u f'(u) = 2g$. Let $x \in \overline{\mathbf{k}}$ be a root of the equation $f'(u) = 0$ and let $y \in \overline{\mathbf{k}}$ be a root of the equation $v^2 = f(x)$. Then the point (x, y) is a singular point on C. Statement 1) now follows.

2) Under this change of variables the equation (1) is transformed to

$(v - h(u)/2)^2 + h(u)(v - h(u)/2) = f(u)$

which simplifies to $v^2 = f(u) + h(u)^2/4$. We note that $\deg_u(f + h^2/4) = 2g + 1$.

3) A singular point (x, y) on C must satisfy $y^2 = f(x)$, $2y = 0$ and $f'(x) = 0$. Hence $y = 0$ and x is a repeated root of the polynomial $f(u)$. ■

We continue the presentation of hyperelliptic curves by defining some important properties.

Definition 26 *Rational points, point at infinity, finite points, opposite, special and ordinary points*

Let $\mathbf{K}$ *be an extension field of* $\mathbf{k}$. *The set of* $\mathbf{K} -$ *rational points on* C *denoted* $C(\mathbf{K})$ *is the set of all points* $P = (x, y) \in \mathbf{K} \times \mathbf{K}$ *which satisfy the equation (1) of the curve* C *together with a special point at infinity denoted* $\mathcal{O}$. $\mathbf{K}$ *is called the base field. The set of points* $C(\overline{\mathbf{k}})$ *will simply be denoted by* C. *The points in* C *other than* $\mathcal{O}$ *are called finite points. For* $P = (x, y) \in C$ *the inverse (or conjugate) of* P *is the point* $\widetilde{P} = (x, -y - h(x))$. *When* P *satisfies* $P = \widetilde{P}$ *it is called a special (or ramified) point. Otherwise the point is called ordinary. The point at infinity* $\mathcal{O}$ *is a special point* $\mathcal{O} = \widetilde{\mathcal{O}}$. ♦

Next we define the Jacobian of an hyperelliptic curve over $\mathbf{k}$. As we will see later, this quantity plays a crucial role in the implementation of hyperelliptic curve cryptosystems. In analogy with the divisor theory for the elliptic curves, we let $\mathbf{D}^0$ denote the divisors of degree 0 and the set of all principal divisors by $\mathbf{D}_{princ}$.

Definition 27 *The Jacobian of the curve C over $\mathbf{k}$*

The quotient group $J_C(\mathbf{k}) = \mathbf{D}^0/\mathbf{D}_{princ}$ is called the Jacobian of the curve C over $\mathbf{k}$. ♦

Here we note that a divisor $D = \sum_{P \in C} n_P(P)$ is said to be defined over $\mathbf{k}$ if $D^\sigma = \sum_{P \in C} n_P(P^\sigma) = D$ for all automorphisms σ of $\overline{\mathbf{k}}$ over $\mathbf{k}$. If D is defined over $\mathbf{k}$, it does not mean that each point in the support of D is $\mathbf{k}-$*rational*. A principal divisor is defined over $\mathbf{k}$ if and only if it is a divisor of a rational function that has coefficients in $\mathbf{k}$. .
In order to have a unique representation for the divisors in $J(C)$ we introduce reduced and semi-reduced divisors:

Definition 28 *Semi-reduced and reduced divisors*

A semi-reduced divisor is a degree 0 *divisor of the form*
$D = \sum_{P \in C \setminus \mathcal{O}} n_P(P) - (\sum_{P \in C \setminus \mathcal{O}} n_P)\mathcal{O}$ *with the following properties:*

(1) $n_P \geq 0$.

(2) *if* $P \neq \widetilde{P}$ *and* $n_P > 0$, *then* $n_{\widetilde{P}} = 0$.

(3) *if* $P \neq \widetilde{P}$ *and* $n_P > 0$, *then* $n_P = 1$.

A semi-reduced divisor is called a reduced divisor when additionally:

4) $\sum_{P \in C} n_P \leq$ *genus.* ♦

Lemma 29 *For each divisor $D \in \mathbf{D}^0$ there exists a semi reduced divisor D_1 ($D_1 \in \mathbf{D}^0$) such that $D \sim D_1$.*

Proof. Let $D = \sum_{P \in C} n_P(P)$. Let (C_1, C_2) be a partition of the set of ordinary points on C such that

1) $P \in C_1$ if and only if $\widetilde{P} \in C_2$

2) if $P \in C_1$ then $n_P \geq n_{\widetilde{P}}$.

Let C_0 be the set of special points on C. Then we can write

$$D = \sum_{P \in C_1} n_P(P) + \sum_{P \in C_2} n_P(P) + \sum_{P \in C_0} n_P(P) - n(\mathcal{O}).$$

We consider the following divisor:

$$D_1 = D - \sum_{P=(x,y) \in C_2} n_P \; div(u-x) - \sum_{P=(x,y) \in C_0} \left\lfloor \frac{n_P}{2} \right\rfloor \; div(u-x).$$

This in turn equals:

$$D_1 = \sum_{P \in C_1} (n_P - n_{\widetilde{P}})(P) + \sum_{P \in C_3} (n_P - 2\left\lfloor \frac{n_{\widetilde{P}}}{2} \right\rfloor)(P) - \sum_{P \in \operatorname{sup} p(D_1)} n_P(\mathcal{O}).$$

Thus it follows that every divisor $D \in \mathbf{D}^0$ can be modified by principal divisors to obtain a semi-reduced $D_1 \sim D$. ■

Lemma 30 *For each divisor $D \in \mathbf{D}^0$ there exists a unique reduced divisor D_1, $D_1 \in \mathbf{D}^0$, such that $D \sim D_1$.*

Proof. *The proof of both the existence and uniqeness can be found in [39].* ■

The statement of the last lemma means that each equivalence class contains a unique reduced divisor and the set of reduced divisors of C over $\mathbf{k}$ forms a complete system of representatives for the Jacobian of C over $\mathbf{k}$. Each semi-reduced divisor $D = \sum_{P \in C \setminus \mathcal{O}} n_P(P) - (\sum_{P \in C \setminus \mathcal{O}} n_P)\mathcal{O}$ defined over $\mathbf{k}$

can be uniquely represented by a pair of polynomials $a, b \in \mathbf{k}[u]$, where $a(u) = \prod_{P \in C} (u - x_P)^{n_P}$ is monic, and $b(u)$ is the unique polynomial such that:

(*i*) $deg\ b < deg\ a$

(*ii*) for all $P \in C$; if $n_P \neq 0$, then $b(x_P) = y_P$

(*iii*) a divides $b^2 + bh - f$.

In this case, $D = gcd(div(a), div(b - v))$, and we write $D = div(a, b)$. Therefore, each reduced divisor D defined over $\mathbf{k}$ has a unique representation of the form $D = div(a, b)$, where $a, b \in \mathbf{k}[u]$ with a monic, $deg\ b < deg\ a < g$, and a divides $b^2 + bh - f$. The degree of D is $deg\ a$. We notice that the opposite of $D = div(a, b)$ is given by $-D = div(a, -h - b)$.

For hyperelliptic curves of genus $g \geq 2$, there is no natural group law for a curve C defined over $\mathbf{k}$. A group law is defined via $J_C(\mathbf{k})$. If $\mathbf{k}$ is a finite field, there are only finitely many divisor class representatives of the form $div(a, b)$, and $J_C(\mathbf{k})$ is a finite abelian group. If $\mathbf{k}$ has order q and the curve C is of genus g over $\mathbf{k}$, then the theorem of Weil implies that $(\sqrt{q} - 1)^{2g} \leq \#J_C(\mathbf{k}) \leq (\sqrt{q} + 1)^{2g}$, so $\#J_C(\mathbf{k}) \approx q^g$. Cantor developed an efficient algorithm for calculating the group law on specific hyperelliptic curves. The restrictions of this algorithm were the assumptions that $h(u) = 0$ and $char(\mathbf{k}) \neq 2$. This algorithm was later generalized by Koblitz. Koblitz's algorithm makes use of the unique reduced representation of the elements of $J_C(\mathbf{k})$. The algorithm contains two steps. Let $D_1 = div(a_1, b_1)$ and $D_2 = div(a_2, b_2)$ be reduced divisors defined over $\mathbf{k}$, so a_1, b_1, a_2, $b_2 \in \mathbf{k}[u]$. The first part of the algorithm finds a semi-reduced $D = (a, b)$ with $a, b \in \mathbf{k}[u]$, such that $D \sim D_1 + D_2$. The second part of the algorithm reduces D to an equivalent reduced divisor D'.

Algorithm 31 *Computation of the composition*

Input: Reduced divisors $D_1 = div(a_1, b_1)$ *and* $D_2 = div(a_2, b_2)$.

Output: A semi-reduced divisor $D = div(a, b)$ *such that* $D \sim D_1 + D_2$.

1) Compute $d_1 = gcd(a_1, a_2) = e_1 a_1 + e_2 a_2$.

2) Compute $d = gcd(d_1, b_1 + b_2 + h) = c_1 d_1 + c_2(b_1 + b_2 + h)$.

3) Let $s_1 = c_1 e_1$, $s_2 = c_1 e_2$ *and* $s_3 = c_2$ *so that*

$$d = s_1 a_1 + s_2 a_2 + s_3(b_1 + b_1 + h).$$

4) Set

$$a = \frac{a_1 a_2}{d^2}$$

and

$$b = \left(\frac{s_1 a_1 b_2 + s_2 a_2 b_1 + s_3(b_1 b_2 + f)}{d} \right) \bmod a. \quad \blacklozenge$$

The complete proof that this part of the algorithm works can be found in [39].

Algorithm 32 *Computation of the reduction*

Input: A semi-reduced divisor $D = (a, b)$.

Output: The unique reduced divisor $D' = div(a', b')$ *such that* $D' \sim D$.

1) Set

$$a' = \left(\frac{f - bh - b^2}{a} \right)$$

and

$$b' = (-h - b) \bmod a'.$$

2) If $\deg(a') >$ *genus then go to step 1).*

3) Make a' *monic.* $\blacklozenge$

Once again, the complete proof that this part of the algorithm works can be found in [39].

5.2 The discrete logarithm problem on hyperelliptic curves

We will in the following make a short review of the attacks which are specific to hyperelliptic curves. The description and analysis of these attacks is not the objective of this paper and we will limit us to give the expected running times of the various attacks. The reason to list them is that they are used in the final stage of the GHS attack on the ECDLP which will be described in the next subsection. Now we describe the discrete logarithm problem on hyperelliptic curve or HCDLP. Let C be a hyperelliptic curve of genus g over $\mathbf{k} = \mathbb{F}_q$. The HCDLP is defined as follows: given C, $D_1 \in J_C(\mathbf{k})$, $r = ord(D_1)$, and $D_2 \in \langle D_1 \rangle$, find the integer $\lambda \in [0, r-1]$ such that $D_2 = \lambda D_1$.

Since the HCDLP is a generalization of the ECDLP it is no surprise that all of the known attack on the ECDLP can be extended to an attack on the HCDLP. This includes the Pohlig and Hellmann, the BSGS, the MOV and the Frey-Ruck attacks. But just as for the ECDLP, these methods only have limited success in solving the HCDLP. And again, the first method that poses a real threat is Pollard's ρ method. The expected running time of the algorithm is $O(g^2 q^{n/2} \log^2 q/m)$ [24], where, as before, m denotes the number of processors involved. However, since the group operations in $E(\mathbf{k})$ can be performed faster than the group operations in $J_C(\mathbf{k})$, it is more efficient to apply the ρ method directly in $E(K)$.

The other alternative is to use index-calculus algorithms. Adleman, DeMarrais and Huang (ADH) [1] presented the first index-calculus algorithm for solving the HCDLP. Their algorithm was described for the case q an odd prime, but this was later extended in [4] to arbitrary q. The algorithm has an expected running time of $L_{q^{2g+1}}[c]$ for $g \to \infty$ and $\log q \leq (2g+1)^{0,98}$, where $c < 2,313$ and $L_n[c] = O(\exp((c+o(1))\sqrt{\log n \log\log n}))$ [24]. The algorithm does not assume that the group order $\#J_C(\mathbf{k})$ is known.

Building on the ADH algorithm, Gaudry presented an algorithm [20] which has an expected running time of $O(g^3q^2\log^2 q + g^2g!q\log^2 q)$. If g is fixed, then this running time is $O(q^{2+\epsilon})$ and the algorithm can be modified to one with running time $O(q^{\frac{2g}{g+1}+\epsilon})$ as $q \to \infty$ [19]. Gaudry's algorithm is faster than the ρ method when $\frac{n}{2} > \frac{2g}{g+1}$ [35], but becomes impractical for large genera, $g \geq 10$, because of the large multiplicative factor $g!$.

For larger g, the algorithm of Gaudry and Enge [11] should be employed. This algorithm has an expected running time of $L_{q^g}[\sqrt{2}] = L_{q^{2g+1}}[1]$ bit operations for $g/\log q \to \infty$, where $L_n[c] = O(\exp((c+o(1))\sqrt{\log n}\sqrt{\log\ \log n}))$. Since its running time is subexponential in q^g, this algorithm is infeasible when q^g is very large, i.e. $q^g \approx 2^{1024}$ [35]. The main reason for the improved running time over the ADH is that the order and structure of $J_C(\mathbf{k})$ is assumed to be known.

5.3 The Gaudry, Hess and Smart (GHS) attack on the ECDLP

The technique of Weil descent to solve the ECDLP was first proposed by Frey [13]. This strategy was detailed further by Galbraith and Smart [16]. These papers were rather general in their scope, but were not detailed enough to give precise and efficient algorithms to solve the ECDLP for specific curves.

The work of Gaudry, Hess and Smart [19] was less general than the earlier works but gave much more powerful and efficient techniques. We refer to the method as the GHS attack. We will in the following give an overview of the attack. More detailed analyses can be found in papers from the references.

Before describing the method we note that almost all research on Weil descent has been performed in characteristic 2. The ideas are easily applied to finite fields $\mathbb{F}_{p^n}$, where p is odd and $n < 1$, but the results in these cases are not as strong as in the case of characteristic 2.

We start now the description of the algorithm. Let l and n be positive

integers. For the remainder of the section we let $q = 2^l$, $\mathbf{k} = \mathbb{F}_q$ and $K = \mathbb{F}_{q^n}$ be the field extension, with $\mathbf{k}-$basis $\{\psi_0, \psi_1, ..., \psi_{n-1}\}$. We consider the elliptic curve E defined over K by the equation

$$E : y+xy = x^3+ax^2+b, \ a \in K, b \in K^*.$$

Let $\sigma : K \to K$ be the Frobenius automorphism defined by $\alpha \to \alpha^q$, and let $b_i = \sigma^i(b)$ for $0 \leq i \leq n-1$. We define

$$m = m(b) = \dim_{\mathbb{F}_2}(Span_{\mathbb{F}_2}\{(1, \sqrt{b_0}), (1, \sqrt{b_1}), ..., (1, \sqrt{b_{n-1}})\})$$

and assume one of the following conditions

i) n is odd, or ii) $m(b) = n$, or iii) $Tr_{K/\mathbb{F}_2}(a) = 0$ [19].

The first step of the in the process is to construct the *Weil restriction* $W_{E/\mathbf{k}}$ of scalars of E over $\mathbf{k}$. We set

$$a = \alpha_0\psi_0 + \alpha_1\psi_1 + ... + \alpha_{n-1}\psi_{n-1}$$

$$b = \beta_0\psi_0 + \beta_1\psi_1 + ... + \beta_{n-1}\psi_{n-1}$$

$$x = x_0\psi_0 + x_1\psi_1 + ... + x_{n-1}\psi_{n-1}$$

$$y = y_0\psi_0 + y_1\psi_1 + ... + y_{n-1}\psi_{n-1}$$

where α_i, $\beta_i \in \mathbf{k}$ are given and x_i, $y_i \in \mathbf{k}$ are variables. Substituting these equations into the equation for our elliptic curve and equating coefficients of ψ_i, we obtain $W_{E/\mathbf{k}}$, which is an $n-$dimensional abelian variety defined over $\mathbf{k}$, the group law on $W_{E/\mathbf{k}}$ being given by the group law on $E(K)$. This process is called *Weil descent*.

The next step is to intersect $W_{E/\mathbf{k}}$ with $n-1$ carefully chosen hyperplanes to obtain the hyperelliptic curve C. The genus g of C is either 2^{m-1} or $2^{m-1} - 1$, where $m = m(b)$.

The final step of the method is to construct an explicit group homomorphism

$$\phi : E(K) \to J_C(k).$$

It was argued in [19] that assuming $\#E(K) = rd$, r a prime and d a small integer, it is highly unlikely that the kernel of ϕ will contain the subgroup of order r of $E(K)$ unless E is defined over a proper subfield of K containing $\mathbf{k}$. Thus, ϕ can be used to reduce instances of the ECDLP to instances of the HCDLP. Namely, given P and $Q \in \langle P \rangle$, then $log_P Q = log_{\phi(P)}\phi(Q)$.

Now, the GHS attack is deemed to be successful if the genus g of C is small enough so that either Gaudry's or Gaudry and Enge's algorithm is more efficient than Pollard's algorithm. The GHS attack fails if either q^g is too large, say $q^g \geq 2^{1024}$, or if $g = 1$, in which case $J_C(k)$ is isogenous with $E(K)$. For the case $q = 2$ this translates to $m \geq 11$ or $m = 1$ [35].

Menezes and Qu [35] proved the following theorem which characterizes the smallest values of $m > 1$ and the elliptic curves which give rise to such m.

Theorem 33 *Let n be an odd prime, t the multiplicative order of 2 modulo n and $s = (n-1)/t$.*

i) The polynomial $x^n - 1$ factors over $\mathbb{F}_2$ as $(x-1)f_1 f_2 ... f_s$, where the f_i's are distinct irreducible polynomials of degree t. For $1 \leq i \leq s$ define

$$B_i = \{b \in K \mid (\sigma - 1)f_i(\sigma)b = 0\}.$$

ii) For all $1 \leq i \leq s$ and all $b \in B_i$, the elliptic curves

$$y^2 + xy = x^3 + \alpha x^2 + b$$

$$y^2 + xy = x^3 + b$$

have $m(b) \leq t + 1$, where α is a fixed element of K of trace one.

iii) If $m(b) = t + 1$ then E must be one of the previous curves for some i and some $b \in B_i$.

iv) The cardinality of the set $\bigcup_{i=1..s} B_i$ is $qs(q^t - 1) + q$.

Proof. See [35]. ■

It was also shown in [35] that if n is a prime in the range $160 \leq n \leq 600$ and $q = 2$ then the GHS attack will be infeasible. Since $\mathbb{F}_{2^n}$ with n prime are the field dimensions of interest when implementing elliptic curve cryptography schemes, we might conclude that the GHS is ineffective on real-life implementations. However, there are a few deployed elliptic curve systems that use the fields $\mathbb{F}_{2^{155}}$ and $\mathbb{F}_{2^{185}}$ in some standards. Curves over the field $\mathbb{F}_{2^{155}}$ were examined in [24] and it was established that the GHS attack could be used to attack approximately 2^{32} isomorphism classes of elliptic curves defined over this field. Since there are about 2^{156} isomorphism classes of elliptic curves over $\mathbb{F}_{2^{155}}$, the probability of finding one where the GHS attack is applicable is negligible. Further analysis of the GHS attack has been given in [33].

A new approach to Weil descent was given in [15]. It was shown that we can sometimes apply the GHS attack to a curve which has a large value of $m(b)$. The idea is to find an isogenous curve $E'(K)$ which has a small value of $m(b')$ and an isogeny $E(K) \rightarrow E'(K)$. The discrete logarithm problem in $E(K)$ can then be mapped in to the discrete logarithm problem in $E'(K)$ and then this can be mapped using the GHS method to the discrete logarithm problem in the Jacobian of a hyperelliptic curve of low genus. Efficient methods to find the isogenous curve and the isogeny are given in [15], as well as a study as to how effective this extension to the GHS method is in practice. This extension to the original attack can still not solve real life problems, but as it was pointed out in [33], the failure of the GHS method does not imply a failure of the Weil descent methodology, there may be other useful curves which lie on the Weil restriction $W_{E/\mathbf{k}}$ that are not constructed by the GHS method.

Summary

In this thesis we presented the known algorithms for attacking the discrete logarithm problem over the elliptic curves, the ECDLP. We started by presenting some basic definitions and facts from the theory of elliptic curves. We proceeded to describe the generic attacks, i.e. algorithms that may be used to solve the ECDLP over general elliptic curves. An in depth analysis of Pollard's ρ and λ algorithms were given. The analysis included both the original method of Pollard and the improvements given by Teske. In addition we have shown the way to parallelize the algorithms, i.e. how to run the attack over a number of processors. The parallelized ρ algorithm is the method of choice when trying to solve the ECDLP in practice. It was used to solve to ECDLP challenges set by the Certicom company.

We have also presented special algorithms for solving the ECDLP. These attacks are special in the sense that they are designed to exploit weaknesses in the structure of some classes of elliptic curves. The algorithms that were analyzed included the Menezes-Okamoto-Vanstone (MOV) algorithm based on the Weil pairing, the Frey-Ruck (FR) algorithm based on the Tate-Lichtenbaum pairing, the algorithm of Smart on the anomalous curves and the relatively new algorithm of Gaudry, Hess and Smart (GHS) based on the Weil descent methodology. These algorithms are effective in attacking classes of elliptic curves which they were designed for, but are easily circumvented in actual implementations.

References

1. L. M. Adleman, J. DeMarrais, M.-D. A. Huang: "A Subexponential Algorithm for Discrete Logarithms over Hyperelliptic Curves of Large Genus over GF(q)", TCS 226(1-2) 7-18(1999)

2. A. O. Atkin: "The number of points on an elliptic curve modulo a prime", Draft, 1998

3. R. Balasubramanian, N. Koblitz: "Improbability that an elliptic curve has subexponential discrete logarithm problem under the Menezes-Okamoto-Vanstone algorithm", Journal of Cryptology, vol. 11, p. 141-145, 1998

4. M. Bauer: "A subexponential algorithm for solving the discrete logarithm problem in the Jacobian of high genus hyperelliptic curves over arbitrary finite fields", 2000

5. I.F Blake, G. Seroussi and N.P. Smart: "Elliptic curves in cryptography", Cambridge University Press, 1999

6. I.F Blake, G. Seroussi and N.P. Smart: "Advances in elliptic curve cryptography", Cambridge University Press, 2005

7. I. Blake, K. Murty, G. Xu: "Refinements of Miller's algortihm for computing Weil/Tate pairing", 2003,

   http://eprint.iacr.org/2004/065.pdf

8. R. P. Brent: "An improved Monte Carlo factorization algorithm", BIT 20, p. 176-184, 1980

9. Certicom Corp, http://www.certicom.com

10. N. D. Elkies: "Elliptic and modular curves over finite fields and related computational issues", Computational perspectives on number theory, Stud. Adv. Math., vol. 7, AMS/ IP, p. 21-76, 1998

11. A. Enge, P. Gaudry: "A general framework for subexponential discrete logarithm algorithms", Acta arith., 102, p. 83-103, 2002

12. A. Enge: "Computing discrete logarithms in high genus hyperelliptic jacobians in provably subexponential time", Mathematics of computation, Volume 71, Nr. 238, p. 729-742

13. G. Frey: "How to disguise an elliptic curve",

    http://cacr.math.uwaterloo.ca/conferences/1998/ecc98/frey.ps

14. G. Frey, M Muller, H.-G. Ruck: "The Tate pairing and discrete logarithm apllied to elliptic curve cryptosystems", 1998

15. S. D. Galbraith, F. Hess, N. P. Smart: "Extending the GHS Weil descent attack", Advances in Cryptology - EUROCRYPT 2002, p. 29–44. Springer-Verlag, 2002

16. S.D. Galbraith, N. P. Smart: "A cryptographic application of Weil descent", Cryptography and Coding Theory, LNCS 1746, p. 191–200. Springer-Verlag, 1999

17. D. Galbraith, K. Harrison, D. Soldera: "Implementing the Tate pairing", 2002

    http://www.hpl.hp.com/techreports/2002/HPL-2002-23.pdf

18. R. Gallant, R. Lambert and S. Vanstone: "Improving the parallelized Pollard lambda search on anomalous binary curves", Mathematics of Computation, Vol. 69, No. 232, p. 1699-1705, 1999

19. P. Gaudry, F. Hess, N. P. Smart: "Constructive and destructive facets of Weil descent on elliptic curves", 2000

    http://www.hpl.hp.com/techreports/2002/HPL-2002-23.pdf

20. P. Gaudry: "An algorithm for solving the discrete log problem on hyperelliptic curves", Eurocrypt 2000

21. D. Hankerson, A. Menezes: "Elliptic curve discret logarithm problem", Auburn University, 2003

22. R. Harasawa, J. Shikata, J. Suzuki, H. Imai: "Comparing the MOV and FR reductions in elliptic curve cryptography", Advances in Cryptology-Eurocrypt, Springer-Verlag, 1999

23. F. Hess: "Generalising the GHS attack on the elliptic curve discrete logarithm problem", LMS J. Comput. Math. 7, p. 167-192, 2004

24. M. Jacobson, A. Menezes, A. Stein: "Solving elliptic curve discrete logarithm problems using Weil descent", 2001

http://eprint.iacr.org/2001/041.pdf

25. M. Jacobson, A. Menezes, A. Stein: "Hyperelliptic curves and cryptography", 2002

http://www.math.uwaterloo.ca/~ajmeneze/publications/hcc.pdf

26. N. Kanayama, T. Kobayashi, T. Saito, Sh. Uchiyama: "Remarks on elliptic curve discrete logarithm problem", IEICE Trans. fundamentals, Vol. E83-A, No. 1, 2000

27. N. Koblitz: "CM-Curves with good cryptographic properties", Advances in Cryptology - CRYPTO '91. Springer-Verlag 576, p. 279-287, 1992

28. N. Koblitz, A. Menezes, S. Vanstone: "The state of the elliptic curve cryptography", Designs, codes and cryptography, 19, p. 173-193, 2000

29. N. Koblitz: "P-adic numbers, p-adic analysis and zeta functions", Graduate texts in mathematics, Vol. 58, Springer-Verlag, 1996

30. N. Koblitz, Elliptic curve cryptosystems, Math. Comp., 1987

31. F. Kuhn, R. Struik: "Random walks revisited: Extension of Pollard's rho algorithm for computing multiple discrete logarithms", 2001

http://cr.yp.to/bib/2001/kuhn-rho.pdf

32. F. Leprevost, J. Monnerat. S. Varrette, S. Vaudenay: "Generating anomalous elliptic curves", 2004

http://lasecwww.epfl.ch/pub/lasec/doc/LMVV05.pdf

33. M. Maurer, A. Menezes, E. Teske: "Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree", 2001

http://eprint.iacr.org/2001/084.ps.gz.

34. A. Menezes, E. Teske: "Cryptographic implications of Hess' generalized GHS attack", Applicable algebra in Engineering, communication and computing, Vol. 16, No. 6, p. 439-460, Springer-Verlag, 2006

35. A. Menezes, M. Qu: "Analysis of the Weil descent attack of Gaudry, Hess and Smart", 2001

www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-48.ps

36. A. Menezes, T. Okamoto, S. Vanstone: "Reducing elliptic curve logarithms to logarithms in a finite field", Information theory, IEEE Transactions, Vol. 39, No. 5, p. 1639-1646, 1993

37. A. Menezes: "Elliptic curve public key cryptosystems", Kluwer Academic Publishers, 1993

38. A. Menezes: "The elliptic curve discrete logarithm problem (ECDLP)", 2001,

www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1028_ecdlp.pdf

39. A. Menezes, Y.-H. Wu, R. Zuccherato: "An elementary introduction to hyperelliptic curves", 1996

http://www.certification.tn/fileadmin/Ecolecrypto/Avanzi/Menezes_

Wu_Zuccherato_-_Introduction_to_Hyperelliptic_Curves.pdf

40. V. Miller: "Use of elliptic curves in cryptography", Advances in cryptology, CRYPTO'85, Springer-Verlag, 1986

41. A. Miyaji: "Curves over Fp suitable for cryptosystems", Lecture notes in computer science, Vol. 718, 1992

42. P. C. van Oorschot, M. J. Wiener: "Parallel collision search with cryptanalytic applications", Journal of Cryptology, 1999

43. J. M. Pollard: "Kangaroos, Monopoly and Discrete Logarithms", Journal of cryptology, 2000

44. H. G. Ruck: "The Tate pairing on elliptic curves", ECC'98, Waterloo, 1998

45. T. Saito, Sh. Uchiyama: "A remark on the MOV algorithm for non-supersingular elliptic curves", 2001

http://cnscenter.future.co.kr/resource/crypto/algorithm/ecc/e84-a_5_1266.pdf

46. R. Schoof: "Nonsingular plane cubic curves over finite fields", Journal of combinatorial theory, Series A, Vol. 46, No. 2, 1987

47. Rene Schoof: "Counting points on elliptic curves over finite fields", Journal de Theorie des Nombres, tome 7, nr. 1 (1995), p. 219-254

48. V. Schoup: "Lower bounds for discrete logarithm and related problems", Advances in Cryptology-EUROCRYPT '97, Springer-Verlag LNCS, p. 313-328, 1997

49. I. A. Semaev: "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p", Mathematics of computation, Vol. 67, No. 221, 1998

50. J. Shikata, Y. Zheng, J. Suzuki, H. Imai: "Optimizing the Menezes-Okamoto-Vanstone (MOV) algorithm for non-supersingular elliptic curves", Advances in cryptology-Asiacrypt '99, p. 86-102, 1999

51. J. Shikata, Y. Zheng, J. Suzuki, H. Imai: "Realizing the Menezes-Okamoto-Vanstone (MOV) reduction for ordinary elliptic curves", 2000
http://www.sis.uncc.edu/~yzheng/publications/files/ieiceE83-2k-4.pdf

52. J. H. Silverman: "The arithmetic of elliptic curves", GTM 106, Springer-Verlag, 1986

53. J. H. Silverman: "Advanced Topics in the Arithmetic of Elliptic Curves", Springer-Verlag, 1995

54. N.P. Smart: "How secure are elliptic curves over composite extension fields", 2000
http://www.iacr.org/archive/eurocrypt2001/20450030.pdf

55. N. P. Smart: "The discrete logarithm problem on elliptic curves of trace one", Journal of cryptology, Vol. 12, No. 3, 1999

56. Standards for efficient cryptography. SEC2: Recomended elliptic curve domain parameters. Version 1.0, http://www.secg.org/, 2000

57. M. Stobauer: "Efficient algorithms for pairing based cryptosystems", diploma thesis, Darmstad university of technology, 2004

58. C. Studholme: "The discrete logarithm problem", 2001
http://www.cs.toronto.edu/~cvs/dlog/research_paper.pdf

59. E. Teske: "Square-root algorithms for the discrete logarithm problem", 2001
www.math.uwaterloo.ca/~eteske/squareroots.ps

60. E. Teske: "Computing discrete logarithms with the parallelized kangaroo method", 2001

    http://www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-01.ps

61. E. Teske: "Speeding up Pollard's rho method for computing discrete logarithms", Technical report No. TI-1/98, Technische Hochschule Darmstadt, 1998

62. E. Teske: "On random walks for Pollard's rho method", Mathematics of computation, Vol. 70, p. 809-825, 2001

63. S. Vanstone: "ECC Holds Key to Next-Gen Cryptography", Certicom Corporation, 2004