Trustless Groups of Unknown Order with Hyperelliptic Curves

Samuel Dobson and Steven D. Galbraith

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

February 19, 2020

Abstract

We suggest an alternative method of generating groups of unknown order for constructions such as cryptographic accumulators, by using the Jacobian groups of hyperelliptic curves (especially of genus 3). We propose that this construction is more efficient in practice than other proposals such as the use of ideal class groups of imaginary quadratic fields. We suggest that both the group operation and the size of element representation is an improvement over class groups. We also offer potential parameter choices for security with respect to best current algorithms for calculating the order of these groups.

1 Introduction

Groups of unknown order have several applications, such as in delay functions [BBF18] and cryptographic accumulators. Accumulators are a construction which allow efficient representation of a set of elements in a much smaller (ideally constant) size than the set itself. Given an accumulator, a membership witness is used to prove that an element is indeed a part of the set represented - with appropriate soundness and completeness properties. Some accumulators also provide additional functionality such as dynamic insertion of new elements into the underlying set, deletions of elements, and non-membership proofs.

Accumulators have attracted attention in recent years due to their potential in blockchain situations. Specifically, there have been proposals to represent the current state of the bitcoin blockchain as a small size accumulator rather than as an unbounded set of unspent transaction outputs (UTXOs) [Dry19, BBF19]. Spending of a UTXO would then require a proof that the UTXO is indeed in the accumulator.

One popular construction is known as the RSA accumulator, where an odd prime p can be accumulated by raising the current state of the accumulator $A_i = A_0^S$ to that prime power, producing a new accumulator $A_{i+1} = A_i^p$. The membership of p in A_{i+1} is witnessed by A_i (membership verification is done via $A_i^p \stackrel{?}{=} A_{i+1}$). An actor who knows the order of the empty accumulator as a group element, can efficiently create membership proofs for elements not contained in the accumulator, a violation of the accumulator security. Thus a group of unknown order is required. Previous suggestions have discussed using a trusted setup to generate such a group of unknown order (i.e. generating an RSA modulus with secret factorisation), or using class groups of imaginary quadratic orders.

In this work we propose an alternative method of obtaining a group of unknown order, without trusted setup, using hyperelliptic curves of genus 2 or 3. This construction is more efficient than using class groups, both in size of element representation (a factor of 4) and efficiency in computing the group operation. We acknowledge that there are potentially polynomial-time algorithms to compute the group order of Jacobians of genus 2 and 3 curves [Pil90, GH00]. However, there is evidence that these algorithms are not practical for

cryptographically-sized parameters and so these groups may be secure for the applications we have in mind. Both genus 2 and 3 curves can be considered, but we suggest that genus 3 are a safer choice since the point counting algorithms become more complex in that case.

1.1 Acknowledgements

We would like to thank Dan Boneh and Benjamin Wesolowski for their valuable comments, feedback and suggestions on this work.

2 Preliminaries

Notation. The $\tilde{O}(x)$-notation is defined as $\tilde{O}(x) = O((\log x)^c \cdot x)$ for some constant c.

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 somehow 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, the group operation should be able to be computed efficiently, and it should be possible to represent elements of the group in a practical form. Despite not knowing the order of the group, it should also be possible to generate random elements in 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 the 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 $u, w \in G$ 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 $\ell \in \text{Primes}(\lambda)$ to the prover.
2. The prover computes $q = \lfloor x/\ell \rfloor$ and sends $Q = u^q \in G$ 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, we shall take the example of $-1 \in G$ (a known element of order 2). 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 setting $Q' = -1 \cdot Q$ in the proof. Then, because ℓ is odd, $-1 \cdot Q^\ell u^r = -w$ will hold despite the fact that $u^x \neq -w$. That is why when using the RSA group, it is important to take it as a quotient over $\langle -1 \rangle$.

Previous work with groups of unknown order has primarily centred around two ideas - RSA groups [RSW96], and class groups of imaginary quadratic orders [BW88]. The former is a group of the form $(\mathbb{Z}/N\mathbb{Z})^*$ for N the product of two primes pq. Calculating the order of this group is equivalent to factorising N. The main drawback of this method is that it requires a trusted setup to generate N (or at least trust that the factorisation of N being used is not known to anyone / has been lost). Class groups do not require a trusted setup to generate, thus have received a lot of attention (for example [Wes19, BBF19, BH01]), and are discussed more below in section 2.3.

Brent [Bre00] briefly mentioned using the Jacobian of a hyperelliptic curve as a group of unknown order. This follows work by Koblitz [Kob89] on the use of Jacobians as groups in which the discrete logarithm problem is infeasible. Unlike the use of class groups of imaginary quadratic fields, this Jacobian idea has received very little further attention.

2.2 Accumulators

Accumulators were originally introduced by Benaloh and de Mare [BdM94], who defined the RSA accumulator. This accumulator was made dynamic by Camenisch and Lysyanskaya [CL02], and universal through the work of [LLX07]. See also [BP97].

An accumulator should be representable with small, ideally constant size, relative to the size of the set they represent. A membership witness for an element x allows anyone with the accumulator to verify that x is indeed in the accumulator. There are a number of additional properties which accumulators may have, giving flexible use in various situations. For example, a dynamic accumulator is one which allows insertion and deletion of accumulated elements, whereas a static accumulator supports neither operation (the set must be fixed at the start). A universal accumulator is one which also supports proofs of non-membership.

When discussing the security of cryptographic accumulators, a few important properties shall be defined. An accumulator is **complete** if for any instance of the accumulator A, any element x and any valid membership or non-membership proof (if x is in A or not, respectively) for x in A, verification of the proof with x and A returns true. An accumulator is **sound** if for any instance of the accumulator A and element x not in the accumulator, it is infeasible for an adversary to generate an accepting proof of membership for x in A, and also for an element y which is in A, it is infeasible for an adversary to generate an accepting proof of non-membership for y in A. Finally we define undeniability which is often used to define a secure accumulator:

Definition 1 (Undeniability) *An accumulator is undeniable if, for any given empty accumulator A_0 generated with security parameter λ, and any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can produce a new accumulator state A, an element x, and both a proof of membership for x in A* ***and*** *a proof of non-membership for x in A, is negligible in λ.*

Lipmaa [Lip12] proposed the static root accumulator as an alternative to the RSA accumulator, without a trusted setup, using groups of unknown order (the class group of an imaginary quadratic field). Further work expanding on this idea of unknown order groups was done by Boneh et al. [BBF19], who propose a dynamic accumulator with efficient (non)membership proof aggregation using groups of unknown order.

It is possible to use the quotient group $(\mathbb{Z}/N\mathbb{Z})^*/\{-1,1\}$ where N is an RSA modulus, to construct an accumulator. But this is only secure if a trusted setup process is used to generate N in such a way that no adversary knows it. Trusted setup is an undesirable property in some settings (including the bitcoin UTXO setting), hence the proposals of class groups.

2.3 Ideal and form class groups

The ideal class group of an imaginary quadratic field has been suggested in the literature as a suitable group of unknown order. We now recall some background theory on these groups. Good references for this section include Cohen [Coh10] and Cox [Cox89].

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

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

where $d < 0$ is a square-free integer of degree two (i.e. d satisfies a quadratic form $f(d) = 0$). The discriminant of K is $\Delta = d$ if $d \equiv 1 \pmod 4$, or $\Delta = 4d$ otherwise ($\Delta \equiv 0, 1 \pmod 4$ since only 0 and 1 are quadratic residues mod 4). Also, 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.

Denote by J_K 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 $(\alpha), (\beta)$. 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 $\mathcal{O}_K$).

In practice, it is most efficient to represent $Cl(\mathcal{O}_K)$ using binary quadratic forms. The ideal class group of an imaginary quadratic field of discriminant $\Delta < 0$ is isomorphic to the **form class group** of the same discriminant. We shall 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 $\Delta = b^2 - 4ac$. Note that for $Cl(\Delta)$, we can represent forms using only two elements (a, b) because then 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. Two forms f, g are considered equivalent, $f \sim g$, if there exists a matrix $\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z})$ 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. Lagrange, and later Gauss and then Zagier, gave algorithms for finding an equivalent reduced form for any binary quadratic form. 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 $b \geq 0$ (Definition 5.3.2 of [Coh10]).

The group law in the form class group 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 $\mathfrak{ab} = \{\sum a_i b_i \,|\, a_i \in \mathfrak{a}, b_i \in \mathfrak{b}\}$). We shall not give the algorithm explicitly here, the reader can consult [Coh10] for more information. The group law on the classes is well-defined but does not usually result in a reduced form, so reduction is an additional step. The identity in $Cl(\Delta)$ is the equivalence class of the principal form $((1, 0, -k)$ when $\Delta = 4k$, $(1, 1, k)$ when $\Delta = 4k + 1)$. Refer to Proposition 5.3.3 of [Coh10].

2.3.1 Cryptographic use

The order of a class group of a negative prime discriminant Δ, where $\Delta \equiv 1 \pmod 4$ and Δ is sufficiently large, is believed to be difficult to compute. Thus simply by selecting a suitable Δ, the class group can be used as a group of unknown order without a trusted setup. This is the idea presented by Wesolowski [Wes19]. In practice, as discussed above, the equivalent form class group $Cl(\Delta)$ would be used rather than computing with ideals.

The use of class groups in cryptography was first suggested by [BW88]. A sub-exponential algorithm for computing the order of the class group was given by [HM89] - for a class group of discriminant Δ, the complexity is $L_{|\Delta|}(1/2)$. To generate a class group of unknown order, we must randomly select an appropriate Δ value, and then give a chosen element in the group which we shall treat as a generator (it is not possible to know if it really is a generator of the whole group or not, which we shall discuss below).

2.4 Hyperelliptic curves

Good references for the theory of hyperelliptic curves are [MWZ96] and [Gal12].

Let k be a field with algebraic closure $\overline{k}$. A hyperelliptic curve C of genus g is a curve of the form $y^2 + h(x)y = f(x)$, where $h, f \in k[x]$, h has degree at most g, and f is monic with $\deg f = 2g + 1$ (we ignore the other general case where $\deg f = 2g + 2$). C has no singular points, that is, no points $(u, v) \in \overline{k} \times \overline{k}$ that satisfy both C and its partial derivatives. For an extension field K/k, we denote by $C(K)$

the set of points $\{P \mid P \in K \times K\}$ satisfying C (the **finite** points), together with the projective point at infinity $\infty \in P^2(k)$. Unlike (genus 1) elliptic curves, these points do not form a group. Instead, the group we use is the class group of degree 0 divisors of the curve, also known as the Jacobian.

For a point P, recall that the opposite point $\tilde{P} = (x, -y - h(x))$. The opposite of the point at infinity is itself, $\infty = \tilde{\infty}$. Points which are their own opposite in this way are **special**, otherwise they are **ordinary**.

The coordinate ring of a hyperelliptic curve 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 $\overline{k}[C]$ (an integral domain) are called polynomial functions on C. Every polynomial function $G(x, y)$ can be written in the form $a(x)-b(x)y$, $a, b \in \overline{k}[x]$. The conjugate of G, written as $\overline{G}$, is $a(x) + b(x)(h(x) + y)$. The **norm** $N(G) = G\overline{G}$.

The function field $K(C)$ of C over field K is the field of fractions of the coordinate ring $K[C]$. Take $R \in \overline{k}(C)$ and $P \neq \infty \in C(\overline{K})$. We say that R is defined at P if and only if there exists polynomial functions $G, H \in \overline{k}[C]$ such that $R = G/H$ and $H(P) \neq 0$ (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 points on C. The divisors form a group Div C. The degree of a divisor is $\deg D = \sum m_P$. The divisors of degree zero $\mathrm{Div}^0(C)$ is a proper subgroup of $\mathrm{Div}(C)$. A **principal** divisor is a divisor of the form $(\gamma) = \sum_{P \in C} m_P P$ for a function $\gamma \in \overline{k}(C)$, where $m_P = \mathrm{ord}_P(\gamma)$ is the order of the zero or pole 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 $\mathcal{P}(C) < \mathrm{Div}^0(C)$. So we may define the quotient group

$$\mathrm{Jac}(C) \cong \mathrm{Div}^0(C)/\mathcal{P}(C) \tag{1}$$

This is known as the (degree 0) Picard group or Jacobian of C, also sometimes denoted by Pic^0. Two divisors $D_1, D_2 \in \mathrm{Div}^0$ are equivalent if $D_1 - D_2 \in \mathcal{P}(C)$. For each element in this group, it is possible to associate to it a unique equivalent divisor in $\mathrm{Div}^0(C)$ called a **reduced divisor**. It is these reduced divisors which are used for computation in this group. A reduced divisor is one of the form $D = \sum m_i P_i - (\sum m_i)\infty$, where all the $m_i \geq 0$ and the points P_i are finite. If $m_P \neq 0$ for some P, then either $m_{\tilde{P}} = 0$ if P is ordinary or $m_{\tilde{P}} = m_P = 1$ if P is special. Finally, $\sum m_i \leq g$. If all these conditions hold, the divisor is reduced.

Representing reduced divisors is done with Mumford representation [Mum07]. A reduced divisor is given uniquely as a pair of polynomials $\langle u(x), v(x) \rangle$, where $\deg v < \deg u \leq g$ and u divides $v^2 + hv - f$ [Can87]. 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 $\mathbb{F}_q$.

If we have a curve C of genus g over the finite field of cardinality q, the $\mathbb{F}_q$-rational points of the Jacobian form a finite group of order bounded by $(\sqrt{q}-1)^{2g} \leq \#\mathrm{Jac}(C) \leq (\sqrt{q}+1)^{2g}$, by the Hasse-Weil bound.

To compute the group law on the elements of the Jacobian of a curve, Cantor gave an algorithm in 1987 [Can87]. See also [CL12]. More efficient explicit formulae have been given for genus 2 (see [Lan05]) and 3 (see [FWG07]).

On elliptic curves, it is well known that division polynomials exist for all $n \in \mathbb{Z}_+$, where the roots of the n-th division polynomial are the points with order n, i.e. $E[n]$. The hyperelliptic curve analogue of these polynomials are division ideals - homogenous ideals vanishing on the torsion subgroups. In this sense, the genus one division ideals are the principal ideals generated by the n-division polynomials. But in general genus, there does exist a system of equations for each ℓ whose solutions give points/divisors of order ℓ [Can94, CFA+05]. A useful result of [Abe18] is that in genus 3, the degrees of Cantor's ℓ-division polynomials are bounded by $O(\ell^2)$.

3 Security of genus 2 and 3 curves

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 curves, calculating the order of the Jacobian is equivalent to computing the zeta function of the curve and thus to counting points on the curve (more information can be found in [CFA+05]). We now survey some previous work in this area, in order to suggest feasible parameter sizes for secure constructions. Note that if the DLP can be efficiently solved, then the group order can also be calculated, because if we solve the DLP for $xG = \mathcal{O}$ (the discrete log of the identity with respect to the group generator), then we find the order of the generator which is a divisor of the size of the group. Thus, previous work on solving the DLP is also important here. See also the discussion by Avanzi, Thériault, and Wang [ATW08].

For large genus, there exist sub-exponential attacks on the DLP, for example [Eng02]. However this does not impact the small genus 2 and 3 case. Gaudry et al. [GTTD07] (and also Nagao [Nag04]) presents algorithms for the discrete logarithm problem on low genus g curves, in time $\tilde{O}(q^{2-2/g})$. This improves over [Gau00] which gave an $O(q^2)$ algorithm, and the single large prime algorithm of [Thé03].

Theorem 1 (Theorem 1, [GTTD07]) *Let $g \geq 3$ be fixed. Let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$ such that the Jacobian group $Jac(C)$ is cyclic. Then the discrete logarithm problem in $Jac(C)$ can be solved in expected time*

$$\tilde{O}\left(q^{2-\frac{2}{g}}\right)$$

as q tends to infinity.

This has better performance for genus 3 than "square root" (of the group order) type algorithms like Pollard's Rho algorithm with complexity $\tilde{O}(q^{1.5})$. In genus 2, Pollard's Rho algorithm is more efficient ($\tilde{O}(q)$).

The base field should be chosen with large prime characteristic, because for small characteristic, algorithms such as that by Kedlaya [Ked01] are possible.

Smith [Smi09] gives a method of translating the DLP on a hyperelliptic curve into that on a non-hyperelliptic curve for 18.57% of genus 3 curves. In these cases, the algorithm by Diem [Die06] can then be used to solve the DLP in time $\tilde{O}(q)$. Laine and Lauter [LL15] improve upon and examine this attack by Diem, including analysis of the logarithmic factors involved, which they estimate to be $(\log_2(q))^2$. They point out, however, that the memory requirement for their attack is high - $\tilde{O}(q^{3/4})$. See also [Lai15], which additionally discusses ways to generate genus 3 curves which avoid isogeny attacks.

In previous work on generating secure hyperelliptic curves for cryptography, a focus was put on generating curves such that the order of the Jacobian had specific properties preventing known attacks. For example, it should have a large prime factor, to avoid attacks such as Pohlig-Hellman. The largest prime factor should also not divide $q^k - 1$ for small k (for example if the curve is supersingular), to avoid MOV-type attacks [FR94]. Additionally, the curve should not have a $p = \text{char } q$ order subgroup over $\mathbb{F}_p$ (it should not be an anomalous curve) [Rüc99]. In the case of groups of unknown order, though, it is (by definition) not possible to know whether the order is resistant to these attacks or not. But fortunately the smooth numbers are very rare, so with very high probability a randomly generated hyperelliptic curve's Jacobian will have a large prime in its order factorisation. This is the same assumption that the security of ideal class groups of unknown order rely on too [CL84].

There has been previous work done on Schoof-Pila-type [Sch85, Pil90] algorithms in genus 2, for example by Gaudry [GH00, GS04]. The basic idea behind these algorithms is to compute elements of order ℓ using division polynomials/ideals, and then analyse the action of Frobenius on them, repeating this process for multiple ℓ. Often it becomes infeasible to perform this analysis for primes higher than some bound, after

which an exponential baby-step giant-step algorithm is used to complete the algorithm. This work has not been extended to genus 3, with [GH00] citing that in the $g > 2$ case, "the main difficulty is that it does not appear possible to avoid manipulation of ideals." 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] use an improved algorithm following the approach of Pitcher (with improvements in the constant factors of the algorithm) to find a curve of secure order in a 127-bit order field - a mixture of schoof-type and exponential BSGS algorithms. In their experiments, they claim around 1,000 CPU hours to compute the order of a curve over this field.

Also worth mentioning is the work by Sutherland in his PhD thesis [Sut07]. It uses generic group algorithms in an $o(\sqrt{N})$ time algorithm for computing the order of elements in a group. This can then be used to probabilistically identify the exponent of a group, and these methods were used to compute the structure of some class groups. However this algorithm only performs well if the order of the group is very smooth, which we do not expect or desire in a group of unknown order.

Previous work has been concerned with the generation of a curve with a secure, known order, for example [GS12] and [HSS01]. [BCHL13] states that "one significant hurdle for genus 2 cryptography to overcome is the difficulty of generating secure genus 2 curves: that is, such that the Jacobian has a large prime or almost prime group order." One such method, for example, is using complex multiplication (CM). Weng [Wen01, Wen03] discuss ways to generate hyperelliptic genus 2 and 3 curves with a prescribed number of $\mathbb{F}_q$-rational points on its Jacobian. Because we wish to trustlessly ensure that the group order is unknown to all parties, these methods should be avoided, and the curve should be generated in a nothing-up-my-sleeve type manner as discussed in Section 4. In this use case, as long as the order is infeasible to compute, it does not matter exactly what it is.

Thus to be fairly conservative, it would be safe to assume an $\tilde{O}(q)$ algorithm for finding the order of the group in the genus 3 case.

3.1 Constructing points of known order

It may be possible to construct points of small, known order on hyperelliptic curves using division ideals. Thus the adaptive root assumption is not met. For example, in the same way that -1 would allow $\mathrm{PoE}(g, -y, x)$ to validate in the multiplicative RSA group [Wes19], here a point of order 2, P, would allow a false proof of $\mathrm{PoE}(G, Y + P, x)$ where $[x]G = Y$.

Assuming there exists no feasible Schoof-type algorithm for counting points on genus 3 curves as above, we implicitly assume that it becomes infeasible to construct points of order larger than some bound s:

Assumption 1 *For a hyperelliptic curve of genus 2 or 3 over a finite field $\mathbb{F}_p$ for sufficiently large p, we assume there exists an $s \in \mathbb{Z}_{>0}$ such that it is infeasible for anyone to compute roots of the ℓ-th division polynomials for $\ell > s$.*

With respect to Assumption 1 we suggest that the group $[N]\mathrm{Jac} = \{[N]P \mid P \in \mathrm{Jac}(C)\}$, where $N = s!$ is a smooth integer chosen to kill off all points of small order, is compatible with the adaptive root assumption.

Another important, related observation is that computing repeated divisions by 2 allows the computation of a point of arbitrary 2^k order. This is clearly coprime to all odd primes, so would allow a malicious prover to easily to find ℓ-th prime roots (given a point Q, find P such that $\ell P = Q$, ℓ an odd prime). Finding square roots repeatedly will almost always cause the field of definition of these roots to blow up (requiring repeated quadratic extensions) as this continues. By ensuring that the hyperelliptic curve generated was of the form $y^2 = f(x)$ for $f(x)$ irreducible above, we ensured that these square roots do not exist. But very similarly, repeated powers of other small primes ≥ 3 can still possibly be calculated. Using the group $[N]\mathrm{Jac}(C)$ will certainly, with very high probability, kill off powers of these low primes dividing the group order. It could

also 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 N above). It is a potential interesting open problem if it is possible to generate an easily verifiable proof that a curve does not have any points of low order.

The issue now is that testing membership of an element in the group is not easy. The original point Y is effectively a proof that $[N]Y$ lies in $[N]$Jac because this can be easily verified. So Y should be sent instead of $[N]Y$ in the protocol, and the verifier can perform the multiplication by N themselves. See Section 4 for a illustrative example in the case of accumulators.

It is worth noting that the impact of being able to find small-order points is highly domain-specific. For example, in the case of a VDF [Wes19, BBF18], to forge a false PoE even in the case that points of known order can be found still relies on knowing the true result of the exponentiation. Thus, this would still require computing the output of the VDF (although relying on a PoE would break the requirement that the VDF output is unique). In the accumulator case, we need the strong RSA assumption rather than the adaptive root assumption, which states that 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 alternatively, simply by disallowing the accumulation of small primes $\ell < s$. Finding roots where $\ell > s$ is hard by Assumption 1 above. So here we do not even require the use of $[N]\text{Jac}(C)$.

The use of $[N]$Jac can impact on efficiency in some ways, due to the extra multiplications required. This is highly protocol-dependent, but in most cases, more than a few extra multiplications should not be needed. For example, in the case of the PoE protocol below, the verifier only needs to perform one extra multiplication by N during verification, compared to the original protocol in $\text{Jac}(C)$. We suggest that this is still efficient enough for practical usage.

3.2 Testing small divisors of orders with the Tate pairing

A good reference for the background of pairings on hyperelliptic curves is [GHV07].

If we can find points of known small order ℓ, the Tate pairing can be used to learn information about the order of other points - specifically whether they are divisible by ℓ or not.

Let k be a positive integer such that $r|q^k - 1$, and let J be the Jacobian of a hyperelliptic curve over F_{q^k}. The reduced Tate pairing is a mapping

$$T_r : J[r] \times J/rJ \to \mu_r$$

Where μ_r is the set of r-th roots of unity. If we have a point Q of small known prime order ℓ (i.e. $Q \in J[\ell]$), then consider the ℓ-th reduced Tate pairing T_ℓ. When paired with any other point P, we get $T_\ell(Q, P) \in \mu_\ell$.

Now if $\ell \nmid \text{Ord}(P)$, then $P = \ell P'$ for some P', thus $P \in \ell J$. And if $P \in \ell J$, then $T_\ell(Q, P) = 1$ for any $Q \in J[\ell]$. So certainly, if the order of P is not divisible by ℓ, the pairing will give the identity. Equivalently, a non-identity result from the pairing always implies the order of P is divisible by ℓ.

Unfortunately, the converse is not so simple. $t_\ell(Q, P) = 1$ alone is not sufficient to imply that the order of P is coprime to ℓ. Rather than knowing only a single point Q of order ℓ, to guarantee the converse and prove the coprimality, it must be shown that $t_\ell(Q_i, P) = 1$ for all $Q_i \in J[\ell]$. Thus we require a basis $Q_1, Q_2 \in J[\ell]$ which we can test. If $t_\ell(Q_i, P) = 1$ for both $i = 1, 2$, then the bi-linearity of the Tate pairing implies it holds for all $Q \in J[\ell]$, and thus that $gcd(|P|, |Q_i|) = 1$.

Finally, we observe that $t_\ell(Q, P) \in \mathbb{F}^*_{q^k}$, which as an extension, blows up as ℓ increases. So we assert

that Assumption 1 still makes this an infeasible approach to learn any significant amount of information about the orders of random points in the Jacobian, or any information at all in $[N]\text{Jac}$.

4 Construction of accumulators from hyperelliptic curves

We suggest that hyperelliptic curves can be used as a more efficient alternative to the ideal/form class group in situations where a group of unknown order is required. Here, we specifically propose this for the accumulator use-case. After choosing appropriate genus (2 or 3) and order q of the base field $\mathbb{F}_q$ - which will be discussed in Section 5 - a random non-supersingular hyperelliptic curve over that field can be generated (see below). The Jacobian of this curve can be used as a group, while even the person generating the curve cannot feasibly calculate the order of this group. Algorithms related to the calculation of the order are discussed in the next section.

The construction proceeds as follows. First, a large enough prime p should be randomly chosen as the order of the base field $\mathbb{F}_p$. To define the curve itself, it suffices that the polynomial $h(x) = 0$ (i.e. the curve will have the form $y^2 = f(x)$), and the coefficients of f be randomly selected in $\mathbb{F}_p$. If we ensure that f is irreducible over $\mathbb{F}_p$, it is guaranteed that the group will have no points of order 2.

After generating a random curve over $\mathbb{F}_p$, we also require a chosen element of its Jacobian to act as a generator point for the group of unknown order. This can be done in a very similar manner, choosing random coefficients for $u(x)$ as the first polynomial in the mumford representation of a divisor. It can then be checked whether a polynomial $v(x)$ exists which satisfies $v^2 - f$ (recovering v can be done in the same way as if from the compressed form a divisor [HSS01]). Of course, the order of the randomly selected point will also be unknown so we cannot know whether it does generate the full group or a subgroup of it. But as discussed more below, there is a high probability that a randomly selected element will not have a low order - so it will generate at least a large order subgroup of the Jacobian.

To ensure that not even the creator of the curve knows its order and that it was indeed generated randomly, we suggest that the coefficients should be chosen in some deterministic "nothing up my sleeve" type manner. This applies to both the function $f(x)$ and the specially chosen divisor polynomial $u(x)$. For example, the coefficients can be taken from the hash of a certain string. The resulting group and generator should then be groups whose order is unknown to all - a trustless setup. This can then be used in constructions such as accumulators, VDFs, and so on. Some important security considerations are discussed in the next section, notably concerning points of small or known order.

Overall, the generation of a new hyperelliptic curve is also 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.

With respect to Assumption 1, the accumulator should use the group $[N]\text{Jac}(C)$ for the newly generated hyperelliptic curve C. The operation of the accumulator within that group is standard, but some protocols will need modification - for example, proofs may require an extra check that an element is indeed in the group. We shall specifically focus on Wesolowski's Proof of Exponentiation (PoE) [Wes19] for illustration, but expect that other protocols and uses of the adaptive root assumption can similarly be modified.

We begin the PoE protocol with input $U \in \text{Jac}(C)$, $W \in [N]\text{Jac}$, and $x \in \mathbb{Z}$, with the claim that $[x][N]U = W$ in $[N]\text{Jac}$.

1. Verifier sends a random prime $(\ell > s) \in \text{Primes}(\lambda)$ to the prover.
2. Prover computes $q = \lfloor x/\ell \rfloor$ and sends $Q = [q]U \in \text{Jac}(C)$ to the verifier.

3. The verifier computes $r = x \bmod \ell$, and accepts if $Q \in \mathrm{Jac}(C)$ and $[N]([\ell]Q + [r]U) = W$

Observe that this modified protocol is compatible with Assumption 1. Specifically, given a valid proof of $[x][N]U = W$, in order to falsely prove $[x][N]U = W + P$ the prover has to be able to find $[1/\ell]P$ 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 polynomial. By Assumption 1, though, this becomes infeasible if $\ell > s$, so even if it is feasible to find points of known order or roots of elements in $\mathrm{Jac}(C)$, it is infeasible to do so in $[N]\mathrm{Jac}(C)$.

Other than the use of a different group (and the above-mentioned additional checks of group membership), the operation of the accumulator on the group is very similar to previous proposals.

5 Concrete parameter choices

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 $\tilde{O}$ complexity, for simplicity we will assume no logarithmic factors - which also gives a more conservative analysis.

Choosing parameters $q \sim 2^{100}$ in the genus $g = 3$ case appears sufficient. 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 $\tilde{O}(2^{133})$.

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 $\deg v < \deg u < g$, the element can be stored concretely as just 5 coefficients of polynomials - a total of 5 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. 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 $\tilde{O}(\ell^6 \log q)$ and memory as $\tilde{O}(\ell^4 \log q)$, 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 $\simeq 2^{95}$ 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 4. 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.

6 Conclusion

The trustless setup of accumulators is especially important in the stateless-blockchain setting, discussed by Boneh, Bünz, and Fisch [BBF19]. Here, it is highly unfavourable to have a trusted setup, as the security model of such systems discourages trust of any third party. Thus, an improvement to the practical efficiency of the trustless setup case with groups of unknown order is a useful advancement in this setting. We propose the use of Jacobian groups of genus 3 hyperelliptic curves as more efficient, practical alternatives in the construction of unknown-order groups. This idea also applies to other constructions relying on such groups.

References

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

[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 Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 561–586, Cham, 2019. Springer International Publishing.

[BCHL13] Joppe W. Bos, Craig Costello, Huseyin Hisil, and Kristin Lauter. Fast cryptography in genus 2. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 194–210, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BdM94] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[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.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 480–494, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

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

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 61–76, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[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, Berlin, Heidelberg, 2012. Springer-Verlag.

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

[Cox89] D.A. Cox. *Primes of the Form X2 + Ny2: 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 Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory*, pages 543–557, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Dry19] Thaddeus Dryja. Utreexo: A dynamic hash-based accumulator optimized for the bitcoin utxo set. Cryptology ePrint Archive, Report 2019/611, 2019. `https://eprint.iacr.org/2019/611`.

[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 Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 19–34, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

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

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

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

[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.

[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 Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, pages 234–247, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[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. Springer, 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.

[Lip12] Helger Lipmaa. Secure accumulators from euclidean rings without trusted setup. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *Applied Cryptography and Network Security*, pages 224–240, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[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.

[LLX07] Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 253–269, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[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.

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

[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 Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 379–407, Cham, 2019. Springer International Publishing.