Trustless Groups of Unknown Order with Hyperelliptic Curves

Samuel Dobson[1], Steven D. Galbraith[1], and Benjamin Smith[2]

[1]Mathematics Department, University of Auckland, New Zealand.
samuel.dobson.nz@gmail.com, s.galbraith@auckland.ac.nz

[2]Inria and École polytechnique, Institut Polytechnique de Paris, Palaiseau, France.
smith@lix.polytechnique.fr

June 26, 2020

Abstract

Groups of unknown order are of major interest due to their applications including time-lock puzzles, verifiable delay functions, and accumulators. In this paper we focus on trustless setup: in this setting, the most popular unknown-order group construction is ideal class groups of imaginary quadratic fields.

We argue that the full impact of Sutherland's generic group-order algorithm has not been recognised in this context, and show that group sizes currently being proposed in practice (namely, approximately 830 bits) do not meet the claimed security level. Instead, we claim that random group orders should be at least 3300 bits to meet a 128-bit security level. For ideal class groups this leads to discriminants of around 6656 bits, which are much larger than desirable.

One drawback of class groups is that current approaches require approximately $2\log_2(N)$ bits to represent an element in a group of order N. We provide two solutions to mitigate this blow-up in the size of representations. First, we explain how an idea of Bleichenbacher can be used to compress class group elements to $\frac{3}{2}\log_2(N)$ bits. Second, we note that using Jacobians of hyperelliptic curves (in other words, class groups of quadratic *function* fields) allows efficient compression to the optimal element representation size of $\log_2(N)$ bits. We discuss point-counting approaches for hyperelliptic curves and argue that genus-3 curves are secure in the trustless unknown-order setting. We conclude that in practice, Jacobians of hyperelliptic curves are more efficient in practice than ideal class groups at the same security level—both in the group operation and in the size of the element representation.

1 Introduction

Interest in groups of unknown order has been fuelled in recent years by applications such as delay functions [BBF18], accumulators [BBF19], and zero-knowledge proofs of knowledge [BFS19]. As the name suggests, a group G has *unknown order* if it is infeasible for anyone to compute the order of G without access to any secret information used to construct G. In the case of *trustless setup*, the order should not even be known to the creator(s) of the group. Specific use-cases may require additional properties, such as Wesolowski's *adaptive root assumption* [Wes19], which is that it should be infeasible to compute random roots, or any elements of known order.

Previously, there have been two proposals for concrete groups of unknown order: RSA groups [RSW96], and ideal class groups of imaginary quadratic fields [BW88]. RSA groups are groups of the form $(\mathbb{Z}/N\mathbb{Z})^\times$, where $N = pq$ is the product of two primes. Computing the order of this group is equivalent to factoring N. The main drawback of RSA groups is that it requires a trusted setup to generate N (or at least trust that the factorisation of N is not known to anyone, or has been destroyed). Class groups, on the other hand, do not require a trusted setup to generate, and thus have received a lot of attention (see e.g. [Wes19, BBF19, BH01]).

In this work we begin by reconsidering parameter sizes for the class group setting in the context of Sutherland's generic group order algorithm [Sut07]. A randomly-generated group is only vulnerable to this algorithm with small probability, *depending on the order.* But inherent to our use-case, since we don't know the order, we cannot check if the group is vulnerable or not without simply attempting to run the algorithm for some time. Suppose a group G is vulnerable to an attack running in time T with probability P. Then, we claim that the security level of the system is at most $\max(T, 1/P)$. In other words, if $\max(T, 1/P) < 2^\lambda$ then there is an adversary that runs in less than 2^λ time that breaks the scheme with probability greater than $1/2^\lambda$.

In Section 4 we show that Sutherland's order-computation algorithm presents a strong opponent to class groups at current discriminant sizes, and propose new parameter sizes in response. For example, Buchmann and Hamdy [BH01] suggest that class groups with 1665-bit discriminants offer 128-bit security; at this size, we expect the order of the class group to be around 2^{833}. This security estimate was used by Boneh et al. [BBF19, BFS19] in their class-group-based instantiations of various protocols. But Sutherland's algorithm will compute the group order in around 2^{55} operations with probability $\sim 2^{-55}$ (corresponding to $u = 15$ below). Thus, we claim that groups with these parameters only offer 55-bit security.

Our second major result is a more compact representation of class group elements. Inspired by a signature-compression method of Bleichenbacher [Ble04], in Section 5 we compress elements of ideal class groups to 3/4 the size of the usual representation. This space saving is particularly welcome in the light of our updated, much larger class group parameters.

In Sections 6, 7, and 8, we present an alternative source of groups of unknown order, without trusted setup: Jacobians of hyperelliptic curves of genus 3. Jacobians are more efficient than class groups at the same security level: the element representation size is smaller by a factor of 2 (or 3/2 if our new compression algorithm is used), and the group operation can be computed more efficiently. We acknowledge that there are potentially polynomial-time algorithms to compute the group order of hyperelliptic Jacobians [Pil90, GH00]. However, there is evidence that these algorithms are already impractical for discrete-log-based cryptographic parameter sizes, i.e., group orders of around 256 bits. Hence, for the extremely large group orders we have in mind, we believe such methods are not a concern. While curves of genus 2 and above can be considered, we suggest that genus-3 curves are a safer choice, since the point counting algorithms are more complex in that case. Naturally, if progress was made to render Schoof-type algorithms for genus 3 practical, then our proposal would be nullified—but at least we have provided motivation for such work. Finally, in Section 9 we explore the use of multiple instances in parallel.

This paper supersedes ePrint 2020/196.

Acknowledgements. We thank Dan Boneh, Benjamin Wesolowski, Steve Thakur, and Jonathan Lee for their valuable comments, feedback and suggestions on an earlier version of this work. We also like to thank Edward Chen for beneficial discussion.

2 Preliminaries

We begin by recalling some elementary definitions and facts on groups of unknown order, ideal class groups, and Jacobians of hyperelliptic curves. Good references include [Coh10] and [Cox89] for class groups, and [MWZ96] and [Gal12] for hyperelliptic Jacobians.

Notation. Recall the soft-O notation: $\widetilde{O}(x) = O((\log x)^c \cdot x)$ for some constant c. We shall also use the L-notation: $L_n(\alpha) = \exp\left[(1 + o(1))(\ln n)^\alpha (\ln \ln n)^{1-\alpha}\right]$.

2.1 Groups of unknown order

As the name implies, a group of unknown order is a group whose order should be infeasible to calculate. The order of the group acts as secret information, which is known either to the creator of the group (trusted setup), or to no-one. In order to be useful, despite not knowing the order of the group, the group operation should be efficiently computable; group elements should have a practical representation; and it should be possible to efficiently generate random elements of the group.

An additional requirement often used follows directly from Wesolowski's [Wes19] adaptive root assumption: that it should be infeasible to find any non-trivial element in the group with known order. More generally, it should be infeasible to compute random roots of any non-trivial element in the group. This assumption is relied upon in constructions such as Proof of Exponentiation (PoE). We briefly recall this construction, as it will be a useful example later.

We have as parameters a group, G, chosen according to security parameter λ. The Proof of Exponentiation takes as input u and w in G and x in $\mathbb{Z}$, and aims to prove that $u^x = w$. It proceeds interactively with a challenge-response format, although can be made non-interactive (see [Wes19, Appendix D] for details).

1. The verifier sends a random prime $\ell \in \mathrm{Primes}(\lambda)$ to the prover.
2. The prover computes $q = \lfloor x/\ell \rfloor$ and $Q = u^q$, and sends Q to the verifier.
3. The verifier computes $r = x \bmod \ell$, and accepts if $Q^\ell u^r = w$

In order to illustrate the necessity of the adaptive root assumption for security of this protocol, suppose we know an element -1 of order 2 in G. Then for any valid proof that $u^x = w$, we can also easily generate a false proof of the contradictory statement $u^x = -w$, by replacing Q with $Q' = -1 \cdot Q$ in the proof. Since ℓ is odd, $(Q')^\ell u^r = -1 \cdot Q^\ell u^r = -w$ will hold, despite the fact that $u^x \neq -w$. That is why when using RSA groups, it is important to use the quotient $(\mathbb{Z}/N\mathbb{Z})^*/\langle \pm 1 \rangle$ to eliminate this element.

As mentioned earlier, previous work with groups of unknown order has been primarily centred around two ideas: RSA groups [RSW96] and class groups of imaginary quadratic orders [BW88] (discussed below in Section 2.2). Brent [Bre00] briefly mentioned using the Jacobian of a hyperelliptic curve as a group of unknown order, following work by Koblitz [Kob89] on the use of Jacobians as groups in which the discrete logarithm problem is infeasible. But unlike class groups of imaginary quadratic fields, this Jacobian idea has received very little further attention.

2.2 Ideal and form class groups

An **imaginary quadratic field** is an algebraic extension

$$K = \mathbb{Q}(\sqrt{d}) = \{a + b\sqrt{d} \mid a, b \in \mathbb{Q}\}$$

where $d < 0$ is a square-free integer. The discriminant Δ of K is d if $d \equiv 1 \pmod 4$, or $4d$ otherwise (so $\Delta \equiv 0, 1 \pmod 4$, since only 0 and 1 are quadratic residues modulo 4). The ring of integers $\mathcal{O}_K$ is generated by $\{1, \omega\}$, where $\omega = \frac{1}{2}(1 + \sqrt{d})$ when $d \equiv 1 \pmod 4$ and $\omega = \sqrt{d}$ otherwise.

Let J_K denote the group of non-zero fractional ideals of $\mathcal{O}_K$. Let $P_K < J_K$ be the subgroup of non-zero principal fractional ideals. Then the ideal class group is the quotient group $Cl(\mathcal{O}_K) = J_K/P_K$. This is the abelian group of fractional ideal classes under the equivalence relation $\mathfrak{a} \sim \mathfrak{b}$ iff $(\alpha)\mathfrak{a} = (\beta)\mathfrak{b}$ for some principal ideals (α) and (β). We denote the class of an ideal $\mathfrak{a}$ as $[\mathfrak{a}]$. The identity of the group is $[(1)]$, and the order of this group is the class number of K (it is always finite when constructed with a ring of integers such as $\mathcal{O}_K$). We shall denote the order of the group as $h(\Delta) = |Cl(\mathcal{O}_K)|$.

In practice, it is most efficient to represent $Cl(\mathcal{O}_K)$ using binary quadratic forms: that is, using the isomorphic **form class group** $Cl(\Delta)$, which is defined as follows. We denote by (a, b, c) the binary quadratic form

$$(a, b, c) = ax^2 + bxy + cy^2 \in \mathbb{Z}[x, y]$$

Its discriminant is $\Delta = b^2 - 4ac$. In $Cl(\Delta)$, we can represent forms using only two elements (a, b), because c is uniquely determined by the equation $c = (b^2 - \Delta)/4a$. A form is **positive definite** if $a > 0$. Just as in the ideal class group, we have an equivalence relation on these quadratic forms. We say forms f and g are equivalent, and write $f \sim g$, if there exists a matrix $\left(\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right)$ in $\mathrm{SL}_2(\mathbb{Z})$ such that $f(x, y) = g(\alpha x + \beta y, \gamma x + \delta y)$.

Equivalent forms always have the same discriminant, so the class group of forms under this relation, denoted $Cl(\Delta)$, is well defined.

We represent each equivalence class in $Cl(\Delta)$ using the unique **reduced form** in that equivalence class. A reduced form (a, b, c) of discriminant Δ is one where $|b| \leq a \leq c$, and if $|b| = a$ or $a = c$, then $b \geq 0$. Lagrange, and later Gauss and then Zagier, gave algorithms for finding an equivalent reduced form for any binary quadratic form. The identity in $Cl(\Delta)$ is the equivalence class of the principal form $(1, 0, -k)$ if $\Delta = 4k$, or $(1, 1, k)$ if $\Delta = 4k + 1$. The group law in $Cl(\Delta)$ is known as composition of forms, and is due to Gauss (it corresponds exactly to multiplication of ideals in $Cl(\mathcal{O}_K)$, defined by $\mathfrak{ab} = \{\sum a_i b_i \mid a_i \in \mathfrak{a}, b_i \in \mathfrak{b}\}$). The algorithm does not usually output a reduced form, so reduction is an additional step. We shall not give these algorithms here (the reader can consult [Coh10] for more information).

It follows from the Brauer–Siegel theorem that for sufficiently large negative discriminant $\Delta \to -\infty$, the class number is on average

$$h(\Delta) \approx \frac{1}{2}\sqrt{|\Delta|} \tag{1}$$

(see [HM00]), and thus $\log h(\Delta) \sim \log \sqrt{|\Delta|}$. So we can conservatively assume a group size of $\approx \frac{1}{2} \log |\Delta|$ bits for cryptographic-sized negative discriminants.

The use of class groups in cryptography was first suggested by Buchmann and Williams [BW88]. A sub-exponential algorithm for computing the order of $Cl(\Delta)$ was given by Hafner and McCurley [HM89], with complexity $L_{|\Delta|}(1/2)$. Thus, the order of a class group $Cl(\Delta)$ of negative prime discriminant $\Delta \equiv 1 \pmod 4$ is believed to be difficult to compute, if Δ is sufficiently large. Wesolowski [Wes19] proposed that by simply selecting a suitable large Δ, the class group $Cl(\Delta)$ can be used as a group of unknown order without a trusted setup. We also need to choose an element in $Cl(\Delta)$ which we shall treat as a generator; it is not possible to know if it really generates the whole of $Cl(\Delta)$ or just a subgroup (we shall discuss this further below).

2.3 Hyperelliptic curves

Let k be a field, and $\overline{k}$ its algebraic closure. A hyperelliptic curve C of genus $g > 1$ is a curve of the form $y^2 + h(x)y = f(x)$, where h and f are polynomials over k with f monic, $\deg f = 2g + 1$, and $\deg h \leq g$. (We will not need the more general case where deg $f = 2g + 2$.) The curve C must have no singular points: that is, there is no (u, v) in $\overline{k}^2$ satisfying both the equation of C and its partial derivatives. For an extension field K/k, we denote by $C(K)$ the set of points P in K^2 satisfying C (the **finite** points), together with the projective point at infinity ∞ in $\mathbb{P}^2(k)$. Recall that every $P = (x, y)$ on C has an **opposite** point $\tilde{P} = (x, -y - h(x))$ (with $\infty = \tilde{\infty}$).

Unlike points on elliptic curves (which correspond to $g = 1$), the points in $C(K)$ do not form a group. Instead, the group we use is the class group of degree-0 divisors on C, also known as the Jacobian.

The coordinate ring of C over K is the quotient ring $K[C] = K[x, y]/(y^2 + h(x)y - f(x))$, where the modulus is the ideal generated by the equation of C. Elements of $K[C]$ are called polynomial functions on C. Every polynomial function $G(x, y)$ can be written in the form $a(x) - b(x)y$ for some a and b in $\overline{k}[x]$. The conjugate of G is $\overline{G} := a(x) + b(x)(h(x) + y)$, and the **norm** of G is $N(G) := G\overline{G}$. The ring $\overline{k}[C]$ is an integral domain.

The function field $K(C)$ of C is the field of fractions of $K[C]$. We say that a function R in $\overline{k}(C)$ is defined at a point $P \neq \infty$ in $C(\overline{K})$ if and only if $R = G/H$ for some polynomial functions G and H in $\overline{k}[C]$ with $H(P) \neq 0$; and in this case, $R(P) = G(P)/H(P)$.

A **divisor** on C is a formal sum of points $D = \sum m_P P$ where $m_P = 0$ for all but finitely many P. The divisors form a group $\operatorname{Div} C$. The degree of a divisor is $\deg D = \sum m_P$. The divisors of degree zero $\operatorname{Div}^0(C)$ is a proper subgroup of $\operatorname{Div}(C)$. A **principal** divisor is a divisor of the form $(\gamma) = \sum_{P \in C} m_P P$ for a function γ in $\overline{k}(C)$, where $m_P = \operatorname{ord}_P(\gamma)$ is the order of vanishing of γ at the point P. Denote by $\mathcal{P}(C)$ the set of principal divisors of C. It is a fact that principal divisors have degree 0, so $\mathcal{P}(C) < \operatorname{Div}^0(C)$, and we may define the **Jacobian**

$$J_C \cong \operatorname{Div}^0(C)/\mathcal{P}(C) \tag{2}$$

(also known as the degree-0 Picard group, denoted by $\mathrm{Pic}^0(C)$). Two divisors D_1 and D_2 in $\mathrm{Div}^0(C)$ are equivalent if $D_1 - D_2$ is in $\mathcal{P}(C)$. Each element in J_C can be represented by the unique **reduced** divisor in the divisor class: this is the divisor of the form $D = P_1 + \cdots + P_r - r\infty$ (with the P_i not necessarily distinct) such that $r \geq g$ and $P_i \neq \tilde{P}_j$ for all $i \neq j$.

We represent reduced divisors using the Mumford representation [Mum07]. A reduced divisor is uniquely specified by a pair of polynomials $\langle u(x), v(x) \rangle$, where u is monic, $\deg v < \deg u \leq g$ and u divides $v^2 + hv - f$. Specifically, the roots of $u(x)$ are the x-coordinates of the points in the support of the divisor. The divisor is **prime** if $u(x)$ is irreducible over $\mathbb{F}_q$. Cantor's algorithm computes the group law on J_C in terms of the Mumford representation [Can87] (see also [CL12]). More efficient explicit formulae have been given when the genus g is small: see [Lan05] for $g = 2$ and [FWG07] for $g = 3$.

If we have a curve C of genus g over the finite field of cardinality q, then the $\mathbb{F}_q$-rational points of J_C form a finite group. The Hasse–Weil bound tells us that $\#J_C \sim q^g$; more precisely,

$$(\sqrt{q} - 1)^{2g} \leq \#J_C \leq (\sqrt{q} + 1)^{2g} .$$

An abelian variety—and specifically, in this case, the Jacobian—is **simple** if it does not contain a (non-zero) proper abelian subvariety. If J_C contains an abelian subvariety A (such as an elliptic curve E, in the case where there exists a nontrivial map $C \to E$), then $\#A(\mathbb{F}_q)$ divides $\#J_C(\mathbb{F}_q)$. This type of splitting may reduce the difficulty of order computations, so we want to restrict to simple Jacobians; fortunately, a general Jacobian is simple.

Recall that if E is an elliptic curve, then for all positive integers n there exists a **division polynomial** $\Psi_n(X)$ such that $\Psi_n(x(P)) = 0$ if and only if $P \neq 0$ is in the n-torsion of E. The hyperelliptic analogue of these polynomials are division *ideals*: homogenous ideals vanishing on coordinates of points in torsion subgroups (in this sense, the genus-one division ideals are the principal ideals generated by the n-division polynomials). Cantor constructs a system of polynomials generating the ℓ-division ideal in [Can94] (see also [CFA$^+$05]); in genus 3, the degrees of Cantor's ℓ-division polynomials are bounded by $O(\ell^2)$ (see [Abe18]).

3 Sutherland's algorithm: the security of generic groups

We now come to the algorithm presented by Sutherland in his thesis [Sut07, Algorithm 4.2], which he calls the *primorial-steps* algorithm. This algorithm computes the order of an element in a generic group; it can be used to probabilistically determine the exponent of a group. Remarkably, it runs in $O(\sqrt{N/\log\log N}) = o(\sqrt{N})$ time (where N is the group order) in the worst case, but in fact the expected runtime depends heavily on the multiplicative structure of N. The algorithm runs particularly quickly when N is smooth, which we do not expect (or desire!) in a group of unknown order. However, Sutherland's algorithm also poses a significant threat to a larger class of groups.

The primorial-steps algorithm is based on the baby-step giant-step (BSGS) algorithm. Suppose we wish to compute the order of α. Instead of computing consecutive powers of α in the baby-steps, we instead compute a new element $\beta = \alpha^E$, such that the order of β is coprime to all primes $2, 3, \ldots, p_n \leq L$ for a chosen bound L. This is done by setting E to be a product of the p_i, each raised to an appropriate exponent $\lfloor \ell \log_{p_i} 2 \rfloor$ (where ℓ is the bit-size of the group element identifiers). The baby-steps are then all powers of β with exponents coprime to P_n, and the giant-step exponents are multiples of P_n, where $P_n = \prod_{i=1}^n p_i$. As in BSGS, a collision allows $|\beta|$ to be learnt, which then allows $|\alpha|$ to be computed very efficiently.

Sutherland shows that if the order N of a group element α is uniformly distributed over $[1, M]$ (for sufficiently large M) and $L = M^{1/u}$, then this is a $O(M^{1/u})$ time and space algorithm that successfully computes N with probability $P \geq G(1/u, 2/u)$ [Sut07, Proposition 4.7]. Here $G(r, s)$ is the semismooth probability function

$$G(r, s) = \lim_{x \to \infty} \psi(x, x^s, x^r)/x$$

where $\psi(x, y, z)$ is the number of integers up to x which are semismooth with respect to y and z (that is, all prime factors less than y, with at most one greater than z).

Table 1: Asymptotic semismoothness probabilities for various values of u, from [Sut07] and [BP96]

u	$G(1/u, 2/u)$	u	$G(1/u, 2/u)$	u	$G(1/u, 2/u)$
2.1	0.9488	5.0	0.4473	12.0	4.255e-12
2.9	0.5038	6.0	1.092e-03	16.0	6.534e-19
3.0	0.4473	10.0	5.382e-09	20.0	2.416e-26

Table 1 gives some numerically computed values for $G(1/u, 2/u)$ from [BP96] and [Sut07]. But in order to estimate the group orders required for 100- or 128-bit security, we need even larger values of u. For this reason, we plotted the known semismoothness probabilities from Table 1 as a function of u, and tentatively extrapolated the trend to larger values (see Figure 1). Our extrapolation suggests that for a success probability of less than 2^{-100}, we should take $u \approx 23$; for 2^{-128}, we should take $u \approx 26$.

In the introduction, we stated that the bit-security level λ is bounded by $2^{\lambda} \leq \max(T, 1/P)$, where there exists an attack running in time T that succeeds with probability P. Therefore, when targeting 128-bit security, we must ensure that the probability that a $\leq 2^{128}$-step algorithm exists for a random group order is less than 2^{-128}. Hence, we suggest that order approximately $M = 2^{128 \times 26} = 2^{3328}$ is required for any random group to meet this security level.

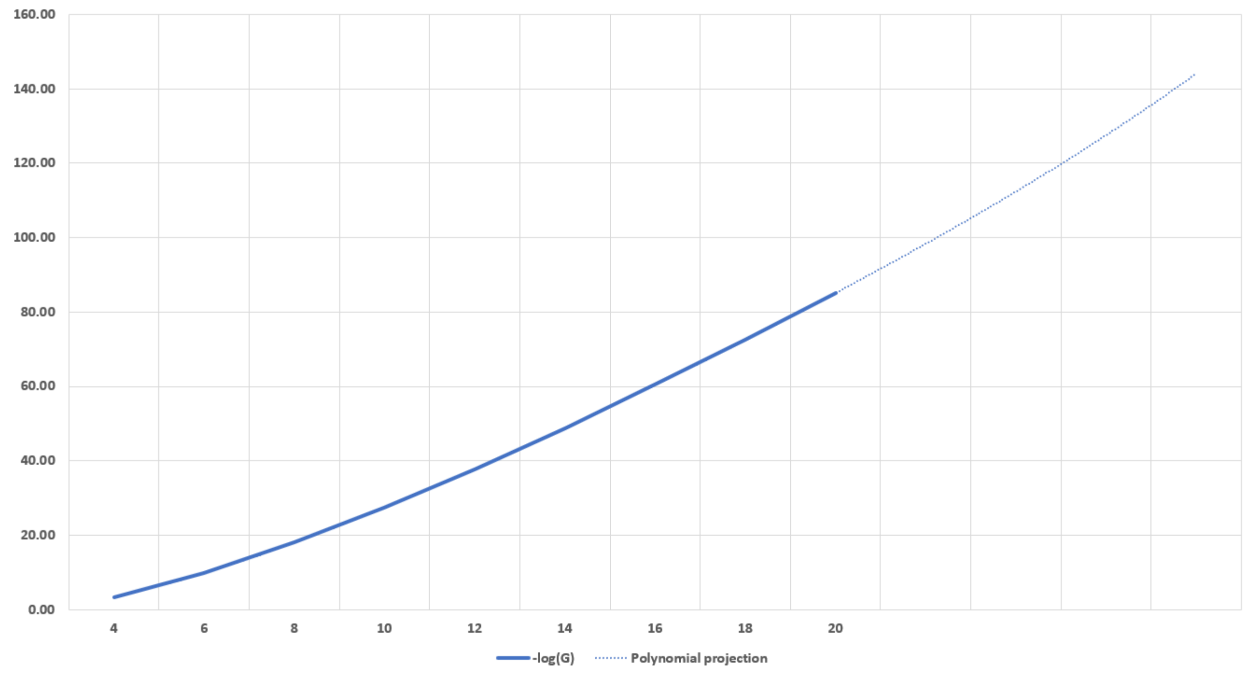

Figure 1: Graph of $-\log G(1/u, 2/u)$ as u grows, from values in [BP96] for $u \leq 20$, with a tenuous polynomial projection for $u > 20$.

We remark that Sutherland's algorithm is less of a threat to unknown order groups with trusted setup. For example, if there is an authority that can be trusted to generate an RSA modulus $N = pq$ where p and q are safe primes, then the order of $\mathbb{Z}_N^{\times}$ cannot be computed using Sutherland's approach.

4 The security of ideal class groups

Until now, cryptographic class group parameters have mainly been proposed with an eye to resisting Hafner and McCurley's subexponential algorithm for computing orders of quadratic imaginary class groups [HM89]. In this section we re-assess the security these parameters in the light of Sutherland's algorithm, and propose new (much larger) parameter sizes targeting the 128-bit security level.

Hafner and McCurley gave their $L_{|\Delta|}(1/2)$ algorithm to compute the order of quadratic imaginary class groups in 1989 [HM89]; for three decades this has been the benchmark for cryptographers estimating the security of quadratic imaginary class groups. The important thing for us to note is that while the Hafner–McCurley algorithm is subexponential, its runtime depends essentially on the size of Δ; in contrast, Sutherland's algorithm has exponential worst-case runtime, but performs much faster with non-negligable probability, depending on the structure of the class group—a factor that the Hafner–McCurley algorithm cannot exploit. When computing the order of a randomly selected class group, therefore, the small probability that Sutherland's algorithm outperforms Hafner–McCurley must be taken into account.

The cryptographic parameter sizes in [HM00] and [BH01] both suppose that Hafner–McCurley is the best known algorithm. Concretely, it is suggested that a 1665-bit negative fundamental discriminant, which means an approximately 833-bit group order (cf. Eq. (1)), should provide 128-bit security. This estimate has since been quoted in various more recent works (e.g. [BBF19] and [BFS19]).

But now suppose we try to compute the order of a random class group with 1665-bit fundamental negative discriminant using Sutherland's algorithm. Hamdy and Möller [HM00] show that for negative fundamental discriminants, it is expected that class numbers are more frequently smooth (although not significantly so) than uniformly random integers of the same size. We may therefore conservatively approximate the smoothness probability of random class group orders as being that of random integers. With the results summarized in Section 3, we find that the probability that such a group class group has less than 128 bits of security ($u = 6.5$) is at least 2^{-14}, and the chance it has less than 64-bit security is 2^{-42}. Thus, with respect to Sutherland's algorithm, the security is much weaker at these discriminant sizes than was previously thought. In fact, taking $u = 15$ (so that a $\sim 2^{55}$ operation algorithm succeeds with probability $\sim 2^{-55}$), we estimate that class groups with 1665-bit discriminants only provide 55-bit security.

Bach and Peralta [BP96] give $G(1/u, 2/u)$ for $u = 20$ as $2.415504 \times 10^{-26} \approx 2^{-85}$. Thus in order to truly obtain even 85-bit security, a discriminant of size around 3400 bits would be required. Using our extrapolation of $G(1/u, 2/u)$ in Figure 1, we estimate that for 100-bit security with respect to Sutherland's algorithm, a discriminant of around 4400 bits would be required. For 128-bit security, we estimate that $u = 26$ should give $G(1/u, 2/u) \approx 2^{-128}$. this implies a group order $N \approx 2^{128\times 26} = 2^{3328}$, and hence we estimate that Δ should be approximately 6656 bits.

We stress that these are tenuous estimates, and that $G(1/u, 2/u)$ is only a lower bound for the sucess probability of Sutherland's algorithm. Still, this should serve at least as a starting point for more accurate estimates in future work.

5 Compression of group elements in class groups

Bleichenbacher [Ble04] proposed a beautiful algorithm to compress Rabin signatures, and his methods can also be used to compress elements in ideal class groups. As far as we can tell, this simple observation has not been made in the literature previously.

As mentioned in Section 2.2, an element of the ideal class group corresponds to a triple of integers (a, b, c) such that $b^2 - 4ac = \Delta$. Since Δ is a fixed and known constant, it suffices to store the pair (a, b). A reduced quadratic form satisfies $|b| \leq a < \sqrt{|\Delta|}$. It follows that the pair (a, b) can be encoded in approximately $\log_2(|\Delta|)$ bits.

Recall that a Rabin signature on a message m under the public (RSA) key N is an integer s such that $s^2 \equiv m \pmod{N}$. Normally s is the same size as N, but Bleichenbacher showed how to bring this down to $\sqrt{N}$. The continued fraction algorithm (or the Euclidean algorithm) can be used to compute integers r and t with $|r|, |t| \leq \sqrt{N}$ such that $s \equiv r/t \pmod{N}$ (see Lemma 1 below). It then follows that $r^2 \equiv mt^2$

(mod N). But $|r| < \sqrt{N}$, so we can recover r from m and t by taking the integer square root of $mt^2 \bmod N$; and given r and t, it is trivial to recover $s \equiv r/t \pmod{N}$. Hence, we may replace s with t, which has half the size,

Now, given a reduced form (a, b) in $Cl(\Delta)$, we have

$$b^2 \equiv \Delta \pmod{a}$$

—a relation reminiscent of the Rabin signature verification equation. As in Bleichenbacher's signature compression, we can use the extended Euclidean algorithm to compute integers s, t such that $b \equiv s/t \pmod{a}$ and $|s|, |t| \leq \sqrt{a}$ (see Lemma 1 below). It then follows that

$$s^2 \equiv \Delta t^2 \pmod{a}\,.$$

We can now use t to encode the the coefficient b in half the space. Given a and t, we compute $\Delta t^2 \pmod{a}$, which is *equal as an integer* to s^2 because $0 \leq s < a$; so s can be recovered as the exact (positive) integer square root. This yields b, up to sign—thus, an additional sign bit is also required.

Lemma 1. *Given integers $a > b > 0$, the extended Euclidean algorithm applied to (a, b) will produce $s_i, t_i \leq \sqrt{a}$ at some stage i during the course of its execution.*

Proof. At each iteration of the extended Euclidean algorithm, we have integers s_i, u_i, and t_i such that

$$s_i = u_i a + t_i b$$

At step i in the algorithm, $|t_i| < |t_{i+1}|$ and $|s_i| > |s_{i+1}|$ [Gal12, Lemma 2.3.3]. Suppose at stage i we have $s_i \leq \sqrt{a}$ for the first time (that is, $s_{i-1} > \sqrt{a}$). Because the sequence of s_i is strictly decreasing, and $s_{-1} = a > \sqrt{a}$, such a stage will always exist. From [Gal12, Lemma 2.3.3] we know that $|s_{i-1} t_i| \leq a$, and recall that $s_{i-1} > \sqrt{a}$. This implies that $|t_i| < \frac{a}{\sqrt{a}} = \sqrt{a}$. Then we are done: we can return s_i and t_i. □

The same idea can be used for class groups. Given the pair (a, b) we compute the corresponding value r/t. A few special cases arise in this context, mostly related to the fact that a is not an RSA modulus:

1. If $r = \sqrt{a}$ then $r^2 \equiv 0 \pmod{a}$, so if the computation above gives 0 during decompression, then we assume we are in this case.

2. If $a = b$, then we exceptionally let $t = 0$ in the compressed form, and recognise this case during decompression. Note that $t = 0$ cannot arise in any other case.

3. If $\gcd(a, t) \neq 1$ then basic Bleichenbacher-style decompression fails. To fix this, we let $g = \gcd(a, t)$, $a' = a/g$, $t' = t/g$, and $b_0 = b \bmod g$. Given a', t', b_0, g, and a sign bit ϵ, we can compute $a = a'g$, $t = t'g$, and $s = \frac{1}{g}\sqrt{t^2\Delta \bmod a}$, using ϵ to choose the correct sign; then $b' = s'/t' \pmod{a}$, $r' = r/g$, and $b' = b \pmod{a'}$. The Chinese Remainder Theorem lets us recover $b \pmod{a}$ from $b \equiv b_0 \pmod{g}$ and $b \equiv b' \pmod{a'}$.

Algorithms 1 and 2 make the compression and decompression procedures completely explicit. Note that $\log_2 b_0 \leq \log_2 g$, so $\log_2 a' + \log_2 g = \log_2 a \approx \log_2 \sqrt{|\Delta|}$ and $\log_2 t' + \log_2 b_0 = \log_2 t' + \log_2 g = \log_2 t \approx \frac{1}{2}\log_2 \sqrt{|\Delta|}$, We have thus compressed the element (a, b) to a $\frac{3}{2}\log_2 |\Delta|$-bit representation, which is three-quarters of the size of (a, b).

6 The security of hyperelliptic Jacobians

In order for a group of unknown order to be useful in its named role, calculating the order of the group should be infeasible. In the case of hyperelliptic Jacobians, there are two relevant classes of algorithms: point-counting algorithms and discrete-log algorithms. Indeed, if the DLP can be efficiently solved, then the group order can also be efficiently computed: if we solve the DLP instance $xG = \mathcal{O}$, where G the generator of (the cryptogaphic subgroup of) J_C, then x is (a divisor of) the exponent of J_C.

Algorithm 1: Element compression for the quadratic imaginary class group $Cl(\Delta)$

```
Input: A reduced form $(a, b)$
Output: A compressed form $(a', t', g, b_0, \epsilon)$
1  $(s, u, t) \leftarrow$ PartialXGCD($|a|, |b|, \sqrt{|a|}$)        // Now $s = au + bt$ with $|s|$ and $|t| < \sqrt{|a|}$
2  if $b < 0$ then $t \leftarrow -t$
3  $g \leftarrow \gcd(a, t)$
4  $a' \leftarrow a/g$
5  if $a = b$ then
6      $t' \leftarrow 0$
7  else
8      $t' \leftarrow t/g$
9  $b_0 \leftarrow b \bmod g$
10 $\epsilon \leftarrow [b >= 0]$
11 return $(a', t', g, b_0, \epsilon)$
```

Algorithm 2: Element decompression for the quadratic imaginary class group $Cl(\Delta)$

```
Input: A compressed form $(a', t', g, b_0, \epsilon)$ and $\Delta$
Output: A reduced form $(a, b)$ and $\Delta$
1  $a \leftarrow g \cdot a'$
2  $t \leftarrow g \cdot t'$
3  if $t = 0$ then return $(a, a)$
4  $x \leftarrow t^2\Delta \mod a$
5  $s \leftarrow \sqrt{x}$                                   // Integer square root
6  $s' \leftarrow s/g$                                    // Exact integer division
7  $b' \leftarrow s' \cdot t^{-1} \pmod{a'}$
8  $b \leftarrow$ CRT($(b', a'), (b_0, g)$)
9  if $\epsilon = False$ then $b \leftarrow -b \mod a$
10 return $(a, b)$
```

6.1 Point-counting algorithms

Let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$, where $q = p^n$; we want to compute $\#J_C$. This is a classic problem (called "point counting") in algorithmic number theory, with many dedicated algorithms. The goal is to compute the zeta function of C, from which we immediately get $\#J_C$. Point-counting algorithms fall naturally into two broad classes: p-adic algorithms and ℓ-adic or Schoof-type algorithms.

The p-adic point-counting algorithms, most notably Kedlaya's algorithm [Ked01] and its descendents, generally compute the action of Frobenius on some p-adic cohomology group connected with C. Kedlaya's original algorithm computes the zeta function of C in time $\widetilde{O}(pg^4n^3)$. Harvey [Har07] has reduced the dependence in p to $p^{1/2}$. For fixed, or very small p (polynomial in n and g), this is polynomial-time; in practice, this is highly efficient for p into the thousands. However, as p grows larger, these algorithms become completely impractical. If we focus on Jacobians of curves C over $\mathbb{F}_{p^n}$ with p large—and especially, on curves over $\mathbb{F}_p$—then we can safely ignore the threat of any Kedlaya-style algorithm.

Schoof-type algorithms compute the characteristic polynomial of Frobenius on subgroups of the (generally irrational) ℓ-torsion for various ℓ using the division polynomials, before combining the results with the Chinese Remainder Theorem to compute the zeta function. Schoof's ground-breaking $\widetilde{O}(\log^5 q)$ algorithm [Sch85] was the first polynomial-time point-counting algorithm for elliptic curves. Its successor, the Schoof–Elkies–Atkin (SEA) algorithm has made elliptic-curve point counting a routine calculation. Pila gave a general extension to higher-dimensional abelian varieties [Pil90], which is polynomial time in p and n, but very badly exponential in g. As far as we know, this general algorithm has never been implemented.

Several Schoof-type algorithms for genus 2 have been implemented and analyzed, for example by Gaudry and Harley [GH00] and Gaudry and Schost [GS04]. This involves computing ℓ-division ideals and analysing action of Frobenius on them, which can become impractical for even moderately small ℓ. Pitcher's PhD thesis [Pit09] gave a Schoof-type point counting algorithm on genus 2 with complexity $O((\log q)^7)$. Gaudry and Schost [GS12] used an improved algorithm, with a mixture of Pitcher's approach and exponential BSGS algorithms, to find a curve of secure order over the 127-bit Mersenne prime field $\mathbb{F}_{2^{127}-1}$. In their experiments, they claimed around 1,000 CPU hours to compute the order of a random curve over this field.

This practical work has not been extended beyond genus 2, though some first steps have been made towards a practical algorithm for the very special class of genus-3 Jacobians with known and efficiently computable real multiplication in [AGS19b], following the analogous algorithm for genus 2 in [GKS11]. Abelard, Gaudry, and Spaenlehauer give some theoretical analysis and projected complexities for the general case in [AGS19a], but the main obstruction, even in genus 3, remains the complexity of division ideal manipulations: as stated in [GH00], "it does not appear possible to avoid manipulation of ideals."

For discrete-log-based cryptography we need Jacobians with known and secure (near-prime) orders, and some work has been done on generating Jacobians *with a known number of points*, using CM theory (see e.g. [GS12] and [HSS01]). In particular, Weng [Wen01, Wen03] discusses ways to generate hyperelliptic genus 2 and 3 curves with prescribed numbers of $\mathbb{F}_q$-rational points in their Jacobians. While this is all well and good for discrete-log-based cryptography, in our setting we need to trustlessly ensure that the group order is unknown to all parties. These curve-generation methods must therefore be avoided; instead, the curve should be generated in a nothing-up-my-sleeve type manner, as discussed in Section 8.

The upshot is that while point-counting for *fixed* genus $g > 2$ is polynomial-time in theory, so far it remains impractical—and even infeasible—in the real world.

6.2 Discrete logarithm algorithms

As we noted above, if we can solve DLP instances in J_C, then we can compute $\#J_C$. Suppose, then, that we want to solve the DLP in J_C, where C is a curve of genus g over $\mathbb{F}_q$. Gaudry et al. [GTTD07], and also Nagao [Nag04], present algorithms for small g running in time $\widetilde{O}(q^{2-2/g})$, improving on the $O(q^2)$ algorithm of [Gau00], and the single-large-prime algorithm of [Thé03].

This has better performance for genus 3 than square-root algorithms like Pollard's Rho algorithm, which has expected runtime in $\widetilde{O}(q^{3/2})$; but in genus 2, Pollard's Rho algorithm is more efficient, in $\widetilde{O}(q)$. Avanzi, Thériault, and Wang [ATW08] give further discussion the security in these cases. Smith [Smi09] gives a method of transferring the DLP from hyperelliptic to non-hyperelliptic genus-3 Jacobians that applies to 18.57% of genus 3 curves; Diem's algorithm [Die06] can then be used to solve the DLP in time $\widetilde{O}(q)$. Laine and Lauter [LL15] examine and improve on Diem's attack (including analysis of the logarithmic factors, which they estimate to be $O(\log^2 q)$), but the memory requirement for their attack is high at $\widetilde{O}(q^{3/4})$. Ways to generate genus-3 hyperelliptic curves that avoid these isogeny-based attacks are discussed in [Lai15].

As $g \to \infty$, there exist sub-exponential (in the group order) attacks on the DLP using index calculus methods (for example, [Eng02]); but these do not impact the case of fixed genus $g = 2$ and 3.

Previous work on generating hyperelliptic curves for cryptography focused on generating Jacobians whose orders have specific properties to prevent known DLP attacks. For example, the order should have a large prime factor, to avoid attacks such as Pohlig–Hellman; the largest prime factor should not divide $q^k - 1$ for small k, to avoid MOV-type attacks [FR94]; and the group order should be prime to $p = \mathrm{char}(\mathbb{F}_q)$ to avoid "anomalous curve" attacks [Rüc99]. Finally, to ensure that we do not weaken the difficulty of point counting using maps to subvarieties, the curve should have a simple Jacobian.

In the context of groups of unknown order, it is (by definition) not possible to know whether the Jacobian meets these conditions or not. Fortunately, the vulnerable group orders are extremely rare: a randomly generated hyperelliptic Jacobian will be simple, and will have a large prime dividing its order, with very high probability. The security of random ideal class groups as groups of unknown order depends on similar assumptions and heuristics [CL84].

6.3 The impact of Sutherland's algorithm

We can use the results summarized in Section 3 to give some success probabilities for computing Jacobian orders of randomly chosen genus-3 hyperelliptic curves over $\mathbb{F}_q$, given different running times.

For example, given $O(q)$ time (corresponding to $u = 3.0$), we would expect to find the group order with probability 0.4473. But even for $O(\sqrt{q})$ time ($u = 6.0$), we can expect to find the group order with probability at least 0.001092. This chance of success, while small at first glance, is still very worrying for cryptographic applications, because it means that at least 1 in every 1000 randomly generated curves would be vulnerable to this algorithm in $O(\sqrt{q})$ time. These generic group algorithms can be further optimised when specialised to Jacobians: we can exploit the fact that negation is effectively a free operation. We can also exploit the fact that even if a curve is not directly vulnerable to Sutherland's algorithm, it may be vulnerable through related curves such as its quadratic twist (this fact was used by Sutherland in [Sut09]). In order to secure against these algorithms and make the probability of a random curve having weak order exponentially small, the group order must be made subexponentially large.

We conclude that Sutherland's algorithm overshadows the other attacks discussed above when it comes to choosing concrete parameters for the unknown-order setting. It is conservative to assume an $\widetilde{O}(q)$ algorithm for finding the order of the group in the genus 3 and above. But to keep the probability of generating one with semismooth order which can be attacked with Sutherland's algorithm, q would need to be made subexponentially large. At these group orders, subexponential attacks on the DLP in higher genus become insignificant - even an $O(q)$ attack on the discrete logarithm in any genus would be irrelevant. So despite the fact that raising the genus beyond 3 may have previously been considered risky, we are forced into parameter choices such that this isn't the case, even for genus as large as 10 or more. By using parameters of this size, we also avoid all criticisms of Lee [Lee20].

6.4 Estimating concrete cryptogaphic parameters

With respect to the algorithms mentioned above, we shall now discuss choices of parameters for practical security levels. The practicality of the above point counting/discrete logarithm algorithms has certainly been debated, but we shall conservatively proceed assuming they can feasibly be used. The existence of algorithms with certain time-complexities is allowable if we choose the parameters appropriately in concrete situations. Despite the use of $\widetilde{O}$ complexity, for simplicity we will assume no logarithmic factors - which also gives a more conservative analysis.

We shall first begin by ignoring Sutherland's algorithm, and discuss only attacks on the group order and DLP as if the group order were not vulnerable to the primorial-steps algorithm. We shall then address this algorithm and propose parameters based on it. Choosing parameters $q \sim 2^{100}$ in the genus $g = 3$ case would appear sufficient to resist point counting on most curves. This would result in the order of the Jacobian of the curve being around 2^{300}. The practical results from [LL15] suggest at this field size, around 2^{113} field multiplications and $1.2 \cdot 10^{14}$ TB of memory would be required to mount their attack - even supposing the chosen genus 3 hyperelliptic curve could be mapped via [Smi09] to a non-hyperelliptic one. The algorithm by [GTTD07] would give time complexity $\widetilde{O}(2^{133})$.

In the genus 2 case, $q \sim 2^{128}$ will give only around 49 bits of security on paper, according to the $O(log^7(x))$ attack. Against such an algorithm, attaining 128-bit security would require an infeasibly large field order. But rather than ruling out the use of genus 2 curves in high security situations entirely, we stress that this is in the very worst case and ignores very large constant factors in the real runtime of such an attack. In practice the security is probably a lot better than this assumption, and there have not been any such large scale computations reported. But even (optimistically) assuming this case has lower security, it could still be useful for some applications.

Compared to class groups, to get a similar level of security against the sub-exponential $L_{|\Delta|}(1/2)$ algorithm, a much larger negative prime discriminant of around 1208 bits would be required according to [HM00]. To compare performance in practice, we generated random hyperelliptic curves of genus 3 over the finite field $\mathbb{F}_p$, $p = 2^{101} - 69$, and selected a random point as the generator point. We then ran basic timing experiments in `MAGMA` for group operations. For the class groups, we used the open source rust implementation by KZen

Networks [KZe]. We used form groups of 996-bit negative prime discriminants in our tests (even smaller than above) and randomly generated reduced forms as the generators. We found that for 100,000 group element additions (point additions in the Jacobian, composition-then-reductions of quadratic forms), the hyperelliptic curve implementation in `MAGMA` performed around 28 times faster on average than the quadratic form test (despite the security level difference). Of course, this is highly implementation dependent and optimisations may be possible in both cases, but the result is in line with our claim that using a genus 3 hyperelliptic curve is more practically efficient.

Because elements of the Jacobian are represented as a pair of polynomials $\langle u, v\rangle$ such that $\deg v < \deg u \leq g$, the element can be stored concretely as just 6 coefficients of polynomials - a total of 6 elements of $\mathbb{F}_q$. This can be further reduced down to just 3 elements and 3 extra bits using the point compression of Hess, Seroussi, and Smart [HSS01]. In that case that $q \sim 2^{100}$, this means elements can be stored in around 303 bits. On the other hand, representing a form or ideal class group can be done using two integers. A reduced form implies that $a \leq \sqrt{|\Delta|/3}$ so for $\Delta \sim 2^{1208}$ we have that $a \sim 2^{603}$ - so that elements are representable in around 1206 bits. The compression from Section 5 would reduce this to around 905 bits. Hence we also propose that Jacobian elements are more compactly represented for the same level of security.

We now come to concrete parameter choice suggestions for the bound s in Assumption 1. Because there has been no previous work done on Schoof-type algorithms in genus 3 (due to the complexity of using division ideals), we shall instead inspect progress in genus 2. Gaudry and Schost [GS12] in their 2012 work calculated modular information for primes up to $\ell = 31$. Memory restrictions prevented higher primes being used at that time. This bound gave them around 30 bits of information when searching in a 127 bit field, giving a cost of the BSGS step of around $O(2^{49})$. While no practical implementation beyond this has been presented in the literature, Abelard's PhD thesis [Abe18] discusses the feasibility of extending this work to higher primes. Due to time complexity growing as $\widetilde{O}(\ell^6 \log q)$ and memory as $\widetilde{O}(\ell^4 \log q)$, it is in fact the running time which becomes more of an issue than the memory. He states that in a 192-bit field, computation with $\ell = 53$ could take around 1000 CPU days and would still result in a search space of $\approx 2^{95}$ in the collision step of the algorithm. Given the current understanding and previous work in genus 2, and considering the inherent difficulty in performing similar work in genus 3, a bound such as $s = 60$ should thus be sufficient. At this size, the memory and time constraints should be large enough to make computation infeasible. This could even be raised to $s = 70$ or higher depending on the use-case.

Let us take $s = 60$ in order to make some concrete efficiency claims. The bit length of $N = 60!$ is 273 bits. Consider the modified PoE from Section 8. The only performance impact the choice of s has is in the final verification step, checking $[N]([\ell]Q + [r]U) = W$. We observe that the choice of prime ℓ respects the security parameter, so would perhaps be around the order of 100 or more bits, and r would be similar. So the additional cost to the verifier is comparable to the existing cost of operation anyway. Thus we claim that this should still be faster than using ideal class groups.

Coming now to Sutherland's algorithms [Sut07], we realise that we cannot safely assume that the group order of a random hyperelliptic curve would be secure against this attack. To keep the probability of generating a weak curve exponentially small in the security parameter λ, the group order must be made much larger - $\sim 2^{3328}$ for $\lambda = 128$. At these group orders, the attacks above and any subexponential attacks on the DLP in higher genus become insignificant - even an $O(q)$ attack on the discrete logarithm in any genus would be irrelevant. So despite the fact that raising the genus beyond 3 may not considered risky, we are forced into parameter choices that this isn't the case for genus even as large as 10 or more.

Specifically, for 128-bit security, we estimate that $u = 26$ should give $G(1/u, 2/u) \approx 2^{-128}$ (from Figure 1). The group order requirement arises from $M = 2^{128*26} = 2^{3328}$ (as in Section 3), which would require a field order q such that $\log q \approx 1109$. Because the primorial-steps algorithm by Sutherland is a generic group algorithm, this requirement on the group order applies to **any** group of unknown order, and is much larger than has been previously recommended in the literature. We also discuss the possibility of using multiple different group instances in parallel below, in order to lower the group order and retain the negligible probability.

7 Elements of known order in hyperelliptic Jacobians

We now consider the problem of constructing points of known order in a hyperelliptic Jacobian of unknown order. This will give us an idea of how to work with Jacobians in situations where Wesolowski's adaptive root assumption is required.

7.1 Constructing points of known order

If $E/\mathbb{F}_q$ is an elliptic curve, then whether or not of the order of $E(\mathbb{F}_q)$ is known, we can try to compute elements of small order ℓ by finding roots of the division polynomial Ψ_ℓ for E. In the same way, we can try to construct points of small, known order on hyperelliptic Jacobians using their division ideals, hence invalidating the adaptive root assumption.

However, if we assume (as we did in Section 6.1) that there exists no feasible Schoof-type algorithm for counting points on genus 3 curves, then we implicitly assume that it becomes infeasible to construct points of order larger than some bound. More formally, we have the following assumption:

Assumption 1. *For each integer $g > 0$, there exists an integer $s(g)$ such that for sufficiently large prime p, it is infeasible to compute roots of the ℓ-division polynomials for a random hyperelliptic curve*[1] *of genus g over $\mathbb{F}_p$ if $\ell > s(g)$.*

Given Assumption 1, we can maintain the adaptive root assumption if we use the subgroup

$$[S]J_C(\mathbb{F}_p) = \{[S]P \mid P \in J_C(\mathbb{F}_p)\}\,,$$

where S is a smooth integer chosen to kill off all points of small order. We can take $S = s!$ with $s > s(g)$ for some very conservative estimate of $s(g)$.

The issue now is that given a point Q in $J_C(\mathbb{F}_p)$, testing subgroup membership $Q \in [S]J_C(\mathbb{F}_p)$ is not easy. However, the original point Q is effectively a proof that $[S]Q$ is in $[S]J_C$, because this can be easily verified; so Q should be sent instead of $[S]Q$ in cryptographic protocols, and the verifier can perform the multiplication by S themselves. See Section 8 for an illustrative example in the case of accumulators.

Another important, related observation is that computing repeated divisions by 2 in J_C allows the construction of points of order 2^k for arbitrarily large k. Since 2^k is coprime to all odd primes ℓ, this would allow a malicious prover to easily find ℓ-th prime roots for these points (that is, given a point Q, find P such that $[\ell]P = Q$). But repeated division by 2 in $J_C(\mathbb{F}_p)$ requires the repeated extraction of square roots in $\mathbb{F}_p$, which quickly requires repeated quadratic field extensions. Using hyperelliptic curves in the form $y^2 = f(x)$ with $f(x)$ irreducible ensures that the required square roots do not exist in $\mathbb{F}_p$. But similarly, repeated powers of other small primes > 2 might still be calculated. Using the group $[S]J_C$ will kill off powers of these low primes dividing the group order. It could also be possible to simply test for these repeated divisions during the curve generation procedure, allowing parties to check for small factors of the group order—and then kill those off with a more tailored choice of S above. It is a potential interesting open problem to generate an easily verifiable proof that a Jacobian does not have any points of low order.

It is worth noting that the impact of finding small-order points is highly domain-specific. For example, in the VDF of [Wes19, BBF18], even if points of known order can be found, forging a false PoE still requires knowing the true result of the exponentiation—and hence stil requires computing the output of the VDF. Relying on a PoE would thus break the requirement that the VDF output is unique, but it would still provide assurance of the delay. In the case of accumulators, we need the strong RSA assumption rather than the adaptive root assumption: it should be hard to find *chosen* prime roots of an element (recall that the membership witness of ℓ in A is the ℓ-th root of A). This case can be addressed differently, by simply disallowing the accumulation of small primes $\ell < s(g)$. Finding ℓ-th roots with $\ell > s(g)$ is hard by Assumption 1 above, so here we do not require the use of $[S]J_C$.

[1] It is important to work with random curves in Assumption 1, since it is easy to construct special curves whose Jacobians have divisors of known order. For example: for any odd prime ℓ, the divisor $(0, 1) - \infty$ represents a nontrivial point of order ℓ in the Jacobian of the curve $y^2 = x^\ell + 1$, because the principal divisor of the function $y - 1$ is $(y - 1) = \ell(0, 1) - \ell\infty$.

Using $[S]J_C$ in place of J_C has an impact on efficiency, due to the extra scalar multiplications required. This impact is highly protocol-dependent, but in most cases only a few extra multiplications should be needed. For example, in the PoE protocol of Section 8.2, the verifier only needs to perform one extra multiplication-by-S during verification when working in $[S]J_C$ instead of J_C. We suggest that this is still efficient enough for practical use.

7.2 Testing small divisors of orders with the Tate pairing

Let J_C be the Jacobian of a hyperelliptic curve C over F_q, let ℓ be a prime (coprime to q), and let k be a positive integer such that $\ell \mid q^k - 1$, The reduced Tate pairing is a bilinear mapping

$$t_\ell : J_C[\ell] \times J_C/\ell J_C \to \mu_\ell \,,$$

where μ_ℓ is the group of ℓ-th roots of unity in $\mathbb{F}_{q^k}^\times$. (A good reference for the background of pairings on hyperelliptic curves is [GHV07].) If we can find points of known small order ℓ, then the Tate pairing can be used to learn information about the ℓ-divisibility of the order of other points.

Suppose we can find a point Q in $J_C(\mathbb{F}_{q^k})$ of small known prime order ℓ. Then for any point P in $J_C(\mathbb{F}_q)$, we can efficiently compute $t_\ell(Q, P)$ in μ_ℓ (assuming k is only polynomially large in $\log q$). Now, if $\ell \nmid \mathrm{Ord}(P)$, then $P = \ell P'$ for some P', so $t_\ell(Q, P) = 1$ for every Q in $J[\ell]$. By the contrapositive, if $t_\ell(Q, P) \neq 1$ for some Q in $J[\ell]$, then ℓ divides the order of P.

Unfortunately, the converse is not so simple: $t_\ell(Q, P) = 1$ for a single point Q of order ℓ is not sufficient to imply that the order of P is coprime to ℓ. Instead, it must be shown that $t_\ell(Q, P) = 1$ for *all* Q in $J_C[\ell]$. Thus, we require a basis $\{Q_1, Q_2\}$ of $J_C[\ell]$ which we can test: if $t_\ell(Q_i, P) = 1$ for $i = 1$ and 2, then the bilinearity of the Tate pairing implies $t_\ell(Q, P)$ for all Q in $J_C[\ell]$, and hence that $\gcd(|P|, \ell) = 1$.

Finally, we observe that the coordinates of Q and the value $t_\ell(Q, P)$ are in $\mathbb{F}_{q^k}$, an extension whose degree blows up with ℓ, so this approach is only useful for very small ℓ. We assert that Assumption 1 makes this an infeasible approach to learn any significant amount of information about the orders of random points in $J_C(\mathbb{F}_q)$, or any information at all in $[S]J_C$ for well-chosen S.

8 Constructions using hyperelliptic curves

We claim that hyperelliptic curves offer a more efficient alternative to class groups in situations where a group of unknown order is required. In this section we consider the construction of hyperelliptic Jacobians for this use-case, with a focus on applications to accumulators.

8.1 Generating hyperelliptic Jacobians of unknown order

After choosing an appropriate genus g and a random prime p of size determined by the security parameter (as discussed in Section 6.4), we need to generate a secure hyperelliptic curve $C : y^2 = f(x)$ over $\mathbb{F}_p$. To select C, we let f be a random irreducible polynomial of degree $2g+1$ over $\mathbb{F}_p$. Ensuring that f is irreducible over $\mathbb{F}_p$ guarantees that J_C has no points of order 2. As we saw in Section 7.1, it may be possible to construct points of small odd order; we could try this for a few small primes ℓ to eliminate C with a small factor in $\#J_C$, but this makes no significant difference to the probability of semismoothness of $\#J_C$. Indeed, our simulations of semismoothness probabilities showed that rejecting random group orders divisible by the first few primes decreased the semismoothness probability by less than a factor of 2.

As in Section 6.2, We impose some simple conditions on C to avoid well-known weak cases. First, J_C is a simple abelian variety; a randomly chosen C will ensure this with overwhelming probability (generic Jacobians are absolutely simple), but we should still be careful to ensure that C does not cover some other curve D (that is, that there is no morphism $C \to C'$ with C' not isomorphic to C or the projective line; for example, C' an elliptic curve), since then J_D would be a nontrivial abelian subvariety of J_C.

We must also exclude curves whose Jacobians have special endomorphism structures, such as the families with efficiently-computable real multiplication exploited in [AGS19b]. Again, a randomly chosen C will land outside these special classes of curves with overwhelming probability, since they form positive-codimensional subspaces of the moduli space. Recent work of Thakur [Tha20] discusses potential classes of curves to avoid.

Having chosen C, we also need an element P of J_C to serve as a generator for the group of unknown order. This can be chosen in a similar way, choosing random coefficients for $u(x)$ as the first polynomial in the Mumford representation $\langle u, v\rangle$ of P, before solving for v such that $v^2 = f \pmod{u}$ (recovering v can be done in the same way as from the compressed form of a divisor [HSS01]); if no such v exists, we try another u. Of course, the order of P is also unknown, so we cannot know whether P generates the full group or a subgroup. But as we will see below, there is a high probability that a randomly selected P will not have a small order—so it will at least generate a large-order subgroup of J_C.

To ensure that not even the constructor of C curve knows $\#J_C$, and that C and P were indeed generated randomly, we suggest that f and u be chosen by deterministic "nothing up my sleeve"-type process. For example, the coefficients of f might be taken from the hash of a certain string. Suppose this process were manipulated by testing multiple different NUMS "seeds", and testing each for a weakness with complexity 2^n for some n. If the probability of encountering a weak curve among random curves is p, then a malicious actor would have to do around $\frac{1}{p}2^n$ work to find such a C. A skeptical verifier, on the other hand, must only test the proposed C just once with just 2^n work in order to detect cheating. So the imbalance of work required to cheat versus to verify is a deterrent for attackers, regardless of the weakness in question. In general, a random curve is in an approximately random isogeny class, and a random isogeny class should be not vulnerable to any of the attacks with very high probability.

Now the order of the Jacobian J_C (and the subgroup generated by P) cannot feasibly be computed, not even by the party who constructed the curve: we have achieved trustless setup. This group can then be used in cryptographic constructions including accumulators and VDFs. Overall, the generation of a new hyperelliptic curve is relatively cheap. Therefore, just as in the case of class groups, it should be feasible to generate a new group of unknown order for each new instance of an accumulator or VDF if desired.

8.2 Wesolowski's Proof of Exponentiation

Given Assumption 1, constructions such as accumulators that rely on the adaptive root assumption should use the group $[S]J_C$. The operation of the accumulator in this group is standard, but some protocols may need modification—for example, proofs may require an extra check that an element is indeed in the group.

To give a specific example of this, we consider Wesolowski's Proof of Exponentiation (PoE) [Wes19]. We expect that other protocols using the adaptive root assumption can be modified in a similar way.

We begin the PoE protocol with input $U \in J_C$, $W \in [S]J_C$, and $x \in \mathbb{Z}$. The claim to be proven is that $[x][S]U = W$ in $[S]J_C$. The protocol proceeds as follows:

1. Verifier selects a random prime $\ell > s$ from Primes(λ) and sends ℓ to the prover.
2. Prover computes $q = \lfloor x/\ell \rfloor$, computes $Q = [q]U$ in J_C, and sends Q to the verifier.
3. The verifier computes $r = x \bmod \ell$, and accepts if Q is in J_C and $[S]([\ell]Q + [r]U) = W$.

This protocol is compatible with Assumption 1. Specifically, given a valid proof of $[x][S]U = W$, in order to falsely prove $[x][S]U = W + P$, the prover must compute $[1/\ell]P$ for whichever ℓ is chosen by the verifier. This may be possible if the order of P is known, or if it is feasible to compute the division ideal—but this is infeasible under Assumption 1, because $\ell > s$.

9 Parallel instances

Even if the accidental random selection of a weak group might highly improbable, it may not be cryptographically improbable. But rather than increasing the group size, we can try using multiple instances in parallel: the chance that all of the instances are weak decreases exponentially in the number of instances. Specifically, if the probability of selecting a curve with less than λ-bit security is 2^{-n}, then we require k

instances such that $2^{-nk} \leq 2^{-\lambda}$. For example, if we take 1665-bit discriminants, then for ≈ 128-bit security it should suffice to choose $k = 9$ instances, because the chance that all 9 are weak would be 2^{-126}.

Naturally, when computing with k instances the computation time (in serial, although this is parallelisable) and storage requirements scale linearly with k. Recall that class group elements can be represented using a pair of integers (a, b). A reduced form implies that $a \leq \sqrt{|\Delta|/3}$, so elements are representable with around two bits less than the size of Δ. We discussed in Section 5 the possibility of compressing class group elements to around 3/4 of their original size. With this, k instances would require $\sim \frac{3}{4}k \log |\Delta|$ bits. Since Δ has roughly twice the bitlength of the group order bound M, we require $\frac{3}{2}k \log M$ space. Above, this would be around 11.2kB. Minimal k is thus most efficient in terms of storage requirement for element representations. However the efficiency of the group law computation decreases as the group order increases, so there may be a tradeoff between space and time-efficiency when choosing k.

We turn now to the security offered by genus-3 hyperelliptic Jacobians, when the order of the field is $q \sim 2^{350}$. The Jacobian order is around 2^{1050}, so the chance of a curve weak to Sutherland's algorithm is around 2^{-18}. In this case we would need $k = 7$ instances to have probability 2^{-126} of a curve weaker than 128-bit occurring. Recall that to store an element of a genus-3 Jacobian in compressed Mumford form, we need 3 field elements and 3 extra bits. The group order has around 3 times as many bits as q, so the space required would be approximately $k \log M$–0around half that of the class group requirement for any particular security level. So at this security would require just under 7.4kB. This is around two-thirds the size required for a class group at equivalent security level above. This difference primarily lies in the compression of the Jacobian points being more efficient than the class group compression from Section 5.

Table 2 shows the group order required for 128-bit security with different numbers k of parallel group instances. It also shows the space requirement for elements of both class groups and Jacobians. It can be clearly seen that a single instance is still more space-efficient than multiple instances, and that the storage requirements of Jacobian instances are more efficient than class groups (after compression).

Table 2: Parameter sizes and element storage requirements as number of parallel instances (k) increases, for 128-bit security

k	occurence	$\sim u$	$\log M$	$\log \Delta$	$\log q$	Cl storage (kB)	Jac storage (kB)
1	$10^{-38.5}$	26	3328	6656	1109	5	3.3
2	$10^{-19.3}$	16	2048	4096	683	6.2	4.1
3	$10^{-12.8}$	12	1536	3072	512	6.9	4.6
4	$10^{-9.6}$	10	1280	2560	427	7.7	5.1

At these group sizes, even an $O(q)$ algorithm to solve the discrete logarithm would be no threat. In genus 10, we could take a 333-bit q. Thus we can consider even higher genus curves safely, despite them being notionally weaker discrete-log-based applications. The storage requirement does not decrease, as the compressed Mumford representation in genus g requires $\approx g \log q + g \sim \log M + g \sim \log M$ bits. But the smaller size of q could possibly improve efficiency of operations. We mention this here for completeness, but we will not follow this line further in this article.

10 Conclusion

Due to the importance of groups of unknown order in many recent constructions - and especially those requiring trustless setup - an improvement to the practical efficiency of these groups is a useful advancement. Upon reviewing the security of recommended parameter choices for class groups of unknown order, we find that the parameters used in recent work are significantly smaller than required for constructions to meet their desired security levels. We thus propose that any group of unknown order must have at least a $\sim$ 3300-

bit order to provide 128-bit security. This would require the discriminant of a class groups $\Delta \sim 6656$ bits. We also propose the use of Bleichenbacher-type compression on class group elements to reduce the storage space. We then propose the use of Jacobian groups of genus 3 hyperelliptic curves as more efficient, practical alternatives in the construction of unknown-order groups, where the aforementioned group order would require a field of order $q \sim 1109$ bits. We hope that this work will motivate further investigation on the practical use of Jacobians as groups of unknown order, or otherwise improvements to point-counting algorithms on such Jacobians.

References

[Abe18] Simon Abelard. *Counting points on hyperelliptic curves in large characteristic: algorithms and complexity.* PhD thesis, 2018.

[AGS19a] Simon Abelard, Perrick Gaudry, and Pierre-Jean Spaenlehauer. Improved complexity bounds for counting points on hyperelliptic curves. *Foundations of Computational Mathematics*, 19(3):591–621, 2019.

[AGS19b] Simon Abelard, Pierrick Gaudry, and Pierre-Jean Spaenlehauer. Counting points on genus-3 hyperelliptic curves with explicit real multiplication. *The Open Book Series*, 2(1):1–19, 2019.

[ATW08] Roberto Avanzi, Nicolas Thériault, and Zheng Wang. Rethinking low genus hyperelliptic jacobian arithmetic over binary fields: Interplay of field arithmetic and explicit formulae. *Journal of Mathematical Cryptology*, 2(3):227–255, 2008.

[BBF18] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. `https://eprint.iacr.org/2018/712`.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *Advances in Cryptology – CRYPTO 2019*, pages 561–586, 2019.

[BFS19] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. Cryptology ePrint Archive, Report 2019/1229, 2019. `https://eprint.iacr.org/2019/1229`.

[BH01] Johannes Buchmann and Safuat Hamdy. A survey on IQ cryptography. In *In Proceedings of Public Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

[Ble04] Daniel Bleichenbacher. Compressing Rabin signatures. In *CT-RSA 2004*, volume 2964 of *LNCS*, pages 126–128, 2004.

[BP96] Eric Bach and René Peralta. Asymptotic semismoothness probabilities. *Mathematics of computation*, 65(216):1701–1715, 1996.

[Bre00] Richard P. Brent. Public key cryptography with a group of unknown order. Technical report, 2000.

[BW88] Johannes Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, Jun 1988.

[Can87] David G. Cantor. Computing in the jacobian of a hyperelliptic curve. *Mathematics of computation*, 48(177):95–101, 1987.

[Can94] David G. Cantor. On the analogue of the division polynomials for hyperelliptic curves. *Journal für die reine und angewandte Mathematik*, 447:91–146, 1994.

[CFA$^+$05] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography.* Chapman and Hall/CRC, 2005.

[CL84] Henri Cohen and Hendrik W. Lenstra. Heuristics on class groups of number fields. In *Number Theory Noordwijkerhout 1983*, pages 33–62. Springer, 1984.

[CL12] Craig Costello and Kristin Lauter. Group law computations on jacobians of hyperelliptic curves. In *Proceedings of the 18th International Conference on Selected Areas in Cryptography*, SAC'11, pages 92–117. Springer-Verlag, 2012.

[Coh10] Henri Cohen. *A Course in Computational Algebraic Number Theory.* Springer Publishing Company, Incorporated, 2010.

[Cox89] D.A. Cox. *Primes of the Form x^2+ny^2: Fermat, Class Field Theory, and Complex Multiplication.* Monographs and textbooks in pure and applied mathematics. Wiley, 1989.

[Die06] Claus Diem. An index calculus algorithm for plane curves of small degree. In *Algorithmic Number Theory*, pages 543–557. Springer Berlin Heidelberg, 2006.

[Eng02] Andreas Enge. Computing discrete logarithms in high-genus hyperelliptic jacobians in provably subexponential time. *Math. Comput.*, 71(238):729–742, April 2002.

[FR94] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comput.*, 62(206):865–874, April 1994.

[FWG07] Xinxin Fan, Thomas Wollinger, and Guang Gong. Efficient explicit formulae for genus 3 hyperelliptic curve cryptosystems over binary fields. *IET Information Security*, 1(2):65–81, 2007.

[Gal12] Steven D. Galbraith. *Mathematics of Public Key Cryptography.* Cambridge University Press, New York, NY, USA, 1st edition, 2012.

[Gau00] Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In *Advances in Cryptology — EUROCRYPT 2000*, pages 19–34. Springer Berlin Heidelberg, 2000.

[GH00] Pierrick Gaudry and Robert Harley. Counting points on hyperelliptic curves over finite fields. In *Algorithmic Number Theory*, pages 313–332. Springer Berlin Heidelberg, 2000.

[GHV07] Steven D. Galbraith, Florian Hess, and Frederik Vercauteren. Hyperelliptic pairings. In *Pairing-Based Cryptography – Pairing 2007*, pages 108–131. Springer Berlin Heidelberg, 2007.

[GKS11] Pierrick Gaudry, David Kohel, and Benjamin Smith. Counting points on genus 2 curves with real multiplication. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology—ASIACRYPT 2011 (Seoul, South Korea)*, volume 7073 of *Lecture Notes in Computer Science*, pages 504–519. Springer, Heidelberg, 2011.

[GS04] Pierrick Gaudry and Éric Schost. Construction of secure random curves of genus 2 over prime fields. In *Advances in Cryptology - EUROCRYPT 2004*, pages 239–256. Springer Berlin Heidelberg, 2004.

[GS12] Pierrick Gaudry and Éric Schost. Genus 2 point counting over prime fields. *Journal of Symbolic Computation*, 47(4):368–400, 2012.

[GTTD07] 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.

[Har07] David Harvey. Kedlaya's algorithm in larger characteristic. *International Mathematics Research Notices*, 2007, 01 2007. rnm095.

[HM89] James Lee Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. volume 2, pages 837–850, 1989.

[HM00] Safuat Hamdy and Bodo Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In *Advances in Cryptology — ASIACRYPT 2000*, pages 234–247. Springer Berlin Heidelberg, 2000.

[HSS01] Florian Hess, Gadiel Seroussi, and Nigel P. Smart. Two topics in hyperelliptic cryptography. In *International Workshop on Selected Areas in Cryptography*, pages 181–189, 2001.

[Ked01] Kiran S Kedlaya. Counting points on hyperelliptic curves using monsky-washnitzer cohomology. *arXiv preprint math/0105031*, 2001.

[Kob89] Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, Oct 1989.

[KZe] KZen Networks. Class groups. `https://github.com/KZen-networks/class-groups/`.

[Lai15] Kim H. M. Laine. *Security of Genus 3 Curves in Cryptography.* PhD thesis, University of California, Berkeley, 2015.

[Lan05] Tanja Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 15(5):295–328, Feb 2005.

[Lee20] Jonathan Lee. The security of groups of unknown order based on jacobians of hyperelliptic curves. Cryptology ePrint Archive, Report 2020/289, 2020. `https://eprint.iacr.org/2020/289`.

[LL15] Kim Laine and Kristin Lauter. Time-memory trade-offs for index calculus in genus 3. *Journal of Mathematical Cryptology*, 9(2):95–114, 2015.

[Mum07] David Mumford. *Tata Lectures on Theta II*, pages 207–213. 01 2007.

[MWZ96] Alfred Menezes, Yi-hong Wu, and Robert J. Zuccherato. *An Elementary Introduction to Hyperelliptic Curves.* Faculty of Mathematics, University of Waterloo, 1996.

[Nag04] Koh-ichi Nagao. Improvement of thériault algorithm of index calculus for jacobian of hyperelliptic curves of small genus. Cryptology ePrint Archive, Report 2004/161, 2004. `https://eprint.iacr.org/2004/161`.

[Pil90] Jonathan Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Mathematics of Computation*, 55(192):745–763, 1990.

[Pit09] Nicole L. Pitcher. *Efficient point-counting on genus-2 hyperelliptic curves.* PhD thesis, 2009.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. 1996.

[Rüc99] Hans-Georg Rück. On the discrete logarithm in the divisor class group of curves. *Math. Comput.*, 68(226):805–806, April 1999.

[Sch85] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of computation*, 44(170):483–494, 1985.

[Smi09] Benjamin Smith. Isogenies and the discrete logarithm problem in jacobians of genus 3 hyperelliptic curves,. *Journal of Cryptology*, 22(4):505–529, Oct 2009.

[Sut07] Andrew V Sutherland. *Order computations in generic groups.* PhD thesis, Massachusetts Institute of Technology, 2007.

[Sut09] Andrew Sutherland. A generic approach to searching for jacobians. *Mathematics of Computation*, 78(265):485–507, 2009.

[Tha20] Steve Thakur. Constructing hidden order groups using genus three jacobians. Cryptology ePrint Archive, Report 2020/348, 2020. `https://eprint.iacr.org/2020/348`.

[Thé03] Nicolas Thériault. Index calculus attack for hyperelliptic curves of small genus. In *Advances in Cryptology - ASIACRYPT 2003*, pages 75–92. Springer Berlin Heidelberg, 2003.

[Wen01] Annegret Weng. A class of hyperelliptic CM-curves of genus three. *Journal of the Ramanujan Mathematical Society*, 16, 01 2001.

[Wen03] Annegret Weng. Constructing hyperelliptic curves of genus 2 suitable for cryptography. *Mathematics of Computation*, 72(241):435–458, 2003.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology – EUROCRYPT 2019*, pages 379–407, 2019.