Getting Rid of Linear Algebra in Number Theory Problems

Paul Kirchner and Pierre-Alain Fouque

IRISA/CNRS, Université Rennes 1, France,
{paul.kirchner,pierre-alain.fouque}@irisa.fr

Abstract. We revisit some well-known cryptographic problems in a black box modular ring model of computation. This model allows us to compute with black box access to the ring $\mathbb{Z}/m\mathbb{Z}$. We develop new generic ring algorithms to recover m even if it is unknown. At the end, Maurer's generic algorithm allows to recover an element from its black box representation. However, we avoid Maurer's idealized model with CDH oracle for the multiplication in the hidden ring by introducing a new representation compatible with ring operations. An element is encoded by its action over the factor basis. Consequently, we can multiply two elements with classical descent computations in sieving algorithms. As the algorithms we propose work without using an expensive linear algebra computation at the end, even though they manipulate large sparse matrices, we can exploit a high-level of parallelism.

Then, we consider general groups such as imaginary quadratic class group and the Jacobian of a hyperelliptic curve, and obtain new methods for group order computation. The repeated squaring problem and the adaptive root problem used in the construction of Verifiable Delay Functions are particularly easy to solve in the black box modular ring, the high degree of parallelism provided by our method allows a reduction in the time to solve them. We improve the smoothing time, and as a result, we break Verifiable Delay Functions and factorize weak keys with lower Area-Time cost.
Finally, we show new AT costs for computing a discrete logarithm over an adversarial basis in finite fields.

1 Introduction

Factorization and Discrete Logarithm (DL) are the two most fascinating important cryptographic problems since the security of the RSA [74] and Diffie-Hellman [25] cryptosystems are related to them. The overall structure of the best known algorithms to solve them was already described in 1922 by Kraichik in his pioneering work [48]: the first step consists in finding a large collection of relations and the second one performs linear algebra computations on a very large sparse matrix. While the first step can be fully parallelized as epitomized by the title of the paper "Factoring by Electronic Mail" [52], the second one is less easy to parallelize [87]. We introduce a new technique to avoid the expensive linear algebra computational stage. The interest in finding parallel algorithms for number theory problems has recently been put in front of the stage by the tremendous research activities in Verifiable Delay Functions. They also introduced new number theoretic problems. Cryptosystems can also be based on various groups such as the group of points of an elliptic curve [44,63] by Koblitz and Miller or the Jacobian of a hyperelliptic curve [45]. Buchman and Williams have also proposed to use the class group of an imaginary quadratic orders [13].

Verifiable Delay Functions (VDF). VDF have been first proposed in [10] by Boneh *et al.*, after a preliminary work by Lenstra and Wesolowski [55]. In 2019, two elegant VDFs have been constructed by Wesolowski in [89] and Pietrzak in [69]. This research area is highly supported by the VDF Alliance as these functions have numerous applications for blockchains and more generally decentralized setting such as randomness beacons. Their construction poses new challenges for the cryptographic community since VDF should be computable in a prescribed amount of time *but not faster* and should be easy to verify. Both VDFs use the hard problem of computing an exponentiation by 2^T in a group of unknown order, a.k.a. *repeated squarings*, in which the naive algorithm has to perform at least T *sequential* squaring operations. It is conjectured that there is no parallelized algorithm that could speed up this computation. This problem has been first proposed by Rivest, Shamir and Wagner in 1996 [75] to construct a time-lock puzzle. These VDFs are non-interactive succinct proof arguments for the computation of an iterated squaring in a group of unknown order, whose security is based on the so-called *adaptive root* and *low order element* problems. An advanced application of these techniques is the Supersonic zk-SNARG scheme [15]. It is believed that the assumptions hold in the group $(\mathbb{Z}/n\mathbb{Z})^\times$ (or the group of squares for n a product of 2 safe primes), but a trusted setup is needed. There are various ways to obtain a trustless setup. The first one is to use a random n, but much larger; the class group of an imaginary quadratic number field [89], and the Jacobian of hyperelliptic curves were also proposed [26].

Several recent works have investigated the security of these assumptions based on idealized settings such as generic group model or the less idealized, algebraic group model. For instance, Rotem and Segev [76] have shown that speeding up repeating squaring is equivalent to factoring in the generic model. However, it is well-known that the security of discrete logarithm schemes is not based on the seminal generic lower bound given by Shoup [81], and the best factoring algorithms are not generic. Our initial goal is to evaluate the security of these assumptions in a more realistic adversarial model.

Cryptographic problems. The repeated squaring problem consists in computing $y = x^{2^T}$ given a random element x where T is a large value typically 2^{30} as proposed in [11] in time less than T squarings. The adaptive root problem in group $\mathbb{G}$ asks the adversary $\mathcal{A}$ to win the following security game. First, $\mathcal{A}$ outputs an element $w \in \mathbb{G}$. Then, the challenger randomly chooses a prime ℓ in a large set of primes $Primes(\lambda)$ depending on the security parameter λ and $\mathcal{A}(\ell)$ must output $w^{1/\ell} \in \mathbb{G}$. We use additive notation for the group embedded in a ring, so we call them repeated doubling and adaptive division in the following.

We also study more classical problems as computing the order of the group $(\mathbb{Z}/n\mathbb{Z})^\times$ for an RSA modulus (equivalent to the factorization problem), computing the order of the class group in an imaginary quadratic number fields, and computing the order of the Jacobian of a hyperelliptic curve when the genus g is large. We look at computing a variant of the discrete logarithm problem in these groups:

Adversarial-basis DL Problem (ABDL). *Given $\mathcal{B} = \{g_1, \ldots, g_n\}$ chosen by the adversary $\mathcal{A}$ in $\mathbb{G}$, the challenger returns $\{xg_1, \ldots, xg_n\}$ to $\mathcal{A}$, who must compute x.*

This problem is close to the Static Diffie-Hellman (SDH) problem studied in [12,19] where the queries to the DH oracle are adaptive. Since the final goal is to compute a discrete logarithm using DH queries, we called it *adversarial-basis DL* problem. Boldyreva in [9] proposed a variant of Static DH and if we can solve our problem, we can solve her problem. Freeman also proposed another variant [30] that is easier than ABDL. While solving Static DH breaks the security of the scheme, ABDL corresponds to a key recovery. To understand the differences between these DH problems, we refer the reader to [46] by Koblitz and Menezes. We emphasize that Static DH can be solved much faster than ABDL [35,38].

Related work. Many algorithms for solving the factorization problem have been proposed since Fermat, followed by Euler and Legendre. Kraichik developed more recent modern techniques that were exploited. Pomerance was then able to find parameters in Kraichik method to define the quadratic sieve [73]. Then, Pollard used number field techniques, introduced for solving the discrete logarithm problem by Coppersmith *et al.* [22] to factorize special numbers. The development of the *Generalized Number Field Sieve* was accomplished by many people as it is stated in [14]. The running time was in $L_n[1/3, 1.923]$. The best constant was achieved by Coppersmith by using multiple number fields in [20] in $L_n[1/3, 1.902]$. Bernstein [7] proposed a circuit for factoring and a new analysis in the AT model to better model hardware cost, extended in [8], where the AT name has been coined. Several devices for speeding up the sieving phase have been proposed such as Twinkle in [53]. To factor specific RSA moduli, H. Lenstra [56] extended Pollard's $p-1$-method [71] and Williams' $p+1$-technique [91] for factorizing numbers where $p-1$ or $p+1$ are smooth for a prime divisor p. The number that needs to be smooth is now the order of an elliptic curve modulo p, *i.e.* in the range $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$, leading to a $L_p[1/2, \sqrt{2}+o(1)]$ complexity.

For the discrete logarithm problem in $\mathbb{F}_Q$, where Q is a prime power, index-calculus techniques were also developed by Kraichik since 1922. Coppersmith *et al.* technique, a.k.a. the Gaussian integer algorithm was the fastest algorithm [22]. In [78], Schirokauer developed a simple $L_Q[1/3, 1.923 + o(1)]$ algorithm. Matyukhin adapted the Multiple Number Field Sieve to the discrete logarithm case [58]. Then, Joux and Lercier made several improvements to the GNFS algorithm for solving the discrete logarithm problem in [37] . When the characteristic is in $L_Q[1/3, \omega(1)]$, the best algorithm is due to Pierrot in [68] with a $L_Q[1/3, 2.156]$ running time. In table 2, we adapt the complexities in the AT cost model after optimizing the parameters.

For hyperelliptic curves, the best algorithm is by Gaudry *et al.* [33] in $q^{2-2/g+o(1)}$ for small genus g, following the general algorithm of Adleman *et al.* [1], which is subexponential when the genus is large. McCurley published a subexponential algorithm for computing an imaginary class group [62] in 1987; Kirchner slightly improved the complexity to $L_{|\Delta|}(1 + o(1))$ [42].

Sutherland proposed an algorithm for computing the group order [83], similar to Pollard's $p - 1$ factoring algorithm. For a subexponentionally small fraction of cyclic groups, the algorithm takes subexponential time. This is a problem if we want a trustless setup, as the party selecting the group could run this attack on a subexponential number of groups, and hope that no one would run it on the chosen group. A workaround for this difficulty could be to select the group using a randomness beacon [55].

Techniques. Black box models have been introduced in cryptography to study the equivalence problems between the RSA and Factorization problems and the DH and Discrete Logarithm problems. In these restricted computational models, the adversary cannot exploit the whole bit representation of the group elements. For instance, we can assume that the element a in a cyclic group $\mathbb{G}$ is encoded as ag and we are not able to get the discrete logarithm a. The adversary can only manipulate the encodings with some prescribed operations such as addition of the group elements, test equality to 0 and 1. After den Boer in [24], Maurer studied in [60] the relation between the Discrete Logarithm problem and the computational DH problem. He proved an outstanding algorithm solving the Discrete Logarithm problem given a Computational DH oracle in a black box group. The method uses a CDH oracle to implement the multiplication in the black box ring. In this model, it is possible to perform straight-line programs on the elements with test equality. Then, he finds an elliptic curve with smooth order, so that one can recover an element via a discrete logarithm computation *on the elliptic curve*. However, his algorithm only works when the group order is known.

We show how to compute the order at a speed which is almost optimal. Our fast algorithm consists in using the CM method to find an elliptic curve with a smooth order, and a known endomorphism ring. Then, solving a norm equation in the imaginary quadratic field leads to the field cardinal.

Repeated doubling can be computed by binary exponentiation. The adaptive division is solved by generating an integer divisible by all small primes, which is done with elliptic curves, just as in elliptic curve factoring method. Overall, all these problems can be solved much faster than a group order computation.

Computational model. Most of our results are in the AT model which allows taking into account parallelism, and some degree of communication cost. The circuit is decomposed as a square circuit of cells and each cell can only exchange data with its adjacent neighbors. Since it is reminiscent to circuit cost evaluation, we call it AT model following [8], and we try to minimize the product Area $\times$ Time. We emphasize that the AT model is designed to model application-specific integrated circuits rather than common hardware. For example, multiplying two matrices has the same AT cost as sorting their entries [86].

The two most time-consuming steps of the NFS algorithm are the sieving and the matrix step. The matrix step identifies linear dependencies between the entries of a sparse large matrix. Many hardware architectures have been proposed in the past to carry out the matrix step, a.k.a. mesh sorting [7] and mesh routing by Lenstra *et al.* [54]. Such devices are believed to provide faster circuit for factorization. Such machines can factor with an AT product (AT cost) of $L[1/3, 1.976 + o(1)]$, while if we consider the best algorithm for factoring [20] due to Coppersmith in the *standard* model, the number of operations is $L[1/3, 1.901 + o(1)]$.

Table 1: Our results, the group order is e and $\ell = \mathrm{gpf}(e)$. $L_x(\alpha)$ is a shorthand for $L_x[1/2, \alpha]$ and *only* the factoring is in the AT cost. Adaptive division takes a negligible number of ring operations only for the current choice. Group order is harder than finding factors. An exponent of $1 + o(1)$ is implicit everywhere. When there is a $\times$, the complexity is the product of the two values.

Groups / Problem solved	Ring operations	$(\mathbb{Z}/n\mathbb{Z})^\times$ AT cost	$\mathrm{Cl}(\mathbb{Q}[\sqrt{\Delta}])$	Hyperelliptic curve on $\mathbb{F}_q$ of genus g	
				small g	large g
Previous work (with limited parallelism)	-	$L_n[1/3, 1.976]$	$L_{\lvert\Delta\rvert}(1)$	$q^{2-2/g}$	$L_e(\sqrt{2})$
Repeated doubling sec. 3.6 Adaptive division sec. 3.7	small	$L_n[1/3, 1.928]$	$L_{\lvert\Delta\rvert}(1)$	q	$L_e(\sqrt{2})$
Group order sec. 3.4	$L_\ell(3/2)$	$L_\ell(\sqrt{2})\times$ $L_n[1/3, 1.928]$	$L_\ell(\sqrt{2})\times$ $L_{\lvert\Delta\rvert}(1)$	q	$L_\ell(\sqrt{2})\times$ $L_e(\sqrt{2})$
adversarial-basis DL sec. 3.5 known order, worst case	$L_\ell(1/2)$	$L_\ell(1/2)\times$ $L_n[1/3, 1.928]$	$L_{\lvert\Delta\rvert}(1.12)$	q	$L_e(1.574)$

Contributions. We extend Maurer black box group model to ring by showing how to implement in practice multiplication with a descent algorithm on a factor basis, and develop new algorithms for classical cryptanalysis problems such as factorization and discrete logarithm without requiring linear algebra.

Our approach is to first use a black box modular ring model for $\mathbb{Z}/m\mathbb{Z}$ for group of known and unknown order. We show algorithms in this computational model to perform basic operations such as computing the group order and recovering the elements using Maurer's algorithm. We develop new efficient parallelized algorithms to solve repeated doubling and adaptive division problems in this model. The resulting algorithms are mostly embarrassingly parallel.

In a second part, we instantiate this algorithm on various groups. It follows that we can compute the group order of a class group and of the Jacobian of a hyperelliptic curve. In the low genus case, we significantly reduce the number of operations needed, from $q^{2-2/g+o(1)}$ to $q^{1+o(1)}$ *in the standard model*. For the others, our algorithms may be slower in the standard model, however they are far more parallel; we can reach a quasi-polynomial circuit depth with only a negligible overhead, and it has lower AT cost.

Then, we study special number fields, which correspond to the factoring problem and discrete logarithm computation. We improve the smoothing time to a constant of 1.004 in the standard model in heuristic claim 7, and 1.08 in AT cost. This is low enough to decrease the complexity in RSA groups and $\mathbb{F}_p$ in the AT model. In particular, we show how to factor an RSA modulus if the greatest factor of $p-1$ is less than ℓ with an AT cost of $L_\ell[1/2, \sqrt{2}+o(1)]L_n[1/3, 1.92757]$. This is an improvement for a $L_n[1/3, -\epsilon]$ proportion of RSA moduli. The algorithm uses the elliptic-curve factorization method in the $p-1$ algorithm, which is enabled by a GNFS-like algorithm.

We then look at the static discrete logarithm in an extension field $\mathbb{F}_Q$, as a large subfield accelerates the smoothing time. It follows that in the AT model, when there is a large subfield, we can solve ABDL problem faster than the discrete logarithm problem the results are summarized in table 2. Our faster smoothing allows faster practical attacks on Diffie-Hellman cryptosystem, see [2]. Outside of low genus hyperelliptic curves, we expect our algorithms to have no practical impact.

Table 2: Complexity in the AT model of the Discrete Logarithm variants in $\mathbb{F}_{p^d}$ and $Q = p^d$, *only when the degree is even*, with the best characteristic χ, and only in a subgroup of size $L_Q[2/3, o(1)]$.

Characteristic / Problem	$L_Q[1/3, \omega(1)]$	$L_Q[2/3, \chi + o(1)]$	$L_Q[2/3, \omega(1)]$
Discrete Logarithm	$L_Q[1/3, 2.2621]$ [6]	$L_Q[1/3, 1.7954]$ [6]	$L_Q[1/3, 1.9761]$ [78]
ABDL sec. 4.2	$L_Q[1/3, 2.2013]$	$L_Q[1/3, 1.7472]$	$L_Q[1/3, 1.9230]$

2 Background

Definition 1. *(Smoothness). Let B be a positive integer. An integer, (resp. polynomial, ideal in a number field) is said to be B-smooth if all its prime power divisors (resp. the degrees of all its irreducible , prime ideals whose norms) are bounded by B.*

Subexponential complexities. For α and r are real numbers with $0 \leqslant r \leqslant 1$, the complexities will be expressed using the $L_n[r;\alpha]$ function equals to

$$\exp\bigl(\alpha(\ln n)^r(\ln\ln n)^{1-r}\bigr), \quad \text{for} \quad n \to \infty,$$

with natural logarithms. Thus, we get $L_n[r;\alpha] + L_n[r;\beta] = L_n[r;\max(\alpha,\beta)]$, $L_n[r;\alpha]L_n[r;\beta] = L_n[r;\alpha+\beta]$, $L_n[r;\alpha]L_n[s;\beta] = L_n[r;\alpha]$ if $r > s$, $L_n[r;\alpha]^k = L_n[r;k\alpha]$ and if $\alpha > 0$, $(\log n)^k L_n[r;\alpha] = L_n[r;\alpha+o(1)]$ for any fixed k so that we can neglect log factors. The following theorem gives an estimation of the probability that an integer $\leqslant x$ is y-smooth in term of the complexity function. It is implied by the Canfield-Erdős-Pomerance theorem [16].

Theorem 1. *Let $0 < s < r \leqslant 1$ and $0 \leqslant \alpha, \beta$. If $x = L_n[r,\alpha]$ and $y = L_n[s,\beta]$,*

$$\frac{\psi(x,y)}{x} = L_n[r-s, -\frac{\alpha}{\beta}(r-s) + o(1)],$$

where $\psi(x,y)$ is the number of integers smaller or equal to x which are y-smooth.

Also, for any constant $k > 0$, and x large enough,

$$\psi(x, \log(x)^k)/x > x^{-k-o(1)}.$$

For example, a uniform integer less than $\leqslant L_n[2/3,\alpha]$ is $L_n[1/3,\beta]$-smooth with probability $L_n[1/3, -\alpha/(3\beta)]$. We abbreviate $L_n[1/2,\alpha]$ to $L_n(\alpha)$.

For the smoothness of polynomials, it is known that ordering polynomials by increasing degree leads to the same result, as long as the smoothness degree is not logarithmic [66]. The probability for a uniform degree d polynomial in $\mathbb{F}_q[x]$ to be 1-smooth, where the factors are in a set of ϵq monic polynomials is larger than $\epsilon^d/d!$.

Complexity analysis. There is essentially only one calculation behind the complexity statements which follow. It consists in remarking that the minimum of $ax + \frac{b}{x}$ is in $x = \sqrt{b/a}$ and equal to $2\sqrt{ab}$. For example, if we have to perform $L_\ell(ax + o(1))$ operations at each iteration, and the algorithm succeeds in one iteration if $2b$ integers sampled uniformly below ℓ are $L_\ell(x + o(1))$-smooth, the total number of operations is

$$L_\ell\left(ax + \frac{b}{x} + o(1)\right) = L_\ell\left(2\sqrt{ab} + o(1)\right).$$

Replacing the addition by a maximum simply removes the factor 2.

Smoothing and descent steps. Nearly all sieving-based algorithms for factorization and discrete logarithm computation rely on a factor basis containing small elements, where the definition of small depends on the case. Let us focus in the case of computing the discrete logarithm is a prime field. Since the probability that a target element $h = g^x \bmod p$ is B_0 smooth is low, individual discrete logarithm computation runs into two steps: the *smoothing step* and the *descent step*. The first step is iterative and relies on testing the smoothness of randomly generated elements $g^t h$ where t is a random and known exponent until a B_1-smooth decomposition is found. This step is usually performed using the Elliptic Curve Method (ECM). The second step starts a recursive process for each element less than B_1 and greater than B_0 found after the first step. For each of these medium-sized elements, we compute a complete decomposition over the factor basis. Each element is the root of a descent tree. We get a relation involving the original element and others whose norm or degree depending on the case, is strictly smaller than that of the root element. These smaller elements become the leaves of the descent tree and as soon as all the leaves are in the factor basis, the discrete logarithm can be computed by a tree traversal. Several optimizations can be added to this principled algorithm. In the first step, we can start to represent elements $g^t h$ as u/v with u, v of half size.

Elliptic curves. Elliptic curves are central in our algorithms. In characteristic larger than 3, a projective elliptic curve E can be put in the projective short Weierstrass form $y^2 z = x^3 + axz^2 + bz^3$. Points on this curve form a group, additions can be computed as polynomials of coordinates. Given a point P on the curve, we can forget its ordinate y and obtain $\pm P$ on the Kummer surface. When we talk about the abscissa of a point, we have implicitly $z \neq 0$, for example $z = 1$. The Frobenius endomorphism over $\mathbb{F}_q$ of an ordinary elliptic curve can be embedded as τ in $\mathbb{Q}[\sqrt{\Delta}]$ where $\Delta < 0$. It follows that the group order is $(\tau - 1)(\overline{\tau} - 1)$, while the field order is $q = \tau\overline{\tau}$. The group order can also be written as $q + 1 - \tau - \overline{\tau}$. The Hasse theorem follows from these properties: $|\tau + \overline{\tau}| \leqslant 2\sqrt{q}$.

If d is not a square, we can consider the elliptic curve $E^d : dy^2 z = x^3 + axz^2 + bz^3$. It is *the* quadratic twist of E, and its group order is $q + 1 + \tau + \overline{\tau}$.

A constant proportion of curves can be put in Montgomery form $by^2 z = x^3 + ax^2 z + xz^3$, where b controls the twist [65]. They have the property of fast doubling, and differential addition which leads to a fast scalar multiplication. Furthermore, we can quickly multiply points $\pm P$ on the Kummer surface, without using the coefficient b.

Using modular forms, it is possible to create an elliptic curve from Δ. The method is fairly involved, and the reader is referred to the papers by Schoof, Atkin and Morain [79,4].

One of the main applications of elliptic curves is factorization [56]. Indeed, if a curve E has a B-smooth order modulo p, a factor of n, then with P a non-trivial point of the curve, $B!P$ is equal to 0 modulo p. We thus compute $B!P$ modulo n, and the gcd of the z coordinate with n should give a non-trivial factor. Taking $L_p(\sqrt{1/2} + o(1))$ curves with $B \approx L_p(\sqrt{1/2})$ leads to a heuristic complexity of $L_p(\sqrt{2}+o(1))$ arithmetic operations. Lenstra, Pila and Pomerance proved a variant of this algorithm using hyperelliptic curves, but with a far higher expected running time [57].

3 Black Box Modular Ring Model

We introduce here the black box modular ring model, and study the complexity of solving various problems in it. It consists in black box access to the ring $\mathbb{Z}/m\mathbb{Z}$, which means that we can add, subtract, multiply, test equality and have access to representations of 0 and 1 while not knowing the number used and m. Our results culminate in the ability to find m, and recover an element from its black box representation.

3.1 Motivation

We first explain the interest of such a model. Finite cyclic groups $\mathbb{Z}/m\mathbb{Z}$ available only through a (mostly) black box are ubiquitous in cryptography and number theory. For example Diffie-Hellman used the units of a finite field $\mathbb{F}_q$, RSA uses $(\mathbb{Z}/n\mathbb{Z})^\times$. In this section, the group is denoted additively, even though some applications typically use a multiplicative notation. It is also possible to consider abelian varieties over finite fields for maximum efficiency, or class group of imaginary quadratic fields. While these abelian groups may not be cyclic, they are almost always of the form $H \times \mathbb{Z}/m\mathbb{Z}$ with H a group with a very small cardinal. It then follows that solving some problem in the abelian group can be efficiently reduced to solving it in the cyclic part, so we will focus on this case only.

While the group and black box aspects are clear, the multiplication is missing. Indeed, when group elements are of the form ag for a known generator g, multiplication is also known as the Computational Diffie-Hellman problem. This led to Maurer's reduction [60] showing how to compute a discrete logarithm using a CDH oracle. This result is our last step, recovering an element, and we recall the algorithm which uses an elliptic curve with smooth order in section 3.5.

Cryptanalysis usually provides a descent algorithm on the given group. It selects generators (g_i) of the group, called a *factor base*, and a descent algorithm, given an element a of the group to find in time T an integer vector x with $a = \sum_i x_i g_i$. Further, the vector x is always short and sparse, so we will assume that only $B^{o(1)}$ entries are non-zero, and they are bounded by $B^{o(1)}$.

Our model then represents an integer a as $(ag_i)_i$, so that group operations have a complexity of B. Then, we can compute ax for any x in the group, simply by using a descent on $x = \sum_i x_i g_i$, so that

$$ax = \sum_i x_i(ag_i).$$

Multiplication of two representations simply consists in repeating this B times. Once the descents are computed, we just need to sort elements; this can be done with AT cost $B^{1/2+o(1)}$ [86].

An important point is that the complexity does not depend on the size of a or b as integers, and indeed it will be exponentially large in our algorithms. Minimizing the number of ring operations is much more important than minimizing additions, or operations outside of the ring. Using double-and-add, we can create an integer with l bits in $2l$ additions.

For practical purpose, we can use another representation, with a multiplication matrix M_a where $ag_i = \sum_j (M_a)_{i,j} g_j$, and M_a is sparse and small. Multiplication matrices can be added and multiplied, and descents are only needed to make them sparser, so that we can slightly decrease their number. However, at least in the worst case, additions may cost much more. This is analogous to, and inspired by a Fully Homomorphic Encryption construction [34].

We will discuss the specificities of each group, and of the descent in the next section.

3.2 Lower bound for computing the order in the black box model

We show that extracting square roots in a black box ring model of $\mathbb{Z}/p\mathbb{Z}$ is harder than finding a factor of a multiple n of p. Indeed, we can use $\mathbb{Z}/n\mathbb{Z}$ as a ring model of $\mathbb{Z}/p\mathbb{Z}$ where equality tests compute the gcd with n. in our computational model. The reduction between the square root computation and the factorization of n is classical and consists in sampling x uniformly in $(\mathbb{Z}/n\mathbb{Z})^\times$ and tests whether $\sqrt{x}^2$ is equal to $\pm x$. If n has at least two factors, with probability at least $1/2$, we can recover a non-trivial square root which allows us to factor n.

As the most efficient algorithm for factoring a multiple of p is Lenstra's Elliptic Curve Method [56] with complexity $L_p(\sqrt{2}+o(1))$, this complexity bound gives a lower bound on the complexity of solving the group order problem in our model. Moreover, this also shows that we can use the ECM algorithm for solving our problems without impacting too much their overall complexities.

In sections 3.3 and 3.4, we develop algorithms to recover the ring order in the black box computational ring model.

3.3 Finding an almost multiple of the ring order

In this subsection, we show how to reduce the general problem of finding the order m of the ring to the problem of finding the order of $\mathbb{Z}/\ell\mathbb{Z}$ where ℓ is an unknown prime. For doing this, it is enough to find an integer n such that $\ell \nmid n$ and $\frac{m}{\ell} | n$, or an "almost multiple" of m. Indeed, as $n(\mathbb{Z}/m\mathbb{Z}) = \mathbb{Z}/\ell\mathbb{Z}$, multiplication by n reduces our problem to $\mathbb{Z}/\ell\mathbb{Z}$. Once ℓ is known, solutions can be recovered with a multiplication by $n^{-1} \bmod \ell$.

The integer n is built using a variant of Lenstra's Elliptic Curve Method. We pick random x_0, y_0 and a, compute $b = y_0^2 - x_0^3 - ax_0$ and work with the projective elliptic curve $y^2z = x^3 + axz^2 + bz^3$ to avoid divisions. The value b is chosen so that $P = (x_0 : y_0 : 1)$ is on the curve. We now define n_i as the z coordinate of $i!P$, which can be computed with $\mathrm{O}(i \log i)$ ring operations. Our candidates for n are the products $N_j = \prod_{i<j} n_i$. Indeed, the probability that

$\ell | n_i$ is roughly the probability that $i!$ divides the order of the i-th curve modulo ℓ. Assuming this order is uniform, the theorem 1 suggests this probability is $\log_i(\ell)^{-(1+o(1))\log_i(\ell)}$. We thus expect the first i where $\ell | n_i$ to be $L_\ell(\sqrt{1/2}+o(1))$, for a total of $L_\ell(\sqrt{2}+o(1))$ computations.

Then, we compute the smallest $N_j = 0 \bmod m$ and N_{j-1} is most likely a candidate. First, with $q = \mathrm{gpf}(m)$, the greatest prime factor of m, we have $j \in L_q\left(\sqrt{1/2}+o(1)\right)\log\log m$ with overwhelming probability, as there are at most $\log m$ prime factors. Most likely $\frac{m}{\gcd(m,N_{j-1})}$ has few prime factors, and in most cases one, that is N_{j-1} is an almost multiple of m. If this is not the case, we resample the ith curve and also the point, and continue. The number of ring operations is $L_\ell(\sqrt{2}+o(1))$.

If N_{j-1} is an almost multiple, we can find ℓ using the algorithm of the next subsection. Repeating with $\ell(\mathbb{Z}/m\mathbb{Z})$, we can find all prime factors of m with their multiplicities, and therefore m.

Heuristic Claim 1 *We can find a multiple, and an almost multiple of the order m of a black box modular ring with $L_{\mathrm{gpf}(m)}(\sqrt{2}+o(1))$ ring operations.*

3.4 Finding the prime group order

We assume here that we are working over $\mathbb{Z}/\ell\mathbb{Z}$, with $\ell > 1000$. Our goal is to find the hidden prime order ℓ. We will describe an easy but slow algorithm for this task, and then a faster, but significantly more complicated one.

Given an elliptic curve and a point P on it, we first show how to find the order of the point when it divides $F = \prod_i p_i^{e_i}$. We compute $\prod_{i \leqslant j} p_i^{e_i} P$ for successive j until this is equal to zero. Next, we compute $p_j^e \prod_{i<j} p_i^{e_i} P$ for successive e until this is equal to zero. The order of P is then the order of $p_j^e P$, which is computed recursively, multiplied by p_j^e. The complexity is then $\mathrm{O}(\log F)$ per prime factor of the order.

The method for finding ℓ consists in remarking that $\ell + 1 = \frac{|E|+|E^d|}{2}$ where E^d is the unique quadratic twist of E, of equation $dy^2z = x^3 + axz^2 + bz^3$ (d is not a quadratic residue and $|E|$ is the order of the elliptic curve E). Then, we sample E randomly until we find that the exponent of both E and E^d are different, and smooth. The easiest way to compute on E^d is to use Montgomery curves ($dy^2z = x^3 + ax^2z + bz^3$), as arithmetic on the curve depends only on the abscissa, and not on the ordinate or the twist parameter d [65].

It is in general true for short Weierstrass curves that the abscissa of kP does not depend on the ordinate of $P = (x_0 : y_0 : 1)$ [63].

It is known that the exponent is often close to the order of the group [82, Section 4], so that guessing the curve orders is easy.

With $\log F = {}_\ell(\sqrt{2}/2 + o(1))$, we can compute the exponent of E with probability $L_\ell(\sqrt{2}/2 + o(1))$. When it is computed, we use $\log F = {}_\ell(\sqrt{2} + o(1))$ on E^d, which is computed with probability $L_\ell(\sqrt{2}/4 + o(1))$.

Heuristic Claim 2 *We can find the prime order ℓ of a black box modular ring with $L_\ell(\frac{5\sqrt{2}}{4} + o(1))$ ring operations.*

We now show how to decrease this to $L_\ell(\sqrt{2}+o(1))$ *multiplications*. Before explaining this faster technique to compute the prime order group, we will present other algorithms that we need to present this technique. The first black box algorithm shows that if we have an elliptic curve with smooth order over $\mathbb{F}_\ell$, we can recover it efficiently if we know its endomorphism ring in time $L_\ell(o(1))$. The second algorithm shows that we can compute a root of a polynomial in the black box ring model. The third one uses this to build elliptic curves with the CM method. This section can be skipped at first read.

Elliptic curves order with known Frobenius discriminant. Suppose we have an ordinary elliptic curve with a known order N, and an endomorphism ring with known Frobenius discriminant Δ. Then, we have $\ell = \tau\overline{\tau}$ and $N = (\tau-1)(\overline{\tau}-1)$ where $\tau \in \mathbb{Q}[\sqrt{\Delta}]$ is the Frobenius endomorphism.

The algorithm starts by factoring $N = \prod_i p_i^{e_i}$. Then, we factor the ideal $(p_i) = \mathfrak{p}_i\overline{\mathfrak{p}}_i$ in the number field $\mathbb{Q}[\sqrt{\Delta}]$. The factorization of the ideal $(\tau-1)$ is for some vector x, $\prod_i \mathfrak{p}_i^{x_i}\overline{\mathfrak{p}}_i^{e_i-x_i}$. We then check for all x with $0 \leqslant x_i \leqslant e_i$ if this ideal is principal. In an imaginary quadratic field, a generator is a shortest vector of the ideal, so that it is polynomial time. The generators are given by the product between a generator and all units, and there are at most 6 units in an imaginary quadratic field. At some point, a generator will be $\tau-1$. We deduce a candidate $\ell = \tau\overline{\tau}$ and check if $\ell = 0$ in the ring.

Alladi found the average of $\sum e_i$ (if N were a uniform smooth number) [3], which leads to a

$$\mathrm{O}\left(\sqrt{\frac{\log \ell}{\log\log \ell}}\right)$$

bound. The inegality of arithmetic and geometric means leads to a bound of

$$2^{\mathrm{O}\left(\sqrt{\frac{\log \ell}{\log\log \ell}}\right)} = L_\ell(o(1))$$

on the number of x candidates, and the complexity of the recovery.

In our case, $-\Delta$ is sufficiently small for a class group computation to be run, which turns the problem into a “random” knapsack problem with density $2^{-\mathrm{O}(\log\log l)} = \log^{-\mathrm{O}(1)} \ell$. The knapsack problem can in turn be solved using lattice reduction [49] or other means [43].

We will now explain how to use the CM method [79] to generate $L_\ell(\sqrt{1/2}+o(1))$ curves with a known Frobenius discriminant Δ.

Root finding. We consider a polynomial $P(x)$ of degree d with coefficients in the black box ring, with d distinct roots. We show how to efficiently find a linear factor of $P(x)$. First, Cantor and Kaltofen found an algorithm which multiplies polynomials in a black box ring efficiently [18]. Second, it can be used to obtain a fast gcd algorithm, for a complexity of $\mathrm{O}\left(d\log^2 d\log\log d\right)$ in degree d [64]. We can do this without division if we drop the condition that the gcd is monic.

We precompute a Montgomery elliptic curve E, with a smooth exponent. We assume that it is ordinary. A putative exponent $e \leqslant 2\ell$ is then computed. For factoring, we start by sampling $s(x) \in \mathbb{Z}[x]/(P(x))$ with uniform coefficients between 0 and $100de^2$. As $e \geqslant \sqrt{\ell/2}$, $s(x)$ is almost uniform in $(\mathbb{Z}/\ell\mathbb{Z})[x]/(P(x))$. We define Q as the point with abscissa $s(x)$ in the ring $(\mathbb{Z}/\ell\mathbb{Z})[x]/(P(x))$. Then, we compute $f(x)/g(x)$ the abscissa of eQ. If the degree of $\gcd(g(x), P(x))$ is 0 or d, we restart. Otherwise, we continue with $\gcd(g(x), P(x))$ or a generator of $(P(x)/\gcd(g, P(x)))$, whichever has the lowest degree.

Let r be a root of $P(x)$. Then r is a root of $g(x)$ iff the point(s) with abscissa $s(x) \bmod (x - r)$ are on a curve with order dividing e. We now prove that it is unlikely if the points with abscissa $s(x) \bmod (x - r)$ are on the quadratic twist of E.

Let t be such that the order of the curve is $\ell + 1 - t$, so that the quadratic twist has order $\ell + 1 + t$. Mestre [79, Theorem 3.1] proved there is a divisor $g > 4\sqrt{p}$ of either $\ell + 1 - t$ or $\ell + 1 + t$. Hasse theorem indicates that $|t| \leqslant 2\sqrt{p}$, so that exactly one of the two orders is divisible by g and therefore

$$\gcd(e, \ell + 1 + t) = \gcd(\ell + 1 - t, \ell + 1 + t) \leqslant \frac{\ell + 1 + t}{2}.$$

This implies that for at least $\frac{\ell-1+t}{4} \geqslant \frac{\ell-1-2\sqrt{\ell}}{4} \geqslant \frac{\ell}{5}$ different abscissae, the corresponding point is not killed by e.

The statistical distance between $s(x)$ and uniform is bounded by $\frac{e^2}{40\ell} \geqslant \frac{1}{30}$. For the algorithm to restart, $d-1$ roots must behave in the same way as the last one. Hence, the probability that $\gcd(g(x), P(x))$ has degree 0 or d is less than

$$\max\left(1 - \frac{1}{5}, \frac{\ell + 1 - t}{2\ell}\right)^{d-1} + \frac{1}{30} \leqslant \frac{5}{6}.$$

If this is not the case, the degree of $P(x)$ is divided by at least 2, and the expected complexity is

$$\mathrm{O}(d \log(\ell d) \log d \log\log d).$$

We can also detect if $P(x)$ has no roots. It is very unlikely that E (or the quadratic twist) has a smooth number of points over a strict extension of $\mathbb{F}_\ell$, so that a restart is almost guaranteed. Stopping after $-10\log(\epsilon)$ restarts leads to a failure probability of ϵ.

CM method. We shall use $-\varDelta \leqslant D = L_\ell(\sqrt{2} + o(1))$, but only when the class group of $\mathbb{K} = \mathbb{Q}[\sqrt{\varDelta}]$ is $L_D(1/2)$-smooth. This heuristically leaves $DL_D(-1/2 + o(1))$ candidates. For each subgroup G of the class group of $\mathbb{K}$ of prime power order, we compute

$$P_G(x) = \prod_{\mathfrak{a} \in G} (x - j(\mathfrak{a}))$$

where $\mathfrak{a}$ any ideal in the class, and $j(\mathfrak{a})$ is the j-invariant of the complex elliptic curve $\mathbb{C}/\mathfrak{a}$. CM theory dictates that $P_G(x)$ has integer coefficients, and $\mathbb{K}[x]/P_G(x)$ is an extension of $\mathbb{K}$ with Galois group G.

We use the previous algorithm to check that $P_G(x)$ has indeed a root, which is equivalent to having $|G|$ roots. This takes a total of $DL_D(1/2 + o(1))$ ring operations.

If D passes all checks, then Artin reciprocity law implies that ℓ is a product of two algebraic integers in $\mathbb{K}$, and the CM method will be able to build a curve. We expect the class of the factors of ℓ to be uniformly distributed in the class group, of cardinal $\mathrm{O}\big(\sqrt{-\Delta}\log(-\Delta)\big)$. This leads to an expectation of

$$\sqrt{DL_D(-1/2 + o(1))} = L_\ell(\sqrt{1/2} + o(1))$$

curves.

The CM method now computes $P_G(x)$ where G is the class group, extracts a root j of it, and builds (at least) two elliptic curves with this j-invariant. As P's coefficients have a bitsize of $|\Delta|^{1/2+o(1)}$ [28], we need $|\Delta|^{1+o(1)}$ additions to transform them into the black-box model. The total number of these operations is

$$D\sqrt{D}L_D(\mathrm{O}(1)) = L_\ell(\sqrt{9/2} + o(1)).$$

The factorization takes $L_\ell(\sqrt{1/2} + o(1))$ ring operations per curve and this can be decreased using the tower of Galois extensions [29,84]. Once the curve is built, we check for $L_\ell(\sqrt{1/2} + o(1))$-smoothness, for a total of

$$DL_D(1/2 + o(1)) + \sqrt{D}L_\ell(\sqrt{1/2} + o(1)) = L_\ell(\sqrt{2} + o(1))$$

ring operations.

If we want to minimize the total number of ring operations, rather than multiplications, we use $D = L_\ell(1 + o(1))$.

Heuristic Claim 3 *We can find the prime order ℓ of a black box modular ring with $L_\ell(\sqrt{9/2}+o(1))$ additions and $L_\ell(\sqrt{2}+o(1))$ multiplications. Alternatively, this can be done in $L_\ell(3/2 + o(1))$ operations.*

3.5 Recovery in prime order group

Given an element x in a black box ring $\mathbb{Z}/\ell\mathbb{Z}$ with known ℓ, we have to find x. This problem was studied by Maurer [60], we give a variant of his algorithm for completeness. The inverse of a is computed with $a^{\ell-1}$.

We first find a cyclic elliptic curve E (with known coefficients, i.e. not in the black box) with $L_\ell(1)$-smooth order and a generator g. This can be done in $L_\ell(1/2+o(1))$. We then sample uniform shift $s \in \mathbb{Z}/\ell\mathbb{Z}$ until $x+s$ is the abscissa of a point on our elliptic curve, which can be checked since we know ℓ. Then, we use Pohlig-Hellman's algorithm [70] to compute its discrete logarithm $\pm a$, with respect to g, in time $L_\ell(1/2 + o(1))$. Finally $x + s$ is recovered as the abscissa of $\pm ag$.

In case the ring operations are very expensive, we can try ℓ^ϵ curves to find a $\log^{1/\epsilon+o(1)}(\ell)$-smooth order, for a total of $\mathrm{O}\Big(\log^{1+1/(2\epsilon)+o(1)}\ell\Big)$ ring operations.

With Pohlig-Hellman's algorithm, this can be extended to any $\mathbb{Z}/m\mathbb{Z}$.

Heuristic Claim 4 *Given m, and an element a in the black box model $\mathbb{Z}/m\mathbb{Z}$, we can find a with $L_{\text{gpf}}(m)(1/2+o(1))$ operations.*

3.6 Repeated doubling

We have to compute, given x in the black box ring, $2^e x$. Without using the multiplication, we can compute this in e repeated doubling of x, a process usually called "binary exponentiation" when $\mathbb{Z}/m\mathbb{Z}$ is represented as a multiplicative group. Here, we can compute 2^e with fast binary exponentiation so that the total ring complexity is $(1+\frac{2}{\log\log e})\log e$.

In our application, x may not be fully represented as an element of the ring, i.e. we may not know all $(xg_i)_i$; nevertheless, a descent on x and the computation of 2^e allows to quickly compute the group element $2^e x$.

3.7 Adaptive division

The task of adaptive division consists in outputting a w, and then for a $q \leqslant Q$ prime uniformly sampled, we have to compute w/q. Without ring operations, we can take w as the product of all primes less than Q so that w/q can be computed, for a total of $\mathrm{O}(Q)$ operations.

We propose to take for w a N_i with the notations of section 3.3 with $i = L_Q(\sqrt{1/2}+o(1))$. Then, it is likely that q divides N_i as integers, and we then compute N_i/q.

Given a straight line program with ring operations computing the integer a [47,88], we can transform it into one computing $\lfloor a/q \rfloor$ with roughly the same size. Indeed, we proceed by induction where numbers are represented with the form $qa+r$. If we want to multiply, we have:

$$(qa_0+r_0)(qa_1+r_1) = q(qa_0a_1+r_0a_1+a_1r_0+\lfloor r_0r_1/q\rfloor)+(r_0r_1) \bmod q.$$

All other operations can be carry out in the same manner. For a small q, which is our case, this is efficient.

Heuristic Claim 5 *We can solve the Adaptive division problem in $L_Q(\sqrt{2}+o(1))$ operations.*

This is much less than Q as the security parameter proposed in [89] seems to suggest since Q is exponential in the security parameter, while we show here efficient algorithms in our black box model. In the next section, we will show how to perform the ring operations in various groups.

4 Applications

We emphasize here that previous index-calculus-like algorithm [48,14,78,1] use typically a Block Wiedemann algorithm [21] with a large dimension m on a sparse matrix. As such, the space needed is proportional to m and the block size

k, which limits the number of processors to around k. Also, $k \geqslant m^{1/\omega}$, the lattice reduction step in the Block Wiedemann algorithm has to be parallelized, which is difficult [85]. Thomé's algorithm needs a time bounded by $m^{1+o(1)}$ and an area of $(mk)^{1+o(1)}$ when the field elements are with $m^{o(1)}$ bits. As a result, even when the algorithms proposed use more operations, they are still of interest. This is especially important for Verifiable Delay functions where the parallel adversary has to compete with a single-thread user.

4.1 Picard groups, the general case

Class group of a quadratic number field. Let $K = \mathbb{Q}[\sqrt{\Delta}]$ be a quadratic number field. The group $\mathbb{G}$ is the class group, usually denoted multiplicatively. Multiplication of group elements corresponds to the multiplication of ideals, followed by a lattice reduction, which allows the representative to have a norm bounded by $\mathrm{O}\left(\sqrt{|\Delta|}\right)$. The factor base consists of all prime ideals of degree one smaller than $B = L_{|\Delta|}(1/2)$. Descent proceeds by randomizing the class, i.e. multiplying the input by a sparse random combination of factor base elements, followed by a smoothness test of the norm. Using ECM this takes negligible time, and we need $L_{|\Delta|}(1/2 + o(1))$ trials. The running time for a ring operation is therefore $L_{|\Delta|}(1 + o(1))$.

The class group cardinal is "random", which leads to a total running time of $L_{|\Delta|}(1 + o(1))$ for a vanishing fraction of quadratic number fields (when the group order is $|\Delta|^{o(1)}$ smooth).

We can use a smoothness bound of $\log^{\omega(1)} |\Delta|$ in ECM, while keeping a complexity of $L_{|\Delta|}(1/2 + o(1))$ for a descent. Thus, we have that for any depth $\log^{\omega(1)} |\Delta|$, there exists a circuit with $L_{|\Delta|}(1 + o(1))$ gates for a ring operation. We also obtain a time of $L_{|\Delta|}(1/3 + o(1))$ for a circuit with area $L_{|\Delta|}(2/3 + o(1))$.

With previous algorithms, we have an AT cost at $L_{|\Delta|}\left(\sqrt{\frac{25}{24}} + o(1)\right)$ when using a factor base with $L_{|\Delta|}\left(\sqrt{\frac{1}{6}} + o(1)\right)$ elements. The time taken is $L_{|\Delta|}\left(\frac{1}{2} + o(1)\right)$.

Jacobian of a hyperelliptic curve. Now we set $K = \mathbb{F}_q(x, y)/(y^2 + h(x)y - f(x))$ where the degree of h is at most g, and the degree of f is $2g + 1$, and the curve is non-singular. The class group of K is then the Jacobian of a hyperelliptic curve of genus g, and is our group $\mathbb{G}$. Multiplication is as above, we first multiply the ideals and then use lattice reduction (in a degree sense) to reduce the degree of the generators to at most g [17].

We take in our factor base $qL_{q^g}(\sqrt{1/2} + o(1))$ (or just q for small g) polynomials with minimal degree. For descent, we randomize as above and factor using any polynomial factoring algorithm until the factors are in the factor base. The number of trials is then less than $L_{q^g}(\sqrt{1/2} + o(1))$, which leads to a multiplication running time of

$$qL_{q^g}(\sqrt{2} + o(1)).$$

We conclude as above: for a vanishing fraction of hyperelliptic curves, we can solve ABDL and find the group order in time

$$qL_{q^g}(\sqrt{2}+o(1)).$$

(For small g, we have a factor base of size q, for a multiplication complexity of only $\mathrm{O}(qg!g\log q)$.) Further, for any hyperelliptic curve, we can compute its order with the second algorithm in time

$$qL_{q^g}(2\sqrt{2}+o(1))$$

and solve ABDL when the group order is known in time

$$qL_{q^g}(\sqrt{3/4}+\sqrt{1/2}+o(1)).$$

When g is small, and the order is known, ABDL is solved in time

$$qg!\log(q)\cdot\log^{1+g/2+o(1)}(q).$$

This is better than the previous time of $q^{2-2/g+o(1)}$ for a discrete logarithm given $|\mathbb{G}|$ by Gaudry *et al.* [33] for any $g \geqslant 3$.

For ng small, say bounded by $\log\log q$, we can solve the same problem over the finite field $\mathbb{F}_{q^n}$ with the same complexity [80,32]. Indeed, we can restrict the coefficients in our factor base to $\mathbb{F}_q$. The descent now has to solve a system of ng polynomial equations with ng variables, which can be done in time $q^{o(1)}$ and with success probability of $q^{-o(1)}$. Overall, we obtain a $q^{1+o(1)}$ running time.

In the same way, a circuit with depth $\log^{\mathrm{O}(1)}(q)$ and with $q^{1+o(1)}$ gates can compute a multiplication. For optimizing the AT cost of a ring multiplication, we restrict to the factor basis to $\lfloor q^{1-1/(2g+1)}\rfloor$ elements as Harley proposed [31], so that a descent is computed in time $q^{g/(2g+1)+o(1)}$. (We do not use the large prime(s) variants [33] as the memory needed appears to increase the cost.) It follows that with area $q^{1-1/(2g+1)+o(1)}$, a ring operation can be computed in time $q^{1/2-1/(4g+2)+o(1)} = q^{g/(2g+1)+o(1)}$, giving an AT cost of $q^{3g/(2g+1)+o(1)}$.

For optimizing the *discrete logarithm* AT cost, we use the double large prime variant of Gaudry *et al.* [33]. We put $\lfloor q^{1-r}\rfloor$ primes in our factor base. The first step is finding an extended base of $\lfloor q^{1-r/2}\rfloor$ primes which can be written with less than $g\log q$ factor base elements. The extended base size is doubled by finding $5q$ relations with $g-2$ factors in the factor base, one in the extended base and the last factor can be added to the extended base. The AT cost for generation is therefore $q^{1+(g-2)r+o(1)}$. A descent is computed by searching a relation with $g-2$ factors[1] in the base and 2 in the extended base. The AT cost for $q^{1-r+o(1)}$ descents is $q^{1-r+(g-2)r+2r/2+o(1)} = q^{1+(g-2)r+o(1)}$. Selecting $r = \frac{3}{2g+1}$ leads to an AT cost of $q^{\frac{5g-5}{2g+1}+o(1)}$ for discrete logarithm. For $g > 2$, this is better than Pollard rho.

Schemes using these groups were proposed [26], Verifiable Delay Functions with trustless setup [89,69] or zk-SNARG [15], assuming the hardness of Adaptive Division and/or Repeated Doubling in $\mathbb{G}$.

[1] It is possible to reduce the area, without changing the AT cost, by reducing this to $\approx g/3$ and the extended base size to $q^{1-\frac{2}{3g}+\frac{6}{g^2}}$ for g large enough.

4.2 Special number fields: factoring, and discrete logarithm

We now consider the case of factoring and discrete logarithm, where we work in $\mathbb{Z}[x]/(h,n)$ and n is often prime in discrete logarithm. $\mathbb{G}$ is chosen as $(\mathbb{Z}[x]/(h,n))^\times$, or a subgroup of it.

We consider two monic irreducible polynomials $f_i \in \mathbb{Z}[x]$, with f_1 of degree d_1, and f_2 of degree d_2, such that they have a known common factor g modulo n. It follows that we have a ring homomorphism from $\mathbb{Z}[x]$ to $\mathbb{Z}[x]/f_i$, and moding out by (n,g) to $\mathbb{Z}[x]/(g,n)$, and the diagram commutes.

$$\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl] \arrow[dr] & \\
\mathbb{Z}[x]/f_1 \arrow[dr] & & \mathbb{Z}[x]/f_2 \arrow[dl] \\
 & (\mathbb{Z}/n\mathbb{Z})[x]/h &
\end{tikzcd}$$

For factoring or computing discrete logarithms in $\mathbb{F}_p$, we have $d_1 = 1$ and take $g = f_1$. f_1 can be chosen as $x - m$ with $m \approx n^{1/d_2}$, f_2 as a base m expansion of n.

The factor base consists as usual of all prime ideals of norm smaller than some bound in the number fields $\mathbb{Q} \otimes \mathbb{Z}[x]/f_i$, and unit group generators. We take B as the norm bound, for a total of at most B elements. More precisely, we compute e the lcm of the class group cardinals, and the representative of $\mathfrak{p}$ is a generator of $\mathfrak{p}^e$. The image of elements of $\mathbb{Z}[x]/f_i$ in $\mathbb{G}$ can simply be computed by a modular reduction, followed by an exponentiation by e. For simplicity, we could force $e = 1$ as most number fields are principal.

When the group order ℓ is known, there is a faster choice, using Schirokauer's ideas [78]. We assume ℓ is prime, and prime with the number of roots of unity and the class number of all fields, and does not ramify. The ℓ-adic Iwasawa logarithm is defined by its morphism property, $\log_\ell(\ell) = 0$ and

$$\log_\ell(1-x) = \sum_{k=1}^{\infty} \frac{-x^k}{k}$$

when it converges, $\frac{\log_\ell(x^e)}{e}$ when it does not, with for example $e = |(\mathbb{Z}/\ell\mathbb{Z})[x]/f_i^\times|$. We select a $\mathbb{F}_\ell$ vector space of dimension d_i with corank the rank of the unit group. Then, we expect that for all principal ideals prime with ℓ, there is only one reduction of the Iwasawa logarithm of the generators modulo ℓ in this space. It implies that the generator is uniquely defined up to a ℓ-th power, and therefore the image of the principal ideal in $\mathbb{G}$ is well-defined. This is extended to all ideals by taking the e-th root of the image of the e-th power.

In order to find a prime factor $p|n$, we use section 3.3 with respect to $\mathbb{Z}/p\mathbb{Z}$ to compute g_i^N in $\mathbb{Z}/n\mathbb{Z}$ for all g_i in the factor base, with N a multiple of $p-1$. Then $\gcd(g_i^N - 1, n)$ reveals a multiple of p, unlikely to be n.

Descent on $a \in \mathbb{Z}/n\mathbb{Z}$ can start if the norm of the factors in $\mathbb{Z}[x]/f_1$ is smaller than $\mathcal{N}(n)^{o(1)}$. A prime factor $\mathfrak{p}$ appearing in the i-th number field can be cleared by finding an element in $\mathbb{Z}[x]$ such that the image in $\mathbb{Z}[x]/f_i$ has a factor $\mathfrak{p}$ and other smaller factors, and there exists a $j \neq i$ such that the image in $\mathbb{Z}[x]/f_j$ has only smaller factors. All these new factors are then cleared recursively, until they are all in the factor base, and this takes negligible time. This is detailed in one case in appendix A.

We now describe how to do the first step efficiently. The first idea is to use lattice reduction to compute a preimage of a of the form z/w with both terms in $\mathbb{Z}[x]/f_1$, so that $\mathcal{N}(z) \approx \mathcal{N}(w) \approx \sqrt{\mathcal{N}(n)}$. While this is well-known, we show how to use it to obtain a much larger gain. Critically, the element can be randomized, so that a low success probability is not a problem.

Smoothing. From Barbulescu [5, 4.5] and Guillevic [36, Theorem 5.3], who followed Pomerance's Early Abort Strategy [72] we obtain:

Heuristic Claim 6 *Given uniformly sampled numbers a, b and a constant $\alpha \leqslant 1$, it is possible to either fail or find a divisor and its factorization larger than respectively a^α and b^α; the prime factors are all smaller than $(a+b)^{o(1)}$ and the expected running time of the algorithm per success is*

$$L_{ab}\left[1/3, \left(\frac{675\alpha}{361}\right)^{1/3} + o(1)\right].$$

The algorithm consists in $\omega(1)$ stages with ECM, with increasing smoothness bounds, and early abort if the factored part is too small. (Barbulescu and Guillevic obtained their results only for one number and $\alpha = 1$, however products of numbers are known to be slightly smoother, and the density of rough numbers allows to extend the result to lower α)

We combine this with Coppersmith's factorization factory [20] to obtain:

Heuristic Claim 7 *After a precomputation of time*

$$L_{ab}\left[1/3, \left(\frac{2025(18+7\sqrt{6})}{-1021+8241\sqrt{6}}\right)^{1/3} + o(1)\right],$$

we can, given uniformly sampled number a, b, either fail or factor both of them with primes factors in $(a+b)^{o(1)}$ and the expected running time per success is

$$L_{ab}\left[1/3, \left(\frac{1350(7+3\sqrt{6})}{-1021+8241\sqrt{6}}\right)^{1/3} + o(1)\right].$$

Proof. We use the previous algorithm with some constant α. In case of success, we run the factorization factory on both numbers in time

$$L_{ab}\left[1/3, \left((1-\alpha)\frac{10+4\sqrt{6}}{9}\right)^{1/3} + o(1)\right].$$

The prime factors are smaller than $(ab)^{1/\log\log(ab)}$ with probability $L_{ab}[1/3, o(1)]$.

Hence, the expected running time per success is

$$L_{ab}\left[1/3, \max\left(\frac{675\alpha}{361}, \frac{(1-\alpha)(10+4\sqrt{6})}{9}\right)^{1/3} + o(1)\right].$$

We choose

$$\alpha = \frac{722}{12150 + 31097\sqrt{6}} \approx 0.54054069$$

which leads to the result, since the precomputation time is

$$L_{ab}\left[1/3, \left((1-\alpha)\frac{12+5\sqrt{6}}{3}\right)^{1/3} + o(1)\right]$$

The constants are respectively ≈ 1.548569 and ≈ 1.003556, which is significantly better than the previous ≈ 1.231966. We used the LLL algorithm [51] to obtain shorter formulas for the exact value α. This is however too slow for several applications.

For minimizing AT cost, we replace Coppersmith's algorithm by Bernstein's [7] (with cost $L_n[1/3, (5/3)^{4/3} + o(1)]$) and obtain with $\alpha = 9025/13399 \approx 0.6735$:

$$\sigma = (16875/13399)^{1/3} \approx 1.079917.$$

In our context, we can use Bernstein and Lange's batch factorization [8], for a fast batch smoothing algorithm. Their circuit factors $L_n[1/3, \frac{7}{16} + o(1)]$ numbers less than n, with an AT cost per number of

$$L_n\left[1/3, \frac{5}{18}\sqrt[3]{117+36\sqrt{10}} + o(1)\right]$$

with a constant less than 1.704, and a time of $L_n\left[1/3, \frac{1}{6}\sqrt[3]{117+36\sqrt{10}} + o(1)\right]$.

Heuristic Claim 8 *We can, given*

$$L_n\left[1/3, \frac{21}{8}\left(\frac{-9+9\sqrt{10}}{1527+5693\sqrt{10}}\right)^{1/3} + o(1)\right]$$

sequences of uniformly sampled number a, b, either fail or factor both of them with primes factors in $(a+b)^{o(1)}$ and AT cost per success in each sequence is

$$L_{ab}\left[1/3, 15\left(\frac{3+\sqrt{10}}{1527+5693\sqrt{10}}\right)^{1/3} + o(1)\right].$$

Proof. We use the same technique with

$$\alpha = \frac{5415+1805\sqrt{10}}{1527+5693\sqrt{10}} \approx 0.5695339.$$

The constants in the claim are respectively ≈ 0.262189 and ≈ 1.021187. We call $\sigma \approx 1.02$ the smoothing constant.

Parameters. We take $Q = n^{\deg\ g}$, $d_i = \delta_i \left(\frac{\log Q}{\log\log Q}\right)^{1/3}$, $B = L_Q[1/3, \beta + o(1)]$. The total complexity is $L_Q(1/3, \kappa + o(1))$. (The Greek letter is chosen according to the beginning of the word.)

Finding representatives. We assume that a preimage of the identity matrix by a sparse small random integer matrix with m rows and at least m columns can be computed in time m^τ. It is known that $\tau \leqslant 2.62902$ in the standard model [42]. When we can use Schirokauer's ideas, it is enough to invert a sparse random matrix modulo ℓ, so we can use $\tau \leqslant 2.2435$ in the standard model [42,27].

The corresponding AT cost is $\tau = 4$ in the first case, and $\tau = 3.5$ in the second, when we are limited to an area of $m^{1+o(1)}$. Indeed, in the first case, we are not interested in the preimages x which have $m^{1+o(1)}$ bits, but the corresponding representative in $\mathbb{G}$ (modulo n), which is $\prod a_i^{x_i}$. As the vectors are computed from the most significant bit to the least significant bit, we can reduce the memory used to $m^{1+o(1)}$ per vector. Using a preconditioner block size of $\lfloor\sqrt{m}\rfloor$ leads to the result [42]: each bit is produced using a multiplication of a dense matrix of dimension $\lfloor\sqrt{m}\rfloor$ and a vector, plus $\mathrm{O}\left(\frac{m}{\lfloor\sqrt{m}\rfloor}\right)$ iterations of a conjugate gradient, each one taking time $m^{1/2+o(1)}$ [86]. In the second case, we use Wiedemann's algorithm [90].

We actually have more area available, so that it is possible to decrease the running time, and the AT cost (see [42,27]). The area available will be at least $m^{2+o(1)}$ in our applications, so we can use a preconditioner block size of $\lfloor m^{3/4}\rfloor$ and a lattice dimension of $\lfloor m^{1/2}\rfloor$, for an AT cost of $m^{3.25+o(1)}$ in the first case. In the second case, we can use a block size (or lattice dimension) of $\lfloor m^{2/3}\rfloor$, for an AT cost of $m^{17/6+o(1)}$.

Factorization. We have to compute each number fields in time $L_n[1/3, \kappa + o(1)]$, which is therefore the number of elements whose smoothness is checked. They are of norm $L_n[2/3, \delta_2\kappa/2 + 1/\delta_2 + o(1)]$, so that with a smoothness bound of $L_n[1/3, \beta']$ we obtain

$$L_n\left[1/3, \kappa - \frac{\delta_2\kappa/2 + 1/\delta_2}{3\beta'} + o(1)\right]$$

relations. We obtain the conditions:

$$\kappa - \frac{\delta_2\kappa/2 + 1/\delta_2}{3\beta'} \geqslant \beta'$$
$$\tau\beta' \leqslant \kappa$$

and changing the first to an equality we get

$$\beta' = \frac{\kappa}{2}\left(1 - \sqrt{1 - \frac{2\delta_2}{3\kappa} - \frac{4}{3\delta_2\kappa^2}}\right).$$

This finds the representative of all $L_n[1/3, \beta']$ smallest prime ideals in $\mathbb{G}$. The representative of an ideal $\mathfrak{p}$ is found by computing a relation where it appears with an exponent of one, and the other ideals in the relation are of norm smaller than the norm of $\mathfrak{p}$. This idea was already sketched in 1993 [50, 9.5].

We now set the running time of a descent to $L_n[1/3, \sigma + o(1)]$. At the bottom of the descent, we search for algebraic integers with coefficient size $L_n[1/3, \kappa/2 + o(1)]$, so that their norm is

$$L_n[2/3, \delta_2\kappa/2 + 1/\delta_2 + o(1)]$$

and the expected number of solutions is:

$$L_n\left[1/3, \sigma - \frac{\delta_2\kappa + 4/\delta_2}{6\beta} + o(1)\right].$$

We therefore add the condition:

$$\sigma = \frac{\delta_2\kappa + 4/\delta_2}{6\beta}.$$

Finally, $\kappa = \sigma + \beta$. Using Lagrangian multipliers, we obtain:

$$\kappa = \frac{4}{\delta_2^2}$$

Combining these equations we get:

$$\kappa = \frac{9\sigma^3 + 2\sqrt{9\sigma^3 + 1} + 2}{9\sigma^2}.$$

Even with the minimum σ, we need $\tau \leqslant 4$, which we can achieve.

Heuristic Claim 9 *The Adaptive Division problem with parameter ℓ, and finding a factor p of n with* $\mathrm{gpf}(p-1) \leqslant \ell$ *can be both solved with an AT cost of*

$$L_\ell[1/2, \sqrt{2} + o(1)]L_n[1/3, 1.92756123].$$

The Repeated Doubling problem in $(\mathbb{Z}/n\mathbb{Z})^\times$ with e doublings can be solved with AT cost

$$\log(e)L_n[1/3, 1.92756123].$$

For ℓ,e low enough, this is better than Bernstein's *factorization* algorithm which has a constant of $(5/3)^{4/3} \approx 1.97605$ [7]. We can use this for smoothing, after the ECM steps, as we expect $\phi(n)$ to be as smooth as a uniform number. This in turn, leads to a slight reduction of the constant.

Time-lock puzzles were built on this group [75], as well as Verifiable Delay Functions with trustless setup [89,69] or zk-SNARG [15], which are assuming the hardness of Adaptive Division and/or Repeated Doubling in $\mathbb{G}$. RSA groups where $\mathrm{gpf}(p-1)$ is (relatively) smooth were proposed by Maurer and Yacobi for a non-interactive key distribution scheme [61,67].

Discrete logarithm in high characteristic finite field extensions. Guillevic [36] invented a fast smoothing algorithm, when we work over $\mathbb{F}_{p^d}$, which reduces the number to be factored to $\approx p^{d-d/r}$ where $r|d$. It consists in multiplying by an element in $\mathbb{F}_{p^{d/r}}$, chosen so that the product is can be written as a fraction of two short elements. They are obtained by lattice reduction. The overall effect is that σ can be multiplied by $(1-1/r)^{-1/3}$.

JLSV [39] used lattice reduction to find polynomials of the same size as in factoring, with $Q = p^d$, as long as the characteristic p is larger than $L_Q[2/3, \omega(1)]$.

Heuristic Claim 10 *The Adversarial-Basis Discrete Logarithm problem in a group of order $\ell | p^d - 1$, $\ell = L_Q[2/3, o(1)]$ can be solved in the time given in table 3.*

Table 3: Complexity of the Adversarial-Basis Discrete Logarithm Problem.

Smallest prime factor of d	Complexity
2, 3 or 5	$L_Q[1/3, \sqrt[3]{64/9}]$
7	$L_Q[1/3, 1.9231]$

Matyukhin's generalized Joux Lercier method [59] uses a different d, with no asymptotical difference.

Discrete logarithm in middle characteristic finite field extensions. We consider now the case where the characteristic is in $L_Q[2/3, \chi]$. We choose the conjugation method for generating the polynomials [6,68,77]. This means that $d_1 = 2d_2$, $Q = p^{d_2}$, and the coefficients of f_1 are $\mathrm{O}(\log Q)$ and the ones of f_2 are bounded by $\sqrt{p}$.

We now treat only the case where at the bottom of the descent, we sieve on linear polynomials, over a space of volume $L_Q[1/3, \kappa]$. It follows that the norm in the first field is $L_Q[2/3, \kappa\delta_2 + o(1)]$ and $L_Q\left[2/3, \frac{\kappa\delta_2}{2} + \frac{1}{2\delta_2} + o(1)\right]$ for the second. Thus, the descent succeeds when:

$$\sigma = \frac{3\kappa\delta_2^2 + 1}{6\delta_2\beta}.$$

It follows that

$$\kappa = \frac{6\sigma^3 + \sqrt{12\sigma^3 + 1} + 1}{6\sigma^2}.$$

With our σ, this gives when $\chi \approx 2.3071$ equal to $\kappa \approx 1.77427$. When d_2 is even, we can use the optimal parameters and $\kappa = 2^{4/3}3^{-1/3} \approx 1.74716$. The

τ required is $3+\sqrt{3}\approx 4.73$. The optimal AT cost for discrete logarithm is $\kappa = 5^{4/3}3^{-1}2^{-2/3}\approx 1.79536$.

We consider now the case where the characteristic is in $L_Q[1/3,\omega(1)]$ and $L_Q[2/3,o(1)]$. When we have to sieve a space of volume $L_Q[1/3,\kappa]$, we are forced to use lattices of degree $d\in\omega(1)$, with $p^d = L_Q[2/3,\delta]$. It follows that the norm in the first field is $L_Q\left[2/3,\frac{2\kappa}{\delta}+o(1)\right]$ and $L_Q\left[2/3,\frac{\delta}{2}+\frac{\kappa}{\delta}+o(1)\right]$ for the second. Thus, the descent succeeds when:

$$\sigma = \frac{\delta^2+6\kappa}{6\delta\beta}.$$

This leads to $\kappa = 2^{5/3}3^{-1/3}\approx 2.201284$ for ABDL as $\tau\leqslant 3+\sqrt{3}\approx 4.73$. The optimal AT cost for discrete logarithm is $\kappa = 5^{4/3}3^{-1}2^{-1/3}\approx 2.26201468$.

We note that if a subfield of the right size exists, even lower complexities can be reached [40,41]. Indeed, if the degree of the finite field has a factor of the right size, we can replace $\mathbb{Z}$ by the integers of a number field, and obtain a complexity identical to the previous case.

These are asymptotically the best algorithms in many (but not all) cases for attacking pairings [23]. When the Pollard Rho algorithm takes roughly the same time as finite field discrete logarithm, we have $p = L_Q[1/3,\chi]$ with a relatively large χ (e.g. from 3 to 8), so that the complexities are slightly higher.

5 Conclusion

We managed to remove the sparse system solving from many number theoretic algorithms by using black-box modular ring algorithms. We introduced a new representation allowing us to perform efficiently all ring operations. We proposed new black-box algorithms to recover the ring order and solve VDF problems. Then, we showed that we can apply these algorithms in various rings and groups. In particular, we proposed novel algorithms for the factorization and adversarial basis discrete logarithm problems in various groups. Finally, we described new smoothing algorithms in the AT cost and standard models which are of independent interest.

Assuming we can lower the cost of smoothing, and that we can compute descents with the same complexity as a relation in a multiple number field setup, we can lower further the complexities in the standard model. For factoring, the optimum is with a smoothing constant of $\sigma = (100/243)^{1/3}\approx 0.74381438$ and the cost of a ring operation would be [2]

$$2.5\sigma = \frac{(5/3)^{5/3}}{2^{1/3}}\approx 1.85953597$$

in the standard, or AT model. Likewise, all Multiple Number Field Sieve ABDL complexities would be decreased, and fast smoothing is already possible when the finite field degree has a small divisor.

[2] This is below the minimum claimed in [54, 2.7]

It seems difficult to adapt the black box modular ring framework to a (flat) torus, so that obtaining the generator of an ideal in a number field with this kind of technique is an interesting challenge.

References

1. L. M. Adleman, J. DeMarrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In *International Algorithmic Number Theory Symposium*, pages 28–40. Springer, 1994.
2. D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 5–17. ACM Press, Oct. 2015.
3. K. Alladi. An Erdos-Kac theorem for integers without large prime factors. *Acta Arith*, 49(1):81–105, 1987.
4. A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of computation*, 61(203):29–68, 1993.
5. R. Barbulescu. *Algorithms of discrete logarithm in finite fields*. Theses, Université de Lorraine, Dec. 2013.
6. R. Barbulescu, P. Gaudry, A. Guillevic, and F. Morain. Improving NFS for the discrete logarithm problem in non-prime finite fields. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 129–155. Springer, Heidelberg, Apr. 2015.
7. D. J. Bernstein. Circuits for integer factorization: a proposal. 2001.
8. D. J. Bernstein and T. Lange. Batch NFS. In A. Joux and A. M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 38–58. Springer, Heidelberg, Aug. 2014.
9. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, Jan. 2003.
10. D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, Aug. 2018.
11. D. Boneh, B. Bünz, and B. Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. https://eprint.iacr.org/2018/712.
12. D. R. L. Brown and R. P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. http://eprint.iacr.org/2004/306.
13. J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, June 1988.
14. J. P. Buhler, H. W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. In *The development of the number field sieve*, pages 50–94. Springer, 1993.
15. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.
16. E. R. Canfield, P. Erdös, and C. Pomerance. On a problem of oppenheim concerning "factorisatio numerorum". *Journal of Number Theory*, 17(1):1–28, 1983.

17. D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of computation*, 48(177):95–101, 1987.
18. D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.
19. J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, July 2010.
20. D. Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6(3):169–180, Mar. 1993.
21. D. Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Mathematics of Computation*, 62(205):333–350, 1994.
22. D. Coppersmith, A. M. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1):1–15, 1986.
23. G. De Micheli, P. Gaudry, and C. Pierrot. Asymptotic complexities of discrete logarithm algorithms in pairing-relevant finite fields. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 32–61. Springer, Heidelberg, Aug. 2020.
24. B. den Boer. Diffie-Hellman is as strong as discrete log for certain primes (rump session). In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 530–539. Springer, Heidelberg, Aug. 1990.
25. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.
26. S. Dobson, S. D. Galbraith, and B. Smith. Trustless groups of unknown order with hyperelliptic curves. Cryptology ePrint Archive, Report 2020/196, 2020. https://eprint.iacr.org/2020/196.
27. W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann, and G. Villard. Solving sparse rational linear systems. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 63–70, 2006.
28. A. Enge. The complexity of class polynomial computation via floating point approximations. *Mathematics of Computation*, 78(266):1089–1107, 2009.
29. A. Enge and F. Morain. Fast decomposition of polynomials with known Galois group. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pages 254–264. Springer, 2003.
30. D. Freeman. Pairing-based identification schemes. Cryptology ePrint Archive, Report 2005/336, 2005. http://eprint.iacr.org/2005/336.
31. P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 19–34. Springer, Heidelberg, May 2000.
32. P. Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. Cryptology ePrint Archive, Report 2004/073, 2004. http://eprint.iacr.org/2004/073.
33. P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of computation*, 76(257):475–492, 2007.
34. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.
35. R. Granger. On the static Diffie-Hellman problem on elliptic curves over extension fields. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 283–302. Springer, Heidelberg, Dec. 2010.
36. A. Guillevic. Faster individual discrete logarithms in finite fields of composite extension degree. *Mathematics of Computation*, 88(317):1273–1301, 2019.

37. A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method. *Math. Comput.*, 72(242):953–967, 2003.
38. A. Joux, R. Lercier, D. Naccache, and E. Thomé. Oracle-assisted static Diffie-Hellman is easier than discrete logarithms. In M. G. Parker, editor, *12th IMA International Conference on Cryptography and Coding*, volume 5921 of *LNCS*, pages 351–367. Springer, Heidelberg, Dec. 2009.
39. A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The number field sieve in the medium prime case. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 326–344. Springer, Heidelberg, Aug. 2006.
40. T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 543–571. Springer, Heidelberg, Aug. 2016.
41. T. Kim and J. Jeong. Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In S. Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 388–408. Springer, Heidelberg, Mar. 2017.
42. P. Kirchner. Fast linear algebra for number fields computation. *To appear.*
43. P. Kirchner and P.-A. Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 43–62. Springer, Heidelberg, Aug. 2015.
44. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.
45. N. Koblitz. Hyperelliptic cryptosystems. *Journal of cryptology*, 1(3):139–150, 1989.
46. N. Koblitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *J. Math. Cryptol.*, 2(4):311–326, 2008.
47. P. Koiran. Valiant's model and the cost of computing integers. *Electron. Colloquium Comput. Complex.*, (003), 2004.
48. M. Kraitchik. *Théorie des Nombres.* Gauthier-Villars, 1922.
49. J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *Journal of the ACM (JACM)*, 32(1):229–246, 1985.
50. A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard. The number field sieve. In *The development of the number field sieve*, pages 11–42. Springer, 1993.
51. A. K. Lenstra, H. W. J. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
52. A. K. Lenstra and M. S. Manasse. Factoring by electronic mail. In J.-J. Quisquater and J. Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 355–371. Springer, Heidelberg, Apr. 1990.
53. A. K. Lenstra and A. Shamir. Analysis and optimization of the TWINKLE factoring device. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 35–52. Springer, Heidelberg, May 2000.
54. A. K. Lenstra, A. Shamir, J. Tomlinson, and E. Tromer. Analysis of Bernstein's factorization circuit. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 1–26. Springer, Heidelberg, Dec. 2002.
55. A. K. Lenstra and B. Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *Int. J. Appl. Cryptogr.*, 3(4):330–343, 2017.
56. H. W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.
57. H. W. Lenstra Jr, J. Pila, and C. Pomerance. A hyperelliptic smoothness test. i. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 345(1676):397–408, 1993.

58. D. Matyukhin. On asymptotic complexity of computing discrete logarithms over GF(p). *Discrete Mathematics and Applications*, 13(1):27–50, 2003.
59. D. V. Matyukhin. Effective version of the number field sieve for discrete logarithm in a field $GF(p^k)$. *Trudy po Diskretnoi Matematike*, 9:121–151, 2006.
60. U. M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 271–281. Springer, Heidelberg, Aug. 1994.
61. U. M. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In D. W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 498–507. Springer, Heidelberg, Apr. 1991.
62. K. McCurley. Cryptographic key distribution and computation in class groups. *Proceedings of NATO ASI Number Theory and applications*, pages 459–479, 1989.
63. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer, Heidelberg, Aug. 1986.
64. R. T. Moenck. Fast computation of gcds. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 142–151, 1973.
65. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
66. D. Panario, X. Gourdon, and P. Flajolet. An analytic approach to smooth polynomials over finite fields. In *International Algorithmic Number Theory Symposium*, pages 226–236. Springer, 1998.
67. K. G. Paterson and S. Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Designs, Codes and Cryptography*, 52(2):219–241, 2009.
68. C. Pierrot. The multiple number field sieve with conjugation and generalized Joux-Lercier methods. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 156–170. Springer, Heidelberg, Apr. 2015.
69. K. Pietrzak. Simple verifiable delay functions. In A. Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, Jan. 2019.
70. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on information Theory*, 24(1):106–110, 1978.
71. J. M. Pollard. Theorems of factorization and primality testing. *Proceedings of the Cambridge Philosophical Society.*, 76(3):521–528, 1974.
72. C. Pomerance. Analysis and comparison of some integer factoring algorithms. *Computational methods in number theory*, pages 89–139, 1982.
73. C. Pomerance. The quadratic sieve factoring algorithm. In T. Beth, N. Cot, and I. Ingemarsson, editors, *EUROCRYPT'84*, volume 209 of *LNCS*, pages 169–182. Springer, Heidelberg, Apr. 1985.
74. R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
75. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. 1996.
76. L. Rotem and G. Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 481–509. Springer, Heidelberg, Aug. 2020.
77. P. Sarkar and S. Singh. New complexity trade-offs for the (multiple) number field sieve algorithm in non-prime fields. In M. Fischlin and J.-S. Coron, editors, *EURO-*

CRYPT 2016, Part I, volume 9665 of *LNCS*, pages 429–458. Springer, Heidelberg, May 2016.

78. O. Schirokauer. Discrete logarithms and local units. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 345(1676):409–423, 1993.
79. R. Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.
80. I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004. http://eprint.iacr.org/2004/031.
81. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.
82. I. E. Shparlinski. Orders of points on elliptic curves. *Contemporary Mathematics*, 369:245–252, 2005.
83. A. V. Sutherland. *Order computations in generic groups*. PhD thesis, Massachusetts Institute of Technology, 2007.
84. A. V. Sutherland. Accelerating the CM method. *LMS Journal of Computation and Mathematics*, 15:172–204, 2012.
85. E. Thomé. Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. *Journal of symbolic computation*, 33(5):757–775, 2002.
86. C. D. Thompson and H. T. Kung. Sorting on a mesh-connected parallel computer. *Communications of the ACM*, 20(4):263–271, 1977.
87. L. Valenta, S. Cohney, A. Liao, J. Fried, S. Bodduluri, and N. Heninger. Factoring as a service. In J. Grossklags and B. Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 321–338. Springer, Heidelberg, Feb. 2016.
88. L. G. Valiant. Completeness Classes in Algebra. In M. J. Fischer, R. A. DeMillo, N. A. Lynch, W. A. Burkhard, and A. V. Aho, editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing*, pages 249–261. ACM, 1979.
89. B. Wesolowski. Efficient verifiable delay functions. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.
90. D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE transactions on information theory*, 32(1):54–62, 1986.
91. H. C. Williams. A $p+1$ method of factoring. *Mathematics of computation*, 39(159):225–234, 1982.

A Descent complexity

We describe how a descent works, in one case, which corresponds to factoring and discrete logarithm with large characteristic. We want to descend an ideal $\mathfrak{p}$ of norm $q \leqslant Q^{o(1)}$, in time $L_Q[1/3, \sigma + o(1)]$.

We start by the case where the norm q is larger than $L_Q[1/3, C]$, with large enough C (say $\log\log Q$). In order to descend an ideal, it is enough to find an element of $\mathfrak{p}$ which has no large factor beside $\mathfrak{p}$. We consider at most $L_Q[1/3, \sigma]$ elements of degree t, so that the size of the coefficients is $\approx q^{1/t}$ [51]. The product of the norms is therefore $\approx Q^{t/d_2} q^{d_1/t} \cdot Q^{t/d_2} q^{d_2/t}$, which is optimized with $t \approx d_2 \sqrt{\frac{\log q}{2 \log Q}}$, $t = \omega(1)$ to

$$\approx Q^{(4+o(1))t/d_2} = L_Q[2/3, (4+o(1))t/\delta_2] = L_Q\left[1, \sqrt{\frac{(8+o(1))\log q}{\log Q}}\right].$$

As we use $\approx L_Q[1/3, \sigma]$ trials, the smoothness bound must be in

$$L_Q\left[1/3, \frac{(4+o(1))t}{3\sigma\delta_2}\right] = L_Q\left[2/3, \sqrt{\frac{(8+o(1))\log q}{9\sigma^2 \log Q}}\right].$$

Since this is in $L_Q[2/3, o(1)]$, ECM runs in negligible time. This leads to $\mathrm{O}(\log Q)$ ideals to be recursively descended (more precisely, $\mathrm{O}\left(\log^{1/3} Q\right)$). Observe that the new t will be chosen near

$$d_2 \sqrt{\frac{4t}{3\sigma\delta_2}} \log^{-1/3} Q \log\log^{1/3} Q \approx \sqrt{\frac{(4+o(1))t\delta_2}{3\sigma}}.$$

The norm of q decreases to below $L_Q[1/3, C]$ in less than $\log\log t \approx \log\log\log Q$ recursive steps.

We now analyze more precisely what happens when the ideal is in the i-th field, its norm is in $L_Q[1/3, \nu]$ and ν is small, less than $\log\log Q$. We use $t = 1$, in order to simplify the analysis. The size of the elements used is now $L_Q[1/3, \frac{\nu+\sigma}{2}]$, the product of the norms in the fields is

$$L_Q[2/3, \delta_2 \frac{\nu+\sigma}{2} + \frac{2}{\delta_2} + o(1)].$$

We set a smoothness bound so that the smoothness probability is $L_Q[1/3, -\sigma + o(1)]$. This is

$$L_Q\left[1/3, \beta \frac{\sigma + \nu + 4/\delta_2^2}{\sigma + \beta + 4/\delta_2^2}\right],$$

so that ν converges exponentially quickly to β.

As the number of branches is bounded by $\log Q$, it follows that the number of prime ideals at the end of the descent is bounded by $\exp((1+o(1)) \ln\log Q \log\log\log Q) = L[1/3, o(1)]$, and it can be completed in time (or AT cost) $L_Q[1/3, \sigma + o(1)]$.