Trustless unknown-order groups

Samuel Dobson[1], Steven Galbraith[1], Benjamin Smith[2]

[1]University of Auckland, New Zealand
[2]Inria and Laboratoire d'Informatique (LIX), CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Abstract *Groups whose order is computationally hard to compute have important applications including time-lock puzzles, verifiable delay functions, and accumulators. Many applications require trustless setup: that is, not even the group's constructor knows its order. We argue that the impact of Sutherland's generic group-order algorithm has been overlooked in this context, and that current parameters do not meet claimed security levels. We propose updated parameters, and a model for security levels capturing the subtlety of trustless setup. The most popular trustless unknown-order group candidates are ideal class groups of imaginary quadratic fields; we show how to compress class-group elements from $\approx 2\log_2(N)$ to $\approx \frac{3}{2}\log_2(N)$ bits, where N is the order. Finally, we analyse Brent's proposal of Jacobians of hyperelliptic curves as unknown-order groups. Counter-intuitively, while polynomial-time order-computation algorithms for hyperelliptic Jacobians exist in theory, we conjecture that genus-3 Jacobians offer shorter keylengths than class groups in practice.*

Keywords: Unknown order groups, ideal class groups, hyperelliptic curves
2010 Mathematics Subject Classification: 94A60, 11Y40

1 INTRODUCTION

Interest in groups of unknown order has been fuelled in recent years by applications such as delay functions [11], accumulators [12], and zero-knowledge proofs of knowledge [16]. As the name suggests, a group G has *unknown order* if it has a compact representation, but it is infeasible for anyone to compute the order of G efficiently 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. Some use-cases may require additional properties, such as the *low order* or the *adaptive root assumptions*. In order to be useful, group operations in G should be efficiently computable; elements of G should have a compact representation; and it should be possible to efficiently sample random elements of G.

Previously, there have been two proposals for concrete unknown-order groups: RSA groups [52], and ideal class groups of imaginary quadratic fields [46, 15]. Brent briefly suggested hyperelliptic Jacobians as unknown-order groups [13]; but unlike RSA and class groups, Jacobians have received little further attention.

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 $(\mathbb{Z}/N\mathbb{Z})^\times$ is equivalent to factoring N. A trusted party can efficiently generate an RSA modulus that resists all known attacks (including Sutherland's algorithm). Sander [54] gave an algorithm to trustlessly generate a modulus N such that (with very high probability) N has two large factors—he calls this an RSA-UFO (unknown factorisation object). However, to match even the lower security of 1024-bit RSA moduli, RSA-UFOs need "bit length (much) greater then 40,000 bits"; this is far too large to be efficient in most unknown-order group applications.

Class groups, on the other hand, can be generated without trusted setup, and so have received a lot of recent attention (see e.g. [46, 64, 12]). Buchmann and Hamdy [14] suggested that 1665-bit discriminants ($\approx$ 833-bit orders) provide security equivalent to 3072-bit RSA (i.e., 128-bit security); more recently, Biasse, Jacobson Jr., and Silvester [9] claim that 1827-bit discriminants ($\approx$ 914-bit orders) are required to reach the same security level.

But the usual notions of security level are not appropriate when evaluating class-group security for applications such as accumulators, where the group is fixed. The computational assumptions underlying security are not defined for a fixed group, and there is no random self-reduction to show that all instances have the same security. We argue that much larger group sizes are needed for secure unknown-order groups in applications where the group is fixed for many users and used for a long period of time.

Precisely, we propose a new security model for unknown-order groups, depending on two parameters (λ, ρ).

Definition 1. *Let* `Gen` *be an algorithm outputting groups. We say that* `Gen` *reaches security level (λ, ρ) if with probability $1 - 1/2^\rho$ over the outputs of* `Gen`*, any algorithm $\mathcal{A}$ given an output G of* `Gen` *requires at least 2^λ bit operations to succeed in computing #G with probability close to 1.*

A similar concept of security is implicit in [4], which considers the security of cryptosystems depending on a prime system parameter p provided by a possibly malicious party, when in practice the users only verify the primality of p—and thus ensure the security level of the system—up to a certain probability (if at all). One

recommendation of [4] is that users "ensure that composite numbers are wrongly identified as being prime with probability at most 2^{-128}", corresponding to $\rho = 128$.

In our context, the probabilistic nature of security is not due to malicious parties, or unreliable verification, but rather to a fundamental mathematical fact: the distribution of random abelian group orders. Our security definition is motivated by Sutherland's generic group order algorithm [58], whose runtime depends on the (unknown) order itself, rather than the supposed size of the order.

The relevance of Sutherland's algorithm to cryptographic unknown-order groups seems to have been overlooked, but in §3.2 we show that it attacks some class groups with parameters that are widely considered secure. For example: Sutherland's algorithm can compute the order of a class group with 1827-bit discriminant (i.e., 914-bit group order) in $\approx 2^{59}$ operations with probability $\approx 2^{-58}$, or in $\approx 2^{114}$ operations with probability $\approx 2^{-20}$. The problem is that among class groups with prime discriminants of a given size, there is a set of weak instances *depending on the order*. A randomly-generated group is only vulnerable with small probability; but since the order is unknown, we cannot check for vulnerability without simply attempting to run the algorithm for the given time (in contrast, in the RSA setting with trusted setup, the group order is known to its generator, who can thus easily choose a group that is not vulnerable). For these reasons, we consider that 1827-bit discriminants (and even the 2048-bit discriminants suggested in [12, 16]) do not meet the requirements for 128-bit security.

We propose new group sizes in response, depending on (λ, ρ). A paranoid choice, $(\lambda, \rho) = (128, 128)$, requires group sizes of around 3400 bits. A more realistic (but still cautious) choice, $(\lambda, \rho) = (128, 55)$, requires 1900-bit group sizes (and so 3800-bit discriminants; more than double the previous suggestion). Table 2 gives sizes of group orders for various combinations of (λ, ρ). An alternative would be to use multiple smaller groups in parallel, however we believe this approach is less efficient than working in a single larger group.

Our second major result is a more compact representation of class group elements. Inspired by a signature-compression method of Bleichenbacher [10], in §3.3 we compress elements of class groups to $3/4$ the size of the usual representation—a particularly welcome saving in the light of our updated, much larger class group parameters.

Our third and more speculative contribution, in §4, is an analysis of Brent's proposal of subgroups of Jacobians of hyperelliptic genus-3 curves as a source of unknown-order groups with trustless setup. We find that Jacobians offer a distinct advantage over class groups at the same security level: the element representation size is smaller (2/3 the size if our new class group compression algorithm is used; if not, 1/2 the size), since point compression for curves is optimal. Using Jacobians also allows us to take advantage of the wealth of algorithms for group operations and exponentiation that have been developed and implemented for hyperelliptic discrete-log-based cryptography, which may be more efficient than their class-group equivalents (though the lack of recent competitive implementations makes it difficult to compare Jacobians and class groups in terms of real-world speed).

We acknowledge that there are, in theory, polynomial-time algorithms to compute the group order of hyperelliptic Jacobians [50, 31]. However, there is evidence that these algorithms are already impractical for discrete-log-based cryptographic group orders of around 256-bits, let alone the much larger group orders that we have in mind. While curves of any genus ≥ 2 might be considered, we suggest that genus-3 curves are the best choice: their point-counting algorithms are already very complex, and their DLP is harder relative to higher-genus curves. Naturally, if Schoof-type algorithms for genus 3 could be made efficient over large prime fields, then these groups would become insecure—but at least we have provided motivation for such work.

Some unknown-order group protocols make stronger assumptions: for example, that finding elements of a given order is hard, or that extracting roots of a given element is hard. In §5 we consider the problem of constructing points of known order in class groups and Jacobians, and explain how we might work with Jacobians when the low-order or adaptive root assumptions are imposed.

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 thank Edward Chen, Luca De Feo, and Jean Kieffer for beneficial discussion, and the anonymous reviewers for their feedback and comments.

Notation. Recall that $\widetilde{O}(x) = O((\log x)^c \cdot x)$ for some constant c, and for subexponential algorithms, $L_x(\alpha) = \exp\left[(1 + o(1))(\log x)^\alpha (\log\log x)^{1-\alpha}\right]$ for $0 \leq \alpha \leq 1$.

2 SUTHERLAND'S ALGORITHM: THE SECURITY OF GENERIC GROUPS

Sutherland's *primorial-steps* algorithm [58, Algorithm 4.2] 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 unknown-order groups; but it also poses a significant threat to a larger class of groups.

Sutherland's algorithm is based on Shanks' baby-step giant-step (BSGS) algorithm, but one can also use Pollard rho. Suppose we wish to compute the order of α. Instead of computing consecutive powers of α in the baby-steps, we 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, by taking E to be the product of the p_i, each raised to an appropriate exponent $\lfloor \log_{p_i}(M) \rfloor$ (where M is an upper bound of the group order). 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 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 $P \geq G(1/u, 2/u)$ [58, Proposition 4.7]. Here $G(r, s)$ is the semismooth probability function

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

for $0 < r < s < 1$, where $\psi(x, y, z)$ is the number of integers up to x 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 from [58] and [6]

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

For each choice of ρ (corresponding to the probability that a weak group is generated), one must determine u such that $1/2^\rho \approx G(1/u, 2/u)$. It then follows that the group size should be at least $u\lambda$ bits so that a $1/u$-th root attack requires at least 2^λ operations. Table 1 gives some numerically computed values for $G(1/u, 2/u)$ from [6] and [58]. Using the method of Banks and Shparlinski [7] to approximate the density of semismooth numbers, we calculate that for a success probability of less than 2^{-100}, we should take $u = 22.5$; for 2^{-128}, we should take $u = 26.5$.

As we mentioned in the introduction, taking $(\lambda, \rho) = (128, 55)$ leads to $\approx$ 1920-bit group orders, because $\rho = 55$ corresponds to $u = 15$, and $15 \cdot 128 = 1920$. A more conservative choice would be $(\lambda, \rho) = (128, 128)$, corresponding to $u \approx 26.5$ and implying 3392-bit groups. Table 2 gives sizes of group orders for various combinations of (λ, ρ). Once again we stress that our setting is different to the usual world of security levels. We are dealing with a fixed class of weak instances of the computational problem.

Table 2: Group size (bits) for various attack success probabilities $2^{-\rho}$ and running costs 2^λ

		ρ					
		40	55	64	80	100	128
λ	55	660	825	880	1045	1265	1430
	80	960	1200	1280	1520	1840	2080
	100	1200	1500	1600	1900	2300	2600
	128	1536	1920	2048	2432	2944	3392

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.

3 IDEAL CLASS GROUPS AS UNKNOWN-ORDER GROUPS

In this section we reconsider ideal class groups as a source of trustless unknown-order groups. We briefly recall the relevant background on class groups in §3.1; detailed references include [19] and [23]. We then reconsider class-group security in §3.2, and give a new compression algorithm for class group elements in §3.3.

3.1 BACKGROUND ON 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 $\Delta \equiv 0, 1 \pmod 4$). The ring of integers $\mathcal{O}_K$ is $\mathbb{Z}[\omega]$, where $\omega = \frac{1}{2}(1 + \sqrt{d})$ when $d \equiv 1 \pmod 4$ and $\omega = \sqrt{d}$ otherwise.

The **ideal class group** is the quotient $Cl(\mathcal{O}_K) = J_K / P_K$, where J_K is the (abelian) group of non-zero fractional ideals of $\mathcal{O}_K$, and $P_K < J_K$ is the subgroup of non-zero principal fractional ideals. In practice, we represent $Cl(\mathcal{O}_K)$ using the isomorphic **form class group** $Cl(\Delta)$ of binary quadratic forms of discriminant Δ (the disciminant of K). We let (a, b, c) denote the binary quadratic form

$$(a, b, c) = ax^2 + bxy + cy^2 \in \mathbb{Z}[x, y] \quad \text{with} \quad b^2 - 4ac = \Delta\,.$$

We can represent this form using only two coefficients (a, b), because c is uniquely determined by $c = (b^2 - \Delta)/4a$. A form (a, b) is **positive definite** if $a > 0$. As with ideal classes, there is an equivalence relation on quadratic forms: f and g are equivalent if $f(x, y) = g(\alpha x + \beta y, \gamma x + \delta y)$ for some α, β, γ, and δ in $\mathbb{Z}$ with $\alpha\delta - \beta\gamma = 1$ (that is, if they are in the same orbit under $\mathrm{SL}_2(\mathbb{Z})$). Equivalent forms always have the same discriminant.

We represent each equivalence class in $Cl(\Delta)$ using the unique **reduced form** in the class. A form (a, b, c) is reduced if $|b| \le a \le c$; and if $|b| = a$ or $a = c$, then $b \ge 0$. Lagrange, and later Gauss and then Zagier, gave algorithms to find the equivalent reduced form for any binary quadratic form. The identity in $Cl(\Delta)$ is the equivalence class of the form $(1, 0, -k)$ if $\Delta = 4k$, or $(1, 1, k)$ if $\Delta = 4k + 1$. The group law in $Cl(\Delta)$, known as composition of forms, is due to Gauss; this does not usually output a reduced form, so reduction is an additional step. We shall not give these algorithms here, but refer the reader to [19].

The order of $Cl(\mathcal{O}_K)$ is the **class number** of K, denoted $h(\Delta)$. It follows from the Brauer–Siegel theorem (see [38]) that for sufficiently large negative discriminants, on average the class number satisfies

$$\log h(\Delta) \sim \log \sqrt{|\Delta|} \quad \text{as} \quad \Delta \to -\infty \tag{1}$$

We can therefore conservatively assume $\approx \frac{1}{2} \log_2 |\Delta|$-bit group sizes for cryptographic-sized negative discriminants.

The use of class groups in cryptography was first suggested by Buchmann and Williams [15]. Hafner and McCurley [37] gave a sub-exponential $L_{|\Delta|}(1/2)$ algorithm for computing the order of $Cl(\Delta)$. 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. Lipmaa [46] proposed that $Cl(\Delta)$ can be used as a group of unknown order without trusted setup, simply by selecting a suitably large Δ and choosing an element in $Cl(\Delta)$ to be treated as a generator (it is not possible to know if it generates the whole of $Cl(\Delta)$, or just a subgroup; we discuss this further below).

3.2 THE SECURITY OF IDEAL CLASS GROUPS

Until now, cryptographic class group parameters have mainly been proposed with an eye to resisting subexponential algorithms for computing orders of quadratic imaginary class groups. 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 [37]; Buchmann extended this to compute the group structure and discrete logarithms. See Biasse et al. [9] for a more up-to-date evaluation of the security of ideal class groups. The important thing to note is that these algorithms all have the same subexponential complexity $L_{|\Delta|}(1/2)$, depending essentially on the size of $|\Delta|$. In contrast, Sutherland's algorithm has exponential worst-case runtime, but performs much faster with a non-negligible probability depending on the structure of the class group—a factor that Hafner–McCurley cannot exploit. When computing the order of a random class group, therefore, the small probability that Sutherland's algorithm outperforms Hafner–McCurley must be taken into account.

The cryptographic parameter sizes in [38] and [14] 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. Biasse, Jacobson Jr., and Silvester [9] improve on previous attacks and suggest 1827-bit discriminants (which implies $\approx$ 914-bit orders) are needed to achieve 128-bit security. These estimates have been quoted in more recent works, including [16] which estimates that 1600-bit discriminants provide 120-bit security, and [12] which proposes a slightly more conservative discriminant size of 2048 bits for 128-bit security.

Suppose we try to compute the order of a random class group with 1827-bit fundamental negative discriminant using Sutherland's algorithm. Sutherland's algorithm has some important practical speedups when specialized from generic groups to class groups—for example, class group element negation is practically free, so time and memory can be reduced by a factor of $\sqrt{2}$ (see [58, Remark 3.1])—but these improvements do not significantly impact security levels. The performance of Sutherland's algorithm on a given quadratic imaginary class group depends entirely on the class number.

Hamdy and Möller [38] show that imaginary 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 of §2, the probability that a random class group with 1827-bit fundamental negative discriminant has less than 128 bits of security ($u = 7.1$) is at least $2^{-14.3}$, and the chance it has less than 64-bit security is 2^{-50}. If a system is using a fixed class group as an accumulator, then we need to ask if these probabilities of weakness are acceptable. We claim that such groups do not satisfy Definition 1 for $(\lambda, \rho) = (128, 128)$, and so the security is weaker at these discriminant sizes than was previously thought.

Bach and Peralta [6] give $G(1/u, 2/u)$ for $u = 20$ as $2.415504 \times 10^{-26} \approx 2^{-85}$. Thus, even for 85-bit security, we require 3400-bit discriminants. Using Banks and Shparlinski [7]'s method of approximating $G(1/u, 2/u)$, we estimate that for 100-bit security with respect to Sutherland's algorithm, a discriminant of around 4500 bits would be required. For 128-bit security, $u = 26.5$ gives $G(1/u, 2/u) \approx 2^{-128}$, which implies a group order $N \approx 2^{128 \times 26.5} = 2^{3392}$, and hence we estimate that Δ should be approximately 6784 bits. We emphasize that $G(1/u, 2/u)$ is only a lower bound for the success probability of Sutherland's algorithm, but this should still serve at least as a guideline.

3.3 COMPRESSING IDEAL CLASS GROUP ELEMENTS

Bleichenbacher [10] proposed a beautiful algorithm to compress Rabin signatures. His method can also be used to compress elements in ideal class groups. As far as we know, this simple observation has not yet been made in the literature.

3.3.1 BLEICHENBACHER'S RABIN SIGNATURE COMPRESSION ALGORITHM

A Rabin signature on a message m under the public key N (an RSA modulus) is an integer σ such that

$$\sigma^2 \equiv m \pmod{N}.$$

Normally σ 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 s and t with $0 < s < \sqrt{N}$ and $|t| \leq \sqrt{N}$ such that $\sigma t \equiv s \pmod{N}$: see Algorithm 1 and Lemma 1 below. The compressed signature is t. Given m and t, let $x = mt^2 \bmod N$; then $s^2 \equiv x \pmod{N}$, but $0 < s < \sqrt{N}$, so we can recover s from m and t by taking the integer square root of x; and then it is trivial to recover $\sigma \equiv s/t \pmod{N}$ (note that if t is not invertible modulo N, then we have found a factor of N, and the signature scheme is broken). We may therefore replace σ with t, which has half the bitlength of σ.

Algorithm 1: `PartialXGCD`.

Input: Integers $a > b > 0$
Output: Integers s in $[0, \sqrt{a}]$ and t in $[-\sqrt{a}, \sqrt{a}]$ such that $s \equiv bt \pmod{a}$

$(s, s', t, t', u, u') \leftarrow (b, a, 1, 0, 0, 1)$
while $s > \sqrt{a}$ **do** `// Invariants:` $0 \leq s < s'$, $|u| < |u'|$, $|t'| < |t|$, $s = au + bt$, $s' = au' + bt'$
  $q \leftarrow s' \operatorname{div} s$ `// Euclidean division without remainder`
  $(s, s', t, t', u, u') \leftarrow (s' - qs, s, t' - qt, t, u' - qu, u)$
return (s, t)

Lemma 1. *Given integers $a > b > 0$, Algorithm 1 returns (s, t) such that $s \equiv bt \pmod{a}$ and $|s|, |t| \leq \sqrt{a}$.*

Proof. Algorithm 1 is a truncated version of the extended Euclidean algorithm, stopping when $s \leq \sqrt{a}$ rather than $s = 0$. The invariants $s' > s \geq 0$, $|u'| > |u|$, $|t'| < |t|$, $s = au + bt$, and $s' = au' + bt'$ are easily verified; in particular, $s \equiv bt \pmod{a}$. Another invariant, $|s't| \leq a$, is proven in [28, Lemma 2.3.3]. Since s takes a sequence of strictly decreasing values, at some point $0 \leq s \leq \sqrt{a}$ and $s' > \sqrt{a}$; this is where the loop terminates. It remains to show that at this point, we also have $|t| \leq \sqrt{a}$: but this follows from the invariant $|s't| \leq a$ and $s' > \sqrt{a}$. □

3.3.2 AN IMPROVED CLASS GROUP ELEMENT COMPRESSION ALGORITHM

Suppose we have a reduced form (a, b, c) in $Cl(\Delta)$. Since $b^2 - 4ac = \Delta$ is a known constant, it suffices to store (a, b). Since the form is reduced, we have $|b| \leq a < \sqrt{|\Delta|}$, so the pair (a, b) can be encoded in approximately $\log_2(|\Delta|)$ bits: this is the traditional "compressed" representation of a class group element.

But we can do better. Since $b^2 - 4ac = \Delta$, we have

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

—a relation reminiscent of the Rabin signature verification equation. The situation is not exactly the same—a is not an RSA modulus, and b is in $(-a, a]$ rather than $[0, a)$—but it is not difficult to adapt signature compression to class group element compression, encoding the coefficient b in half the space.

First, we reduce to the case where $b \geq 0$: we store the sign of b as $\epsilon = 1$ if $b < 0$, and 0 otherwise, and replace b with $|b|$. We will treat the special cases $a = b$ and $(a, b) = (1, 0)$ later; in the meantime, we may suppose that $0 < b < a$. Using Algorithm 1, we compute integers s and t such that $bt \equiv s \pmod{a}$ and $|s|, |t| \leq \sqrt{a}$; then

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

Given a and t, we can compute $x = \Delta t^2 \bmod a$, and then $x = s^2$ *as integers* because $0 \leq s < a$; so s can be recovered as the exact (positive) integer square root. Now $bt \equiv s \pmod{a}$, and the Bleichenbacher approach suggests compressing b to t and recovering b as $s/t \pmod{a}$; but since a is not an RSA modulus, we may—and often do—have $\gcd(a, t) \neq 1$, and then t cannot be inverted modulo a.

To fix this, we compress (a, b) to $(a', g, t', b_0, \epsilon)$, where $g = \gcd(a, t)$, $a' = a/g$, $t' = t/g$, $b_0 = |b| \bmod g$, and ϵ and t are defined as above. To decompress, we compute $a = a'g$, $t = t'g$, $x = t^2\Delta \bmod a$, and $s = \sqrt{x}$ (if $x = 0 \equiv a \pmod{a}$, then s is taken to be $\sqrt{a}$, not 0). Let $b' \equiv s'/t' \pmod{a}$, where $s' = s/g$; then $b' \equiv b \pmod{a'}$, and we can compute $b \pmod{a}$ from $b \equiv b_0 \pmod{g}$ and $b \equiv b' \pmod{a'}$ using the Chinese Remainder Theorem. If $\epsilon = 1$ then we correct the sign, replacing b with $-b$; and after computing c (if required) as $(b^2 - \Delta)/(4a)$, we are done.

For the special case $a = b$, we exceptionally let $t = 0$; this is not ambiguous, because $t = 0$ cannot occur in any other case. The compressed form is then $(a', g, t', b_0, \epsilon) = (1, a, 0, 0, 0)$. For $(a, b) = (1, 0)$, we compress to $(0, 0, 0, 0, 0)$. Again, this is unambiguous: no other element of $Cl(\Delta)$ compresses to this value.

Algorithms 2 and 3 make the compression and decompression procedures completely explicit. Note that

$$\log_2 a' + \log_2 g = \log_2 a \approx \log_2 \sqrt{|\Delta|}$$

and

$$\log_2 t' + \log_2 b_0 \leq \log_2 t' + \log_2 g = \log_2 t \approx \tfrac{1}{2} \log_2 \sqrt{|\Delta|}\,.$$

Algorithm 2 therefore compresses the form (a, b, c) to a $\frac{3}{4} \log_2 |\Delta|$-bit representation, three-quarters of the size of the traditional (a, b). When a party receives a compressed group element it is necessary for them to execute Algorithm 3 before performing group operations on the element.

Algorithm 2: Element compression for $Cl(\Delta)$

Input: A reduced form (a, b, c) in $Cl(\Delta)$ (c may be omitted)
Output: A compressed form $(a', g, t', b_0, \epsilon)$

if $(a, b) = (1, 0)$ **then return** $(0, 0, 0, 0, 0)$
if $a = b$ **then return** $(1, a, 0, 0, 0)$
$\epsilon \leftarrow \begin{cases} 1 & \text{if } b < 0 \\ 0 & \text{otherwise} \end{cases}$
$b \leftarrow |b|$
$(s, t) \leftarrow \texttt{PartialXGCD}(a, b)$ `// Now` $s \equiv bt \pmod{a}$, `with` $|s|$ `and` $|t| < \sqrt{|a|}$
$g \leftarrow \gcd(a, t)$
$(a', t') \leftarrow (a/g, t/g)$
$b_0 \leftarrow b \bmod g$
return $(a', g, t', b_0, \epsilon)$

4 HYPERELLIPTIC JACOBIANS AS UNKNOWN-ORDER GROUPS

We now revisit Brent's proposal of hyperelliptic Jacobians as a concrete source of unknown-order groups, focusing on genus $g = 3$. Hyperelliptic Jacobians can be seen as the ideal class groups of quadratic function fields. We will argue that even despite the existence of theoretical polynomial-time point-counting algorithms, these Jacobians may still present a more efficient alternative to class groups at the same security levels.

Algorithm 3: Element decompression for $Cl(\Delta)$

Input: A compressed form $(a', g, t', b_0, \epsilon)$ and Δ
Output: A reduced form (a, b, c) in $Cl(\Delta)$ (c may be omitted)

if $(a', g, t', b_0, \epsilon) = (0, 0, 0, 0, 0)$ **then return** $(1, 0, -\Delta/4)$
if $t' = 0$ **then return** $(a, a, (a^2 - \Delta)/(4a))$
$(a, t) \leftarrow (g \cdot a', g \cdot t')$
$x \leftarrow t^2\Delta \mod a$
if $x = 0$ **then**
  $x \leftarrow a$
$s \leftarrow \sqrt{x}$ // Integer square root
$s' \leftarrow s/g$ // Exact integer division
$b' \leftarrow s' \cdot t^{-1} \pmod{a'}$
$b \leftarrow \mathtt{CRT}((b', a'), (b_0, g))$ // $b \equiv b' \pmod{a'}$ and $b \equiv b_0 \pmod{g}$
if $\epsilon = 1$ **then**
  $b \leftarrow -b$
return $(a, b, (b^2 - \Delta))$

4.1 HYPERELLIPTIC CURVES

We briefly recall the relevant background here; details can be found in [47] and [28]. The reader familiar with hyperelliptic curves may skip this section.

Let q be an odd prime power. A hyperelliptic curve C of genus $g > 1$ over $\mathbb{F}_q$ is defined by an equation $y^2 = f(x)$, where f is a monic, squarefree polynomial of degree $2g + 1$ over k.[1] A (finite) point on C is an (x, y) in $\overline{\mathbb{F}}_q^2$ satisfying the defining equation of C; there is also a unique point at infinity, denoted ∞.

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 degree of a divisor is $\deg D = \sum m_P$. The divisors form a group $\mathrm{Div}(C)$, and the divisors of degree zero form a proper subgroup $\mathrm{Div}^0(C)$ of $\mathrm{Div}(C)$. We let $\mathcal{P}(C)$ denote the set of principal divisors of C: that is, divisors of the form $(\gamma) = \sum_{P \in C} \mathrm{ord}_P(\gamma) P$ for some γ in the function field $\overline{\mathbb{F}}_q(C) = \overline{\mathbb{F}}_q(x)[y]/(y^2 - f(x))$ (here $\mathrm{ord}_P(\gamma)$ is the order of vanishing of γ at P). Principal divisors have degree 0, so $\mathcal{P}(C) \subset \mathrm{Div}^0(C)$; the **Jacobian** is the quotient group

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

(also known as the degree-0 Picard group, denoted by $\mathrm{Pic}^0(C)$).

We compute with elements of J_C using the Mumford representation [48]. Each divisor class contains a unique *reduced* divisor in the form $P_1 + \cdots + P_r - r\infty$ (with the P_i not necessarily distinct) with $r \leq g$ and $P_i \neq \tilde{P}_j$ for all $i \neq j$. The reduced divisors correspond to pairs of polynomials $\langle u(x), v(x) \rangle$, where u is monic, $\deg v < \deg u \leq g$ and $v^2 \equiv f \pmod{u}$. The roots of $u(x)$ are the x-coordinates of the points in the support of the divisor. The divisor classes defined over $\mathbb{F}_q$—that is, such that u and v have coefficients in $\mathbb{F}_q$—form a finite group, denoted $J_C(\mathbb{F}_q)$. The Hasse–Weil bound tells us that $\#J_C \sim q^g$; more precisely,

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

The group law on $J_C(\mathbb{F}_q)$ can be computed using Cantor's algorithm [17] (see also [22]). Efficient explicit formulae exist for $g = 2$ (see [44]) and $g = 3$ (see [26]).

4.2 THE SECURITY OF HYPERELLIPTIC JACOBIANS

Let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$, where $q = p^n$, and J_C its Jacobian. Recall that $\#J_C(\mathbb{F}_q) \approx q^3$. For $J_C(\mathbb{F}_q)$ to be useful as an unknown-order group, calculating $\#J_C(\mathbb{F}_q)$ should be infeasible. Besides generic algorithms, two classes of algorithms specific to hyperelliptic Jacobians are relevant here: point-counting and discrete-log algorithms.

As a base-line, C must be chosen such that $J_C(\mathbb{F}_q)$ resists Sutherland's algorithm with acceptable probability, as in §2. Sutherland's algorithm has some important practical optimizations when specialized to hyperelliptic Jacobians—for example, we can exploit the fact that negation is effectively free to decrease storage and runtime by a factor of up to $\sqrt{2}$ (see [58, Remark 3.1]), and even if a Jacobian is not directly vulnerable to Sutherland's algorithm, its order may be deduced from that of vulnerable twists, as in [59]—but these improvements do not significantly impact security levels.

[1] We do not need the more general case of nonsingular curves $y^2 + h(x)y = f(x)$, where $\deg f = 2g + 1$ or $2g + 2$, and $\deg h \leq g$.

To reach acceptable security levels against Sutherland's algorithm using genus 3 curves then q must be subexponentially large. Looking at Table 2, the cautious $(\lambda, \rho) = (128, 55)$ level requires 1920-bit groups, or $q \approx 2^{640}$; the paranoid $(128, 128)$ level requires 3392-bit groups, or $q \approx 2^{1131}$. Fields of this size also address the concerns of Lee [45].

4.2.1 POINT-COUNTING ALGORITHMS

Computing $\#J_C(\mathbb{F}_q)$ is a classic problem (called "point counting") in algorithmic number theory: the goal is to compute the zeta function of C, from which we immediately get $\#J_C(\mathbb{F}_q)$. The many dedicated point-counting algorithms fall naturally into two broad classes: p-adic algorithms and ℓ-adic "*Schoof-type*" algorithms. The p-adic algorithms (notably Kedlaya's algorithm [41] and its descendants [39]) have complexity polynomial with respect to g and n, but exponential in $\log p$. Taking $q = p$, we can ignore these algorithms.

Schoof-type algorithms compute $\#J_C(\mathbb{F}_q)$ in polynomial time *for fixed* g. Indeed, from a theoretical point of view, the existence of Schoof-type algorithms may make the use of hyperelliptic Jacobians as unknown-order groups seem perverse. But Schoof-type algorithms are totally impractical over large prime fields, even in genus as small as 3. To understand why, we need to look at how they operate.

First, consider the case of elliptic curves ($g = 1$). Schoof's ground-breaking $\widetilde{O}(\log^5 q)$ algorithm [55], the first polynomial-time point-counting algorithm for elliptic curves, computes the characteristic polynomial of Frobenius for a series of small prime ℓ, using polynomial arithmetic modulo the division polynomials Ψ_ℓ, before combining the results with the Chinese Remainder Theorem to compute the zeta function.[2] Its successor, the Schoof–Elkies–Atkin (SEA) algorithm [56], runs in time $\widetilde{O}(\log^4 q)$, and has made elliptic-curve point counting routine.

Pila generalised Schoof's algorithm to higher-dimensional Abelian varieties [50], including all Jacobians of curves. Pila's algorithm is polynomial-time in p and n, but badly exponential in g; as far as we know, it has never been implemented. The task gets a little simpler when we specialize from general abelian varieties to hyperelliptic Jacobians. The crucial objects are the analogue of the division polynomials: these are multivariate *division ideals* vanishing on coordinates of points in torsion subgroups (in this sense, Ψ_ℓ generates the ℓ-division ideal in genus $g = 1$). Cantor constructs generators for the ℓ-division ideal in [18] (see also [20]); in genus 3, the degrees of Cantor's ℓ-division polynomials are bounded by $O(\ell^2)$ (see [1]).

Schoof-type point-counting is already challenging in genus 2. Several genus-2 algorithms have been implemented and analysed, beginning with Gaudry and Harley [31] and Gaudry and Schost [34]. Pitcher's PhD thesis [51] gave a genus-2 algorithm with complexity $O((\log q)^7)$. Gaudry and Schost [35] used an improved algorithm, with a mixture of Pitcher's approach and exponential birthday-paradox 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 on average to compute the order of a random genus-2 curve over this 127-bit field. Computing ℓ-division ideals and analysing the action of Frobenius on them can become impractical for even moderately small ℓ: the computations mentioned above, with an 8GB limit on RAM, used primes $\ell \leq 31$ (the earlier [34] used ℓ up to 19). They also used small prime powers $\ell^k = 2^{16}$, 3^6, 5^4, and 7^2. These ℓ and ℓ^k are not sufficient to determine the group order; to finish the order computation, they used one- or two-dimensional random walks (a low-memory square-root algorithm: see [33] for details). We underline the fact that finishing this point-counting computation is a situation where in practice, an exponential algorithm is more practical than a polynomial-time one!

We have found no practical work for general genus-2 curves going beyond $\ell = 31$ in the literature. Abelard's PhD thesis [1] discusses the feasibility of continuing with larger ℓ. With time complexity $\widetilde{O}(\ell^6 \log q)$ and space complexity $\widetilde{O}(\ell^4 \log q)$, running time becomes more of an issue than memory: for 192-bit q, the computation for $\ell = 53$ could take around 1000 CPU-days, yet still leave a search space of $\approx 2^{95}$ elements in the exponential "collision" step of the algorithm.

This practical work has not been extended to genus 3. The main obstruction is the complexity of computing with division ideals. Some theoretical analysis and projected complexities appear in [2]. Some first steps have been made for the very special class of genus-3 Jacobians with known and efficiently computable real multiplication endomorphisms in [3], following the analogous genus-2 algorithm in [32], but this approach does not apply to general genus-3 hyperelliptic Jacobians.

Concretely, taking $q \sim 2^{100}$ in genus 3 would appear sufficient to resist point counting on most curves C, and result in $\#J_C(\mathbb{F}_q) \sim 2^{300}$; the much larger group and field sizes require to resist Sutherland's algorithm render point counting irrelevant as an attack.

The upshot is that while point-counting for *fixed* genus $g > 2$ is polynomial-time in theory, it remains impractical—even infeasible—in the real world. This is already true of the relatively small field sizes relevant to discrete-logarithm-based cryptography; it is even more so for the much larger, subexponential-sized fields required to protect against Sutherland's algorithm in the unknown-order setting.

[2] 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. For odd prime ℓ, the degree of Ψ_ℓ is $(\ell - 1)/2$.

Remark 1. *Some work has been done on generating genus-2 and genus-3 Jacobians* with a known number of points *using CM theory, for applications in DLP-based cryptography, notably by Weng [62, 63] (see also e.g. [35] and [40]). Obviously, these curve-generation methods must be avoided for unknown-order applications.*

Remark 2. *One might hope that progress in computing higher-genus modular polynomials might yield a SEA analogue improving substantially on pure Schoof-style point counting. However, any SEA analogue in genus $g > 3$ would actually be* slower *than pure Schoof. Indeed, the number of isogenies splitting $[\ell]$ (and hence the degree of the ideal that a SEA analogue would use at the prime ℓ) is in $O(\ell^{g(g+1)/2})$; this exceeds the degree of the ℓ-division ideal, which is in $O(\ell^{2g})$. Even for $g = 3$, the asymptotic complexity of SEA is no better than that of Schoof.*

4.2.2 DISCRETE LOGARITHM ALGORITHMS

If the DLP can be efficiently solved in a subgroup $\langle G\rangle \subset J_C(\mathbb{F}_q)$, the order of $\langle G\rangle$ can also be efficiently computed: if $xG = O$, where is the cryptographic subgroup, then x is (a multiple of) the order. Suppose, then, that we want to solve the DLP in $J_C(\mathbb{F}_q)$, where C is a curve of genus g over $\mathbb{F}_q$. Gaudry et al. [36] and Nagao [49] present algorithms for small g running in time $\widetilde{O}(q^{2-2/g})$, improving on the $O(q^2)$ algorithm of [30], and the single-large-prime algorithm of [61]. This has better performance for genus 3 than square-root algorithms like Pollard Rho, which has expected runtime in $\widetilde{O}(q^{3/2})$; but in genus 2, Pollard Rho is more efficient, in $\widetilde{O}(q)$. Avanzi, Thériault, and Wang [5] further discuss security in these cases.

Smith [57] 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 [24] can then be used to solve the DLP in time $\widetilde{O}(q)$. Laine and Lauter [42] 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 $\widetilde{O}(q^{3/4})$. The practical results from [42] suggest that even for $q \sim 2^{100}$, discrete logarithms require around 2^{113} field multiplications and 1.2×10^{14}TB of memory, assuming the reduction of [57] applies; if not, the algorithm of [36] would require on the order of 2^{133} operations. Genus-3 hyperelliptic curves avoiding isogeny-based attacks are constructed in [43].

As g tends to infinity, there exist subexponential attacks on the DLP using index calculus (for example, [25]); but these have no impact for fixed genus 2 and 3.

4.3 GENERATING HYPERELLIPTIC JACOBIANS OF UNKNOWN ORDER

Algorithm 4 (`Gen`) takes security parameters (λ, ρ) (as in Definition 1), and outputs a generator P for a group G such that Sutherland's algorithm running in time 2^λ succeeds in computing $\#G$ with probability less than $1/2^\rho$. The group G is realized as (a sub-group of) a genus-3 hyperelliptic Jacobian. Having chosen a suitable prime p as a function of (λ, ρ), the algorithm samples a uniformly random monic irreducible degree-7 polynomial $f(x)$ in $\mathbb{F}_p[x]$ and polynomials u and v such that $\langle u, v\rangle$ is the Mumford representation of a divisor class P in $J_C(\mathbb{F}_p)$, where C is the curve defined by $y^2 = f(x)$. Being random, P generates a large-order subgroup of $J_C(\mathbb{F}_p)$ with high probability.

Taking f to be random makes the probability that C is a "weak" curve overwhelmingly small. The order of a random Jacobian should have a large prime factor, protecting against Pohlig–Hellman; the largest prime factor should not divide $q^k - 1$ for small k, protecting against MOV-type attacks [27]; and the group order should be prime to $p = \mathrm{char}(\mathbb{F}_q)$ to avoid "anomalous curve" attacks [53]. Randomly-sampled Jacobians do not have special endomorphisms, such as the efficiently-computable real multiplication exploited for faster point counting in [3], because these special classes of curves form positive-codimensional subspaces of the moduli space. Recent work of Thakur [60] further discusses classes of curves to avoid. The security of random ideal class groups as groups of unknown order depends on similar assumptions and heuristics [21].

Taking f irreducible over $\mathbb{F}_p$ ensures that $J_C(\mathbb{F}_p)$ has no points of order 2. As we will see in §5, 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 small factors in $\#J_C(\mathbb{F}_p)$, but this makes no significant difference to the probability of semismoothness of $\#J_C(\mathbb{F}_p)$, and thus to Sutherland's algorithm. Our simulations showed that rejecting random group orders divisible by the first few primes decreased the semismoothness probability by less than a factor of 2.

To ensure that not even the constructor of C knows $\#J_C(\mathbb{F}_p)$, and that C and $P = \langle u, v\rangle$ 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 taking multiple "seeds", and testing each resulting curve for weakness. If the probability of encountering a weak curve among random curves is δ, and testing for weakness costs 2^n operations, then a malicious actor requires around $\delta^{-1}2^n$ operations to generate a weak C. A sceptical verifier, on the other hand, must only test the proposed C just once to detect cheating, at a cost of just 2^n operations. This imbalance of the cost of cheating versus verifying is a deterrent for attackers, regardless of the weakness in question.

Algorithm 4: Gen. Constructs a random unknown-order (subgroup of a) genus-3 hyperelliptic Jacobian.

Input: (λ, ρ)

Output: A prime p, a hyperelliptic genus-3 curve $C/\mathbb{F}_p$, and $P \in J_C$ such that $\langle P \rangle$ has unknown order.

Determine n such that a random genus-3 curve over an n-bit prime field has λ-bits of security with probability $1 - 1/2^\rho$

$p \leftarrow$ a random n-bit prime

Sample random $u(x) = x^3 + u_2x^2 + u_1x + u_0$ in $\mathbb{F}_p[x]$

Sample random $v(x) = v_2x^2 + v_1x + v_0$ in $\mathbb{F}_p[x]$

repeat

  Sample random $w(x) = x^4 + w_3x^3 + w_2x^2 + w_1x + w_0$ in $\mathbb{F}_p[x]$

  $f(x) \leftarrow v(x)^2 + u(x)w(x)$

until $\gcd(f(x), f'(x)) = 1$ ***and*** *f is irreducible*

$P \leftarrow \langle u, v \rangle$

return (p, C, P) *where C is the hyperelliptic curve $y^2 = f(x)$ over $\mathbb{F}_p$*

Now the order of the Jacobian $J_C(\mathbb{F}_p)$ (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.

Elements of $J_C(\mathbb{F}_q)$ are represented as pairs of polynomials $\langle u, v \rangle$ with $\deg v < \deg u \leq g$, so elements can be stored concretely with 6 elements of $\mathbb{F}_q$, and further compressed to just 3 $\mathbb{F}_q$-elements and 3 extra bits using the method of [40].[3] For $(\lambda, \rho) = (128, 55)$, with $\approx$ 640-bit fields, this means that group elements can be stored in $\approx$ 1920 bits; elements of a class group of equivalent security require $\approx$ 2860 bits with the compression of §3.3 (or $\approx$ 3840 bits without it). Moving to the more paranoid security level of $(\lambda, \rho) = (128, 128)$, hyperelliptic Jacobian elements require $\approx$ 3392 bits while class-group elements require $\approx$ 5088 bits (or $\approx$ 6784 bits without the compression of §3.3). We therefore claim that genus-3 Jacobians offer more compact elements than class groups at the same security level.

5 ELEMENTS OF KNOWN ORDER

We now briefly consider the problem of constructing points of known order in groups of unknown order. This will give us an idea of how to work with Jacobians when the low-order or adaptive root assumptions are imposed.

5.1 LOW-ORDER ASSUMPTIONS AND COFACTORS

Common additional requirements on unknown-order groups include

- the *low-order assumption*: finding an element P of a given order s in G is hard (see [11, Def. 1]); and
- the *adaptive root assumption*: extracting roots of a given element—that is, given Q and s, find P such that $Q = [s]P$ in G—is hard (see [11, Def. 2] and [64]).

These assumptions only make sense if the adversary must solve arbitrary instances in a randomly-sampled G; it is not possible to define security for a fixed G. Example 1 describes Wesolowski's Proof of Exponentiation (PoE), a protocol which requires these assumptions.

Example 1 (PoE)**.** *Let G be a group, 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 $u^x = w$ in significantly less time than it takes to compute u^x. It proceeds interactively as follows (although it can be made non-interactive: see [64]).*

1. *Verifier sends a random prime $\ell \in Primes(\lambda)$ to Prover.*
2. *Prover computes $q = \lfloor x/\ell \rfloor$ and $Q = u^q$, and sends Q to Verifier.*
3. *Verifier computes $r = x \bmod \ell$, and accepts if $Q^\ell u^r = w$*

To see why the security of this protocol requires the low-order assumption, suppose we know an element ϵ of order 2 in G (for example, if G is an RSA group, then we can take $\epsilon = -1$). Then for any valid proof that $u^x = w$,

[3] Given that hyperelliptic Jacobians are a function-field analogue of ideal class groups of quadratic fields, it is natural to ask why the almost-ideal hyperelliptic Jacobian element compression algorithm of [40] does not have an efficient class-group analogue. To compress a Jacobian element $\langle u, v \rangle$, the algorithm of [40] begins by factoring u, a polynomial over a finite field; we know how to do this efficiently. A class-group analogue compressing (a, b) would need to factor the integer a, which is a much harder problem.

we can easily generate a false proof of the contradictory statement $u^x = \epsilon w$, by replacing Q with $Q' = \epsilon Q$ in the proof. Since ℓ is odd, $(Q')^\ell u^r = \epsilon Q^\ell u^r = \epsilon w$ holds despite the fact that $u^x \neq \epsilon w$. This 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.

Suppose we are given an algorithm Gen constructing unknown-order groups reaching the (λ, ρ) security level. If we can specify a set $\mathcal{S}$ containing the integers s such that we can construct elements of order s or extract s-th roots in groups G output by Gen in $< 2^\lambda$ operations with probability $> 2^{-\rho}$, then the low-order and adaptive-root assumptions hold in the subgroup

$$[S]G = \{[S]P \mid P \in G\} \qquad \text{where} \qquad S := \operatorname{lcm}(\mathcal{S})\,. \tag{3}$$

We will propose conservative choices for $\mathcal{S}$ for concrete groups below. In the meantime, to give some concrete intuition, if we take $\mathcal{S} = \{1, \ldots, 60\}$ then S is an 84-bit integer, so multiplication by S is efficient.

The operation of protocols such as accumulators in $[S]G$ is standard, but some protocols may need modification: for example, proofs may require an extra check that an element is indeed in the group. The issue here is that given a point Q in G, testing subgroup membership $Q \in [S]G$ is not easy. However, the original point Q is effectively a proof that $[S]Q$ is in $[S]G$, 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.

Using $[S]G$ in place of G 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. To give a specific example, we revisit the PoE from Example 1 in Example 2. The verifier only needs to perform one extra multiplication-by-S during verification when working in $[S]G$ instead of G. We suggest that this is efficient enough for practical use, and that other protocols using the adaptive root assumption can be modified in a similar way.

Example 2 (PoE in $[S]G$)**.** *We begin the PoE protocol with input $U \in G$, $W \in [S]G$, and $x \in \mathbb{Z}$. The claim to be proven is that $[x][S]U = W$ in $[S]G$. The protocol proceeds as follows:*

1. *Verifier selects a random ℓ from Primes$(\lambda) \setminus \mathcal{S}$ and sends ℓ to Prover.*
2. *Prover computes $q = \lfloor x/\ell \rfloor$, computes $Q = [q]U$ in G and sends Q to Verifier.*
3. *Verifier computes $r = x \bmod \ell$, and accepts if Q is in G and $[S]([\ell]Q + [r]U) = W$.*

The security of this protocol depends on the choice of $\mathcal{S}$. 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 the ℓ chosen by the verifier. This may be possible if the order of P is known, but this is supposed to be infeasible because ℓ is not in $\mathcal{S}$.

Remark 3. *The impact of finding small-order elements is highly domain-specific. For example, in the VDF of [64, 11], even if points of known order can be found, forging a false PoE still requires knowing the true result of the exponentiation—and hence still 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. For accumulators, we need an analogue of 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 ℓ dividing elements of $\mathcal{S}$. Finding ℓ-th roots with ℓ not in $\mathcal{S}$ is supposed to be infeasible, so here we do not need to use $[S]G$.*

5.2 ELEMENTS OF KNOWN ORDER IN CLASS GROUPS AND JACOBIANS

For class groups, it is well-known that the factorization of Δ reveals the 2-torsion structure of $Cl(\Delta)$, and even allows the explicit construction of elements of order 2. Similarly, for Jacobians, if $C/\mathbb{F}_q$ is defined by $y^2 = f(x)$, then the factorization of $f(x)$ reveals the 2-torsion structure of $J_C(\mathbb{F}_q)$, and lets us construct explicit points of order 2. This motivates the restriction to negative prime Δ when using $Cl(\Delta)$ as an unknown-order group, and our restriction to irreducible f in Algorithm 4.

Belabas, Kleinjung, Sanso, and Wesolowski [8] give several constructions of special discriminants Δ together with a known ideal of small odd order in $Cl(\Delta)$. Similarly, we can construct hyperelliptic Jacobians equipped with a point of small order, as in Example 3.

Example 3. *Let ℓ be an odd prime, and set $g = (\ell - 1)/2$. Choose a polynomial $c(x)$ over $\mathbb{F}_q$ of degree $\leq g$ such that $f(x) := x^\ell + c(x)^2$ is squarefree. Then $C : y^2 = f(x)$ is a hyperelliptic curve of genus g, and the divisor $D = (0, c(0)) - (\infty)$ represents a nontrivial element of order ℓ in $J_C(\mathbb{F}_q)$ (because ℓD is the principal divisor of the function $y - c(x)$). Taking $\ell = 7$ gives a family of genus-3 Jacobians with a known element of order 7.*

These discriminants and curves generally do not occur when Δ or f is chosen in a "nothing-up-my-sleeve" way. In any case, the risk of choosing groups with constructible small-order elements can be eliminated by using a smooth cofactor S.

There are three curve-specific methods for constructing elements of known small order, or deducing information about small divisors of the order of a given element, which do not apply to class groups. The first is to use the division ideals. This is practical for small primes like 2, 3, and 5 (reinforcing the need for the cofactor S above). However, if we assume that there exists no feasible Schoof-type algorithm for counting points on genus 3 curves, then we implicitly assume that it is infeasible to construct ℓ-division ideals for ℓ larger than some bound that is polynomial with respect to the security parameters.

The second method is to use repeated divisions by 2 in $J_C(\mathbb{F}_q)$ to construct points of order 2^k for arbitrarily large k. Since 2^k is coprime to all odd primes ℓ, this allows malicious provers to easily find ℓ-th roots for these points (that is, given a point Q, find P such that $[\ell]P = Q$). But repeated division by 2 in $J_C(\mathbb{F}_q)$ requires the repeated extraction of square roots in $\mathbb{F}_p$, which quickly requires repeated quadratic field extensions and the field computations blow-up exponentially. Using hyperelliptic curves in the form $y^2 = f(x)$ with $f(x)$ irreducible ensures that the required square roots do not exist in $\mathbb{F}_p$.

Similarly, we might calculate repeated divisions by very small odd primes. Using the group $[S]J_C(\mathbb{F}_q)$ will kill off powers of these small 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 these off with a tailored choice of S. It is an interesting open problem to generate an easily verifiable proof that a Jacobian does not have any points of low order.

The third method involves the Tate pairing. Let C be a hyperelliptic curve over $\mathbb{F}_q$, let ℓ be a prime (coprime to q), and let k be the smallest positive integer such that $\ell \mid q^k - 1$. The reduced ℓ-Tate pairing is a bilinear mapping

$$t_\ell : J_C[\ell] \times J_C(\mathbb{F}_{q^k})/\ell J_C(\mathbb{F}_{q^k}) \longrightarrow \mu_\ell \subset \mathbb{F}_{q^k}^\times ,$$

where μ_ℓ is the group of ℓ-th roots of unity (see [29] for background on pairings on hyperelliptic curves). If we can find points of known order ℓ, then the ℓ-Tate pairing can give information about the ℓ-divisibility 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 $t_\ell(Q, P) = 1$. By the contrapositive, if $t_\ell(Q, P) \neq 1$, 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 does not imply $\ell \nmid \mathrm{Ord}(P)$. Instead, it must be shown that $t_\ell(Q, P) = 1$ for *all* Q in $J_C[\ell]$. Thus, we require a basis $\{Q_1, \ldots, Q_{2g}\}$ of $J_C[\ell]$ which we can test: if $t_\ell(Q_i, P) = 1$ for $1 \leq i \leq 2g$, then the bilinearity of the Tate pairing implies $t_\ell(Q, P)$ for all Q in $J_C[\ell]$, and hence that $\gcd(\mathrm{Ord}(P), \ell) = 1$.

The utility of this approach is limited by the difficulty of constructing points of order ℓ, but also by the field extension degree k (since the coordinates of Q and the value of $t_\ell(Q, P)$ are in $\mathbb{F}_{q^k}$); and k, being the order of q modulo ℓ, tends to blow up with ℓ. If q is well-chosen, then in practice we can learn very little information about the orders of random points in $J_C(\mathbb{F}_q)$, or any information at all for points in $[S]J_C(\mathbb{F}_q)$ for a suitable S. For the Jacobian case, we conjecture that $\mathcal{S} = \{1, \ldots, 60\}$ is sufficient for a (128, 128) security level, based on the discussion in §4.2.1. For class groups, $\mathcal{S}$ can either be empty in the case of a prime discriminant, or $\mathcal{S} = \{2\}$ if a non-prime discriminant is used.

REFERENCES

[1] Simon Abelard. "Counting points on hyperelliptic curves in large characteristic: algorithms and complexity". PhD thesis. Université de Lorraine, 2018. URL: https://tel.archives-ouvertes.fr/tel-01876314.

[2] Simon Abelard, Perrick Gaudry, and Pierre-Jean Spaenlehauer. "Improved complexity bounds for counting points on hyperelliptic curves". In: *Foundations of Computational Mathematics* 19.3 (2019), pp. 591–621.

[3] Simon Abelard, Pierrick Gaudry, and Pierre-Jean Spaenlehauer. "Counting points on genus-3 hyperelliptic curves with explicit real multiplication". In: *ANTS-XIII. Proceedings of the Thirteenth Algorithmic Number Theory Symposium*. Ed. by Renate Scheidler and Jonathan Sorenson. Vol. 2. The Open Book Series 1. Mathematical Sciences Publishers, 2019, pp. 1–19. DOI: 10.2140/obs.2019.2.1.

[4] Martin R. Albrecht, Jake Massimo, Kenneth G. Paterson, and Juraj Somorovsky. "Prime and Prejudice: Primality Testing Under Adversarial Conditions". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM, 2018, pp. 281–298. DOI: 10.1145/3243734.3243787.

[5] Roberto Avanzi, Nicolas Thériault, and Zheng Wang. "Rethinking low genus hyperelliptic Jacobian arithmetic over binary fields: Interplay of field arithmetic and explicit formulae". In: *Journal of Mathematical Cryptology* 2.3 (2008), pp. 227–255.

[6] Eric Bach and René Peralta. "Asymptotic semismoothness probabilities". In: *Mathematics of computation* 65.216 (1996), pp. 1701–1715.

[7] William D. Banks and Igor E. Shparlinski. "Integers with a large smooth divisor". In: *Integers* 7.A17 (2007), pp. 1–11.

[8] Karim Belabas, Thorsten Kleinjung, Antonio Sanso, and Benjamin Wesolowski. *A note on the low order assumption in class group of an imaginary quadratic number fields*. Cryptology ePrint Archive, Report 2020/1310. https://eprint.iacr.org/2020/1310. 2020.

[9] Jean-François Biasse, Michael J. Jacobson Jr., and Alan K. Silvester. "Security Estimates for Quadratic Field Based Cryptosystems". In: *Information Security and Privacy – ACISP 2010*. Ed. by Ron Steinfeld and Philip Hawkes. Vol. 6168. Lecture Notes in Computer Science. Springer, 2010, pp. 233–247. DOI: 10.1007/978-3-642-14081-5_15.

[10] Daniel Bleichenbacher. "Compressing Rabin Signatures". In: *Topics in Cryptology – CT-RSA 2004*. Ed. by Tatsuaki Okamoto. Vol. 2964. Lectures Notes in Computer Science. Springer, 2004, pp. 126–128. DOI: 10.1007/978-3-540-24660-2_10.

[11] Dan Boneh, Benedikt Bünz, and Ben Fisch. *A Survey of Two Verifiable Delay Functions*. Cryptology ePrint Archive, Report 2018/712. https://eprint.iacr.org/2018/712. 2018.

[12] Dan Boneh, Benedikt Bünz, and Ben Fisch. "Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains". In: *Advances in Cryptology – CRYPTO 2019*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. Lecture Notes in Computer Science. Springer, 2019, pp. 561–586. ISBN: 978-3-030-26948-7.

[13] Richard P. Brent. *Public Key Cryptography with a Group of Unknown Order*. Tech. rep. Oxford University, 2000.

[14] Johannes Buchmann and Safuat Hamdy. "A Survey on IQ Cryptography". In: *Public Key Cryptography and Computational Number Theory. Proceedings of the International Conference organized by the Stefan Banach International Mathematical Center Warsaw, Poland, September 11-15, 2000*. Ed. by Kazimierz Alster, Jerzy Urbanowicz, and Hugh C. Williams. De Gruyter Proceedings in Mathematics. De Gruyter, 2001, pp. 1–15. DOI: 10.1515/9783110881035.

[15] Johannes Buchmann and Hugh C. Williams. "A key-exchange system based on imaginary quadratic fields". In: *Journal of Cryptology* 1.2 (June 1988), pp. 107–118. ISSN: 1432-1378. DOI: 10.1007/BF02351719.

[16] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lectures Notes in Computer Science. Springer, 2020. DOI: 10.1007/978-3-030-45721-1_24.

[17] David G. Cantor. "Computing in the Jacobian of a hyperelliptic curve". In: *Mathematics of Computation* 48.177 (1987), pp. 95–101.

[18] David G. Cantor. "On the analogue of the division polynomials for hyperelliptic curves." In: *Journal für die reine und angewandte Mathematik* 447 (1994), pp. 91–146. URL: http://eudml.org/doc/153593.

[19] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Vol. 138. Graduate Texts in Mathematics. Springer, 2010. DOI: 10.1007/978-3-662-02945-9.

[20] 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.

[21] Henri Cohen and Hendrik W. Lenstra. "Heuristics on class groups of number fields". In: *Number Theory Noordwijkerhout 1983*. Ed. by Hendrik Jager. Vol. 1068. Lecture Noetes in Mathematics. Springer, 1984, pp. 33–62.

[22] Craig Costello and Kristin Lauter. "Group Law Computations on Jacobians of Hyperelliptic Curves". In: *Selected Areas in Cryptography – SAC 2011*. Ed. by Ali Miri and Serge Vaudenay. Vol. 7118. Lecture Notes in Computer Science. Springer, 2012, pp. 92–117. DOI: 10.1007/978-3-642-28496-0_6.

[23] David 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. ISBN: 9780471506546. URL: https://books.google.co.nz/books?id=pSMlAQAAIAAJ.

[24] Claus Diem. "An Index Calculus Algorithm for Plane Curves of Small Degree". In: *Algorithmic Number Theory – ANTS 2006*. Ed. by Florian Hess, Sebastian Pauli, and Michael Pohst. Vol. 4076. Lecture Notes in Computer Science. Springer, 2006, pp. 543–557. DOI: 10.1007/11792086_38.

[25] Andreas Enge. “Computing Discrete Logarithms in High-genus Hyperelliptic Jacobians in Provably Subexponential Time”. In: *Mathematics of Computation* 71.238 (Apr. 2002), pp. 729–742. DOI: 10.1090/S0025-5718-01-01363-1.

[26] Xinxin Fan, Thomas Wollinger, and Guang Gong. “Efficient explicit formulae for genus 3 hyperelliptic curve cryptosystems over binary fields”. In: *IET Information Security* 1.2 (2007), pp. 65–81.

[27] Gerhard Frey and Hans-Georg Rück. “A Remark Concerning M-divisibility and the Discrete Logarithm in the Divisor Class Group of Curves”. In: *Math. Comput.* 62.206 (Apr. 1994), pp. 865–874. ISSN: 0025-5718. DOI: 10.2307/2153546.

[28] Steven D. Galbraith. *Mathematics of Public Key Cryptography*. 1st edition. Cambridge University Press, 2012. DOI: 10.1017/CBO9781139012843.

[29] Steven D. Galbraith, Florian Hess, and Frederik Vercauteren. “Hyperelliptic Pairings”. In: *Pairing-Based Cryptography – Pairing 2007*. Ed. by Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto. Vol. 4575. Lecture Notes in Computer Science. Springer, 2007, pp. 108–131. ISBN: 978-3-540-73489-5.

[30] Pierrick Gaudry. “An Algorithm for Solving the Discrete Log Problem on Hyperelliptic Curves”. In: *Advances in Cryptology – EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 19–34. DOI: 10.1007/3-540-45539-6_2.

[31] Pierrick Gaudry and Robert Harley. “Counting Points on Hyperelliptic Curves over Finite Fields”. In: *Algorithmic Number Theory – ANTS 2000*. Ed. by Wieb Bosma. Vol. 1838. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 313–332. DOI: 10.1007/10722028_18.

[32] Pierrick Gaudry, David Kohel, and Benjamin Smith. “Counting Points on Genus 2 Curves with Real Multiplication”. In: *Advances in Cryptology – ASIACRYPT 2011 (Seoul, South Korea)*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Heidelberg: Springer, 2011, pp. 504–519. DOI: 10.1007/978-3-642-25385-0_27.

[33] Pierrick Gaudry and Éric Schost. “A Low-Memory Parallel Version of Matsuo, Chao, and Tsujii’s Algorithm”. In: *Algorithmic Number Theory – ANTS 2004*. Ed. by Duncan Buell. Vol. 3076. Lecture Notes in Computer Science. Springer, 2004, pp. 208–222. DOI: 10.1007/978-3-540-24847-7_15.

[34] Pierrick Gaudry and Éric Schost. “Construction of Secure Random Curves of Genus 2 over Prime Fields”. In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan L. Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 239–256. DOI: 10.1007/978-3-540-24676-3_15.

[35] Pierrick Gaudry and Éric Schost. “Genus 2 point counting over prime fields”. In: *Journal of Symbolic Computation* 47.4 (2012), pp. 368–400.

[36] Pierrick Gaudry, Emmanuel Thomé, Nicolas Thériault, and Claus Diem. “A Double Large Prime Variation for Small Genus Hyperelliptic Index Calculus”. In: *Mathematics of Computation* 76.257 (2007), pp. 475–492.

[37] James Lee Hafner and Kevin S. McCurley. “A rigorous subexponential algorithm for computation of class groups”. In: *Journal of the American Mathematical Society* 2.4 (1989), pp. 837–850.

[38] Safuat Hamdy and Bodo Möller. “Security of Cryptosystems Based on Class Groups of Imaginary Quadratic Orders”. In: *Advances in Cryptology – ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lectures Notes in Computer Science. Springer, 2000, pp. 234–247. DOI: 10.1007/3-540-44448-3_18.

[39] David Harvey. “Kedlaya’s Algorithm in Larger Characteristic”. In: *International Mathematics Research Notices* 2007 (Jan. 2007). DOI: 10.1093/imrn/rnm095.

[40] Florian Hess, Gadiel Seroussi, and Nigel P. Smart. “Two topics in hyperelliptic cryptography”. In: *Selected Areas in Cryptography – SAC 2000*. Ed. by Douglas R. Stinson and Stafford Tavares. Vol. 2259. Lectures Notes in Computer Science. Springer, 2001, pp. 181–189.

[41] Kiran S. Kedlaya. “Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology”. In: *Journal of the Ramanujan Mathematical Society* 16 (2001), pp. 323–338.

[42] Kim Laine and Kristin Lauter. “Time-memory trade-offs for index calculus in genus 3”. In: *Journal of Mathematical Cryptology* 9.2 (2015), pp. 95–114.

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

[44] Tanja Lange. "Formulae for Arithmetic on Genus 2 Hyperelliptic Curves". In: *Applicable Algebra in Engineering, Communication and Computing* 15.5 (Feb. 2005), pp. 295–328. DOI: 10.1007/s00200-004-0154-8.

[45] Jonathan Lee. *The security of Groups of Unknown Order based on Jacobians of Hyperelliptic Curves*. Cryptology ePrint Archive, Report 2020/289. https://eprint.iacr.org/2020/289. 2020.

[46] Helger Lipmaa. "Secure Accumulators from Euclidean Rings without Trusted Setup". In: *Applied Cryptography and Network Security – ACNS 2012*. Ed. by Feng Bao, Pierangela Samarati, and Jianying Zhou. Vol. 7341. Lecture Notes in Computer Science. Springer, 2012, pp. 224–240. DOI: 10.1007/978-3-642-31284-7_14.

[47] Alfred Menezes, Yi-Hong Wu, and Robert J. Zuccherato. *An Elementary Introduction to Hyperelliptic Curves*. Appendix in Algebraic Aspects of Cryptography by Neal Koblitz, Springer-Verlag, 1998, pages 155-178. 1996.

[48] David Mumford. *Tata Lectures on Theta II*. Birkhäuser, Jan. 2007, pp. 207–213. DOI: 10.1007/978-0-8176-4578-6_13.

[49] Koh-ichi Nagao. *Improvement of Thériault Algorithm of Index Calculus for Jacobian of Hyperelliptic Curves of Small Genus*. Cryptology ePrint Archive, Report 2004/161. https://eprint.iacr.org/2004/161. 2004.

[50] Jonathan Pila. "Frobenius maps of abelian varieties and finding roots of unity in finite fields". In: *Mathematics of Computation* 55.192 (1990), pp. 745–763.

[51] Nicole L. Pitcher. "Efficient point-counting on genus-2 hyperelliptic curves". PhD thesis. University of Illinois at Chicago, 2009, p. 124. ISBN: 9781109242188.

[52] Ronald L. Rivest, Adi Shamir, and David A. Wagner. *Time-lock puzzles and timed-release crypto*. Technical Report MIT/LCS/TR-684. 1996.

[53] Hans-Georg Rück. "On the Discrete Logarithm in the Divisor Class Group of Curves". In: *Math. Comput.* 68.226 (Apr. 1999), pp. 805–806. ISSN: 0025-5718. DOI: 10.1090/S0025-5718-99-01043-1.

[54] Tomas Sander. "Efficient Accumulators without Trapdoor Extended Abstract". In: *Information and Communication Security – ICICS 1999*. Ed. by Vijay Varadharajan and Yi Mu. Vol. 1726. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, pp. 252–262. DOI: 10.1007/978-3-540-47942-0_21.

[55] René Schoof. "Elliptic curves over finite fields and the computation of square roots mod p". In: *Mathematics of Computation* 44.170 (1985), pp. 483–494.

[56] René Schoof. "Counting points on elliptic curves over finite fields". In: *Journal de Théorie des Nombres de Bordeaux* 7.1 (1995), pp. 219–254. URL: http://www.numdam.org/item/JTNB_1995__7_1_219_0/.

[57] Benjamin Smith. "Isogenies and the Discrete Logarithm Problem in Jacobians of Genus 3 Hyperelliptic Curves," in: *Journal of Cryptology* 22.4 (Oct. 2009), pp. 505–529. ISSN: 1432-1378. DOI: 10.1007/s00145-009-9038-1.

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

[59] Andrew V. Sutherland. "A generic approach to searching for Jacobians". In: *Mathematics of Computation* 78.265 (2009), pp. 485–507.

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

[61] Nicolas Thériault. "Index Calculus Attack for Hyperelliptic Curves of Small Genus". In: *Advances in Cryptology – ASIACRYPT 2003*. Ed. by Chi-Sung Laih. Vol. 2894. Lecture Notes in Computer Science. Springer, 2003, pp. 75–92. DOI: 10.1007/978-3-540-40061-5_5.

[62] Annegret Weng. "A class of hyperelliptic CM-curves of genus three". In: *Journal of the Ramanujan Mathematical Society* 16.4 (Jan. 2001), pp. 339–372.

[63] Annegret Weng. "Constructing hyperelliptic curves of genus 2 suitable for cryptography". In: *Mathematics of Computation* 72.241 (2003), pp. 435–458.

[64] Benjamin Wesolowski. "Efficient Verifiable Delay Functions". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. Lecture Notes in Computer Science. Springer, 2019, pp. 379–407. DOI: 101007/978-3-030-17659-4_13.