UNIVERSITATEA “ALEXANDRU IOAN CUZA” IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Factorization and DLP in a Subexponential Setting

propusă de

Bogdan-Gabriel Ursu

Sesiunea: Iulie, 2015

Coordonator științific

Prof. Dr. Ferucio Laurențiu Țiplea

UNIVERSITATEA ALEXANDRU IOAN CUZA IAȘI

FACULTATEA DE INFORMATICĂ

Factorization and DLP in a Subexponential Setting

Bogdan-Gabriel Ursu

Sesiunea: Iulie, 2015

Coordonator științific

Prof. Dr. Ferucio Laurențiu Țiplea

Declaraţie Privind Originalitatea şi Respectarea Drepturilor de Autor

Prin prezenta declar că Lucrarea de Licenţă cu titlul **Factorization and DLP in a Subexponential Setting** este scrisă de mine şi nu a mai fost prezentată niciodată la o alta facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imagini etc. preluate din proiecte *open-source* sau alte resurse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, 24 Iunie 2015 Absolvent Bogdan-Gabriel Ursu

_____

(semnătura în original)

Declaraţie de Consinţământ

Prin prezenta declar că sunt de acord ca Lucrarea de Licenţă cu titlul **Factorization and DLP in a Subexponential Setting**, codul sursă al programelor şi celelate conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.
De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza, Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, 24 iunie 2015 Absolvent Bogdan-Gabriel Ursu

_____

(semnătura în original)

Contents

Introduction The domain concerned by the present paper dates back to ancient times, when the need for secure communication meant turning to the art of cryptography. As times passed by, cryptography has increasingly ceased being an art, turning into a field of science. In the seventeenth century, Fermat noticed that a good way of factorizing big integers was by trying to construct pairs of squares, equal to one another modulo the number wanted factorized. A long time passed by, and, during the last century, the Belgian mathematician Kraitchik proposed a method using the so-called Kraitchik's polynomial. In 1977, along with the realisation of the computational difficulty of integer factorization, the RSA was published. The RSA cryptosystem employed with carefully selected parameters is still in widespread use today. As an alternative to the RSA, the ElGamal encryption algorithm has been proposed in 1985. This paper briefly discusses the NFS, whose variants are asymptotically the fastest known algorithms for integer factorization. Then we describe a method far better than square root attacks to computing discrete logarithms in prime and finite fields. We then present Coppersmith's algorithm for prime fields, which although published in 1986 remains the fastest method of computing logarithms in prime fields.

The work done in this subfield of cryptography, along with the increase in computational power every year, means that security parameters have to become larger continuously. Advanced methods for computing discrete logarithms in finite fields, like the function field sieve and its refinements continuously establish new records. For the moment, the state of the art in computing logarithms in finite fields is due to the work of R. Barbulescu et al., who have published an heuristic algorithm which is quasy-polynomial for fields of small characteristic.

Momentarily, the current trend in cryptography is to rely on the ECDLP, which is rather resistant to sub-exponential attacks, mainly due to an inability to properly define smoothness on elliptic curves. Unlike in the prime field case, we are unable to lift points on the elliptic curve defined over a finite field to points of the same elliptic curve but defined, for instance, over $\mathbb{Q}$. That is, it is possible to lift elements, but their representation is impractically large. We present one of the proposed alternatives to elliptic curves, the hyperelliptic curves. For these algebraic curves there exist efficient index-calculus attacks in the case of small and large genus, and of particular interest here are curves of genus 2, on which square root attacks perform better than index calculus.

Contributions The theoretical part of this paper is represented by a survey of the domain. Due to the vastness of this field, we have been forced to restrict ourselves to some specific algorithms or techniques. Special attention has been given to the field of hyperelliptic curves, which encompasses approximately half of the present work. As a practical application, we have implemented the linear sieve variant of the Coppersmith's algorithm, which is usually employed by modern algebra systems when, due to the choice of the prime number, we cannot easily find a convenient unique factorization domain. Concerning the case when the additive group corresponds to the Jacobian of a hyperelliptic curve, we provide an implementation of Rho-Pollard, along with an adaptation of Gaudry's index calculus variant for small genus curves. In order to successfully achieve this, we instantiate our own

curves based on a method described by Koblitz.

1 Smooth Integers

Sub-exponential algorithms concerning factorization and Index Calculus methods share a common history, with ideas used to solve one problem then applied to the other. This is especially true since one of the asymptotically fastest algorithms to determine discrete logarithms is an adaptation of the number field sieve. Usually, what happens is that we desire to compute the factorization of a given integer or the logarithm of an element in a generic group that provides us with no additional algebraic properties. The approaches used in the factorization and index calculus algorithms are similar, yet in some respects it may seem that they are intertwined problems behaving as opposite sides of the same coin. For example, in factorization algorithms we usually coalesce a series of exponent vectors into a matrix, and then recover with high probability a solution by finding a linear dependency between the rows of the matrix. On the other hand, in index calculus methods, we build a matrix that is slightly larger than the former, we hope that it is a full column rank one, and then we attempt to solve a linear system of equations. Sometimes during the index calculus phase we might need to use a fast factorization algorithm in order to find relations, and so forth. In index calculus we attempt to detect smoothness by lifting elements, whilst in some fast factorization approaches we do the opposite.

Smoothness

Definition 1.1. An integer a is said to be $y - smooth$ if $\forall p$ a prime integer with p/a, we have $p \leq y$

For a random integer a, what is the probability that it will be $y - smooth$? Let $\psi(X, Y)$ denote the number of integers $b \leq X$ that are $Y - smooth$, $X \in \mathbb{Z}$. Then the probability of a being smooth is $\frac{\psi(X,Y)}{X}$ and the number of random picks from $[1, X]$ until we get a smooth integer is $\frac{X}{\psi(X,Y)}$ [9] and $\frac{\pi(Y)*X}{\psi(X,Y)}$ computational steps(if we use trial division). If one uses a rudimentary factorization algorithm, like the Kraitchik polynomial or the continuous fraction method, one needs to examine around $\pi(X, Y)$ integers to ensure that a linear dependency is uncovered. This means around $\frac{\pi(Y)^2*X}{\psi(X,Y)}$ steps, and trying to analytically minimise Y while fixing X yields that the minimum value for Y is $e^{\frac{1}{2}\sqrt{log(X)*log(log(X))}}$ and the minimum number of steps of obtaining the dependencies is $e^{2\sqrt{log(X)*log(log(X))}}$ [9].

Definition 1.2. L-notation: The L function is defined as $L_n(\alpha, c) = e^{c*log(n)^{\alpha}*log(log(n))^{1-\alpha}}$

Remark: We usually take α in $[0, 1]$, for $\alpha = 0$ we get that $L_n(0, c) = log(n)^c$, thus polynomial in the size of n. On the other hand, for $\alpha = 1$, we have that $L_n(0, c) = n^c$, which is exponential in the size of n, even if we fix c.
When $\alpha \in (0, 1)$ we say that we are between polynomial and exponential, thus sub-exponential complexity.

Property 1.3. Behaviour of the L-function, asymptotically as n $\to +\infty$

1. $L[\alpha_1, c_1] = o(L[\alpha_2, c_2])$, when $(\alpha_1 \leq \alpha_2)$ or $(\alpha_1 = \alpha_2$ and $c_1 < c_2)$.
2. $L[\alpha_1, c_1] * L[\alpha_2, c_2] = L[\alpha_2, c_1 + o(1)]$
3. $L[\alpha_1, c_1] * L[\alpha_2, c_2] = L[\alpha_2, c_1 + c_2]$

Definition 1.4. $L_n[\alpha] = L_n[\alpha, c]$ for some constant c.

Theorem 1.5. *A uniformly random integer smaller than X is* $L_X(\alpha, c) - smooth$ *with probability* $\frac{1}{L_X(1-\alpha, \frac{1-\alpha}{c})}$ *when* $X \to +\infty$.

This result is due to Canfield, Erdos and Pomerance, therefore we will refer to it as the CEP theorem. It should be noted, that according to [9], this result has been proven under the assumption that smooth integers are randomly distributed in $\mathbb{Z}$.
This crucial theorem gives us an idea as to the density of the smooth numbers in the ring of integers. Namely, they are relatively quite common. This is the fundamental reason for which factorization of integers and index calculus methods in $\mathbb{F}_p$ are not only feasible, but actually the best known methods for breaking the factorization or the DLP.

2 A special case of the NFS

The main difference between the number field sieve and other factorization methods is the fact that breaking from approaches like the continued fraction method or the quadratic sieve, we attempt to use other algebraic structures than the ring of integers. All this is in the hope of using stronger algebraic properties.

In the NFS, we still use Fermat's approach of constructing squares $a^2 \equiv b^2 \ mod \ n$, in the hope that rewriting it as $(a-b)(a+b) \equiv \ 0 \ mod \ n$ and computing $gcd(a-b, n)$ we will stumble upon a nontrivial factor of n. However, unlike the other aforementioned factorization methods, we do not have one side of all the collected relations already a square, thus we will deal with non-squares on both sides. This presentation follows the outline from [2], along with theoretical results from [7] and [11].

Setup of the NFS, n is the number we desire to factorize

1. First choose d such that $1.5 * (\frac{d}{log(2)})^d < n$. Write n in base m notation, from [2] we know that from the choice of d, $n < 2 * m^d$. This means that $n = m^d + c_{d-1} * m^{d-1} + ... + c_0$.
2. Let $f(x) = x^d + f_{d-1} * x^{d-1} + ... + f_1 * x + f_0$. From $f(m) = n$ we have that $f(m) \equiv 0 \ mod \ n$. When f is reducible, then from [2], a nontrivial factorization of $f(x) = g(x)h(x)$ implies a

nontrivial factorization of n as $g(m) * h(m)$. If this is the case, we are very fortunate and we have decomposed n. Otherwise, we can assume that f is irreducible.

3. If f is irreducible, then let α be one of its roots, $\alpha \in \mathbb{C} \setminus \mathbb{Q}$ (it is well known that reducibility over $\mathbb{Q}$ implies reducibility over $\mathbb{Z}$). Therefore, f is the minimal polynomial of α over $\mathbb{Q}$.

4. We are going to need a homomorphism ϕ, from the number field $\mathbb{Q}[\alpha] \rightarrow \mathbb{Z}_{/n\mathbb{Z}}$. The only condition we need to impose here is that $\phi(\alpha) = m$. Since ϕ is a ring homomorphism, we get for free a very nice property, that we are going to use extensively from now on:

$$\phi(\sum_{i=0}^{d-1} c_i * \alpha^i) = (\sum_{i=0}^{d-1} c_i * m^i) \; mod \; n$$

From basic field theory, the fact that f is irreducible ensures maximality of the ideal generated by f, and we have the isomorphism $\mathbb{Q}[\alpha] \simeq \mathbb{Q}[x]_{/(f(x))}$. From this, it follows that every element $\theta \in \mathbb{Q}[\alpha]$ can be written as $\theta = \sum_{i=0}^{d-1} c_i \alpha^i$ which means that the relation above is actually the definition of our homomorphism. In fact, $\mathbb{Q}[\alpha]$ can be viewed as a vector space over $\mathbb{Q}$, or as a direct sum: $\mathbb{Q}[\alpha] \simeq \mathbb{Z} * 1 \oplus \mathbb{Z} * \alpha \oplus ... \oplus \mathbb{Z} * \alpha^{d-1}$. Concerning ϕ, it can be shown that it is a ring homomorphism.

Target relations Our goal is to compute a set $\mathbf{S} = \{(a, b) \; / \; (a, b) = 1\}$ for which the following two relations hold:

1. $\prod_{(a,b)\in\mathbf{S}}(a - b * m) = v^2$, for some $v \in \mathbb{Z}$
2. $\prod_{(a,b)\in\mathbf{S}}(a - b * \alpha) = \gamma^2$, for some $\gamma \in \mathbb{Z}[\alpha]$

Definition 2.1. A complex number is an algebraic integer if and only if it is a root of some monic polynomial $f \in \mathbb{Z}[x]$.

Theorem 2.2. [7]*: For any algebraic integer $\beta \in \mathbb{C}$ and any $f \in \mathbb{Z}[x]$ monic and of minimal degree with $f(\beta) = 0$, f is irreducible over $\mathbb{Q}$.*

This result is quite important concerning the way we keep track of our algebraic integers. Instead of looking at a multitude of polynomials over $\mathbb{Q}$, we can choose to consider only polynomials over $\mathbb{Z}$. In the following, we are going to define a measure of size for $\mathbb{Q}[\alpha]$, which would allow us to distinguish between $small$ and $large$ integers.

Definition 2.3. The norm function on $\mathbb{Q}[\alpha]$ is defined as $N(\theta) = N(\sum_{i=0}^{d-1} c_i \alpha^i) = \prod_{\alpha_j=1} \sum_{i=0}^{d-1} c_i \alpha_j^i$.

We are allowed to do this because $\mathbb{Q}[\alpha]$ is a field and therefore an Euclidean domain as well, algebraic structures on which we may define the norm function. The norm function N behaves as a morphism with respect to multiplication. From [7], we have two additional properties. Firstly,

$N(\theta) \in \mathbb{Q}, \forall \theta \in \mathbb{Q}[\alpha]$. Moreover, if θ is an algebraic integer, $N(\theta) \in \mathbb{Z}$. The first property can be proven with ease, whilst the proof for the second is more theoretical and we will not describe it here.

Theorem 2.4. [7]*: The set of algebraic integers in* $\mathbb{C}$*, denoted by* $\mathbb{A}$*, is a ring.*

The intersection $\mathbb{Q}[\alpha] \cap \mathbb{A}$ is the number ring the provides the underlying algebraic structure for the number field sieve. Please note that this is actually the set of algebraic integers of $\mathbb{Q}[\alpha]$. Moreover, $\mathbb{Q}[\alpha] \cap \mathbb{A} \supseteq \mathbb{Z}[\alpha]$. When this inclusion is strict, many difficulties that prevent us from forming squares in $\mathbb{Z}[\alpha]$ arise. We are not going to describe the algorithm in the case of strict inclusion, we assume that $\mathbb{Q}[\alpha] \cap \mathbb{A} = \mathbb{Z}[\alpha]$.

Remark: Any $a - b * \alpha$ is an algebraic integer, because of the ring structure of the number ring associated to $\mathbb{Q}[\alpha]$ (and because α is of course an algebraic integer). Moreover, since we have introduced the norm function of our measure of size, we can naturally define a notion of smoothness in $\mathbb{A} \supseteq \mathbb{Z}[\alpha]$:

Definition 2.5. An algebraic integer θ is $y - smooth$ if $N(\theta) \leq y$.

Let us conclude this chapter by pointing towards the terminology employed by [12]: A number field is a finite field extension of $\mathbb{Q}$ and a number ring is a subring of a number field. Therefore, in our setup $\mathbb{Q}[\alpha]$ is the number field and $\mathbb{Z}[\alpha]$ is one of its number rings.

2.1 Sieving, preliminaries

This section follows the approach given in [2]. We should start by describing the sieving process for integers, considering the case where we need to sieve for the values of the polynomial $G(a, b) = a - b * m$. The goal is to detect (a, b) for which $G(a, b)$ is $y - smooth$. Here, it suffices to see that if we fix a, then for any p prime with $p/G(a, b)$, p will also divide $G(a, b + k * p)$. This property is successfully used to sieve for the values of a polynomial in both the quadratic and multiple polynomial quadratic sieve.

Sieving for the polynomial $G(x, y)$ **will therefore be done by fixing** a **and sieving for** b**.** (at the end of each such cycle, a will be incremented - this method is called line-by-line sieving. In practice, more efficient sieves are being used).

Property 2.6. [2]: $N(a - b * \alpha) = b^d * f(a/b)$.

Proof: $N(a - b * \alpha) = \prod_{\alpha_j=1} \sum_{i=0}^{d-1} c_i \alpha_j^i$ Denote by $sum_j = \sum_{i=0}^{d-1} c_i \alpha_j^i$. Then, if by convention $\alpha = \alpha_1$, we have that $sum_1 = a - b * \alpha$ and more importantly, $sum_j = a - b * \alpha_j$. Therefore, we get that $N(a - b * \alpha) = (a - b * \alpha_1) * ... * (a - b * \alpha d)$.
Since f can also be written as $f = (x - \alpha_1) * ... * (x - \alpha_j)$, by division with b^d we get the required result. By homogenizing f to $F(x, y) = y^d * F(x, y)$, we obtain $N(a - b * \alpha) = F(a, b)$.

From [7], the norm also satisfies the following multiplicative property: $N(\theta_1\theta_2) = N(\theta_1)N(\theta_2)$. As a very simple corollary of this property, it holds that when an element $\theta \in \mathbb{Q}[\alpha]$ is a square, its norm will also be a square in $\mathbb{Q}$. Attempting to sieve for smooth values of $\prod_{(a,b)\in\mathbf{S}}(a - b * \alpha)$ using this property of the norm defeats nevertheless the purpose, since having the norm as a square does not necessarily imply that the initial element is a square. (A very simple counterexample in $\mathbb{Z}[i]$ is 3, since $N(3) = 9$ but 3 is not the square of a Gaussian integer).

Theorem 2.7. *[7]: For every $\mathbb{Q}[\alpha]$, its corresponding ring of algebraic integers $\mathbb{Q}[\alpha] \cap \mathbb{A}$ is a Dedekind domain.*

We have assumed maximality of the number ring $\mathbb{Z}[\alpha]$ in $\mathbb{Q}[\alpha] \cap \mathbb{A}$. We are going to use this theorem to argue that any prime ideal of $\mathbb{Z}[\alpha]$ is maximal. Let $P \in \mathbb{Z}[\alpha]$ be a prime ideal, and consider $P \cap \mathbb{Z}$, the intersection which will also be an ideal in $\mathbb{Z}$. But the only ideals in $\mathbb{Z}$ are of the form $p * \mathbb{Z}$ where $p \in \mathbb{Z}$ is a prime. This implies that $P \cap \mathbb{Z} = p * \mathbb{Z}$. Consider now $\mathbb{Z}[\alpha]_{/P}$, which is a field by the maximality of P, and by the definition of the characteristic, $char(\mathbb{Z}[\alpha]_{/P}) = p$.

Property 2.8. Using the previous remarks, we say that a prime $\mathbf{p} \in \mathbb{Z}[\alpha]$ lies over a unique prime p in $\mathbb{Z}$ (with $\mathbf{p} \cap \mathbb{Z} = p * \mathbb{Z}$).

It should be clear now how to move from a prime in $\mathbb{Z}[\alpha]$ to a prime in $\mathbb{Z}$. What about the reverse? First, observe that for $p \in \mathbb{Z}$, we can identify a prime ideal that lies over it by looking at the kernel of a ring homomorphism: $\psi : \mathbb{Z}[\alpha] \to \mathbb{F}_p$. Keeping in mind that $\mathbb{F}_p \simeq \mathbb{Z}_p$, we particularise ψ in a similar fashion to ϕ, by specifying the image of α.

Remark: $(f \ mod\ p)(\psi(\alpha)) = \psi(f(\alpha)) = 0$ implies that $\psi(\alpha)$ is a zero of f. We now conclude the discussion on how to uniquely identify prime elements of $\mathbb{Z}[\alpha]$. We associate the ideal generated by each such prime to the ring homomorphism ψ_{q,r_q} where:

1. $\psi : \mathbb{Z}[\alpha] \to \mathbb{F}_q$
2. $r_q = \psi(\alpha)$

Next, we should analyse the behavior of primes in the $\mathbb{Z}[\alpha]$ with respect to the algebraic integer $a - b * \alpha$ where $(a, b) = 1$. Let us consider a prime ideal $\mathbf{p} \in \mathbb{Z}[\alpha]$, $\mathbf{p} = \ker(\psi_{p,r_p})$.
$\mathbf{p}/(a - b * \alpha)$ implies that we have $p \nmid b$. Otherwise, $p \in \mathbf{p}$ and p/b would mean $b \in \mathbf{p}$ from which $a \in \mathbf{p}$, and we get the contradiction $(a, b) \neq 1$. Now $(a - b * \alpha) \in \mathbf{p}$ would mean that $a = b * \alpha$ in $\mathbb{Z}[\alpha]_{/\mathbf{p}}$ thus $a * b^{-1} = \alpha$ in $\mathbb{Z}[\alpha]_{/\mathbf{p}}$. $a * b^{-1}$ is therefore a root of $(f \mod p)$. We arrive thus at the following property, found in [11]:

Property 2.9. A prime $\mathbf{p} \in \mathbb{Z}[\alpha]$ divides $a - b * \alpha$ if and only if $f(a * b^{-1}) \equiv 0 \mod p$.

Since we have $p \nmid b$ and from the properties of the norm, we can rewrite the aforementioned result to say that:

Property 2.10. [11]: A prime $\mathbf{p} \in \mathbb{Z}[\alpha]$ divides $a - b * \alpha$ if and only if $p/N(a - b * \alpha)$. Furthermore, the exponent of $\mathbf{p} \in \mathbb{Z}[\alpha]$ is the same as the exponent of p in $N(a - b * \alpha)$.

As a side note, for primes of degree 1 in a number ring, the function that denotes the exponent of these primes is a morphism. In $\mathbb{Z}[\alpha]$, all primes are of degree 1. Let us give some additional explanations regarding what does it mean for a prime of $\mathbb{Z}[\alpha]$ to divide an integer prime. Firstly, let us remark that we cannot say for example that $\mathbf{p} = (\tilde{p})$ for some $\tilde{p} \in \mathbb{Z}[\alpha]$ and then attempt to use simple divisibility arguments for (probably complex) $\tilde{p}$. This is because $\mathbb{Z}[\alpha]$ is most likely not a principal ideal domain (our assumption for the maximality of the number ring $\mathbb{Z}[\alpha]$ only implies that we have unique ideal factorization for $\mathbb{Z}[\alpha]$).

2.2 Sieving

We already know how to sieve for the values of the polynomial $G(a, b) = a - b * \alpha$, by choosing a factor base consisting of prime integers smaller than our smoothness bound. The question now is how do we sieve for the values of $(a - b * \alpha)$? We are going to do this by sieving for the polynomial $F(a, b) = N(a, b)$. To this end, we are going to need a second factor base, which will contain all primes $\mathbf{p} \in \mathbb{Z}[\alpha]$, that are chosen as in the following.

Second factor base construction A prime $\mathbf{p} \in \mathbb{Z}[\alpha]$ belongs to the second factor base if the ring homomorphism ψ corresponding to its kernel $\mathbf{p} = \ker(\psi_{p,r_i})$ satisfies that: $p \leq$ smoothness bound of the first factor base.

2.3 Exponent vectors constructions

Denote by $\vec{v}(a - b\alpha)$ a vector which for each pair p, r has the entry $\vec{v}(a - b\alpha)_{p,r}$. By definition,

1. $\vec{v}(a - b\alpha)_{p,r} = 0$, when $ab^{-1} \not\equiv r \mod p$. (This entry in the vector is 0 if the prime $\mathbf{p}$ corresponding to (p, r) does not divide $a - b\alpha$).

2. $\vec{v}(a - b\alpha)_{p,r} =$ exponent of p in $F(a, b)$, $ab^{-1} \equiv r \mod p$. (This entry corresponds to the exponent of the prime (p, r) in $a - b\alpha$).

Why does this method of assembling squares work? Because we have the opposite implication, that the exponent of $\mathbf{p}$ corresponds precisely to the exponent of the corresponding p in $N(a - b\alpha)$. We have also stated that the order function of degree 1 primes behaves as a morphism. This means that we can collect a multitude of vectors corresponding to smooth $a - bm$ and smooth $a - b\alpha$ and then use linear algebra modulo 2 to recover a relation of the form $\phi(\gamma^2) = v^2 \mod n$ for $\gamma \in \mathbb{Z}[\alpha]$ and

$v \in \mathbb{Z}$. Then we can write that $\phi(\gamma)^2 = v^2 \mod n$ and with high probability recover a nontrivial factor of n.

Non-maximality of $\mathbb{Z}[\alpha]$ When $\mathbb{Z}[\alpha] \subset \mathbb{Q}[\alpha] \cap \mathbb{A}$, a multitude of problems arise, starting with the fact that we do have unique ideal factorization in $\mathbb{Q}[\alpha] \cap \mathbb{A}$ but not in $\mathbb{Z}[\alpha]$. To compensate for this, it is sufficient in practice to use the two following properties, which state that in order to form squares in $\mathbb{Z}[\alpha]$ we need that the sum of the chosen exponent vectors be 0 modulo 2 and that we might also need certain Jacobi symbols to be positive. The theoretical motivation for this is beyond the scope of this presentation, an interested reader should consult [11].

Property 2.11. [2]: For $S = \{(a, b \in \mathbb{Z}) \ /(a, b) = 1, \ N(a - b\alpha \leq B)\}$, if $\prod_{(a,b)\in S}(a - b\alpha) = \gamma^2, \ \gamma \in \mathbb{Q}[\alpha] \cap \mathbb{A}$. then $\sum_{(a,b)\in S} \vec{v}(a - b\alpha) \equiv 0 \mod 2$.

Property 2.12. [2]: For $q \in \mathbb{Z}$ an odd prime and $s \in \mathbb{Z}$,with $f(s) \equiv 0 \mod q$ and $f^{'}(s) \not\equiv 0 \mod q$. If $q \nmid a - bs \ \forall (a, b) \in S$ and $f^{'}(\alpha)^2 \prod_{(a,b)\in S}(a - b\alpha)$ is a square in $\mathbb{Z}[\alpha]$, then: $\prod_{(a,b)\in S}(\frac{a-bs}{q}) = 1$.

Before concluding this section and moving to index calculus methods, let us give the heuristic complexity of the NFS, according to a proof from [2]: $L_n[\frac{1}{3}, \sqrt[3]{\frac{32}{9}}]$.

3 Index Calculus Outline

The structure follows the presentation from [8], we are first given a cyclic group $(\mathbf{G}, *)$ of order n, along with a generator g. Then, for a $\beta = g^\alpha$, our goal is to determine $log_g(\beta) = \alpha$. The algorithm consists of four stages, namely:

1. **Factor Base selection**
   Choose a subset $\mathbf{S}$ of $\mathbf{G}$ such that a significant portion of elements of $\mathbf{G}$ can be expressed solely as a product of elements from $\mathbf{G}$

2. **Collect linear relations**
   Compute linear relations involving the logarithms of elements from the factor base $\mathbf{S}$. The simplest approach of doing this is to randomly select an element k and then try to factor g^k using only elements from the factor base. If this is possible, apply logarithm to both sides of $g^k = \prod_{s\in\mathbf{S}} s$ to obtain the linear relation: $k \equiv \sum_{s\in\mathbf{S}} log(s)$ mod n.

   We need to perform this step repeatedly, trying to obtain $|\mathbf{S}|$ independent relations, which will tell us that the system of linear equations has a unique solution.

3. **Compute the logarithms of all elements in the factor base**
   Solve the aforementioned system of equations to determine the exact value of $log_g(s)$ for each $s \in \mathbf{S}$.

4. **Determine y**

   Compute y using a probabilistic approach, by first choosing random exponents k and then trying to factor $\beta * g^k$ using only elements of the factor base: $\beta * g^k = \prod_{s \in \mathbf{S}} s$. Taking logarithms of both sides as before, yields $k + \alpha = k * log_g(\beta) = \sum_{s \in \mathbf{S}} log(s) \ mod \ n$ and therefore $\alpha = \sum_{s \in \mathbf{S}} log(s) - k \ mod \ n$.

   Note that we could also approach this step as we have done with step 2, by randomly choosing k and computing $\beta^k = \prod_{s \in \mathbf{S}} s$, then taking logarithms to obtain $k * \alpha = k * log_g(\beta) = \sum_{s \in \mathbf{S}} log(s) \ mod \ n$. Nevertheless, it is inefficient to choose this route since we could avoid dividing by k in the first place.

3.1 Choice of the factor base for $\mathbb{F}_p$ and $\mathbb{F}_{p^n}$

Firstly, for $\mathbb{F}_p$, one may chose a smoothness bound according to the CEP theorem and then select as a factor base, the first smooth integer primes. In the case of $\mathbb{F}_{p^n}$, since we may view every element as a polynomial, it is satisfactory to choose a factor base consisting of all irreducible polynomials of degree smaller than the chosen smoothness bound. [8]

Example in a prime field Let us consider the following setup: we work in $\mathbb{Z}_{/1031\mathbb{Z}}$, and we are given a generator of the multiplicative subgroup, $g = 1029$. The smoothness bound has been arbitrarily chosen as $B = 11$, therefore our factor-base consists all primes smaller or equal to 11, $S = \{2,\ 3,\ 5,\ 7,\ 11\}$. The precomputation phase requires us to obtain the logarithms of the factor-base elements, we do this by looking for smooth elements of the form g^k, with k chosen randomly. We randomly picked $k \in \{631,\ 106,\ 830,\ 824,\ 596\}$ from $\mathbb{Z}_{/1030\mathbb{Z}}$. Now we compute:

- $g^{631} \equiv 140 \equiv 2^2 * 5 * 7 \mod 1031$
- $g^{106} \equiv 20 \equiv 2^2 * 5 \mod 1031$
- $g^{830} \equiv 1000 \equiv 2^3 * 5^3 \mod 1031$
- $g^{660} \equiv 140 \equiv 2^2 * 3 * 5 * 11 \mod 1031$
- $g^{596} \equiv 140 \equiv 2^6 * 3 * 5 \mod 1031$

All these linear dependencies make up a $5x5$ matrix, we end the precomputation phase of computing logarithms for each factor-base element by solving the linear system:

$$\begin{pmatrix} 2 & 0 & 1 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 3 & 0 & 3 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 \\ 6 & 1 & 1 & 0 & 0 \end{pmatrix} x = \begin{pmatrix} 631 \\ 106 \\ 830 \\ 660 \\ 596 \end{pmatrix}. \text{ We obtain x=} \begin{pmatrix} -514 \\ 486 \\ 104 \\ -505 \\ 232 \end{pmatrix}$$

With the precomputation phase over, let us find the logarithm of 521, which is a prime number and will not factor immediately. Assuming that we randomly picked 2, we see that $g^2 * 521 \equiv 2 * 11 \mod 1030$. This means that we can compute $log_{1029}(521) \equiv -514 + 232 - 2 \equiv -284 \mod 1030$. In practice, it is usually much faster to compute discrete logarithms once the precomputation phase has been completed, since we need only one smooth element. It should be noted that this behaviour comes in contrast with the workings of the quadratic or the number field sieve: here we may reuse the results from the precomputation phase to compute individual logarithms faster.

4 Coppersmith's Index Calculus in $\mathbb{Z}_p^*$

Generally, index calculus relies at some point on lifting some intermediary results to an algebraic structure where we can define an advantageous notion of smoothness. For computations in prime fields, the idea is to lift residue classes modulo the prime integer p to integers in $\mathbb{Z}$, which can be easily tested for smoothness. For the rest of this section, unless stated otherwise, the material covered is according to [1].

For $X, Y \in \mathbb{Z}$, recall that $\frac{\psi(X,Y)}{X}$ represented the probability of randomly picking an $Y - smooth$ integer $\leq$ than X, from the interval $[1, X]$. Let $p \in \mathbb{Z}$ be a prime, according to [1], the following theorem holds:

Theorem 4.1. [1]*:* $\frac{\psi(X,Y)}{X} = L_p[\frac{1}{2}, -\frac{\alpha}{2*\beta}]$ *when* $X = p^\alpha$ *and* $Y = L[\frac{1}{2}, \beta]$

Proof: directly from the **CEP** theorem.

In the following, we would be going through the outline of the Coppersmith algorithm. Although seemingly simplistic, this is actually the fastest algorithm we currently know that is capable of solving the DLP in $\mathbb{Z}_p^*$. Its efficiency comes from the ingenious combination of two strategies that are not necessarily compatible with each other:

1. The first strategy can even be traced back to the continued fraction method of integer factorisation. There, one used a_i/b_i, the continued fraction convergents of $\sqrt{n}$. Then, one looked for smooth values of $Q_i(x) = a_i^2 - b_i^2 * n$, in a bid to obtain favorable exponent vectors. The chance of gaining smooth values was higher because of the known inequality $|Q_i| < 2 * \sqrt{n}$.

2. The second strategy is using a sieve, which can only happen when using favorable sets. Modern sieving methods have rendered the continued fraction method obsolete, at the cost of ever increasing values of the polynomials used, like in the Quadratic or the MPQS sieves.

Coppersmith's algorithm employs an efficient sieve while keeping the numbers we require to be smooth around $\sqrt{p}$.

4.1 Setup and choice of the factor base

In the following, we go through the outline from [1].

1. Select $\epsilon > 0$, choose $H = [\sqrt{p}] + 1$ and $J = H^2 - p$.
2. S=$\{q \,/q \in \mathbb{Z}\, prime,\ q \leq L_p[\frac{1}{2}, \frac{1}{2}]\} \cup \{H + c \,/0 \ < \ c \ < L_p[\frac{1}{2}, \frac{1}{2} + \epsilon]\}$. S contains the set of small primes, along with integers near H.
   $|S| \approx L_p[\frac{1}{2}, \frac{1}{2} + \epsilon]$

Definition 4.2. Let $\mathbf{M} = \{c_1, c_2 / c_1 \leq c_2, c_i \in \mathbb{Z}_+^*, c_2 < L_p[\frac{1}{2}, \frac{1}{2} + \epsilon]\}$, the set of all ordered pairs c_i, c_j and let $\mathbf{\Theta} = \{\theta \in \mathbb{Z}_p / \theta = ((H + c_1) * (H + c_2) \mod p)$ for some c_1, c_2 from $\mathbf{M}\}$, the set of residues $\mod p$

Property 4.3. For $H + c_1,\ H + c_2 \in S$ we have $((H + c_1) * (H + c_2) \mod p) \leq \sqrt{p} + \epsilon_2$, with ϵ_2 small.

Property 4.4. If one sets $X = p^{\frac{1}{2} + \frac{\epsilon}{2}}$ and $Y = L_p[\frac{1}{2}, 1/2]$, by applying the previous theorem, one gets $\frac{\phi(X,Y)}{X} = L_p[\frac{1}{2}, -\frac{1+\epsilon}{2}]$. Here we arrive at the heuristic assumption posed by the Coppersmith's algorithm - we assume that this probability is close to the actual probability of finding $y - smooth$ integers in $[1, X]$.

. The set $\mathbf{\Theta}$, has, under the heuristic assumption aforementioned before, the same density of $Y - smooth$ numbers as $[1, X]$. Moreover, it is upper bounded strictly by $2\sqrt{p}$.

Estimate: Let us estimate the number of $Y - smooth$ numbers we expect to produce by analysing each element of $\mathbf{\Theta}$. Indeed, the cardinality of our set is $|\mathbf{\Theta}| = \binom{L_p[\frac{1}{2}, 1/2+\epsilon]}{2} \approx \frac{L_p[\frac{1}{2}, 1/2+\epsilon]^2}{2} = \frac{L_p[\frac{1}{2}, 1+2*\epsilon]}{2}$, and the number of $Y - smooth$ integers one expects to produce is:

$$L_p[\tfrac{1}{2}, 1 + 2 * \epsilon] * \phi(X, Y)/X = L_p[\tfrac{1}{2}, 1 + 2 * \epsilon] * L_p[\tfrac{1}{2}, -1/2 - \epsilon/2] = L_p[\tfrac{1}{2}, 1/2 + 3 * \epsilon/2]$$

4.2 Sieving $\mathbf{\Theta}$

The most basic choice is a line-by-line sieve, if one wishes, more elaborate types of sieving are probably possible. Fix c_1 and a prime $q < L_p[\frac{1}{2}, \frac{1}{2}]$. For a given power of q, say $q^k < L_p[\frac{1}{2}, \frac{1}{2}]$ we are going to sieve for possible values of c_2.

1. Fix c_1 and $q^k < L_p[\frac{1}{2}, \frac{1}{2}]$
2. Choose size of array to sieve as $L_p[\frac{1}{2}, 1/2 + \epsilon]$
3. First compute $d \equiv -(J + c_1 H) * (H + c_1)^{-1} \mod q^k$, if the multiplicative inverse exists.
4. Step through the array, every $c_2 \equiv d \mod q^k$ will have the corresponding θ_{c_1,c_2} a smooth integer. [1]

4.3 Linear relations

Using trial division or more efficient algorithms on the numbers we identify as smooth integers with high probability, we obtain, from [1]: $\theta_{c_i,c_j} = (H + c_1)(H + c_2) = J + (c_1 + c_2)H + c_1c_2 = \prod q_i^{k_i} \mod p$, and by taking logarithms of both sides with respect to a generator g of $\mathbb{Z}_p^*$ we obtain:

$$log_g(H + c_1) + log_g(H + c_2) = \sum k_i * log_g * q_i \mod p - 1.$$

Remark : At this point, a befuddled reader might remember the outline of the index calculus algorithm and ask how are we going to compute $log_g(H+c_i)$. In reality, we cannot do that separately, that is why we have included it into the factor base S!

Another problem that we encounter at this point is that of the zero divisors of $p-1$, since the ring of exponents $\mathbb{Z}_{/(p-1)\mathbb{Z}}$ is not even an integral domain. What saves the day is the Chinese Remainder Theorem, applied on the ideals $(q_i^{k_i})$ where $p - 1 = \prod q_i^{k_i}$(of course, k_i will be the greatest power such that $q_i^{k_i}/p - 1$, otherwise our ideals will not be relatively prime). Indeed, even if factorizing p is unfeasible, we can first identify the small prime factors of $p-1$. Then we can just assume that our element $\alpha \mod p - 1$ is invertible. If we find it to be a zero divisor of p-1, compute $(\alpha, p - 1)$ and retrieve a non-trivial factor of p-1.

Complexity The heuristic complexity of the preprocessing phase for Coppersmith's algorithm is $L_p[\frac{1}{2}, 1 + 2 * \epsilon]$. [1]

4.4 Computation of the logarithm

Going back to the DLP, assume that we are given a group element β of $\mathbb{Z}^*_{/p\mathbb{Z}}$, and we are being asked to compute $\alpha \in \mathbb{Z}_{/(p-1)\mathbb{Z}}$ such that $g^\alpha = \beta$. The algorithm goes through three stages, which we are only going to sketch here.

1. Using an efficient integer factorization algorithm, randomly find a suitable r such that $g^r\beta$ can be written as the product of medium-sized primes p_i.

2. For each p_i, compute $log_g(p_i)$.

3. Combine the logarithms to recover $log_g(\beta)$.

Prime decomposition Factorizing means that we are going to obtain both small primes (smaller than $L_p[1/2, 1/2]$) and medium large primes (bigger than the small primes but still smaller than $L_p[1/2, 2]$). We need to keep track of both the small and the large primes, observe that we already know the logarithms of the small primes (sometimes not all of them) from the precomputation phase.

Computing logarithms of medium primes Assume that we need to compute just the logarithm $log_g p_i$ of the medium prime p_i. This is being done in three steps:

- By sieving a $L_p[\frac{1}{2}, \frac{1}{2}]$ interval, we find a smooth integer a around $\frac{\sqrt{p}}{p_i}$.
- By sieving a $L_p[\frac{1}{2}, \frac{1}{2}]$ interval, recover smooth integer b around $\sqrt{p}$ such that $abp_i - p$ is smooth.

Intuitively, the rationale for such an approach is that with such a choice of a and b, abp_i will not be significantly larger than p, thus $abp_i - p$ is small and more likely to be smooth. For more details regarding the complexity requirements of this stage, please refer to [1].

4.5 Implementing the linear sieve

During this paragraph, we are going to give more details regarding the implementation of the linear sieve designed as part of the practical work for this paper. Firstly, some clarifications regarding the general outline should be given:

- In their current form, the linear relations concerning base logarithms that we have collected are not bound to any generator of $\mathbb{Z}_{/p\mathbb{Z}}$. In order to make this binding, we introduce an extra relation for the chosen generator, namely $log_g g = log_g g$. Assuming that we introduce this as the last row in our matrix, the right-hand side of the linear algebra step will be the vector $(0, 0, ...0, 1)$. [1]

- However, in order to do this, we assume that the given generator is a small prime integer ($\leq L_p[\frac{1}{2}, \frac{1}{2}]$). If this is not the case, an easy workaround exists: find a generator g' such that $g' \leq L_p[\frac{1}{2}, \frac{1}{2}]$. Then we compute using the linear sieve two discrete logarithms, one of them being $log_{g'} g$ and the other being the logarithm of the target element to the base g'.

- For small values of p, when we sieve with respect to the prime power q^k, it might hold that $(H + c_1)(H + d + \gamma q^k) - (H + c_1)(H + d) \geq p$. When this type of overlapping happens, we need to stop the sieving process for q^k.

- Not all elements of the form $H + c_i$, $c_i \leq L_p[\frac{1}{2}, 1/2 + \epsilon]$ should be included into the factor-base, in fact these elements are used as helpers to obtaining relations between the small primes.

We have already mentioned some strategies for tackling the problem of the underlying algebraic structure for the linear algebra step: the ring $\mathbb{Z}_{/(p-1)\mathbb{Z}}$. My approach is to reduce the computations to finite fields, using the CRT and Hensel Lifting, which is going to be detailed next.

The difficulty with finite fields When we are dealing with finite fields that are not of prime order, we might be tempted to use the same strategy, and perform the linear algebra step in the corresponding finite field. This is a common mistake, take for example $\mathbb{F}_{p^n}$. We might easily describe a mapping $\phi : \mathbb{Z}_{/p^n\mathbb{Z}} \to \mathbb{F}_{p^n}$, but in general, we cannot find a mapping from the finite field back to $\mathbb{Z}_{/p^n\mathbb{Z}}$ (which will also preserve addition and multiplication). The solution is described below.

Hensel Lifting Hensel lifting is a method of computing roots of a polynomial modulo powers of a prime q recursively, receiving as input a root of the polynomial modulo a power q^k (k is usually 1). First we solve the equation $Mx_0 = b \mod q$, where q is a factor of $p-1$, and M, b have integer entries. To compute x_1 such that $Mx_1 = b \mod q^2$, first observe that $x_1 \equiv x_0 \mod q$, therefore $x_1 = x_0 + qy_0$.

Thus $b \equiv Mx_1 \equiv M(x_0 + qy_0) \equiv Mx_0 + qMy_0 \mod q^2$ which implies that $b - Mx_0 \equiv qMy_0 \mod q^2$. First notice that $Mx_0 - b = cq$, which means that $c = \frac{Mx_0 - b}{q}$. But $b - Mx_0 - qMy_0 = dq^2$. Simplify by q to obtain $-c \equiv My_0 \mod q$ which is the equation we now need to solve. [1]

Unlike the classic Hensel Lifting algorithm, which performs squaring, this approach is more inefficient. However, it is still polynomial in the size of p.

A toy example Let us consider $\mathbb{Z}_{/1283\mathbb{Z}}$, where 1283 is a safe prime. We are asked to compute the logarithm to the base 2 of 1094. Experimentally, for such small values of the prime field, we need to consider primes larger than $L_p[\frac{1}{2}, \frac{1}{2}]$ for our factor-base. Here we take this bound to be four times the default value, 24. We let all the other parameters follow the description given by [1]. Performing the linear equations collection step, we obtain an invertible sparse matrix of 58 by 29. We have implemented the algorithm using the SAGE computer algebra system. This decision was primarily driven by SAGE's implementation of sparse linear algebra over finite fields.
In $\mathbb{Z}_{/1283\mathbb{Z}}$, we happily discover that 2 is a primitive element. Upon the completion of the preprocessing phase we add an additional row to our matrix, corresponding to the relation $log_2 2 = log_2 2$. Sometimes it might occur that at the end of the preprocessing we do not recover the logarithm of every element in the factor-base. Indeed, we do not need all of them, having most of them suffices. Solving the linear systems we obtain the following two vectors:
$\begin{pmatrix}1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$, in $\mathbb{Z}_{/2\mathbb{Z}}$ and $\begin{pmatrix}1 & 573 & 565 & 386 & 406 & 373 & 223 & 280 & 176\end{pmatrix}$, in $\mathbb{Z}_{/641\mathbb{Z}}$
We use the CRT and obtain the logarithms of the factor-base elements:
$\begin{pmatrix}1 & 1214 & 565 & 1027 & 406 & 1014 & 864 & 280 & 817\end{pmatrix}$, in $\mathbb{Z}_{/(p-1)\mathbb{Z}}$.

Computing the individual logarithm We are now ready to move to the second part of the algorithm and compute $log_2 1094$.

1. Choose $r = 1145$ and discover that $2^r 1094 \equiv 1074 \equiv 2 * 3 * 179 \mod p$.
2. $p_i = 179$ is a medium prime, we discover it by first finding small $a = 7$.
3. We find $b = 46$, where $log_2 b = 1 + 817 = 818$. b satisfies $ab * 179$ smooth.
4. However, $ab * 179 - p = 56355$ is terribly large. For $p \gg p_i$, we do not have this problem and we will encounter a small smooth value. We will proceed anyway:
5. $56355 = 3 * 5 * 13 * 17^2$ and its logarithm is $1214 + 565 + 1014 + 2 * 864 \equiv 675 \mod p - 1$.

6. $log_2 179 \equiv 675 - 1027 - 818 \equiv 112 \mod p-1$.

7. $log_2 1094 = 1 + 1214 + 112 - 1145 \equiv 182 \mod p-1$.

Applicability of the linear sieve In practice, most computer algebra systems prefer Coppersmith's Gaussian integers sieve to the linear sieve. The reason for this is the very low density of the linear equations collected. The strategy of solving the equations in prime fields and then recomposing them is supported by an additional argument: most cryptosystems rely on safe or strong primes because Pohlig-Hellman reduction would transfer the DLP in the group of high order to the DLP in the proper subgroup of highest order. For safe primes, the linear sieve performs only two linear algebra steps, for 2 and the equivalent Sophie-Germain prime, and then recomposes them without the need to resort to any lifting.

4.6 Timing the linear sieve against Rho-Pollard

We have implemented Rho-Pollard for prime fields and tested it against our linear sieve, in SAGE 6.7. In order to obtain a speedup, we also take into account the fact that the multiplicative group of every prime field is cyclic. This implies that we obtain an automorphism $\phi : \mathbb{Z}^*_{/p\mathbb{Z}} \to \mathbb{Z}^*_{/p\mathbb{Z}}$, $\phi(x) = x^{-1}$. Some basic measurements for the unoptimized algorithm, in seconds:

$\mathbb{Z}_{/p\mathbb{Z}}$	Linear Sieve	Rho-Pollard
$\mathbb{Z}_{/1283\mathbb{Z}}$	0.41	0.0009
$\mathbb{Z}_{/114467\mathbb{Z}}$	7.55	0.03
$\mathbb{Z}_{/10021163\mathbb{Z}}$	36.21	16.77
$\mathbb{Z}_{/140296907\mathbb{Z}}$	116.85	176.95

For groups larger than $\mathbb{Z}_{/140296907\mathbb{Z}}$, we expect Rho Pollard to start behaving much worse than the linear sieve. Therefore, this graphic is only meant to illustrate the existence of the point from which index calculus behaves better.

5 Coppersmith's $\mathbb{Z}$[i] method

Using the general outline of the algorithm described before, the authors in [1] provide a particularly useful method of using the Gaussian integers to compute DLs.
When $r \in \{-1, -2, -3, -7, -11, -19, -43, -67, -163\}$, the ring $Z[\sqrt{r}]$ is an UFD.

Setup: Consider a small negative integer r, a quadratic residue $\mod p$(i.e. $r \in \mathbb{Z}^*_-, r = \gamma^2 \mod p$, for some $\gamma \in \mathbb{Z}_p$). We will assume that $Z[\sqrt{r}]$ is an UFD.

Factor base: $S = \{x + y\sqrt{r} / \ x + y\sqrt{r} \text{ prime} \in \mathbb{Z}[\sqrt{r}],\ N(x + y\sqrt{r}) < L_p[\frac{1}{2}, 1/2]\} \cup \{q \in \mathbb{Z}$ prime, $q < L_p[\frac{1}{2}, 1/2]\}$.

What this means is that we are taking $L_p[\frac{1}{2}, 1/2] - smooth$ primes in our degree 2 extension of $\mathbb{Z}$ (something which also justifies the writing of the small primes from $\mathbb{Z}[\sqrt{r}] \setminus \mathbb{Z}$), along with small integer primes. Moreover, some integer primes will factor into two primes of $\mathbb{Z}[\sqrt{r}]$. This adds some redundancy which we will neglect for the moment. We will also need:

1. Let T and V be integers such that $T^2 \equiv rV^2 \mod p$, with $T, V \leq \sqrt{p}$. This equation can be solved using the extended gcd algorithm for $t - rv \equiv 0 \mod p$, with fixed r. Afterwards, if t and v are not quadratic residues $\mod p$, we have no solution and we should try with a different setup.

2. A generator $g = g_1 + g_2\sqrt{r}$ of the multiplicative group of $\mathbb{Z}[\sqrt{r}]_{/(T+V\sqrt{r})}$. This generator will be used as the basis for our logarithms.

5.1 Sieving

Take pairs of integers (c, d), such that $c, d < L_p[\frac{1}{2}, 1/2]$ for which $cV - dT$ is $L_p[\frac{1}{2}, 1/2] - smooth$. Linearly, we sieve line-by-line, fixing c and sieving for d. For every smooth $cV - dT$, observe that:

$$cV - dT = V(c + d\sqrt{r}) - d(T + V\sqrt{r}) \implies cV - dT = V(c + d\sqrt{r}) \mod (T + V\sqrt{r})$$

For c, d small, it holds with high probability that $c + d\sqrt{r}$ smooth. In this situation, since we are working in an UFD, we can decompose $c + d\sqrt{r}$ into small complex primes. Then decompose $cV - dT$ as the product of real primes and by taking logarithms in base g, get linear relations of the elements in our factor base:

$$cV - dT = V(k_1k_2...k_{nr}) \mod (T+V\sqrt{r}) \implies r_1r_2...r_{n_1} = V(k_1k_2...k_{n_2}) \mod (T+V\sqrt{r}) \implies$$

$$\implies \sum_{j=1}^{n_1} log_g r_j = log_g(V) + \sum_{l=1}^{n_2} log_g k_l,\ with\ k_l \in \mathbb{C}\ \forall l,\ r_j \in \mathbb{R}\ \forall j$$

Obtaining the logarithm of an element u is done by finding a pair c, d such that $u/(cV - dT)$ and then by sieving through the values of $d' = d + ku$ with $k \leq L_p[\frac{1}{2}, 1/2]$ to find smooth elements $(cV - d'T)/u$. Once a smooth element $(cV - d'T)/u$ has been uncovered, we check that $c + d'\sqrt{r}$ is also smooth. If this happens, that would mean that during the linear relation search we have already uncovered the logarithms of $cV - d'T$ and since $(cV - d'T)/u$ is smooth, we can now compute the logarithm of u. The approach underlined here might also require us to compute the logarithm of the given generator with respect to g in order to recover arbitrary logarithms. There are more theoretical

details which we do not cover and which are required for this algorithm, the interested reader may consult [1].

6 Index Calculus on Hyperelliptic Curves

6.1 Preliminaries

In the following, we use the notations and definitions from [10], on which we expand on below.

Definition 6.1. [10]: For an algebraic variety V and the finite set $f_1, ...f_m \in \bar{K}[X]$, generators for $I(V)$, we say that $P \in V$ is a nonsingular (smooth) point if the matrix $(\frac{\partial f_i}{\partial X_j}(P))_{1\leq i\leq m, 1\leq j\leq n} = n - dim(V)$.

If K is our base field and $\bar{K}$ is its algebraic closure, we denote by $\bar{K}[X]$ the ring in n variables $\bar{K}[X_1, ...X_n]$. The dimension of an algebraic variety V, denoted by $dim(V)$, is the transcendence degree of $\bar{K}(V)$ over V (it is usually helpful to view this as the number of degrees of freedom of $\bar{K}(V)$ over V).

Definition 6.2. [10]: An algebraic curve is a projective variety of dimension 1.

Although we can easily define algebraic curves of dimension 1 in more than 2 variables, in the case of elliptic and hyperelliptic curves, m=1 and n is therefore 2.

Another characterization of smoothness comes from the concept of the maximal ideal at every given point P. For every P, define $M_P = \{f \in \bar{K}[V]/\ f(P) = 0\}$. It can be shown that it is indeed maximal. The localization of $\bar{K}[V]$ at P will be performed by considering a set S of "nonzero denominators" and will give us the local ring $S^{-1}\bar{K}[V]$ denoted as $\bar{K}[V]_P$, where $\bar{K}[V]_P = \{\frac{f}{g}/\ f, g \in \bar{K}[V],\ g(P) \neq 0\}$.

Proposition 6.3. [10]: For any algebraic curve C, if $P \in C$ smooth, $K[X]_P$ is a discrete valuation ring, and this valuation is $ord_P : \bar{K}[V]_P \to \mathbb{N} \cup +\infty$, $ord_p(f) = \sup\{d \in \mathbb{N}/\ f \in M_p^d\}$. We extend this to $\bar{K}(C)$ defining $ord_p(f/g) = ord_p(f) - ord_p(g)$. A uniformizer for C at P is any function $t \in \bar{K}(C)$ with $ord_p(t) = 1$.

Remark that any uniformizer at P is a generator of M_P.

Definition 6.4. [10]: The divisor group $Div(C)$ of an algebraic curve C is the free abelian group generated by the points of C. A divisor is therefore a formal sum of points from C, $D \in Div(C) \implies D = \sum_{P\in C} n_p(P)$, $n_p \in \mathbb{Z}$ and $n_p \neq 0$ only for finitely many P.

Definition 6.5. [10]: The degree of a divisor $D = \sum_{P\in C} n_p(P)$ is defined as $deg(D) = \sum_{P\in C} n_P$.

The divisors of degree 0 form a subgroup of $Div(C)$, namely $Div^0(C) \leq Div(C)$. Now let f be a nonzero function from the function field of C, $f \in \bar{K}(C)^*$. We can associate to f a divisor, $div(f)$ such that $div(f) = \sum_{P \in C} ord_P(f)(P)$. Moreover, we say that a divisor D is principal if there exists $f \in K(C)^*$ such that $D = div(f)$. We also define an equivalence relation, $D_1 \sim D_2$ if $D_1 - D_2$ is principal.

Theorem 6.6. *[10]: Riemann Roch: Given a smooth algebraic curve C and a canonical divisor K_C on C, there exists an integer $g \geq 0$, called the genus of C, such that for every divisor $D \in Div(C)$, $\ell(D) - \ell(K_c - D) = deg(D) - g + 1$.*

$\ell(D) = dim_{\bar{K}} \mathcal{L}(D)$, where $\mathcal{L}(D) = \{f \in \bar{K}(C)^* : div(f) \geq -D\} \cup \{0\}$.

Now, since $\ell(D) \geq 0$, we have the following corollary:
Corollary:

1. $deg(K_c) = 2g - 2$
2. $\ell(D) \geq deg(D) - g + 1$
3. $\ell(D) = deg(D) - g + 1$ when $deg(D) \geq 2g - 2$

Remark: A canonical divisor is the divisor of a differential form on C. Discussing its properties is beyond the scope of this presentation, in our proofs we are only going to use the fact that such a divisor exists.

The rest of this section is related to basic properties of hyperelliptic curves. The material presented is largely from [14], along with the proofs for every important result. In some places, we have inserted various remarks or slight variations from the source material, with the purpose of making the original reasoning more accessible.

Definition 6.7. [14]: A curve C, defined over K, given by the equation $C : y^2 + h(x)y = f(x)$ is a hyperelliptic curve of genus g if it is smooth and the polynomials f, h satisfy: $deg(f) = 2g + 1$ and $deg(h) \leq g$. Moreover, f can be assumed to be monic.

Most often, a line intersects the hyperelliptic curve in $2g + 1$ points. Setting the genus $g = 1$, one gets an elliptic curve. If the characteristic of the base field is not 2, one can form squares getting $y^2 = f(x)$ by performing the change of variables $x \to x, y \to (y - \frac{h(x)}{2})$. However, it is not possible to define a law of addition for points as in the elliptic curve case, because taking two points now leaves us with $2g - 1$ other points on the same line.

Definition 6.8. [14]: Let $P = (x, y) \in C$. The map $\omega(P) = \omega(x, y) = (x, -y)$ in the case of $char(K) \neq 2$ is called the hyperelliptic involution.

Remark that the involution composed with itself is the identity. In the case of an elliptic curve, $w(P) = -P$. Moreover, $\omega(P_\infty) = P_\infty$.

Definition 6.9. [10]: The Picard's group of C, denoted by $Pic(C)$, is the quotient group $Div(C)/_{principal\ divisors\ group}$, and $Pic^0(C) = Div^0(C)/_\sim$. $Pic^0_K(C)$ is the subgroup of $Pic^0(C)$ fixed by the Galois action of $Gal(\bar{K}/K)$.

Remark: The set of principal divisors forms an abelian group, this can be seen when considering function multiplication. If $D_1 = div(f)$ and $D_2 = div(g)$, with $f, g \in K(C)^*$ then we can coalesce the zeros of f and g by multiplication. Hence, $D_1 + D_2$ is also principal, and it is equal to $div(fg)$.

Property 6.10. A hyperelliptic curve has only one point at infinity, which we will denote by P_∞. Moreover, $P_\infty = [0, 1, 0]$.

To see this, let's consider the homogenization of C in $\mathbb{P}^2$. We obtain the projective curve: $y^2 z^{2g-1} + h(x) y z^{2g-deg(h)} = f(x)$. The affine part of the curve rests in the affine chart given by $z = 1$ and dehomogenization is done dividing by z. Setting $z = 0$ and trying to dehomogenize with respect to z yields $x = 0$ and $y \neq 0$ and we are done.

Proposition 6.11. [10]: Let C be a smooth curve and and $f \in \bar{K}(C)^*$ Then there are only finitely many zeros and poles of f.

Another result that we are going to use extensively is that any principal divisor D has $deg(D) = 0$. From this it follows that any $f \in \bar{K}(C)^*$ has an equal number of zeros and poles, counting multiplicity.

Remark: Following the approach from [14], we are first going to consider the behaviour of hyperelliptic curves when taken over an algebraically closed field. In order to highlight this, we are going to make an analogy to further results over finite fields by denoting the algebraically closed field as $\bar{K}$.

Property 6.12. Remark on the order of vanishing: A counterintuitive property of algebraic curves is that the order of vanishing can behave unexpectedly. What we need for the following is that, on a hyperelliptic curve C, if $P(x) \in K(C)$ a polynomial, the number of zeros cannot exceed its $deg(P(x))$. However, the multiplicity of these zeros can be bigger than the degree.

Example: For $P = (a, 0)$, $x - a$ has a double zero at P. For more details, see [14].

Property 6.13. [6]: Let C be a smooth hyperellptic curve given by the equation $y^2 = f(x)$, with $char(K) \neq 2$. Then $f(x)$ has no multiple roots.

Proof: Assume a is a multiple root for $f(x)$. Then $f'(a) = 0$ and the point $P = (a, 0)$ is a singular (a non-smooth) point, because the rank of $(\frac{\partial C}{\partial x}, \frac{\partial C}{\partial y})$ at P is 0.

6.2 Mumford Representation

The following two seemingly uninteresting propositions yield very interesting applications concerning computation using divisors on hyperelliptic curves. Indeed, these are the two building blocks we need in order to see that some divisors can be represented as polynomials. Once we have this result, we can anticipate and say that the smoothness criterion on hyperelliptic curves will be the degree of the said polynomials.

Proposition 6.14. [14]: Let $A(x) = \prod_j (x - a_j)^{c_j}$. it holds that:
$div(A(x)) = \sum_j c_j * [(P_j) + (\omega(P_j)) - 2(P_\infty)]$, where $P_j = (a_j, \sqrt{f(a_j)})$

Proof: If $f(a_j) \neq 0 \implies x - a_j$ has exactly two zeros: in P and $\omega(P)$ thus P_∞ will be a double pole.
When $f(a_j) = 0$, there is only one zero in $P = (a_j, 0)$ and things get more complicated. First, recall that C smooth $\implies$ $f(x)$ has no multiple roots. Now examine the order of $x - a_j$ at P. Observe that $x - a_j = y^2 h(x)$ where $h = \frac{x-a_j}{f(x)}$. Now,

1. First see that because $f(x)$ has no multiple roots, $h(P) \neq 0$ and clearly $h(P) \neq \infty$.
2. $M_P = (x - a_j, y)$, and since h is invertible in $\bar{K}[C]_P$ see that y is a generator of M_P and thus a unformizer at P.

Thus, the order of $x - a_j$ at P is 2. Now use the fact that $deg(fg) = deg(f) + deg(g),\ \forall f, g \in \bar{K}(C)^*$ and we are done.

Proposition 6.15. [14]: Let $V(x)$ be a polynomial such that $f(x) - V(x)^2 = \prod_j (x - a_j)^{d_j}$. It can be shown that:
div$(y - V(x)) = \sum_j d_j * [((a_j, V(a_j))) - (P_\infty)]$. Moreover, $V(a_j) = 0 \implies d_j = 1$

Proposition 6.16. [14]: Let $D \in Div^0(C)$. Then $D + \omega(D)$ is principal.

We write $D = \sum_j c_j(P_j) \implies D = \sum_j c_j[(P_j) - (P_\infty)]$. If $P_j = P_\infty$ the corresponding term reduces, so assume $P_j \neq P_\infty,\ \forall j$. Thus $D + \omega(D) = \sum_j c_j[(P_j) + (\omega(P_j)) - (P_\infty)]$, which according to an aforementioned proposition is the divisor of a rational function in x.

Definition 6.17. [14]: Let $P_j = (a_j, b_j)$. A divisor of the form $D = \sum_j c_j[(P_j) - (P_\infty)]$ is called semi-reduced if it satisfies:

1. $c_j \geq 0\ \forall j$
2. $b_j = 0 \implies c_j = 0$ or $c_j = 1$
3. when $b_j \neq 0$ then: if (P_j) appears in the sum, $(\omega(P_j))$ does not.

Remark: Behind the second condition rests a much more natural reasoning. Actually, it is exactly the last condition wearing different clothes. Indeed, one may ask what happens if $\omega(P_j) = P_j$? Simply put, in this situation P_j may not appear at all, and if it does appear, it may appear only once. Any extra appearances would break constraint number 3.

Property 6.18. [14]: Let $V(x)$ polynomial such that $f(x) - V(x)^2 = \prod_j (x - a_j)^{d_j}$. Since $f(x) - V(x)^2$ is a polynomial in x, it cannot have any finite poles $\implies \forall j,\ d_j \geq 0$. From an aforementioned proposition, if $b_j = 0 \implies d_j = 1$.. Also, from that proof, the order of vanishing for $y - V(x)$ at a $P = (a, b)$ $b \neq 0$is the same as the order of vanishing for $(y + V(x))(y - V(x))$. Therefore, if P is a root of $y - V(x)$, $P(a, b)$ cannot be a root of $y + V(x) \implies b + V(a) \neq 0$. Assume that $\omega(P) = (a, -b)$ could be a root for $y - V(X)$. Then $-b - V(a) = 0$ which is a contradiction.

Definition 6.19. [14]: A semi-reduced divisor is reduced if $\sum_j c_j \leq g$.

Definition 6.20. [14]: Let $D_1 = \sum_j c_j[(P_j) - (P_\infty)]$ and $D_2 = \sum_j d_j[(P_j) - (P_\infty)]$ with $c_j, d_j \geq 0\ \forall j$. Their greatest common divisor is defined as $gcd(D_1, D_2) = \sum_j min(c_j, d_j)[(P_j) - (P_\infty)]$.

Proposition 6.21. [14]: Let $D = \sum_j c_j[(P_j) - (P_\infty)]$ be a semi-reduced divisor, $P_j = (a_j, b_j)$ and $U(x) = \prod_j (x - a_j)^{c_j}$.
Consider $V(x)$ to be a polynomial such that $b_j = V(a_j)\ \forall j$. Then

$$D = gcd(div(U(x)), div(y - V(x))) \iff f(x) - V(x)^2 \text{ is a multiple of } U(x).$$

Proof: This is an attempt to provide us with a converse for the second proposition in this chapter.
$\implies$:
Indeed, we can write $div(y - V(x)) = \sum_j d_j[(P_j) - (P_\infty)]$. If $D \leq div(y - V(x))$ then $d_j \geq c_j\ \forall j$. Thus, $\forall j$, the multiplicity of P_j in $f(x) - V(x)^2$ is greater than its multiplicity in $U(x)$. Moreover, $U(x)$ has no other zeros, thus it is a divisor of $f(x) - V(x)^2$. Conversely, the proof is similar.

Remark: According to [14], for the points where $b_j = 0$, we have $c_j = 1$ from reducibility of D. Although $div(x - a_j)$ contains $2(P_j) - 2(P_\infty)$, since $div(y - V(x))$ contains $(P_j) - (P_\infty)$ only once, the gcd will contain $(P_j) - (P_\infty)$ only once.

Theorem 6.22. [14]*: There is a one-to-one correspondence between semi-reduced divisors* $D = \sum_j c_j[(P_j) - (P_\infty)]$ *and pairs of polynomials* $(U(x), V(x))$ *that satisfy all of the below:*

1. $U(x)$ *is monic.*
2. $deg(U(x)) = \sum_j c_j$ *and* $deg(V(x)) < deg(U(x))$.
3. $V(x)^2 - f(x)$ *is a multiple of* $U(x)$.

Moreover, D *will correspond to* $gcd(div(U(x)), div(y - V(x)))$.

Proof: It is immediate to see that for any pair $U(x), V(x)$, there is only one gcd. The fact that this divisor is also semi-reduced results from the previous proposition.
Conversely, for one such semi-reduced divisor D, we can definitely find a pair $(U(x), V(x))$. Uniqueness comes from the condition $deg(V(x)) < deg(U(x))$. Indeed, assume that there exist two polynomials $V_1(x)$ and $V_2(x)$ that along with $U(x)$ satisfy the constraints of the theorem. The order of vanishing of $V_i(x)$ is at least c_j at each P_j thus the multiplicity of P_j in $V_1(x) - V_2(x)$ is at least c_i. But then $V_1(x) - V_2(x)$ has at least $\sum_j c_j = deg(U(x))$ zeros, which can only happen when the difference vanishes at every point. (See remark on the order of vanishing).

Finding the polynomial pair $(U(x), V(x))$, when given the semi-reduced divisor $D = \sum_j c_j[(P_j) - (P_\infty)]$. [14]

1. First, let us recover $V_j(x)$ where $V_j^2(x) \equiv f(x) \mod (x-a_j)^{c_j}$ and $V_j(a_j) = b_j$. (The second constraint keeps us under the GCD proposition).

2. If $b_j = 0 \implies f(a_j)^2 = 0 \implies f(a_j) = 0$ and by Proposition 13.2, $c_j = 1$.

   (a) Thus, $f(x) = f(a_j) = 0 \mod x - a_j \implies V_j(x) = 0 \mod x - a_j$, and we can choose any $V_j \in (x - a_j)$.

   Note: For every polynomial $P(x)$, we have $P(x) \equiv P(a) \mod (x - a)$. This comes from the Remainder theorem: For every field K with $a \in K$ and $P(x) \in K[x]$, $P(a)$ will be the remainder in the division of $P(x)$ by $x - a$.

3. On the other hand, when $b_j \neq 0$, take first $W_1(x) = b_j$, in an attempt at satisfying the constraints from 1. This only means that $W_1(x) \equiv f(x) \mod x - a_j$. To find V_j, we are going to use an approach that is similar to Hensel lifting, called Newton's method.

   (a) Assume that you know W_n such that $W_n(x)^2 \equiv f(x) \mod (x - a_j)^n$.

   (b) Observe that $W_{n+1}(x) \equiv f(x) \mod (x - a_j)^n$ as well. This means that W_{n+1} differs from W_n by a multiple of $(x - a_j)^n$.

   (c) As such, write $W_{n+1}(x) = W_n(x) + k * (x - a_j)^n$. We now need to find k.

   (d) By squaring, one gets $W_{n+1}^2(x) = W_n^2(x) + 2kW_n(x) * (x - a_j)^n + k^2 * (x - a_j)^{2n}$. Since we need $W_{n+1}(x)^2 \equiv f(x) \mod (x - a_j)^{n+1}$, we can now apply this constraint, obtaining:

   (e) $f(x) \equiv W_n(x)^2 + 2kW_n(x) * (x - a_j)^n \mod (x - a_j)^{n+1}$.

   (f) $2kW(x) = P(x) \mod (x - a_j)^{n+1}$ where $P(x) = \frac{f(x) - W_n(x)^2}{(x-a_j)^n}$. Note that $P(x) \in K[x]$ since $f(x) - W_n(x)^2 \in ((x-a_j)^n)$ by the induction hypothesis. Now, taking into account that $W_n(a_j) = b_j$, we find that $k = P(a)/2b_j$.

4. Polynomials $(x-a_j)^{c_j}$ and $(x-a_i)^{c_i}$ have been constructed in such a way that they have no common factors when $i \neq j$. Therefore, the ideals they generate are relatively prime and we can apply the Chinese Remainder Theorem for each V_j to obtain $V(x)$.

Property 6.23. [14]: Let D be a principal divisor. Then $\omega(D)$ is also principal.

D principal $\implies \exists f \in \bar{K}(C)^*$ such that $D = div(f)$. Then, consider $P \in C$, either a zero or a pole of f. Then P will remain as such for $f \circ \omega$, since $\omega \circ \omega(P) = P \implies f \circ \omega(\omega(P)) = P$.

Proposition 6.24. [14]: $\forall D \in Div^0(C)$, there exists a unique reduced divisor D_1, such that $D - D_1$ is a principal divisor.

Proof of Existence:
As a corollary to Riemann-Roch, it holds that $\ell(D) \geq deg(D) - g + 1 \; \forall D$. Thus, for our given D, $\ell(D + g(P_\infty)) \geq deg(D) + g - g + 1 = 1 \implies \exists f \in \bar{K}(C)^*$ such that $div(f) + D + g(P_\infty) \geq 0$. Our $D_1 = div(f) + D$. Let us show that D_1 can be transformed into a reduced divisor:

1. $div(D_1) = 0$ and $D_1 + g(P_\infty) \geq 0 \implies D_1$ has only one negative coefficient, that of P_∞. Assume there are more than g points in the sum, different from P_∞. Then their coefficients are nonnegative $\implies deg(D) \neq 0$.

2. Therefore, we can write D_1 as $D_1 = \sum_j c_j[(P_j) - (P_\infty)]$. Obviously, $c_j \geq 0 \; \forall j$. Also, because there are maximum g other points in the sum, $\sum_j c_j \leq g$.

3. Assume that both P and $\omega(P) \in D_1$, n_P and $n_{\omega(P)}$ times, respectively. Then substract $min(n_P, n_{\omega(P)}) * [(P) + (\omega(P)) - 2(P_\infty)]$, which is a principal divisor.

4. One last thing is to show that $\forall P = (a_j, b_j) \in D_1$, the corresponding c_j is either 0 or 1. Substract accordingly by a multiple of $div(x - a_j) = (P) + (\omega(P)) - 2(P_\infty)$.

Uniqueness: Let $D - div(F) = D_1$ and $D - div(G) = D_2$ be two reduced divisors. Then $D_1 + \omega(D_2) = D + \omega(D) - div(F) - \omega(div(G))$ is principal because the RHS is principal $\implies$ $D_1 + \omega(D_2) = div(H)$. Thus $(D_1 + g(P_\infty)) + (\omega(D_2) + g(P_\infty)) = div(H) + 2g(P_\infty)$. From the first part of the theorem, the LHS is ≥ 0 thus $div(H) + 2g(P_\infty) \geq 0$ which implies that $H \in \mathcal{L}(2g(P_\infty))$. Using the third part of the Riemann-Roch corollary, since $deg(2g(P_\infty)) \geq 2g - 2$, we get that $\ell(2g(P_\infty)) = g + 1$.
Recall that $div(x) = (0, \sqrt{f(0)}) + (0, -\sqrt{f(0)}) - 2(P_\infty)$. Therefore, $x^j \in \mathcal{L}(2j(P_\infty))$. As such, the set:

$$\{1, x, x^2, ..., x^g\} \subseteq \mathcal{L}(2g(P_\infty))$$

and as an independent set, it will form a basis for $\mathcal{L}(2g(P_\infty))$. This means that $H \in \bar{K}(C)^* \cap \bar{K}[x]$, i.e. H is a polynomial in x, therefore $D_1 + \omega(D_2) = \sum_j c_j[(P_j) + (\omega(P_j)) - 2(P_\infty)]$. Since D_1 and D_2 are reduced, if $P_j \in D_1 \implies \omega(P_j) \in \omega(D_2)$ and vice-versa. We have thus that $D_1 = D_2$.

The Jacobian of C

The divisor classes of degree 0 modulo principal divisors of C form the Picard group of C, $Pic^0(C)$. We can give a law of addition on this group, giving it the structure of an algebraic variety, called the Jacobian variety of C, which we will denote as $J(C)$. For $P \in C$ define $\psi : C \to Pic^0(C)$, $\psi(P) = (P) - (P_\infty)$, which is a reduced divisor.

Assume that for distinct P_1, $P_2 \in C$, where $D_1 = (P_1) - (P_\infty)$ and $D_2 = (P_2) - (P_\infty)$, we have $D_1 - D_2 = div(F)$. From an almost identical argument as the uniqueness part of the preceding theorem, since D_1, D_2 are reduced, we will have $D_1 = D_2$. ψ is therefore injective. According to [14], in the case of elliptic curves ψ is an isomorphism. As an additional remark, please note that as far as algebraic varieties are concerned, a bijective morphism of curves is not necessarily an isomorphism.

Theorem 6.25. [14]*: There is a one-to-one correspondence between divisor classes of degree 0 on C and pairs* $(U(x), V(x))$ *of polynomials satisfying:*

1. *U is monic*
2. $deg(V) < deg(U) \leq g$
3. $V^2 - f(x)$ *is a multiple of U.*

Proof: We already know that all semi-reduced divisors are in a bijective correspondence with pairs (U, V). Additionally, we know now that in every divisor class of degree 0 we can find a reduced divisor. Additionally, since $deg(U) = \sum_j c_j$, and since a semi-reduced divisor $D = \sum_j c_j$ if reduced if it has $\sum_j c_j \leq g$, we know have $deg(U) \leq g$.

Remark: We are going to use this constraint on $deg(U)$ to devise a way of "reducing" the Mumford representation, with the goal of obtaining a valid Mumford representation that has $deg(U) \leq g$.

Definition 6.26. For a divisor class of degree 0, we denote the pair $(U(x), V(x))$ to be its Mumford representation, where (U, V) satisfies the conditions of the preceding theorem.

6.3 Reduction and Addition using the Mumford Representation

At this point, we have some very nice characterizations regarding structural properties of the group of degree 0 divisor classes. We know that for each class in the Picard's group, starting from a representative we can find only one reduced divisor in the same class. From the proof of that theorem, we have also found a way of reducing divisors of a special form to a semi-reduced divisor, by subtracting multiples of divisors of the form $(P) + (\omega(P)) - 2(P_\infty)$. This could only work for divisors whose only negative coefficient is the coefficient of P_∞. However, it is not clear how could one start from an arbitrary divisor D and then algorithmically start computing the reduced divisor of the same class, since in order to do that one would need a way to generically compute $div(F)$ where

$F \in \mathcal{L}(D + g(P_\infty))$. The other idea of doing this is by looking at the Mumford representation of D.

Reduction Algorithm: Given the semi-reduced $D \in Div^0(C)$ and its Mumford representation (U, V), modify U, V such that it corresponds to the reduced divisor in the class of D. [14]

1. While $deg(U) > g$
   (a) $U_1 = \frac{f - V^2}{U}$.
   (b) $V_1 = -V \mod U_1$.
   (c) multiply U_1 by the inverse of its leading coefficient to make it monic.
   (d) $U = U_1$ and $V = V_1$.
2. Return (U, V).

Theorem 6.27. [14]: *For a semi-reduced $D \in Div^0(C)$ and its Mumford representation (U, V), the output of the reduction algorithm is the Mumford representation of the reduced divisor that corresponds to the class of D.*

Proof: Let $D = \sum_j c_j[(P_j) - (P_\infty)]$, where $P_j \neq P_\infty$, $\forall j$ and $P_j = (a_j, b_j)$. From the GCD theorem, recall that $U(x) = \prod_j (x - a_j)^{c_j}$. Therefore, $div(U(x)) = \sum_j [(P_j) + \omega(P_j) - 2(P_\infty)]$. Since D semi-reduced, $div(U(x)) = D + \omega(D)$.
Now, from the gcd correspondence, we also know that $U(x)/(f(x) - V(x)^2)$. Recall that when $b_j \neq 0$ the order of vanishing for P_j in $y - V(x)$ is the same as its order of vanishing in $(y - V(x))(y + V(x))$, which means that it cannot be a zero for both $y - V(x)$ and $y + V(x)$. However, if $y - V(x)$ vanishes at P_j then $y + V(x)$ vanishes at $\omega(P_j)$.

Clearly, $f(x) - V(x)^2$ contains the zeros P_j of $U(x)$ along with some other zeros Q_k that we group into a divisor $E = \sum_k e_k[(Q_k) - (P_\infty)]$, $e_k \geq 0\ \forall k$. Therefore, $div(y - V(x)) = D + E$ and $div(y + V(x)) = \omega(D) + \omega(E)$.

Since $div(U) + div(U_1) = D + E + \omega(D + E)$ and $div(U) = D + \omega(D)$, we have $div(U_1) = E + \omega(E)$. Now observe that $gcd(div(y + V), div(U_1)) = \omega(E)$. We still need to show that:

1. $\omega(E) \sim D$, which is true from $D - div(y - V) + div(U_1) = \omega(E)$.
2. (U_1, V_1) corresponds to $\omega(E)$
3. $deg(U_1) < deg(U)$, recall that from the definition of a hyperelliptic curve, $deg(f) = 2g + 1$. Also, from D semi-reduced, $deg(V) < deg(U)$. Therefore, when $deg(U) \geq g + 1$, we have $deg(U) + deg(U_1) = deg(f - V^2) = 2g + 1 \implies deg(U_2) < deg(U)$.

Cantor's Algorithm (Addition Algorithm) Let $D_1, D_2 \in Div^0(C)$, with (U_1, V_1) and (U_2, V_2) their Mumford representations.

1. Compute $d = gcd(U_1, U_2, V_1 + V_2)$ and polynomials h_1, h_2, h_2 such that $d = U_1h_1 + U_2h_2 + (V_1 + V_2)h_3$.
2. Compute $V_0 = \frac{U_1V_2h_1+U_2V_1h_2+(V_1V_2+f)h_3}{d}$
3. $U = \frac{U_1U_2}{d^2}$ and $V \equiv V_0 \mod U$
4. Apply Reduction procedure on (U, V).

Theorem 6.28. *The output of Cantor's algorithm is the Mumford representation of* $D_1 + D_2$.

Proof: The proof is rather involved and it is beyond the scope of this presentation. The interested reader may consult [14].

6.4 The Jacobian $J_K(C)$ when $K = \mathbb{F}_{p^n}$

We are going to start with a brief analysis of what are the elements of $J_K(C)$. Since we are not going to work only with $\bar{K}$, as we did before, let us specify once more some of the conditions that apply on our hyperelliptic curve $C : y^2 = f(x)$. We ask that $f(x)$ has coefficients in $K = \mathbb{F}_{p^n}$, and that it has no multiple roots. The second condition refers to multiplicity in $\bar{\mathbb{F}}_{p^n}$. Moreover, we take $p \neq 2$. As stated earlier, the elements of $Div^0_K(C)$ form the subgroup of $Div^0(C)$ fixed by the Galois action of $Gal(\bar{K}/K)$. We need to look in particular at the elements of $Pic^0_K(C)$, since we have a unique correspondence for the unique reduced divisor in a particular class and its Mumford representation.
Remark: From [10], it makes absolutely no sense to consider divisors defined over a field K if the curve itself is not defined over that field K. This is why this condition is essential. Moreover, if for a divisor D and $\sigma \in Gal(\bar{K}/K)$, $D = D^\sigma$, this does not necessarily imply that P is $k - rational$ ($P \in C(K)$) for all $P \in D$.

Property 6.29. Let $\sigma \in Gal(\bar{K}/K)$. If $D \in Div^0(C)$ is reduced and $D \sim D^\sigma$, D^σ is also reduced.
Proof: Clearly, if $g \geq \sum_j c_j \geq 0$, this will also hold for D^σ. Now assume that in D^σ we have both $\sigma(P)$ and $\omega(\sigma(P))$. The hyperelliptic involution can be viewed as the curve morphism $[X, -Y]$ and we get the contradiction.

Proposition 6.30. There is a one-to-one correspondence between $J_{\mathbb{F}_{p^n}}(C)$ and pairs (U, V) with coefficients in $\mathbb{F}_{p^n}$ which satisfy:

1. U monic
2. $dev(V) < deg(U) \leq g$.
3. U divides $V^2 - f(x)$. [14]

Proof: Let $\sigma \in Gal(\bar{\mathbb{F}}_{p^n}/\mathbb{F}_{p^n})$. We want to work in $Pic^0_{\mathbb{F}_{p^n}}(C)$, therefore we take $D \in Div^0_K(C)$, such that $D \sim D^\sigma$.This implies $D = D^\sigma + div(F)$. If $R \sim D$ is the reduced divisor for D, then $R = D + div(G)$. Therefore, $R^\sigma = D^\sigma + div(G)^\sigma \implies R^\sigma \sim D^\sigma$. Since R^σ is reduced, and because $R \sim R^\sigma$, we have that $R \in Pic^0_{\mathbb{F}_{p^n}}(C) \implies R \in J_{\mathbb{F}_{p^n}}(C)$. However, recall that in each degree 0 divisor class we have only one reduced divisor, thus $R = R^\sigma \implies U^\sigma = U$ and $V^\sigma = V$.

Theorem 6.31. *[10]: Hasse-Weil's bound for algebraic curves:* $(p^n + 1 - 2g\sqrt{p^n}) \leq |C(\mathbb{F}_{p^n})| \leq (p^n + 1 + 2g\sqrt{p^n})$.

Theorem 6.32. *[14]: Generalized Hasse-Weil's bound:* $(\sqrt{p^n} - 1)^{2g} \leq |J_{\mathbb{F}_{p^n}}(C)| \leq (\sqrt{p^n} + 1)^{2g}$.

This property ensures that $J_{\mathbb{F}_{p^n}}(C)$ is finite, and since it can be viewed as an abelian group, we can use a well-known theorem to write it as a direct sum. Some index-calculus algorithms on hyperelliptic curves, starting with Adleman's approach, do not assume that this structure is known and employ some techniques to determine it.

Property 6.33. Let $(U(x), V(x)$ be a pair corresponding to a semi-reduced divisor. Then, $(1, 0)$ behaves as a neutral element for the group $J_{\mathbb{F}_{p^n}}(C)$, namely:

1. $(U, V) + (1, 0) = (U, V)$
2. $(U, V) + (U, -V) = (1, 0)$
3. Moreover, the pair $(1, 0)$ corresponds to the reduced zero divisor. [14]

Proof: Denote the result of the addition through the pair (U_{res}, V_{res}). From Cantor's algorithm, $gcd(U, 1, V) = 1 \implies h_1 = 0, h_3 = 0 \implies V_{res} = V$ and then $U_{res} = U$. For the second part, $gcd(U, U, 0) = U \implies$ either $h_1 = 1$ or $h_2 = 1$ with the other being a zero. For $h_1 = 1$, we compute first $V_0 = -V$ and then since $U_{res} = 1$, we have that $V_{res} = 0$.

The last assertion can be easily analysed, indeed 1 has neither zeros nor poles and it divides $-f(x)$, while V has to be the zero polynomial, since $deg(V)$ must be strictly smaller than 0.

A constructive approach to computing $|J_{\mathbb{F}_{p^n}}(C)|$ Although there exist point counting algorithms for the Jacobian of a hyperelliptic curve, they are not used in practice. The common approach is to construct curves for which we already know the cardinality of the Jacobian, either by using the CM method or the Zeta function. In the following, we are going to detail the later, as found in [6].

Remark on the Hyperelliptic Involution

Our proofs and computations have been simplified by our initial assumption that the characteristic of the base field is not 2. However, when $char(K) = 2$, our hyperelliptic involution ω has a different expression, namely: $\omega((a_j, b_j))) = (a_j, -b_j - h(a_j))$. For details on how does this impact each individual result, please consult [6].

Proposition 6.34. Given $(U(x), V(x))$, defined over $\mathbb{F}_{p^n}$, corresponding to a semi-reduced divisor, let us factor $U(x) = \prod_i U_i(x)$, where U_i has coefficients in $\mathbb{F}_{p^n}$. Let $V_i = V \mod U_i$. Then each pair $(U_i(x), V_i(x))$ corresponds to a semi-reduced divisor D_i with $\sum_i D_i = D$. Moreover, if D is reduced, so is every D_i. [14]

Proof: Each pair $(U_i(x), V_i(x))$ satisfies $deg(V_i) < deg(U_i)$ and since $V^2 \equiv f \mod U$, we also have $V_i^2 \equiv f \mod U_i$. Therefore, $(U_i(x), V_i(x))$ corresponds to a semi-reduced divisor.
When $U_i = \prod_j (x - a_j)^{c_j}$, from the GCD theorem it holds that: $gcd(div(U_i), div(y - V_i)) = \sum_j c_j[(P_j) - (P_\infty)]$, with $P_j = (a_j, V_j(a_j))$. Observe that since $U_i(a_j) = 0 \implies V_i(a_j) = V(a_j)$ we have that each zero of D_i is a zero of D. For a given P_j, the sum of its multiplicities in each D_i is equal with the multiplicity in D because $U(x) = \prod_i U_i(x)$ and we are done.

6.5 Rho-Pollard

In this section we are going to investigate some properties our Jacobian $J_{\mathbb{F}_{p^n}}(C)$ should have in a cryptographic setting. Then, we will give a description of an exponential attack on the discrete logarithm. This particularization of Rho-Pollard for hyperelliptic curves has [3] as the main source.

Definition 6.35. By hyperelliptic curve discrete logarithm problem, denoted as **HCDLP**, we understand that given $D_1, D_2 \in J_{\mathbb{F}_{p^n}}(C)$, we need to find an integer λ, such that $D_2 = \lambda D_1$. Also known is the fact that D_2 belongs to the subgroup generated by D_1, along with the order n of this subgroup (λ can be uniquely described $\mod n$).

Pohlig-Hellman Reduction For $J_{\mathbb{F}_{p^n}}(C)$, its order should be almost prime [4]. By this we mean that we expect the order to have a very large prime factor p. This is absolutely necessary since after a preliminary run of a point counting algorithm, one will successfully reduce the HCDLP in $J_{\mathbb{F}_{p^n}}(C)$ to the DLP in the subgroup of order p, using Pohlig-Hellman reduction.

The Rho-Pollard algorithm relies on the birthday paradox and solves the DLP in a generic group of N elements: randomly drawing $\sqrt{N}$ elements of the group, we are going to get a collision with probability around $\frac{1}{2}$(a better approximation for the number of random picks is $\sqrt{\frac{\pi}{2}N}$).

Strategy: Taking a function F, we construct chains of divisors by starting with a divisor $E_0 = e_0^1 D_1 + e_0^2 D_2$ and then iterating F on this divisor, recursively: $E_i = F(E_{i-1})$. This is being done as a time memory trade-off, instead of storing all the random points, we can store only the beginning and the end of a chain. Once a collision $E_i = E_j$ has been found, one can recover the DLP by noticing that $E_i - E_j = 0 \implies E_i - E_j = (e_i^1 - e_j^1)D_1 + (e_i^2 - e_j^2)D_2 = 0$. Thus, when $e_i^2 \neq e_j^2$ we have that $\lambda \equiv \frac{e_i^1 - e_j^1}{e_j^2 - e_i^2} \mod N$. The probability that $e_i^2 = e_j^2$ is $\frac{1}{N}$, which is small for N large.

Additive version Our assumption of gaining a practical improvement over the brute-force attack relies on the randomness of F. We present the following method of constructing F, utilised by [3]. We precompute r random divisors $T^j = t_j^1 D_1 + t_j^2 D_2$, where $0 \leq j < r$. For each divisor, we define a hash function $\mathcal{H} : J_{\mathbb{F}_{p^n}}(C) \to \{0 \,..\, r-1\}$, and we let F be $F(R) = R + T^{\mathcal{H}(R)}$. A good choice for r is $r \geq 8$.

Multiplicative version This time, instead of choosing r random divisors, we choose r random elements μ_j from $\mathbb{Z}_{/r\mathbb{Z}}$ and we let $F = \mu_{\mathcal{H}(R)} R$. This time we cannot retrieve the DLP if a collision comes from the elements of the same chain, since one only retrieves $\prod_{j=0}^{r-1} \mu_j^{f_j} = 1 \mod N$, which is not useful. Collision between elements of different chains will yield the DLP though: $\xi E_0 = F_0 \implies \lambda \equiv \frac{\xi e_0^1 - f_0^1}{f_0^2 - \xi e_0^2} \mod N$, with $\xi = \prod_{j=0}^{r-1} \mu_j^{f_j}$. [3]

A combination of both additive and multiplicative random walks, i.e. the mixed walks might also ensure a greater level of randomness.

Using the hyperelliptic involution Assume the following scenario, in which the collision $E_i = \omega(E_j)$ is found. For E_j, consider the corresponding pair (U_j, V_j) in $J_{\mathbb{F}_{p^n}}(C)$, for which we recall that its inverse is $(U_j, -V_j)$. Assume that E_j is reduced and defined over $\mathbb{F}_{p^n}$. Then $E_i + \omega E_j$ is the zero divisor. This means that the automorphism ω also transfers to the Jacobian, namely $\omega((U, V)) = (U, -V)$.

Therefore, a collision like $e_i^1 D_1 + e_i^2 D_2 = e_j^1 \omega(D_1) + e_j^2 \omega(D_2)$ implies that $e_i^1(U_1, V_1) + e_i^2(U_2, V_2) = e_j^1(U_1, -V_1) + e_j^2(U_2, -V_2)$. Considering the additive inverse in $J_{\mathbb{F}_{p^n}}(C)$, we observe that $e_i^1(U_1, V_1) + e_i^2(U_2, V_2) = -e_j^1(U_1, V_1) - e_j^2(U_2, V_2)$ which means that $\lambda \equiv -\frac{e_i^1 + e_j^1}{e_j^2 + e_i^2} \mod N$.
Speedup: Using the hyperelliptic involution, we obtain a theoretical improvement of $\sqrt{2}$. Problems here come from the apparition of useless cycles.

Automorphisms Let $\sigma : C \to C$ be an automorphism of order m on the curve. Then we might not get a collision $E_i \neq E_j$, but applying the automorphism, we end up with $E_i = \sigma^w(E_j)$. We will assume that $J_{\mathbb{F}_{p^n}}(C)$ is cyclic of prime order N, moreover, we consider the case where it is equal to $< D_1 >$. Automorphisms on the curve extend naturally to automorphisms on the Jacobian, if $\sigma(P_i) = P_j$ then $\sigma[(P_i) - (P_\infty)] = [(P_j) - (P_\infty)]$, providing that the image of P_∞ is P_∞.

Since σ on the Jacobian behaves like a group automorphism, observe that $\sigma(D_1) \in < D_1 >$, thus there exists an integer θ such that $\sigma(D_1) = \theta D_1$. This means that $\forall D \in < D_1 >$ with $D = \lambda D_1$, we have $\sigma(D) = \sigma(\lambda D_1) = \lambda\theta D_1 = \theta D$. Therefore, we only need to compute θ and, equipped with this information, when we uncover a collision of the type:
$e_i^1 D_1 + e_i^2 D_2 = e_j^1 \sigma^w(D_1) + e_j^2 \sigma^w(D_2)$. with D_1, D_2 and σ defined over the base field $\mathbb{F}_{p^n}$, we have

that: $e_i^1 D_1 + e_i^2 D_2 = e_j^1 \theta^w(D_1) + e_j^2 \theta^w(D_2)$ and as such, we recover the discrete logarithm as:

$$\lambda \equiv \frac{e_i^1 - \theta^w e_j^1}{\theta^w e_j^2 - e_i^2} \mod N.$$

We work thus in the group $< D_1 >_{/\sim}$ where we define the relation of equivalence $\sim$ as: $D_i \sim D_j \iff \exists w \in \mathbb{Z}_{/N\mathbb{Z}}$ such that $D_i = \sigma^w D_j$. If σ is an automorphism of order m, then we have m classes of equivalence relative to w and thus the size of the quotient group is $\frac{N}{m}$. Therefore, the complexity of Rho-Pollard on the classes of equivalence is $\sqrt{\pi \frac{N}{2m}}$, with the theoretical speedup of $\sqrt{m}$. [3]

A method for computing θ Recall that we have showed that $\sigma(D_1) = \theta D_1$, but it may seem that finding θ reduces to solving another discrete logarithm, which would be rather unfortunate. There is still a workaround, and the solution requires us to use the fact that the order of the automorphism σ is m. Recall that in this setting $| < D_1 > | = N$ and $\theta \in \mathbb{Z}_{/N\mathbb{Z}}$. Therefore, $D_1 = \sigma^m(D_1) = \sigma(\sigma(...(\sigma(D_1)))) = \theta^m D_1$. This means that the order of θ in $\mathbb{Z}_{/N\mathbb{Z}}$ is a divisor of m. When N is prime, what we are required to do is solve the polynomial equation $P(T) = T^m - 1 = 0$, which has as solutions the $m - th$ roots of unity. If m is not exceedingly large (this occurs in practice), finding the roots is feasible, therefore we can exhaustively check all the m possibilities, until we find the appropriate choice for θ.

Well-defined mappings If for the hyperelliptic involution we can find an easy way of implementing an additive walk, when one tries to make use of other automorphisms a certain issue should be kept in mind. Assume that we iterate F over the equivalence classes and let us consider the case where $\overline{E}_{i+1} = \overline{R}_{i+1}$ but $F(E_{i+1}) \neq F(R_{i+1})$. This happens when F is not well defined with respect to $\sim$. If we do not attempt to parallelize the computations then we do not encounter problems since the chains are still deterministically constructed. However, in a concurrent context we encounter nondeterminism of the walks and we need to consider a well defined function, for example $F_{new}(E_i) = \overline{F(E_i)}$. [3]

Non-cyclic Jacobians The Jacobian of an hyperelliptic curve is most of the times non-cyclic. We do however, choose it in such a way that it contains a large enough prime factor d. The question is, will the subgroup generated by D_1 be invariant under the action of the automorphism σ of large order m? If $ord(D_1) = d$ then since $d^2 \nmid |J_{\mathbb{F}_{p^n}}(C)|$ we have that all elements of order d belong to $< D_1 >$ (they should belong to a group of order a multiple of d, and there is no other choice).
Now, observe that $\sigma(D_1)^d = \sigma(D_1^d) = \sigma(1) = 1$ and thus $\sigma(D_1)^d = 1$. Since d is prime this means that the order of $\sigma(D_1)$ is d. By the previous argument, $\sigma(D_1) \in < D_1 >$ and therefore $< D_1 >$ is stable under the action of σ. [3]

6.6 Gaudry's Algorithm for curves of small genus

Now, having all these tools for operating with the Jacobian of a hyperelliptic curve, let us give a first description of an algorithm for index calculus. We will first define the DLP on the Jacobian of a curve along with notions regarding divisor primality and smoothness. Unless stated otherwise, the results of this chapter follow the description in [4].

Definition 6.36. A divisor D, given by its Mumford representation $(U(x), V(x))$, is said to be a **prime divisor** if $U(x)$ is irreducible over $\mathbb{F}_{p^n}$. [4]

We can now decompose a semi-reduced divisor into its prime divisors using the last proposition of the previous section, just as we would do if we worked in $\mathbb{Z}[x]$.

Definition 6.37. A divisor D, given by the pair $(U(x), V(x))$ is said to be $S - smooth$ if all of its prime divisors have the corresponding polynomial $U(x)$ of degree at most S. If $S = 1$, a $1 - smooth$ divisor will have $U(x)$ split completely over $\mathbb{F}_{p^n}$. [4]

Let us see why $1 - smooth$ divisors have a splitting $U(x)$. If we decompose $U(x)$ into polynomials defined over $\mathbb{F}_{p^n}$ of degree at most 1, we obtain the following writing: $U(x) = \prod_j (x - a_j)$, where $a_j \in \mathbb{F}_{p^n}$ $\forall j$.

Let us move the smoothness property towards divisors rather than just polynomials.

Definition 6.38. A divisor of the form $D = (P_1) + ... + (P_g) - g(P_\infty)$ is $S - smooth$ if and only if each $P_i \in D$ is defined over an extension field $\mathbb{F}_{p^{nk}}$ where $k \leq S$. [4]

First, recall that D defined over $\mathbb{F}_{p^n}$ does not necessarily mean that each $P_j \in \mathbb{F}_{p^n}$. Nevertheless, we can start from $D = D^\sigma$, where $\sigma \in Gal(\bar{\mathbb{F}}_{p^n}/\mathbb{F}_{p^n})$. Let $P_j = (a_j, b_j)$. Since $U(x)$ can be factored into irreducible polynomials of degree at most S, we will have that the roots a_j of $U(x)$ may not be in $\mathbb{F}_{p^n}$, but they will necessarily be in $\mathbb{F}_{p^{nS}}$. Therefore, at least $a_j \in \mathbb{F}_{p^{nS}}$.
Now, since P_j is in $\sum_j c_j[(P_j) - (P_\infty)] = gcd(div(U_i), div(y - V_i))$, we will have that $b_j = V(a_j)$. But both $V(x)$ and $f(x)$ are defined over $\mathbb{F}_{p^n}$, which means that $b_j \in \mathbb{F}_{p^{nS}}$. Concluding, we have that each P_j must be defined over $\mathbb{F}_{p^{nk}}$ where $k \leq S$ and k is chosen accordingly.

Choice of the factor base The factor base is the set of all prime divisors $D = (U, V)$ for which $deg(U) = 1$. This means that $G = \{(U, V)$, where $U(x) = x - a_j$, $a_j \in \mathbb{F}_{p^n}$ and $V(x)^2 \equiv f(x) \mod U(x)\}$. Therefore, we list all the polynomials $U(x)$ and then recover $V(x)$ by computing $f(x) \mod U(x) \in \mathbb{F}_{p^n}$. Now we only need to compute a square root in $\mathbb{F}_{p^n}$, and we have constructed the factor base.

Seen as a set of divisors, our factor base contains only reduced divisors which are in the same class as the divisors of the form $(P) - (P_\infty)$, where $P \in C(\bar{\mathbb{F}}_{p^n})$.

An additive random walk Start with $R_0 = r_0^1 D_1 + r_0^2 D_2$ as the initial point of the random walk. We are going to pseudo-randomly pick elements of $< D_1 >$, precomputing r random divisors $T^j = t_j^1 D_1 + t_j^2 D_2$, $0 \leq j < r$. Again, we let F be $F(R) = R + T^{\mathcal{H}(R)}$. The hash function $\mathcal{H} : J_{\mathbb{F}_{p^n}}(C) \to \{0 \,..\, r-1\}$ is recommended to be given by the last bits of the internal representation of the reduced divisors. For each divisor R_i, unless we have a collision like in the Rho-Pollard version, we will test it for smoothness, and if it is indeed smooth, we are going to factor it with respect to the elements of the factor base. This can be done using trial division with respect to the U polynomial, which might be costly for large cardinalities of the base. The second option is to use a factorization algorithm for polynomials, which is more practical. Moreover, we are going to give a criterion for smoothness:

Property 6.39. When the degree of $gcd(x^{p^n} - x, U(x))$ is equal to the degree of $U(x)$, $U(x)$ splits completely over $\mathbb{F}_{p^n}$. [4]

Proof: Considering the Frobenius action, we see that $x^{p^n} - x$ has as roots all the elements of $\mathbb{F}_{p^n}$. Therefore, computing its gcd with $U(x)$ we get the product of all the degree 1 polynomials which are factors of $U(x)$.
We are going to use this property as a criterion for the $1 - smoothness$ of reduced divisors. Please note that such an approach will be rather unhelpful when p^n is large, thus its applicability depends on the choice of curve.

Linear Relations The random walk is going to yield a number of linear relations of the form: $e_i^1 D_1 + e_i^2 D_2 = \sum_{j=1}^{|G|} f_i^j F_j$, where $F_i^j \in G$ and f_j are integers modulo N. This happens when the $U(x)$ polynomial of $e_i^1 D_1 + e_i^2 D_2$ splits completely over $\mathbb{F}_{p^n}$. We thus obtain a sparse matrix M. A nonzero vector $v \in \ker(M^t)$ corresponds to a linear dependency between the lines of M, and thus we get that $\sum_i v_i(e_i^1 D_1 + e_i^2 D_2) = 0$ and therefore $\lambda \equiv \frac{\sum_i v_i e_i^1}{\sum_i v_i e_i^2} \mod N$. We get the discrete logarithm when the denominator is non-zero, which happens with probability $\frac{N-1}{N}$.

It is interesting to note that here we encounter a first difference between the outline of Gaudry's algorithm and the original index calculus or Coppersmith's variants. Instead of trying to obtain a matrix that has full column rank, we attempt to obtain a linear dependency between the rows of the matrix, which is more or less the exact opposite. In this respect, the linear algebra step is more similar to the QS or the NFS rather than the first index calculus sketch. As a side effect, we will not encounter the same behaviour as before, in which the preprocessing stage could have been reused to compute other individual logarithms. This is because, in this instance, we do not recover the logarithms for the factor-base elements, but only the logarithm of the target element.

Hyperelliptic involution improvement In fact, one might reduce the size of the matrix M by a factor of 2, just by reducing the size of the factor base by 2. For a given polynomial pair (U, V), eliminate $\omega((U, V)) = (U, -V)$ from the factor base. During the construction of M, when we find

that an element contains a factor of $(U, -V)$, say $\beta(U, -V)$, we put $-\beta$ as the corresponding value for (U, V) in M.

Proposition 6.40. [4]: $|J_{\mathbb{F}_{p^n}}(C)| \approx p^{ng}$ when p^n is much larger than $4g^2$.

For the complexity analysis, we will have to assume that the genus is quite small compared to the size of the base field, $\mathbf{p^n \gg 4g^2}$. Recall the generalized Hasse-Weil bound, $(\sqrt{p^n} - 1)^{2g} \leq |J_{\mathbb{F}_{p^n}}(C)| \leq (\sqrt{p^n} + 1)^{2g}$. This means that $|J_{\mathbb{F}_{p^n}}(C)| \approx p^{ng}$ with an error not greater than $2gp^{n(g-\frac{1}{2})}$, since from the binomial theorem, $p^{ng} + \sum_{k=0}^{2g-1} \binom{2g}{k} p^{\frac{nk}{2}} \leq p^{ng} + 2gp^{n(g-\frac{1}{2})}$. This error is small with respect to p^{ng} when $\frac{2gp^{n(g-\frac{1}{2})}}{p^{ng}}$ is small, which indeed holds under the assumption we have made.

Proposition 6.41. [4]: The Jacobian of a curve of genus g over $\mathbb{F}_{p^n}$ has a proportion of $\frac{1}{g!}$ $1-smooth$ divisors when $p^n \to +\infty$.

Let us make a very rough analysis for the number of $\mathbb{F}_{p^n} - rational$ points on our curve. For every $x \in \mathbb{F}_{p^n}$, we have, when x is a quadratic residue, two corresponding solutions for y. However, only half the elements of $\mathbb{F}_{p^n}$ are quadratic residues, therefore we get an estimate of $p^n + 1$ $\mathbb{F}_{p^n} - rational$ points, since we have also counted the point at infinity. Remember that reduced divisors had at most g zeros, corresponding to points on the curve. Since the order of these g zeros is unimportant, we have that $\frac{(p^{ng}+1)!}{g!}$ divisors made up of $\mathbb{F}_{p^n} - rational$ points are 1-smooth, and hence their proportion is $\frac{1}{g!}$.

Remark: Albeit the approach was different this time, it seems that the number of smooth divisors of $J_{\mathbb{F}_{p^n}}(C)$ can be approximated using divisors whose points are actually in $\mathbb{F}_{p^n}$ (even though the former are only defined over $\mathbb{F}_{p^n}$ which means that they do not necessarily have all their points in $\mathbb{F}_{p^n}$).

Complexity

1. There are p^n monic polynomials of degree 1 we consider as an $U(x)$, for each of them we need to solve the equation $V^2 \equiv f(x) \mod U(x)$.
2. Generate random elements for the random walk, several simple operations in the Jacobian.
3. We need to generate the matrix M until it has $|G| + \epsilon$ rows.
   (a) Computing a new element of the random walk E_{i+1} requires an addition in the Jacobian. We also perform two additions in $\mathbb{Z}_{/N\mathbb{Z}}$ to keep track of the decomposition into $e^1_{i+1}D_1 + e^2_{i+1}D_2$. We will then test this for $1 - smoothness$ by computing $gcd(x^{p^n} - x, U(x))$. This step uncovers a smooth divisor once every $g!$ iterations.
   (b) Once we have a smooth divisor, we need to factor it, if we do trial division we need $|G|$ operations using polynomials in the base field, so approximately p^n polynomial divisions.

4. Performing the linear algebra computations can be done using a variation of the Lanczos algorithm which recovers elements of the kernel, better than Gauss' elimination $|G|^3$, in $g|G|^2$ steps.

According to [4], the complexity of the algorithm is $O(p^{2n} + g!p^n)$ of $glog(p^n)$ operations.

Automorphisms The fact that using the hyperelliptic involution we can reduce the size of the matrix by half suggests that all automorphisms could be used to help with the linear algebra step. Assume once more that $J_{\mathbb{F}_{p^n}}(C)$ is cyclic of prime order N with D_1 as its generator. Recall that $\sim$ is the relation of equivalence where $D_i \sim D_j \iff \exists w \in \mathbb{Z}_{/N\mathbb{Z}}$ such that $D_i = \sigma^w D_j$. From each of the m equivalence classes we are only going to keep one element in the factor base, denote it by g_w, where $w = \overline{1, m}$. Now, let us consider an arbitrary $D = P_1 + P_2 + ... + P_k = e^1 D_1 + e^2 D_2$, with P_l prime divisors of degree 1, $l = \overline{1, k}$. For all l, there exists w_l such that $P_l = \sigma^{w_l}(g_{w_l})$. and thus we discover that $D = \sum_{l=0}^{k} \theta^{w_l} g_{w_l}$. We reduce therefore the matrix by a factor of m, which would hopefully translate to a speedup of m^2 during the chosen linear algebra step.

7 Thériault's Algorithm

In the following, we are going to describe an improvement of Gaudry's algorithm, under the assumption that the genus of the curve is greater than 3 but it is even smaller with respect to the size of the base field: $p^n > (g-1)!$. The main reference for this section is [13].

Factor base Again, we are only going to consider divisors of degree 1 (the degree of the corresponding U polynomial is 1) for our factor base. However, we do not choose all such divisors but decide instead to limit ourselves to a subset of reduced divisors B_D. If $deg(U) = 1$, each element of the factor base will be equivalent to a divisor $D = (P_1) - (P_\infty)$. Using the aforementioned correspondence it is clear that we are choosing a subset of $\mathbb{F}_{p^n} - rational$ points, which we denote by B. We choose B such that $|B| = p^{nr}$ and then approximate the cardinality of the set of reduced divisors B_D as equal to B. Therefore, we may give a slightly modified definition of smoothness, namely:

Definition 7.1. A reduced divisor D is smooth relative to B if $D = \sum_{i=1}^{g} D_i$, with $D_i \in B_D$.

Proposition 7.2. When $\frac{2}{3} < r < 1$ there are $\frac{p^{nrg}}{g!} + \mathcal{O}(\frac{g^2 p^{nr(g-1)}}{g!})$ smooth divisors in $J_{\mathbb{F}_{p^n}}(C)$. [13]

Proof: Recall that a divisor D, smooth relative to B is in the same divisor class as the divisor $\sum_{i=1}^{g}[(P_i) - (P_\infty)]$ where $(P_i) - (P_\infty) \in B$. Since U splits completely, the points P_i must be $\mathbb{F}_{p^n} - rational$. Therefore, we may look at our divisors as multi-sets of g points. The number of divisors made up of g distinct points P taken from B is $\binom{p^{nr}}{g} = \frac{1}{g!}\prod_{i=0}^{g-1}(p^{nr} - i) = \frac{p^{nrg} - (g-1)p^{nr(g-1)}}{g!} + \mathcal{O}(p^{nr(g-2)})$, while the number of smooth divisors with g points but only $g-1$ distinct ones is $(g-1)\binom{p^{nr}}{g-1} = \frac{g-1}{(g-1)!}\prod_{i=0}^{g-2}(p^{nr} - i) = \frac{p^{nr(g-1)}}{(g-2)!} + \mathcal{O}(p^{nr(g-2)})$ (the $(g-1)$ factor arises because

any of the $g-1$ points might be the repeated one). On the other hand, the number of $B-smooth$ divisors with $g-1$ distinct points is $\binom{p^{nr}}{g-1} = \frac{p^{nr(g-1)}}{(g-1)!} + \mathcal{O}(p^{nr(g-2)})$ and the number of $B-smooth$ divisors of less than $g-1$ distinct points is $\mathcal{O}(p^{nr(g-2)})$. We may not conclude without first showing that:

$$\mathcal{O}(p^{nr(g-2)}) \subseteq \mathcal{O}(\frac{g^2 p^{nr(g-1)}}{g!})$$

Indeed, $p^n > (g-1)!$ and from that $gp^{nr(g-1)} > (g-1)!$ when $r(g-1) \geq 1$ ($\frac{2}{3}2 \geq 1$ for $g=3$). This implies that $\frac{g^2 p^{nr(g-1)}}{g!} = \frac{g^2 p^{nr(g-2)} p^{nr}}{g!} \geq \frac{g^2 p^{nr(g-2)}}{g} \geq p^{nr(g-2)}$. Now we may sum up all the terms and we are done.

Proportion of B-smooth divisors Recall that due to the generalized Hasse-Weil bound, we know that the cardinality of $J_{\mathbb{F}_{p^n}}(C)$ is equal to p^{ng} with an error not greater than $2gp^{n(g-\frac{1}{2})}$. Therefore, the proportion of $B-smooth$ divisors is $\frac{\frac{g^{rs}}{g!}+\mathcal{O}(\frac{g^2 p^{nr(g-1)}}{g!})}{p^{ng}+\mathcal{O}(gp^{n(g-\frac{1}{2})})} \approx \frac{1}{g! p^{n(1-r)g}}$. Furthermore, assuming that our pseudo-random walk is indeed random, we expect to analyse approximately $g! p^{n(1-r)g}$ divisors before we encounter a $B-smooth$ one.

Definition 7.3. An $\mathbb{F}_{p^n} - rational$ point P is called a large prime if $P \notin B$ (therefore it does not appear in the expression of any divisor in B_D).

Note that the reduced divisor $(P) - (P_\infty)$ is a prime divisor, its polynomial U is linear.

Definition 7.4. An almost-smooth divisor D is a reduced divisor that contains only $\mathbb{F}_{p^n} - rational$ points plus an additional large prime.

Proposition 7.5. When $\frac{2}{3} < r < 1$ there are $\frac{p^{n(rg+1-r)}}{(g-1)!} + \mathcal{O}(\frac{g^{rg}}{(g-1)!})$ almost-smooth divisors in $J_{\mathbb{F}_{p^n}}(C)$. [13]

Proof: Associate each $almost-smooth$ divisor to its large prime component and use the proposition regarding $B-smooth$ divisors. From Hasse-Weil's bound, there are $(p^n+1) - \mathcal{O}(2g\sqrt{p^n})$ $\mathbb{F}_{p^n} - rational$ points on the hyperelliptic curve. This amounts to $p^n - p^{nr} + \mathcal{O}(\sqrt{p^n})$ choices for the large prime, since g is a small constant and $1 \in \mathcal{O}(\sqrt{p^n})$. The number of $B-smooth$ divisors made up of a maximum of $(g-1)$ points comes from substituting g for $(g-1)$ in the last proposition, getting $\frac{p^{nr(g-1)}}{(g-1)!} + \mathcal{O}(\frac{(g-1)^2 p^{nr(g-2)}}{(g-1)!})$. The total quantity is therefore $\frac{p^{n(r(g-1)+1)}}{(g-1)!} - \frac{p^{nrg}}{(g-1)!} + \mathcal{O}(\frac{p^{n(rg+\frac{1}{2}-r)}}{(g-1)!}) + \mathcal{O}(\frac{(g-1)^2 p^{n(rg+1-2r)}}{(g-1)!}) + \mathcal{O}(\frac{(g-1)^2 p^{nr(g-1)}}{(g-1)!}) + \mathcal{O}(\frac{(g-1)^2 p^{n(rg-2r+\frac{1}{2})}}{(g-1)!})$. Now, observe first that $\mathcal{O}(\frac{(g-1)^2 p^{nr(g-1)}}{(g-1)!}) \subseteq \mathcal{O}(\frac{(g-1)^2 p^{n(rg+1-2r)}}{(g-1)!})$, because $r<1$. Moreover, for $\sqrt{p^n} \geq (g-1)^2$ (according to our assumption), we also get $\mathcal{O}(\frac{(g-1)^2 p^{n(rg-2r+\frac{1}{2})}}{(g-1)!}) \subseteq \mathcal{O}(\frac{(g-1)^2 p^{n(rg+1-2r)}}{(g-1)!})$. We are therefore left with an estimate of $\frac{p^{n(r(g-1)+1)}}{(g-1)!} - \frac{p^{nrg}}{(g-1)!} + \mathcal{O}(\frac{p^{n(rg+\frac{1}{2}-r)}}{(g-1)!}) + \mathcal{O}(\frac{(g-1)^2 p^{n(rg+1-2r)}}{(g-1)!})$. Now, $\mathcal{O}(\frac{p^{nrg}}{(g-1)!})$ obviously contains the last two sets, and we are done.

Using large primes The rationale for the last two propositions has been to show that the number of almost-smooth divisors is significantly greater than the number of smooth divisors, by a factor of more than $p^{n(1-r)}$. Since practicalities concerning the computation of the order of the Jacobian usually require us to work in small genus curves (that and Adleman's first index calculus attack on hyperelliptic curves of large genus), we will normally work in small genus curves. What this means is that in order to ensure that $|J_{\mathbb{F}_{p^n}}(C)|$ is of cryptographic size, we will need to define our hyperelliptic curve over a large finite field.

We get therefore that p^n is reasonably large, which means that we will encounter far more almost-smooth divisors than smooth ones. In the following, we are going to detail a rather simple approach to using the former, as proposed by [13].

Assume that we have two almost-smooth reduced divisors D_1 and D_2, where both D_1 and D_2 contain a large prime P_i relative to the chosen factor-base of $\mathbb{F}_{p^n} - rational$ points. For both these divisors, the multiplicity of P_i must be strictly positive, from the definition of almost-smooth divisors this multiplicity is 1. Then, one might be tempted to compute $D_1 - D_2$ and then apply Cantor reduction, in order to obtain as a result, a smooth divisor (when $D_1 \neq D_2$ this divisor should not be equivalent to the zero divisor). But what does $D_1 - D_2$ mean? The answer will yield another useful application of the hyperelliptic involution automorphism. First recall that if a semi-reduced divisor D_1 is written using the Mumford polynomials (U, V), we have proven that $-D1 = (U, -V)$. Moreover $-D1 = \omega(D_1)$ therefore we may define $D_1 - D_2$ using Cantor's algorithm for addition $D_1 + \omega(D_2)$.

Linear algebra step with large primes For the following, we need to assume that for each P_i in the factor-base, we also include $\omega(P_i)$, although this is by no means mandatory from a practical point of view. Before analysing the large primes case, let us consider the number of relations we need to collect if we use only smooth divisors. According to [13], this should be in $\mathcal{O}(|B| = p^{nr})$, but in practice, for a small epsilon, a matrix with $p^{nr} + \epsilon$ relations would normally suffice. We have shown that the proportion of smooth divisors is $\mathcal{O}(g! p^{n(1-r)g})$, therefore we expect that we shall have to analyse $\mathcal{O}(g! p^{n(1-r)g+nr})$ smooth divisors until we can find a linear dependency.
Let us see what happens when we introduce large primes into the equation. Without losing generality, assume that there is only one large prime that we can choose (this might have happened according to the choice of r). If we were to build the entire matrix, before solving the linear system of equations, we need to get rid of the extra column, the one corresponding to the large prime, eventually by Gaussian elimination, in order to obtain a smaller matrix that corresponds only to relations involving smooth divisors. Instead of going through this additional troublesome steps, Thériault's method employs a different approach.

For each almost-smooth divisor encountered, if the large prime P_i that belongs to it has not been

previously seen, we add the almost-smooth divisor to a list of intersections belonging to the large prime, and we do not attempt to use it otherwise for the moment. In the future, let us say that we recover another almost-smooth divisor D_2 that contains the large prime P_i. Since the intersection list is not empty, containing at least D_1, we will compute the reduction of $D_1 + \omega(D_2)$. This will be a smooth divisor which we can add to our matrix. [13]

Ensuring an appropriate number of intersections In order to build a matrix of full column rank, we need to have slightly more relations than elements in our factor-base, which amounts to $\mathcal{O}(p^{nr})$. The parameter that must be fine-tuned in order to make amends for this requirement is r, which according to [13], should be $\frac{g+log_{p^n}((g-1)!)}{g+1}$ for the variant using only smooth divisors. For the algorithm that uses almost-smooth divisors and intersections lists, r should be equal to $\frac{g-1/2+log_{p^n}((g-1)!/g)}{g+1/2}$. For shortness, we are not going to dwell into any more details here.

Beyond large primes An interesting idea for future work is to extend the definition of an almost-smooth divisor. We could allow our reduced divisor to have a positive multiplicity different from 1 in the corresponding large prime P, $D = n_P(P) + (\sum_{i=1}^{g-n_p}(P_i)) - g(P_\infty)$, where $P_i \in B$. In order for this to work, we will have to maintain n_P different intersection lists. Since every list contains no more than one divisor, this is not a problem in terms of occupied memory. Moreover, since g is quite small, we expect that the number of intersection lists is quite small. What is not clear at the moment is how this impacts the complexity of the algorithm. In the worst case scenario where this strategy does not yield more intersections than in Thériault's approach, we expect to perform g more operations, which is not a significant slowdown.
At the moment, the main difficulty with using different multiplicities for the large primes concerns the space requirements of the algorithm. In [13], the author remarks that in order to keep intersection lists for all large primes, we need to store the corresponding almost smooth divisors, along with $e^1, e^2 \in \mathbb{Z}_{/|J_{\mathbb{F}_{p^n}}(C)|\mathbb{Z}}$ for which $D = e^1 D_1 + e^2 D_2$. This actually requires more space than we need for the linear algebra step.

7.1 The genus 1 case

When the genus of a hyperelliptic curve is 1, we are actually dealing with an elliptic curve. The problem of index calculus for elliptic curves is an ongoing research topic, and the main problem is how to define smoothness on such varieties. Reduced divisors would contain only one point. This means that the $U(x)$ polynomial splits completely over $\mathbb{F}_{p^n}$. If we were to apply Gaudry's algorithm in such a setting, we would discover that we have to include all the reduced divisors into the factor-base, which is obviously impractical. The other option, Thériault's algorithm, would still require that we select at least $\frac{2}{3}$ of the total number of $\mathbb{F}_{p^n} - rational$ points. This is obviously not an improvement in any way. There are attempts to tackle index-calculus on elliptic curves, a good

first reference in this direction is a survey by Silverman. In the following, we are going to briefly present two methods of transferring the DLP on an elliptic curve to the DLP in a group where index-calculus is possible. First, let us describe the ECDLP:

Definition 7.6. [6]: ECDLP Recall that when C is an elliptic curve defined over $\mathbb{F}_{p^n}$, the points defined over a finite field $C(\mathbb{F}_{p^n})$ form an abelian group with respect to the addition law on elliptic curves. Therefore, the ECDLP is defined in terms of basic group theory: Let $P \in C(\mathbb{F}_{p^n})$ be a point of large order and $Q \in C(\mathbb{F}_{p^n})$. We are being asked to find $\lambda \in \mathbb{Z}_{/|C(\mathbb{F}_{p^n})|\mathbb{Z}}$ such that $Q = \lambda P$.

MOV attack The MOV attack uses the Weil pairing to transfer the DLP on the elliptic curve to the DLP in finite fields, where there exists a myriad of index-calculus variations. Similar to the MOV attack is the approach due to Frey and Rück, which uses the Tate-Lichtenbaum pairing in precisely the same fashion, albeit with less restrictions and faster computations.

Definition 7.7. [14] The Weil pairing is a map $e : E[m] \times E[m] \to \mu_m$, where $E[m]$ is the $m - torsion$ group and μ_m is the group containing the $m - th$ roots of unity, with $(m, p) = 1$ and $m \nmid (p^n - 1)$, which satisfies the following properties:

1. e_m is bilinear in each variable.
2. e_m is nondegenerate in each variable, meaning that when $e_m(S, T) = 1, \forall S \in E[m]$, T must equal P_∞ and vice-versa.
3. $e_m(T, T) = 1, \forall T \in E[m]$.
4. $e_m(T, S) = e_m(S, T)^{-1}, \forall S, T \in E[m]$.

The following paragraphs follow the presentation from [5]. Firstly, observe that we can safely assume, without loss of generality, that the order m of the base point P is prime. Let us look at the order k of p^n in $\mathbb{Z}_{/m\mathbb{Z}}$, the smallest integer such that $p^{nk} \equiv 1 \mod m$. Now, remark that $m/(p^{nk} - 1)$. From Cauchy's theorem, this means that there exists an element of order m in $\mathbb{F}^*_{p^{nk}}$. Now consider the subgroup generated by said element, this will be a cyclic subgroup of order m of $\mathbb{F}^*_{p^{nk}}$, which is of course isomorphic to μ_m. We are now ready to describe the MOV attack, assume that you are given P, Q and must recover λ such that $Q = \lambda P$. We are going to randomly choose an element $T \in E[m]$, then compute $e_m(P, T)$ and $e_m(Q, T) = e_m(P, T)^\lambda$. Therefore, we can use a sub-exponential index calculus attack in $\mathbb{F}^*_{p^{nk}}$ to compute $\lambda \equiv log_{e_m(P,T)} e_m(Q, T) \mod m$.
The order of p^n in $\mathbb{Z}_{/m\mathbb{Z}}$, is called the embedding degree, and it is important because it defines the size of the extension field in which we solve the DLP. In fact, the parameters are going to suffer an expansion of $\mathcal{O}(k)$, and when k is very large, it becomes too costly to solve the DLP, even with the sub-exponential approaches.
It is also possible to perform another type of transfer for the DLP. Instead of trying to find a mapping

into a finite field, we use the Weil restriction of the elliptic curve to obtain another algebraic variety. The GHS attack, for instance, reduces this variety to an hyperelliptic curve, on which we can hope to use index calculus.

7.2 The genus 2 case

Both algorithms for index calculus on hyperelliptic curves require that the genus remains small. An important observation here is that when the genus becomes 2, it is actually too small, and the square root attacks perform better than what we hoped to be a more efficient approach. This is due to the proportion of reduced divisors of a curve of genus 2, on which the factor-base becomes simply too large with respect to the total number of divisors for us to have an efficient approach. Methods like reducing the factor-base might yield good results in this case, but it depends on the order of the automorphism utilized.

8 Practical considerations for HCDLP

In this section, we are going to describe the implementation of an index calculus attack on hyperelliptic curves, along with an example for the Rho-Pollard algorithm. Before we do this however, we need a way to compute the cardinality of the Jacobian. There exist algorithms which allow for this computation to be done in the case of a randomly picked hyperelliptic curve. Nevertheless, we are going to use the approach found in [6].

Definition 8.1. [6]: For a hyperelliptic curve C of genus g defined over $\mathbb{F}_{p^n}$ and $M_r = J_{\mathbb{F}_{p^{nr}}}(C)$, we define the zeta function as $Z(C_{/\mathbb{F}_{p^n}}; T) = e^{\sum_{r=1}^{\infty} \frac{M_r T^r}{r}}$.

We present the following theorem, which yields practical importance for the zeta function in our context, due to Weil:

Theorem 8.2. [6]: *For a hyperelliptic curve C of genus g, defined over $\mathbb{F}_{p^n}$, it holds that the zeta function can be expressed as a rational function:*

1. $Z(C_{/\mathbb{F}_{p^n}}; T) = \frac{P(T)}{(1-T)(1-p^nT)}$, *with* $P(T)$ *a polynomial of degree* $2g$, *of the form* $P(T) = \sum_{j=0}^{g} a_j T^j + \sum_{j=0}^{g-1} a_j p^{n(g-j)} T^{2g-j}$, *with* $a_0 = 1$.
2. $P(T) = \prod_{i=1}^{g} (1 - \alpha_i T)(1 - \overline{\alpha_i} T)$, *with* α_i *a complex root of* $P(T)$, *smaller in absolute value than* $\sqrt{p^n}$.
3. $|J_{\mathbb{F}_{p^{nr}}}(C)| = \prod_{i=1}^{g} |(1 - \alpha_i^r)|^2$. *Since* $\overline{1 - \alpha_i} = 1 - \overline{\alpha_i}$, *we see that* $|J_{\mathbb{F}_{p^n}}(C)| = P(1)$.

By taking logarithms, one may see that $log(Z(C_{/\mathbb{F}_{p^n}}; T)) + log(1 - T) + log(1 - p^nT) = log(P(T))$, and by differentiating, $\sum_{r=1}^{\infty} M_r T^{r-1} - \frac{1}{1-T} - \frac{p^n}{1-p^nT} = \frac{P'(T)}{P(T)}$. If $|p^nT| < 1$, we know

that $\frac{1}{1-T} = \sum_{r=0}^{\infty} T^r$ and $\frac{p^n}{1-p^nT} = \sum_{r=0}^{\infty} p^{n(r+1)}T^r$. We obtain thus that:

$$\frac{P'(T)}{P(T)} = \sum_{r=0}^{\infty}(M_{r+1} - 1 - p^{n(r+1)})T^r \implies P'(T) = P(T)\sum_{r=0}^{\infty}(M_{r+1} - 1 - p^{n(r+1)})T^r$$

In order to compute the roots of $P(T)$, we must first recover the coefficients of $P(T)$, namely a_i, when $i = \overline{1,g}$. According to [6], in order to do that we have to exhaustively compute M_i, for $i = \overline{1,g}$.

8.1 An example for Rho-Pollard

We have implemented an additive variant of Rho-Pollard in SAGE, making allowances for the early abort when we detect infinite cycles. Moreover, we use the hyperelliptic involution in the hope of obtaining a theoretical improvement of $\sqrt{2}$, usually amortized by other costs, as in the additional search for collisions. We have chosen the genus 2 smooth curve $C : y^2+2xy = -x^5-x^4+x^2-2x+1$, defined over $\mathbb{F}_3$. We then compute the number of $\mathbb{F}_3 - rational$ and $\mathbb{F}_{3^2} - rational$ points, 3 and 19 respectively. Using the aforementioned procedure we find that the cardinality of $J_{\mathbb{F}_{3^3}}(C)$ is $869 = 11 * 79$. The cardinality is rather smooth, but this is not important for our discussion. For computing the order of a random divisor, we employ a double-and-add procedure that uses the known cardinality of the Jacobian. This is much faster than square-root strategies, polynomial if we already know the factorization of the cardinality. We consider the field extension $\mathbb{F}_3[x]_{/(x^3+2x+1)} \simeq \mathbb{F}_3[a] \simeq \mathbb{F}_{3^3}$, where $a^3 + 2a + 1 = 0$. We recover a reduced divisor $(x + a^2 + 1, a^2 + 2a + 2)$ of order 869.

Hash function As suggested in [4], the hash function should be very fast and provide randomness for our additive walk. In our implementation, we are summing up the ASCII codes of the characters that appear in the object string representation, followed by an operation modulo the number of random divisors.

Setup We are required to compute the logarithm of $D_t = (x^2 + (2a^2 + 2)x + 2a^2 + 2 * a + 1, ax + a^2 + a + 1)$. First, preinitialize the random walk by computing 5 random divisors:

- $R_0 = \left(x^2 + \left(2a^2 + a\right)x + 2a^2 + 2a + 2, \left(a^2 + 2a + 1\right)x + 2a^2\right) = 709D + 783D_t$
- $R_1 = \left(x^2 + a + 2, a^2 + a\right) = 468D + 712D_t$
- $R_2 = \left(x^2 + (a + 1)x + a^2, (a + 1)x + 2a^2 + a + 2\right) = 801D + 270D_t$
- $R_3 = \left(x^2 + (a + 2)x + 2a^2 + a + 1, ax + a\right) = 656D + 718D_t$
- $R_4 = \left(x^2 + \left(2a^2 + a\right)x + 2a^2 + 2a, (a + 1)x + 2a\right) = 11D + 792D_t$

The starting point for our walk is also random, $(x^2+(a+1)x+a+1, (a^2+a+1)x+2a^2+a+1) = 811D + 55D_t$. We construct the following chain of divisors:

- $D_0 = \left(x^2 + (a + 1)x + a + 1, \left(a^2 + a + 1\right)x + 2a^2 + a + 1\right) = 811D + 55D_t$, $Hash(D_0) = 2$
- $D_1 = \left(x^2 + x + 2a^2 + a + 1, 2a^2x + 2a^2 + a + 1\right) = D_0+801D+270D_t = 743D+325D_t$, $Hash(D_1) = 0$

- $D_2 = \left(x^2 + \left(2a^2 + 2a + 1\right)x + 2a^2 + a + 1, (2a + 2)x + 2a^2 + a + 2\right) = D_1 + 709D + 783D_t = 583D + 239D_t$, $Hash(D_2) = 0$
- $D_3 = \left(x^2 + 2a^2x + 2, \left(a^2 + 2\right)x + 2a + 1\right) = D_2 + R_0 = 423D + 153D_t$, $Hash(D_3) = 1$
- $D_4 = \left(x^2 + \left(a^2 + a + 1\right)x + a^2 + 2a + 2, \left(2a^2 + a + 2\right)x + 2a^2 + a + 2\right) = 22D + 865D_t$, $Hash(D_4) = 2$
- $D_5 = \left(x^2 + (2a + 1)x + 2a^2 + a, \left(a^2 + 2a + 1\right)x + 2a^2 + 1\right) = 823D + 266D_t$, $Hash(D_5) = 0$
- $D_6 = \left(x^2 + \left(2a^2 + 2a + 2\right)x + a^2 + 1, \left(2a^2 + a + 1\right)x + a\right) = 663D + 180D_t$, $Hash(D_6) = 2$
- $D_7 = \left(x + a^2 + 2a + 2, a^2 + a + 2\right) = 595D + 450D_t$, $Hash(D_7) = 1$
- $D_8 = \left(x^2 + \left(2a^2 + a\right)x + a^2 + 2, \left(2a^2 + a + 1\right)x + a\right) 194D + 293D_t$, $Hash(D_8) = 4$
- $D_9 = \left(x^2 + \left(2a^2 + a\right)x + 2a^2 + 2, (2a + 1)x + 2a^2 + a + 1\right) 205D + 216D_t$, $Hash(D_9) = 1$
- $D_{10} = \left(x^2 + (2a + 1)x + 2a^2 + a, \left(2a^2 + a\right)x + a^2 + 2\right) = 673D + 59D_t$.

We detect a collision using the hyperelliptic involution, namely $D_{10} = \omega(D_5)$. We therefore recover $\lambda \equiv -\frac{673+823}{59+266} \mod 869 \equiv 220$. The algorithm can be easily adapted to check for collisions using other automorphisms. Another observation is that, since we work with reduced divisors, we do not have to worry about well defined mappings, should the need to parallelize arise.

8.2 Index Calculus

In Rho Pollard, we require no knowledge of the cardinality of the Jacobian, although this is a known piece of information in our toy example. It might seem that this is also the case for index calculus, nevertheless one finds that it is not sufficient to know only the cardinality of the base point, since this might bear no influence on the order of factor-base elements. The matrix construction phase does not need to take into account whether the base divisor is an element of the factor-base or not. We are going to work on an adaptation of Gaudry's algorithm, for our genus 2 curves. The divisors that are not part of the factor-basis are those divisors for which the $U(x)$ polynomial is either irreducible, or if it is reducible, there are no corresponding points on the curve when we attempt to find a solution for the y coordinate.

For genus 2 curves, the matrix we collect is going to be extremely sparse, with no more than two non-zero entries on each row. The size of the matrix is halved using the hyperelliptic involution. This is rather natural, we have elected to match the prime divisors of degree 1 to their position in the matrix using the unique solution of the $U(x)$ polynomial (a point on the hyperelliptic curve). In our implementation, for practical reasons we will use the integer representation of the point (it is easy to represent any finite field element as an integer). If in our matrix we take each row as the representative of a linear relation, it means that we are required to compute one element from the left null-space of the matrix. Recall that in the presentation of Gaudry's algorithm we have pointed out the difference between it and other index calculus methods. Another consequence of computing only a linear dependency between the rows of our matrix is that we limit ourselves to computing one individual discrete logarithm, without recovering any logarithms for factor-base elements.

A toy example Let us consider $C : y^2 + (2x^2 + 1)y = 2x^5 + x^4 + 2x^2 + 2x$, defined over $\mathbb{F}_3$ (we are nonetheless going to look at $J_{\mathbb{F}_{3^2}}(C)$). There are 5 $\mathbb{F}_3 - rational$ points and 19 $\mathbb{F}_{3^2} - rational$ points, from which we determine that $|J_{\mathbb{F}_{3^2}}(C)| = 209$. From the 19 $\mathbb{F}_{3^2} - rational$ points, we leave out the point at infinity, halve the set using the hyperelliptic involution, and we obtain 9 $1 - smooth$ divisors for our factor-base. We are required to find the logarithm of $D_t = (x^2 + ax + 2a + 2, (2a + 1)\, x + a)$ with respect to $D = (x + a + 2, 1)$, where $ord(D) = 209$. In order to keep the presentation short, we will not exhibit an entire additive walk, but only those divisors that are $1 - smooth$.

- $D_0 = \left(x^2 + (a+1)\, x + 1, (a+1)\, x + a\right) = 191D + 28D_t = -\left(x + 2a, a + 2\right) - \left(x + 2a + 1, 1\right)$
- $D_1 = \left(x^2 + 2ax, (a+2)\, x\right) = 145D + 172D_t = (x, 0) - (x + 2a, a + 2)$
- $D_2 = \left(x^2 + 2x + a + 2, a + 2\right) = 66D + 23D_t = (x + a, a + 2) - (x + 2a + 2, 2a + 2)$
- $D_3 = \left(x^2 + 2x + 2, 1\right) = 85D + 202D_t = (x + a + 2, 1) - (x + 2a, a + 2)$
- $D_4 = \left(x^2 + (a+1)\, x + 2a + 1, (2a+2)\, x + a\right) = 6D + 53D_t = (x + 2, 2) - (x + a + 2, 1)$
- $D_5 = \left(x^2 + (a+2)\, x + a + 1, (a+1)\, x + 2a + 2\right) = 0D + 4D_t = -\left(x + 1, 2a + 2\right) - \left(x + a + 1, a + 1\right)$
- $D_6 = (x + 1, 2a + 2) = 161D + 105D_t = (x + 1, 2a + 2)$ (already prime)
- $D_7 = \left(x^2 + 2a + 2, a + 2\right) = 180D + 75D_t = (x + a, a + 2) + (x + 2a, a + 2)$
- $D_8 = \left(x^2 + ax, (a+1)\, x\right) = 138D + 16D_t = (x, 0) + (x + a, a + 2)$
- $D_9 = \left(x^2 + (a+1)\, x + a, (a+1)\, x + 2a + 2\right) = 157D + 195D_t = -\left(x + 1, 2a + 2\right) - \left(x + a, a + 2\right)$
- $D_{10} = \left(x^2 + (2a+2)\, x + 1, (2a+2)\, x + 2a + 1\right) = 78D + 46D_t = (x + a, a + 2) + (x + a + 2, 1)$
- $D_{11} = \left(x^2 + (a+2)\, x + 2a, (2a+2)\, x + a\right) = 32D + 190D_t = (x + 2, 2) - (x + a, a + 2)$

We look at the decomposition into prime divisors of degree 1 and obtain the following matrix. The number of the column assigned to each divisor from the factor base is given on a first-come first-served basis:

$$\begin{pmatrix}
208 & 208 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
208 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 208 & 0 & 0 & 0 & 0 \\
208 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 208 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 208 & 208 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 208 & 0 & 0 & 0 & 208 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 208 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$

We proceed by looking at the factorization of $209 = 11 \times 19$, and we recover the following two elements from the left kernel:

$(0, 10, 0, 5, 4, 0, 8, 4, 1, 8, 10, 7)$, modulo $\mathbb{Z}_{/11\mathbb{Z}}$ and $(0, 14, 0, 15, 2, 0, 4, 10, 5, 4, 6, 17)$, modulo $\mathbb{Z}_{/19\mathbb{Z}}$.

We combine these using the CRT to obtain the following vector from the left null-space of the matrix:

$(0, 109, 0, 148, 59, 0, 118, 48, 100, 118, 120, 150)$, modulo $\mathbb{Z}_{/209\mathbb{Z}}$

It is now time for the final step in the recovery of the logarithm, in which we produce a "collision" for Rho-Pollard. First let us compute the numerator, as $109 * 145 + 148 * 85 + 59 * 6 + 118 * 161 + 48 * 180 + 100 * 138 + 118 * 157 + 120 * 78 + 150 * 32 = 35 \mod 209$. The denominator is computed similarly, as $109 * 172 + 148 * 202 + 59 * 53 + 118 * 105 + 48 * 75 + 100 * 16 + 118 * 195 + 120 * 46 + 150 * 190 = 155$. Finally, the logarithm is determined as:

$$\lambda \equiv -\frac{35}{155} \equiv 20 \mod 209$$

9 Conclusions

In parallel with the latest breakthroughs in the theory of algebraic curve cryptosystems, there is also an ongoing effort to find better attacks. Index calculus is at the forefront of such efforts, which in particular focus on solving the ECDLP for elliptic curves that are currently part of cryptographic standards. At the moment, there are no successful attacks, nevertheless assuming such an attack is found, alternatives are easily available. One of these alternatives is represented by hyperelliptic curves of genus 2, chosen in such a way that they have a big enough embedding degree and a large enough prime in the factorization of the order of the Jacobian. We have made a survey of hyperelliptic curves, and implemented Gaudry's index calculus variant, along with a generic square root attack, Rho-Pollard. We have not included the usage of automorphisms in neither algorithms since they depend on the choice of the curve, these techniques may nevertheless be easily integrated with our implementation. A possible direction of development here revolves around work on Thériault algorithm, there already exist various improvements that rely on the usage of more that just one large prime.

As far as index-calculus on finite fields is concerned, the amount of work being done here is tremendous. In fact, our paper could be considered but a shy introduction into this subfield, with possible future work on improvements of NFS or FFS. Moreover, it would be interesting to see how do these improvements impact factorization solvers. Finally, an important aspect we have unfortunately neglected in this paper relates to linear algebra algorithms, either solving linear systems or finding an element belonging to the kernel of a matrix. For example, one might attempt to improve Lanczos's block algorithm or some other techniques, such as Berlekamp–Massey algorithm or even iterative methods for finite rings.

References

[1] Don Coppersmith, Andrew M. Odlyzko, and Richard Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1):1–15, January 1986.

[2] Richard Crandall and Carl Pomerance. *Prime numbers: a computational perspective. Second Edition*. 2010.

[3] I. Duursma, P. Gaudry, and F. Morain. Speeding up the discrete log computation on curves with automorphisms, 1999.

[4] Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. Springer-Verlag, 2000.

[5] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.

[6] N. Koblitz, A.J. Menezes, Y.H. Wu, and R.J. Zuccherato. *Algebraic Aspects of Cryptography*. Algorithms and Computation in Mathematics. Springer Berlin Heidelberg, 2012.

[7] David Marcus. *Number Fields*. 1977.

[8] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, and R. L. Rivest. *Handbook of Applied Cryptography*. 1997.

[9] Carl Pomerance. A tale of two sieves. *Notices Amer. Math. Soc*, 43:1473–1485, 1996.

[10] J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer New York, 2010.

[11] Peter Stevenhagen. The number field sieve. 2004.

[12] Peter Stevenhagen. Number rings. 2012.

[13] Nicolas Thériault. Index calculus attack for hyperelliptic curves of small genus, 2003.

[14] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography, Second Edition*. Chapman & Hall/CRC, 2 edition, 2008.