CM55: special prime-field elliptic curves almost optimizing den Boer's reduction between Diffie–Hellman and discrete logs

Daniel R. L. Brown[*]

February 24, 2015

Abstract

Using the Pohlig–Hellman algorithm, den Boer reduced the discrete logarithm problem to the Diffie–Hellman problem in groups of an order whose prime factors were each one plus a smooth number. This report reviews some related general conjectural lower bounds on the Diffie–Hellman problem in elliptic curve groups that relax the smoothness condition into a more commonly true condition.

This report focuses on some elliptic curve parameters defined over a prime field size of size $9 + 55 \times 2^{288}$, whose special form may provide some efficiency advantages over random fields of similar sizes. The curve has a point of Proth prime order $1 + 55 \times 2^{286}$, which helps to nearly optimize the den Boer reduction. This curve is constructed using the CM method. It has cofactor 4, trace 6, and fundamental discriminant -55.

This report also tries to consolidate the variety of ways of deciding between elliptic curves (or other algorithms) given the efficiency and security of each.

Contents

[*]Certicom Resarch. Contact: `dbrown@certicom.com`

1 Introduction

An elliptic curve, defined over a prime field $\mathbb{F}_p$ of size $9+55\times 2^{288}$ and having a ($\mathbb{F}_p$-rational) point with order $1+55\times 2^{286}$, which is a Proth prime, can be constructed using the complex multiplication method. The potential benefits of these special curve parameters (CM55) are:

- it makes den Boer's reduction [dB90] between the DLP and DHP almost as tight as possible (even if the DLP becomes weaker than expected);
- the special form of the underlying field may permit more efficient field arithmetic than for average random prime field of similar size;
- the complex multiplication properties of the curve might make the Gallant–Lambert–Vanstone method [GLV01] useful; and
- heuristic rarity properties of the curve possibly reduce the risk of some types of malicious curve selection attacks.

Some potential security risks of the CM55 curve parameters are:

- the significantly lower than average magnitude of its discriminant;
- the significantly lower than average magnitude of its trace;
- the special form of its field;
- larger size than usually deemed sufficient for security (usually 256 bits), with a resulting waste of data transmission and computation;
- possible non-optimally efficient field of the given size, with a resulting waste of computation; and
- a possible vulnerability to an unpublished attack that affects only a rare class of curves including this curve.

This report states a theorem which adjusts den Boer's algorithm in such a way to plausibly drop a smoothness condition making it applicable to most elliptic curve groups.

Appendices to this report discuss ways to compare elliptic curves (and more general cryptographic algorithms), and some possible approaches to efficient implementation of a CM55 curve.

2 Previous Work and Motivation

2.1 den Boer's Reduction between DHP and DLP

At CRYPTO 88, den Boer [dB90] proved that, in certain groups, solving the Diffie–Hellman problem (DHP) is almost as hard solving the discrete logarithm problem (DLP) by giving a reduction algorithm that solves the DLP using an oracle for solving the DHP. Specifically, if a group has a hard discrete logarithm problem and a prime order n such that $n-1$ is smooth, then den Boer's reduction proves that the group has a hard Diffie–Hellman problem, which is necessary for the security of many key agreement schemes, and some other cryptographic schemes.

2.2 Complex Multiplication Method of Generating Elliptic Curves

Atkin and Morain [AM93] introduced the complex multiplication (CM) method for generating elliptic curves of a given order n. It is straightforward to apply the CM method to optimize den Boer's reduction by starting with n such that $n-1$ is quite smooth. To elaborate, set $n = 1+s$ where s is smooth, then apply the CM method to find the curve over the prime field. If applied naively, the CM method results in a curve whose defining field size has the form $p = n + r$ where r is some random-looking integer with $|r|$ with bit-size similar the $\sqrt{n}$, under the heuristic that Cornacchia's algorithm provides random-looking solutions. Therefore, the two main deficiencies of this naive application of the CM method to den Boer's work are the following.

- It results in an elliptic curve with discriminant of lower magnitude than is typical for a random elliptic curve, which is deemed a potential risk of having a easier than usual discrete logarithm problem.
- It results in a field size will have its least significant half of binary expansion essentially random, which is potentially less efficient than is possible for specially selected fields of similar size.

The CM55 curve parameters described in this paper are an attempt to somewhat resolve this latter deficiency.

Remark 2.1. Potentially, field efficiency helps the adversary as much as the user, so to some extent field inefficiency may not be the deficiency it is conventionally deemed. (See §B.7 for further discussion.)

2.3 Earlier Publication of the CM55 Parameters

The CM55 curve parameters were described in patent of Brown, Gallant and Vanstone [BGV07], which also mentioned the benefit of improving den Boer's reduction and the benefit of having a field of special size.

This report elaborates on those observations. In other words, this report dusts off an old curve in the attic, and to give it a second look.

3 Provable Security: Reducing DLP to DHP

In this section, some variants of the den Boer reduction algorithm are stated and proven. These algorithms solve the ECDLP using an oracle for the ECDHP. These reductions show that a hard ECDLP suffices to avoid direct attacks on the ECDHP.

In particular, the reductions rule out the possibility that ECDHP is easy, or at least feasible, while the ECDLP is still hard. Although the possibility of such a gap appears unlikely, a similar-sized gap between the decision DHP and the computational DHP is known to exist for certain groups, including elliptic curve groups with low enough embedding degrees.

Remark 3.1. Some cryptographic schemes, such as many signature schemes including ECDSA, are not known to depend directly on the hardness of the ECDHP. In particular, in an ECDSA-only system, the CM55 curve has little advantage over other curves. (Some alternative signature schemes, such as the Goh–Jarecki signature scheme [GJ03], are related to the ECDHP. So, CM55 may provide some advantages to these alternative signature schemes.)

3.1 A Special Case of den Boer's Algorithm

The following theorem is essentially a special case of a theorem due to den Boer. It makes a positive use of the algorithm first described by Pohlig and Hellman. It turns this PH attack algorithm for solving the discrete logarithm into a reduction between two conjecturally hard problems.

As in den Boer's theorem, we are trying to keep two things low: (1) the number of calls to the DH oracle; and (2) the amount of computation in the algorithm. In den Boer's original theorem the benefit of the second objective helps to avoid making any assumptions about the cost of the discrete logarithm problem, which may make sense in the finite field setting, where index calculus algorithm show promise for improvement.

In the elliptic curve setting, the second objective has a less likely benefit: in the event the discrete logarithm problem becomes significantly easier, but still intractable, the second objective ensures that the theorem is still applicable.

This report next restates and reproves the den Boer theorem, making it specific to the CM55 group order.

Theorem 3.1 (den Boer, special case)**.** *Suppose that G is a group of prime order $n = 1 + 55 \times 2^{286}$, and g is a generator of the group, and multiplicative notation is used for the group operation. Consider the function $M : G \times G \to G : (g^x, g^y) \mapsto g^{xy}$, which outputs Diffie–Hellman shared secrets, and the function $L : G \to \mathbb{Z}_n : g^z \mapsto z$, which outputs discrete logarithms. Then there exists an efficient algorithm A (described in the proof) that computes function L using at most:*

- *286 calls to the function M,*
- *341 calls to each of the following operations:*
  - *an exponentiation in the group G,*
  - *an exponentiation modulo n,*
  - *an exponentiation modulo $n - 1$, and*

– *a Chinese remainder theorem (CRT) calculation modulo $n - 1$.*

Proof. Algorithm A receives input g^z and needs to compute $L(g^z) = z$. Let us use the fact that 3 is a primitive element of the field of integers modulo n. If $g^z = 1$, which A can easily detect, then $z = 0$. Otherwise, $z \neq 0$, and $z = 3^y \bmod n$ for some y whose value matters only modulo $n - 1$. Write:

$$y \equiv (x, w) \bmod (2^{286}, 55) \tag{1}$$

which are the CRT coordinates of y. We will use such CRT coordinates for other values. We will have $0 \leq w < 55$ and $x = x_0 + x_1 2 + \dots x_{285} 2^{285}$ with $x_j \in \{0, 1\}$. Clearly, it suffices to find x and w, since that determines y, and then z.

Let $h_0 = g^z$, which is the input to L, and let $h_j = M(h_{j-1}, h_{j-1})$ for $1 \leq j \leq 286$, which requires 286 applications of the function M. Note that:

$$h_j = g^{z^{2^j}} = g^{3^{y \times 2^j}} = g^{3^{(2^j x, 2^j w)}} \tag{2}$$

Computation of these h_j uses all the calls available to A. The rest of the work can only use the normal multiplication in the group G, which corresponds to addition of the g exponents.

Compute the values:

$$k_i = g^{3^{2^{286} \times i}} = g^{3^{(0, 2^{286} i)}} \tag{3}$$

for $0 \leq i < 55$. Note that these values do not depend on z and can be pre-computed. It follows that

$$h_{286} = k_w \tag{4}$$

So the secret w can be determined by the index w of the value k_w in the list of values k_i that matches h_{286}.

Our next goal is to determine x, which we will do one bit at a time. If $x_0 = 0$, then

$$h_{285} = g^{3^{(0, 2^{285} w)}}. \tag{5}$$

Since we know w, we can compute the value on the right using CRT, modular exponentiation, and the square-and-multiply algorithm for exponentiation in the group G. When we compute this value, if it matches h_{285}, then we conclude $x_0 = 0$, otherwise we conclude $x_0 = 1$. In either case, we have determined x_0. Next, we test if $x_1 = 0$ by computing if

$$h_{284} = g^{3^{(2^{284} x_0, 2^{284} w)}}. \tag{6}$$

Again we can compute the right hand side conventionally. The boolean truth value of (6) matches the boolean truth value of the equation $x_1 = 0$. So we can determine x_0. We find x_2, by using x_0, x_1 and h_{283}. Keep iterating in the same manner to find all the bits x_j.

Note that in addition to the 286 calls to M, algorithm A does about $55 + 286 = 341$ of each of the following operations:

- a CRT calculation for $n - 1$,
- an exponentiation modulo $n - 1$,
- an exponentiation modulo n, and

- an exponentiation in the group G

which is what the theorem claims for A. $\square$

The immediate consequence of Theorem 3.1 is that the cost of the ECDHP is at least $286^{-1} \approx 2^{-8.26}$ times the cost of the ECDLP, approximately, almost regardless of how difficult the ECDLP is. More precisely, this lower bound on the ECDHP holds provided that the cost of the ECDLP dominates the cost of the reduction (the 341 calls to the four operations listed in the theorem). In the next section, we consider the interplay between higher-cost reductions and assumptions of a higher cost for the ECDLP.

Note that the algorithm we used above is simpler than den Boer's original algorithm for a few reasons. First, it works over a prime order group. Second, it mainly takes advantage of one prime factor two (but repeated multiple times). Third, it treats the prime factors 5 and 11 together as a single factor of 55. Fourth, rather than using Shanks' baby-step-giant-step (BSGS) algorithm for each factor, it just uses a table look-up.

In particular, the actual den Boer algorithm would have been more efficient in determining (4), but would have used slightly more calls to M. Following den Boer's approach of using the BSGS algorithm, we could determine k_w using just 15 exponentiations instead of 55 exponentiations, as follows. For $0 \le i < 7$ and $0 \le f < 8$, compute the pair of values

$$k_i = g^{3^{(0,2^{286} \times i)}} \tag{7}$$

$$h_{286,f} = h_{286}^{3^{(0,2^{286} \times 8f)}}, \tag{8}$$

and then find a pair (i, f) with $k_i = h_{286,f}$ which will determine that $w = i - 8f \bmod 55$. This is only a small gain in efficiency, but in the next section, we try to extend this approach to further reduce the calls made to M. Furthermore, instead of using BSGS, we will use Pollard rho.

Remark 3.2. The fact $2^{286} | n - 1$ in Theorem 3.1 makes the tightness of low cost reduction nearly optimal in the sense of using the very few calls to M. In other words, the n is nearly optimal in regards to certain types of security.

Remark 3.3. Theorem 3.1 assumes a perfect DH oracle M. Previous other work, such Shoup's [Sho97], considers the problem of handling imperfect DH oracles in such reductions. A first step in handling imperfect DH would be to blind the inputs, just in case the oracle always fails for the inputs of the kind our algorithm generates. Then some kind of error handling might apply. Future work may fill in the necessary details to adapt these techniques to the improved specific result in Theorem 3.1.

3.2 More Costly Reductions with Fewer Oracle Calls

In this section, a slight variant of the den Boer reduction is described. This variant uses fewer DH oracle calls, but a greater amount of computation. By allowing the amount of computation to approach what one usually conjectures to be the cost of the discrete logarithm problem, one can minimize the number of oracle calls.

This results in a tighter reduction, but relies on an extra assumption, namely about the hardness of the ECDLP. A further benefit is that it works for a much wider class of group sizes, which we discuss briefly after stating the result for the CM55 group. A downside of this approach is that to apply optimally (accurately), one has to be quite (moderately) careful about the complexity of the algorithms. We will use some constants to abstract away some of this level of detail.

It will suffice for stating the theorem to describe the average running time of the Pollard rho as taking $C_1\sqrt{m}$ iterations on average in a cyclic group of size m. (Note: we will not assume that m is prime.)

This time around, we will use additive notation in the proof. This will make it look less like den Boer's original proof, but will make it more like the traditional elliptic curve cryptography notation, which is where we intend to apply the result. It will also make it more like Boneh and Lipton's results [BL96] for the black-box field problem. It will also reduce the power tower notation from three levels of superscripts to two.

Theorem 3.2. *Let $\mathbb{G}$ be a group of prime order $n = 1 + 55 \times 2^{286}$, with additive notation and generator G. Let D be an DH oracle such that $D(aG, bG) = abG$ for all $a, b \in \mathbb{Z}$. Let d be an integer with $1 \leq d \leq 286$. Given $A \in \mathbb{G}$, one can compute $z \in \mathbb{Z}$ such that $A = zG$, using d calls to D, and at most, on average,*

$$(286 - d) + \left(C_1\sqrt{55} \times 2^{143-d/2}\right) \tag{9}$$

calls to each of the following four operations:

- *two pseudorandom functions,*
- *a scalar multiplication in the group $\mathbb{G}$,*
- *an exponentiation modulo n, and*
- *an exponentiation modulo $n - 1$.*

Proof. To test $z = 0$, check if $A = 0G$. Otherwise, there exists an integer y, which the algorithm will try to find, such that $z = 3^y \bmod n$ with $0 \leq y < n - 1 = 55 \times 2^{286}$. Write:

$$y = x(2^{286-d} \times 55) + w \tag{10}$$

for integers x and w with $0 \leq x < 2^d$ and $0 \leq w < m = (n-1)/2^d = 55 \times 2^{286-d}$. The algorithm will next try solve for w using a version of the Pollard rho algorithm, and then solve for x one bit at time, using the usual Pohlig–Hellman algorithm.

Let $H_0 = A$, and let $H_j = D(H_{j-1}, H_{j-1})$ for $1 \leq j \leq d$, which requires d applications of the DH oracle D. Note that:

$$H_j = z^{2^j}G = 3^{y \times 2^j}G. \tag{11}$$

In particular,

$$H_d = 3^{w \times 2^d}G. \tag{12}$$

We will try to run Pollard rho over the m element set:

$$\mathbb{M} = \{3^{2^d \times i}G : 0 \leq i < m - 1\}. \tag{13}$$

For this, will need two pseudorandom functions:

$$f : \mathbb{M} \to \mathbb{Z}_m \tag{14}$$

$$F : \mathbb{M} \to \{G, H_d\} \tag{15}$$

Let $R_0 = G$. For $j \geq 0$, let

$$R_{j+1} = 3^{2^d \times f(R_j)} F(R_j) \tag{16}$$

Compute and track quadruples $(R_{j-1} R_j, R_{2j-1}, R_{2j})$ until $R_j = R_{2j}$, which we can call a Floyd collision. In the event that $F(R_{j-1}) = F(R_{2j-1})$, which should happen with probability about one half, start over with new pseudorandom functions f and F (perhaps changing any keys that the functions use). Eventually, we should get a Floyd collision in which $F(R_{j-1}) \neq F(R_{2j-1})$ and therefore either:

$$3^{2^d f(R_{j-1})} G = 3^{2^d (f(R_{2j-1}) + w)} G; \tag{17}$$

or the same with $j - 1$ and $2j - 1$ swapped. But this means that either,

$$w \equiv f(R_{j-1}) - f(R_{2j-1}) \bmod m \tag{18}$$

or the modular negation. The expected number of R_j that need to be computed is given by $C_1\sqrt{m}$, in accordance with our definition of the constant C_1.

Once w is determined, we proceed similarly to the proof of Theorem 3.1 to determine the bits of x, one at a time. Let $x = x_0 + x_1 + \cdots + x_{d-1}2^{d-1}$. We can determine x_0 by looking at:

$$H_{d-1} = 3^{(x_0 \times 2^{d-1} \times 55) + (w \times 2^{d-1})} G \tag{19}$$

Since we now know w, after having applied Pollard rho, there are only two possible value of x_0, one of which can pick to compute the right-hand side of the equation above. If it matches, then we know our guess at x_0 is correct. Then continue by using H_{d-2} to determine x_1 and so on, finishing by using H_1 to determine x_{d-1}. $\square$

Now we take a common conjecture for elliptic curves in general, and specify it to the CM55 curve(s).

First some more details about the Pollard rho algorithm, in its conventional application to solving the ECDLP. Each iteration of the pseudorandom operation costs about C_2 group operations on average, amortized over the execution of the whole algorithm. I think a series work, especially by Teske [Tes01], addresses empirically and theoretically how low C_2 can go (I hope to update this report accordingly upon reviewing such research). For a concrete example, suppose $C_2 = 2$.

Conjecture 3.3. *All algorithms that solve the ECDLP over the curve CM55 cost at least the equivalent of:*

$$C_3 C_2 C_1 \sqrt{n} \tag{20}$$

group operations, for margin-of-error constant $C_3 > 2^{-16}$.

If Pollard rho is the optimal algorithm for solving the ECDLP, then $C_3 = 1$. But perhaps some other algorithm is better, in which case $C_3 < 1$. The lower bound 2^{-16} in the conjecture is chosen so that the ECDLP in CM55 would fare about as well as an arbitrary curve group with order approximately 2^{256} such that Pollard rho is the fastest ECDLP in this group. This accounts for the suspicion about low magnitude discriminant by estimating any improvement in the ECDLP due to the low discriminant produces only a 2^{16} times speedup in the ECDLP. Of course, it is possible that the low discriminant could have a much greater effect, or even that all elliptic curves are more effected. So, the content of the conjecture is a quantified expression of confidence in the CM55 parameters. The main justification for this confidence, at a qualitative level, is that low discriminant curves have long been suspected, and even used in BitCoin, but no attack has been published. The exact quantitative level of confidence in this conjecture has been selected more for convenience of the CM55 parameters.

Remark 3.4. Gallant, Lambert and Vanstone [GLV00] showed how to speed up generic attacks on the ECDLP in groups equipped with an efficient endomorphism, specifically Koblitz subfield curves defined over a binary field. I suspect that such a speed-up relies on the endomorphism being more efficient than a group operation. In other words, it appears to me that generic group algorithms will not be helped substantially by endomorphism less efficient than a group algorithm. I expect any endomorphism of CM55 curves to be less efficient than the group operation. If CM55 is lucky, then its endomorphism will help the user more than the adversary. I hope that further research will resolve these types of uncertainties.

While writing this paper, I have begun to wonder about Silverman's xedni calculus attack for solving the ECDLP, which Jacobson, Koblitz, Silverman, Stein and Teske [JKS+99] show to be infeasible in general. In particular, I wonder if the xedni attack might somehow be more effective for low discriminant elliptic curves. Recall that the xedni calculus strategy involves lifting the finite field points of the curve to rational points, and thus into the complex numbers. The CM method finds finite field curves already related to curves over the complex numbers. Perhaps this pre-existing link between the finite field curve and complex curve in the CM method can be exploited in a variant of the xedni calculus. I have no expertise in the area, but the vague argument above, plus the suspicion expressed by many others about low discriminant curves is enough to convince me to state my conjecture for C_3 as low as 2^{-16} in the case of CM55.

Now let C_4 be the average number of CM55 group operations needed to perform each the scalar multiplication in the reduction algorithm of Theorem 3.2. We expect C_4 to be larger than C_2. Again, one can amortize some of the costs, do some upfront calculations and so on. Let us estimate $C_4 = 350$ for the sake of a concrete example. Note that it will turn out to be the case that the largeness of the difference between C_4 and C_2 affects the tightness of the lower bound on the ECDHP.

The next step is to optimize d to gain the highest lower bound on the cost of the ECDHP based on Theorem 3.2. For the sake of simplicity, we will ignore the cost of the pseudorandom functions, and exponentiations modulo n and $n - 1$ in the reduction algorithm. A more thorough analysis should include these costs.

Let X be the cost of a hypothetical ECDHP algorithm, in units equivalent to group operations. Then Theorem 3.2 and Conjecture 3.3 say that

$$dX + C_4(286-d) + C_4C_1\sqrt{55} \times 2^{143-d/2} \geq C_3C_2C_1\sqrt{n}. \tag{21}$$

For d enough smaller than 286, the term $C_4(286-d)$ will be dominated by the term to its right by virtue of the factor $2^{143-d/2}$, so can be dropped with negligible loss of accuracy. This allows us to write a lower bound of:

$$X \leq C_3C_2C_1\sqrt{n}\frac{1-\frac{C_4}{C_3C_2 2^{d/2}}}{d} \tag{22}$$

To make this concrete, put $(C_1, C_2, C_3, C_4) = (1, 2, 1, 350)$, and try to optimize d. This gives an optimal value of $d = 21$, and places X at least $0.042 \approx 2^{-4.58}$ times the cost of Pollard rho.

The tightness of the reduction compared Theorem 3.1 has been slightly improved in the analysis above. Furthermore, because the value of d above is smaller, Theorem 3.2 can be generalized to groups of order n where $n-1$ is only divisible by 2^d. This gain in generality is largely due to the rather strong assumption in the corresponding generalization of Conjecture 3.3 to other groups.

The event of ECDLP being much more quickly solvable than Pollard rho can be represented by setting C_3 even smaller. For example, setting $C_3 = 2^{-30}$, which is below the estimate in the conjecture, the optimal d seems to be 87, and the tightness of the reduction is $2^{-6.5}$. Again, Theorem 3.2 should generalize to $n-1$ having a factor of 2^{87}. For smaller C_3, one will get larger d, and one has to be more precise with the estimates to ensure accuracy.

3.3 Relaxing the Smoothness Requirements

In this section, we state a version of den Boer's theorem that does not rely on any smoothness conditions. Instead, it relies on the existence of a factor of a certain size, similar to the Brown–Gallant (BG) algorithm [BG04] (which Cheon [Che10] extended). Indeed, in some sense, it is an application of the BG algorithm because a full DH oracle can of course be used to implement the static DH oracle used in the BG algorithm. The main difference is the algorithm here uses fewer oracle calls, because a full DH oracle is available, not just a static DH oracle.

Theorem 3.4. *Let $\mathbb{G}$ be a group of prime order $n = 1 + uv$, with additive notation and generator G. Let D be an DH oracle such that $D(aG, bG) = abG$ for all $a, b \in \mathbb{Z}$. Given A one can compute $z \in \mathbb{Z}$ such that $A = zG$, using at most $2\log_2(u)$ calls to D, and at most, on average, $2\sqrt{v} + 2\sqrt{u}$ calls to each of the following four operations:*

- *a scalar multiplication in the group $\mathbb{G}$,*
- *an exponentiation modulo n, and*
- *an exponentiation modulo $n-1$.*

Sketch: The proof is almost the same as the proof of Theorem 3.2 with $u = 2^d$ and $m = v$, except we will use BSGS instead of Pollard rho, just to make the theorem statement simpler by avoiding the constant C_1, and instead of computing $H_d = 3^{w \times 2^d} G$ by using d calls to D, effective applying d squaring operations to the logarithm, we instead a square-and-multiply strategy to compute $H = 3^{wu} G$. (Again, $z = 3^y \bmod n$ and $y = xv + w$ where $0 \leq x < u$ and $0 \leq w \leq v$.

We then solve for w using BSGS using about $2\sqrt{v}$ scalar multiplications and modular exponentiations. We must then solve for x, which can also be done using BSGS at a cost of $2\sqrt{u}$ scalar multiplications and modular exponentiations. □

The tightness of the resulting reductions has an analysis similar to that of Theorem 3.2. The two main differences are as follows.

- There must exist a factor u of the optimal size. Small factors are likely, for random n, but are likely a little off from optimal in size.
- Theorem 3.2 has $u = 2^d$ which minimizes the number of calls to M, thus making the reduction tighter than average, by one bit, according to Theorem 3.4 but perhaps improvable to only half a bit tighter.

Theorem 3.4 does not concern CM55 directly, but is useful for comparing CM55 to other curves. In particular, it shows that CM55 can achieve a tighter reduction than typical curves, but the reduction is not dramatically tighter.

Remark 3.5. When first writing about the CM55 in the patent [BGV07], I did not contemplate a result such as Theorem 3.4. Consequently, I was under the mistaken belief that the CM55 parameters represented a more substantial improvement in the tightness of the reduction between the DHP and DLP.

Therefore, the main direct benefit of group orders like that of CM55 group is that they nearly optimize the tightness of these kinds of bounds over a large range of situations.

Remark 3.6. This also helps in arguing that the CM55 parameters were not selected maliciously but rather according to a principle of optimizing provable security. More precisely, if we assume that a weakness in the curve does not correlate to optimizing the den Boer reduction, then by optimizing the den Boer reduction we reduced some of the wiggle room to allow a malicious search. These issues will be discussed in further detail later in §5.

Remark 3.7. Very loosely speaking, if the ECDLP in general collapses from about $\sqrt{n}$ to far enough below $\sqrt[4]{n}$ group operations in difficulty, then Theorem 3.4 can no longer offer much protection for the ECDHP. Theorem 3.1 would still apply. For other curves, ECDHP might collapse further to no security, much like the ECDDHP collapses when pairings become efficient. Under these unlikely disastrous conditions, CM55 would retain about 2^{65} group operations for the security the ECDHP, which is quite low, but still out of reach for many adversaries. Perhaps there exists a curve similar to CM55, but with an extra margin of error, that can better tolerate such a collapse.

In this situation, a Maurer–Wolf reduction [MW96] would be useful. For example, the auxiliary curves of Muzereau, Smart and Vercauteren [MSV04] can be used for the NIST curves, to provided some kind of lower bound on the ECDH in the event of a $\sqrt[4]{n}$ attack on the ECDLP. In this case, the ECDHP solving algorithm might not collapse to the level of being called efficient, but instead to something feasible, such as 2^{55} groups operations.

4 Resisting Other Attacks on CM55 (if Trusted)

This section reviews well-known (suspected potential) attacks, that can affect certain elliptic curves, but can normally be avoided by an appropriate choice of elliptic curve. For each attack, this section considers its applicability to the CM55 curves.

For the purposes of assessing the risk of well-known suspected potential attacks, such as on low-magnitude discriminant curves, this section presumes either that the CM55 curve was not selected maliciously, or that its selector, that is me, was incapable of actualizing any of these potential attacks.

A skeptical reader, who is cautious about elliptic curves being maliciously promoted, yet is also in some way interested in using the CM55 curve, might seek to verify the arguments in §5 to allay some (but not all) of these concerns.

4.1 Resistance to the MOV Attack

The Menezes–Okamoto–Vanstone (MOV) attack [MOV93] solves the discrete logarithm in an elliptic curve group if the elliptic curve group has a sufficiently low embedding degree. The embedding degree B for an order n subgroup of an elliptic curve defined over a prime field of size p is the smallest non-negative integers such that $p^B \equiv 1 \bmod n$. In other words, it is the order of p modulo n. For example, if $n = p + 1$, then $B = 2$, since $p^2 \equiv (-1)^2 \equiv 1 \bmod n$.

The MOV attack works by converting the discrete logarithm problem in the elliptic curve group into the discrete logartihm problem in an order-n subgroup of the multiplicative group of the the finite field of size p^B. For low values of B, the finite field problem can be solved using an index calculus algorithm, such as the finite field sieve algorithm, which is faster then generic algorithms like Pollard rho.

The CM55 curve appears to have an MOV embedding degree $B = (n - 1)/2$, which is the second highest possible value (the next below maximal value of $(n - 1)$). This value is well above the threshold to make MOV attack effective, which could be anywhere between 20 to 100. Indeed, even storing a random elenent of the finite field of sizae p^B is infeasible, let alone running any algorithm in the field.

4.2 Resistance to the SASS Attack

The Smart–Araki–Satoh–Semaev (SASS) attack, [BSS99, §V.3] solves the discrete logarithm problem in elliptic curve groups whose size equals that of the underlying field. In other words, it affects trace one curves.

The CM55 curve has size $p - 5$, not p, so is not affected by the SASS attack. It has trace six, not trace one.

4.3 Risk of Low Magnitude Discriminant

The discriminant is -55 which is significantly lower magnitude than it would be for a random curve.

Conventional wisdom is that low-magnitude discriminant are risky in that the DLP might be easier to solve. Indeed, this potential risk was incorporated into the bounds of

Conjecture 3.3.

Perhaps future research can find a curve with a discriminant large enough to allay these concerns, yet, like CM55, still with a tight den Boer reduction, and with an efficient underlying field.

4.4 Risk of Low Magnitude Trace

The CM55 curves all have trace 6. This is significantly lower magnitude than for random curves, with a p-value of about 2^{-145}.

Low trace curves are not usually considered a typical risk, but this may be partly due to the relatively few proposals for low trace curves. In fact, both the MOV and SASS attacks can be described as affecting low trace curves, namely trace zero and trace one curve, respectively.

Proposing the low-trace CM55 curve in this report may prompt further investigation into low-trace curves. Perhaps the MOV and SASS attacks have a common generalization to other low-trace curves. If so, then this would be bad for CM55, but I think would improve the general understanding of ECC.

4.5 Weak q-Strong DH Assumption

The q-strong DH assumption [BB04] is optimally weak in this group. In particular, per [BG04], it is not optimal for use in cryptographic schemes that expose the raw DH secret, such as some variants of the Ford–Kaliski key retrieval scheme [FK00], and the Passmaze protocol [Bro05].

Remark 4.1. The security against the q-strong DHP is an example of security characteristic that is negatively correlated to the goal of optimizing the den Boer reduction. Any user considering whether or not to trust CM55, must not only be careful not to rely on the q-strong DHP, but must assess the risk that some other attack correlates negatively the tightness of the den Boer reduction.

4.6 Strong Static DHP

The static Diffie–Hellman Problem, formalized by Brown and Gallant [BG04], like the ordinary DHP, becomes harder for CM55 in the sense of having a tighter reduction. This is an example of a positive correlation between two security properties.

Unlike the q-strong DHP, most ECDH-based protocols rely to a degree on the static DHP to be hard: namely, any protocols that allow re-use of ECDH private keys. Of course, such protocols can just rely on the ordinary DHP too, but only if they assume perfect key generation. Given the many examples of poor key generation in deployed products discovered by Lenstra's team [LHA$^+$12], the assumption of perfect key generation seems a little too strong. If static DH keys are biased but still unguessable, then the security appears to rely on the static DHP being hard.

Put another way, Brown and Gallant showed that the static ECDH problem suffers from weak key generation only if the ECDLP does too, by establishing a reduction between the two problems. In the CM55 curve, this reduction is nearly optimally tight. The known ways in which the ECDLP suffers from weak key generation are fairly well-understood, so the value of this reduction is in reducing the threat that some other kind of weak keys have to be avoided for static ECDH problem.

To justify why weak keys might be a greater threat for ECDHP than for ECDLP, consider the case of the ephemeral keys used in ECDSA and Schnorr signatures. These suffer from weak key generation attacks, where the weakness is very mild and has nearly no impact on the ECDLP. In other words, there is a large gap between in the conditions for secure key generation in the context of imperfect key generation. In particular, there is no reduction between the hardness of the problems under weak key generation settings. (Just to be clear, some security proofs for these signatures would assume perfect key generation.) If these weak key generation for ElGamal-based signature had remained unpublished, then perhaps some implementations would have taken some liberties, say for efficiency's sake, with the generation of ephemeral secret keys in signatures, and thereby would have been vulnerable to an attack. In other words, these weaknesses would have been unforeseen.

In this light, the Brown–Gallant reduction between the static DHP and DLP, helps to ensure that that there is no such gap, and helps to assure one that the current practices in key generation are sufficient, and there are no unforeseen gaps. Maybe nobody foresees such a gap, and maybe our foresight is good enough. The CM55 curves nearly optimize this reduction, so benefit well from this security analysis.

Of course, the first line of defense against weak key generation is strong key generation. In particular, the kind of bias of question is almost certainly avoided by merely using a proper pseudorandom function. Therefore, this kind of reduction merely provides a second line of defense. Put another way, a corrupted implementation, or fault injection attack, cannot just flip a bit of the private key to defeat the security of ECDH. (By contrast, this might work against a signature scheme, or one-time pad). Put yet another way, rather than relying on the full-fledged pseudorandomness of the key generation, one can instead rely on the more modest, and hence more plausible, properties that ensure that keys do not the ECDLP weaker. In particular, for resisting known attacks, this means that the keys need to be unguessable, and to not have a sufficient bias to make the Pollard kangaroo algorithm significantly faster than Pollard rho.

5 Rarity and Absence of Malice

This section is preliminary, hypothetical and speculative, so may contain inaccuracies.

5.1 Malicious Algorithm Promotion

Various mailing list emails, web sites, and journals have cast suspicion against various NIST cryptographic standards. These suspicions postulate that the cryptographic algorithms have been maliciously promoted. In other words, NIST or its partners knew, it is alleged, of attack against these curves, yet promoted them anyway (presumably: with the belief that others will not discover the secret attacks, or else that they possess a secret key providing exclusive access to the attack; and apparently, with a lack concern about suspicions such as these.)

Similar suspicions for all cryptographic algorithms go far back. Special values, sometimes described as "nothing-up-my-sleeve" (NUMS) numbers, have been incorporated into several algorithms. In their role as NUMS numbers, these special values serve as a mechanism to persuade users that the values have not been chosen maliciously. For example, the MD5 algorithm includes constants derived from values of the sine function evaluated at integer radian values. Sometimes, these special values serve a second purpose, which is to be random-looking, in the hope to reduce the risk that some special non-random pattern might render the algorithm insecure.

5.2 Suspicions About the "Verifiably Random" NIST Curves

Some of these suspicions about NIST have been directed towards the ten "verifiably random" the NIST recommended curves [NIS99, NIS05]. Indeed, these curves do not comply with the usual nothing-up-my-sleeve approach. These curves are derived from the output of a pseudorandom function applied to a seed value. Unfortunately, the seed values look completely random and have yet to be unexplained. In particular, there is no way to confirm that the seed has not been selected maliciously. Therefore, the way the curves have been promoted does not provide any kind of NUMS effect.

Remark 5.1. Of course, the label "verifiably random" is a slight misnomer (one of many in cryptography[1]). The Brainpool team "verifiably pseudorandom" may be more accurate, since the curve is derived from the output of a pseudorandom function.

This "verifiably random" process provides some assurance against sparse attacks like the MOV and SASS attacks. In fact, if the attacks are also negligibly sparse like the MOV and SASS attacks are, then it seems infeasible to choose the seed maliciously, provided that the pseudorandom function has some minimal one-way properties. Under the assumption that all secret attacks are as sparse as the MOV and SASS attacks, then the NIST "verifiably random" curves do provide a NUMS effect.

Consequently, it seems that the main way that the "verifiably random" NIST curves could have been maliciously selected is to hypothesize a class of weak curves whose size is large enough that a random curve would land in the class with non-negligible probability. For lack of a better term, let us call this hypothetical non-sparse weakness, and the class of curves possessing it, the spectral weakness, and spectral class, respectively.

One strong argument against the existence of a spectral weakness is that the cryptanalytic effort spent on ECC has been sufficient to publish all substantial weaknesses on prime-field curves. The Pohlig–Hellman attack, the Pollard rho attack, the MOV attack and the SASS attack were all discovered by 1999 or earlier. The SASS attack was the most recently discovered, just shortly before the NIST curves were proposed. If the SASS attack had not been published but were kept secret by NIST, the NIST curve generation "verifiably random" process would have made it infeasible to make the curves vulnerable to the SASS attack.

One weak argument for the plausibility of the spectral weakness is on to consider elliptic curves defined over extension fields. Significant and surprising recent progress has

[1] For example, in standard mathematical terminology, the term "random", as in random variable, and random event, does not refer to a uniform probability distribution. In cryptography, the term "random" is often used for such "uniform' distribution", as in randomness extractor, which is a misnome, or at least a conflict with its use in mathematics. As such, "verifiably random", is a misnomer within misnomer, in such a way that the two wrongs cancel into a right: in math random variable can be constant.

been made there, combining the methods of Weil descent and sub-field indexed calculus. Although this is not expected to carry over to prime field curves, it may still be sufficient grounds for suspicion of spectral weakness. In other words, one should not merely dismiss the threat of a spectral weakness as paranoia. Rather, one should instead address it rationally, using some form of risk management. Indeed, the reason for considering provable security is to minimize the risk of secret attacks.

In particular, Theorems 3.1–3.4 are an effort to avoid certain types of hypothetical unknown attacks, rather than just known attacks, just like any other type of provable security result. For consistency with this effort, one should examine the resistance to as many other types of hypothetical unknown attacks. In this section, the wrinkle, compared to the straightforward provable security, in the threat model is that the prover (or informal arguer) is a potential adversary. So, the approach here will be try to appeal, as much as possible, to the user's logic, rather the user's beliefs. To that end, we the follow the usual approach in provable security, by stating what assumptions the user is asked to believe, and state what formal arguments the user can verify by logic alone.

5.3 Rarity and Other Arguments

Bernstein and Lange [BL14] introduced the term "rigidity". This term is used for an idea similar, at least in purpose, to that of "nothing-up-my-sleeve" constants used in algorithms like MD5.

It turns out that similar arguments can be applied to CM55, but I will term the arguments "rarity" arguments, not "rigidity", for the following reasons.

- I do not want misrepresent what Bernstein and Lange mean by "rigidity".
- Elliptic curves arise from geometry, so using the geometric term "rigidity" for a non-geometric concept mixes metaphors.
- The natural opposite of rigidity, flexibility or manipulability, suffers from a grammatical tense problem: it is the process of curve selection that is manipulable, not the curve set itself. In other words, although a fixed curve has potentially been manipulated, it is no longer manipulable once it has been selected.
- The general notion that rigidity seems to describe does not resolve all the issue of manipulability, because more clever manipulation can never be fully ruled out, especially of the definition of rigidity. In other words, the term "rigidity" exaggerates the security benefits of the concept.

I think the term "rarity" avoids the deficiencies above. Furthermore, I delibertate extend, generalize, vary and modify some aspects of the Bernstein–Lange idea of rigidity, yielding a formalized argument in Lemma 5.1

5.3.1 Rarity Sets: Informal Definition

The intuitive idea of rarity for an elliptic curve is to argue that an elliptic curve belongs to a small set of curves (it is rare). Thereby, one hope is that the set is small enough that even if the selector (say, me) knows of a secret attack (akin to the the spectral weakness hypothesized above), the chance that the secret attack affects the curves in the rare set is negligible.

The argument above is not rigorous at all (or less kindly: it is pure nonsense), because every curve belongs to a set of curves of size one. To improve upon the rigor, we first need to qualify what constitutes a being rare. An informal definition of rarity is membership in a rarity set R, which has

- an easy and simple definition,
- a small, and easily provable, upper bound on its cardinality,
- a definition in terms of identified directly beneficial properties, or reasonable proxies of the properties.

The intuitive purpose of using reasonable proxies is to keep the rarity set easily definable. For example, defining the rarity set as taking the optimal value of certain parameters introduces a potentially difficult optimization problem. So, a proxy may be a substitute that is easier to optimize, or easier to establish a threshold for.

A rarity set which fails to meet the third property can be said to have the lesser property of artificial rarity. For example, the Curve25519 has a curve coefficient that is minimum element of a certain subset. If the small resulting size of the coefficient had no direct benefits, such as efficiency, then I would deem it to be artificially rare. But, if there was an approximate correlation between efficiency and small coefficient size, then choosing a small size coefficient could be deemed as a good proxy for a beneficial property, namely efficiency, and this condition would not be artificial. For another example, the Brainpool curves use a seed and a minimized counter in a pseudorandom function. Both of these are artificial rarity criteria because a small counter value or seed value, once passed through a pseudorandom function does not confer any direct beneficial property to the curve (that is, other than rarity), under the intuitive argument that the pseudorandom function should make that impossible.

Remark 5.2. Just to be clear, a rarity argument, on its own, is inferior to selecting a curve randomly in the sense that random selection can ensure avoiding secret attacks that are sparse, like the known MOV and SASS attacks, as we will soon see.

Unfortunately, combining a randomness and rarity argument only appears possible for artificial rarity.

A rarity argument might also contain one more element: a pool P from which the rare set is to be viewed as a subset. The pool is background context against to which the curve C and rarity set is to be compared.

Remark 5.3. We have described P and R as sets. In some cases, the pool may be better described as a process to generate curves, with some curves being more likely than others. In other words, P could be distribution of curves instead of a set of curves. We can formally approximate this situation by allowing P to be a multi-set, with more likely members appearing more times in the multi-set.

In most cases, the pool of curves can be taken for granted by context, without much ado. For example, it can take as the set of curves in which the generic attacks, Pollard rho and Pohlig–Hellman, are infeasible, which means the curve size has prime factor than some minimum bound such as 2^{160} or 2^{256} or 2^{255}.

5.3.2 Formalizing Secret Attacks

A rarity argument attempts to dissuade a user about a secret attack. This is an attack about which the user lacks information. To formalize this lack of information, we can use Shannon's theory of information, which uses the more fundamental notion of probability to formalize information. Therefore, from the user's perspective, the attack is a random variable. (Of course, from the attacker's perpective, the attack is a different random variable.) Just to be clear: a random variable does not have to be uniformly distributed.

Specifically, let A be a random variable consisting of the set of elliptic curve vulnerable to the secret attack. The secret attack can have a computational cost τ and success rate σ that vary with the curve, so we could write A_{τ_0,σ_0} for the set of curves which are vulnerable to the secret attack such that $\tau < \tau_0$ and $\sigma > \sigma_0$. This results in a family of random variables. In the following argument, we argue about one of these random variables, which we just abbreviate to A. (We may occasionally refer to the secret attack itself as A.)

Remark 5.4. When one varies the parameters of the attack as in subscripted set A_{τ_0,σ_0}, one may also want to vary the definition of the pool and rarity sets to keep things consistent.

5.3.3 Uncorrelated Secret Attacks

Next, we define two other random variables derived from A:

$$q_{A,P} = \frac{|A \cap P|}{|P|} \tag{23}$$

$$q_{A,R} = \frac{|A \cap R|}{|R|} \tag{24}$$

These are the frequencies with which the secret attack affects curves in the pool and in the rarity set, respectively. We say that the secret attack is (P, R)-uncorrelated if the expected values of these two frequencies are the same:

$$E(q_{A,P}) = E(q_{A,R}). \tag{25}$$

In particular, if the secret attack is (P, R)-uncorrelated then the rarity set has not been chosen to be more positively correlated than random curves drawn from the pool. If we replace the equality in (25) by an approximation, then we say that A is weakly (P, R)-uncorrelated.

Given the set A, there exists R such that A is not (P, R)-uncorrelated. For example, if $R = A$, then $q_{A,R} = 1$, which is presumably greater than $q_{A,P}$. More generally, an attacker may be able to choose R to boost the probability $q_{A,R} \gg q_{A,P}$.

A user might suspect that an adversary would not want to leak information about its secret attack to others. In particular, the adversary might define R in such a way that it appears independent of A. This might cause A to be (P, R)-uncorrelated. In other words, this provides a weak argument for the plausibility of A being (P, R)-uncorrelated.

In our informal definition of a rarity set, we expressed a preference to avoid artificial rarity criteria. This preference helps support a belief that A is (P, R)-uncorrelated. If R has a definition that includes an artificial criterion, then perhaps the artificial criterion has been selected maliciously in order to increase $q_{A,R}$ compared to $q_{A,P}$.

With the formalism of the sets R and P, the random variable A, and the property being (P, R)-uncorrelated, we can define three arguments to help persuade a user to trust a curve $C \in R$: an equivalence argument, a rarity argument, and an exhaustion argument.

5.3.4 Equivalence Argument

The equivalence argument goes as follows. If a curve C is selected uniformly at random from the rarity set R, and D is selected uniformly at random from the pool P, and the secret attack is (P, R)-correlated, then C being vulnerable to the secret attack is equally probable to the curve D being vulnerable. In other words, under the assumption that A is (P, R)-uncorrelated, using a random curve from R is equivalent to using a random curve from P.

Remark 5.5. Regarding the CM55 curves, which we have defined in terms of field size and curve order, there is still some freedom in the choice of actual curve. This choice could be made randomly or pseudorandomly, and then the argument above can be applied.

Just to be clear: the equivalence argument cannot be applied rigorously to non-random curves.

5.3.5 Rarity Argument

The rarity argument hinges on the following trivial lemma.

Lemma 5.1. *If A is (weakly) (P, R)-uncorrelated, then the probability that $|A \cap R| \geq 1$ is at most (approximately)*

$$E(q_{A,P})|R| \tag{26}$$

Proof. Simply:

$$\begin{aligned} P(|A \cap R| \geq 1) &= \sum_{c \geq 1} P(|A \cap R| = c) \\ &\leq \sum_{c \geq 0} cP(|A \cap R| = c) \\ &= E(|A \cap R|) \\ &= E(q_{A,R})|R| \\ &= E(q_{A,P})|R|. \end{aligned} \tag{27}$$

with the last equation replaced by an approximation if A is weakly uncorrelated. □

So,

- if $|R|$ is small,
- if the user believes $E(q_{A,P})$ is sufficiently small, and
- if the user believes that A is weakly (P, R)-uncorrelated,

then Lemma 5.1 will persuade the user (to the extent of the beliefs above) that there is low probability that R contains any curves C affected by the secret attack with $C \in A$.

Remark 5.6. The purpose of invoking (P, R)-uncorrelatedness is to make the first belief more independent of the set of R. In other words, the user can formulate a belief about $q_{A,P}$ based on all the existing effort that has gone into studying elliptic curve cryptography.

Remark 5.7. I take this rarity argument to be part of what Bernstein and Lange mean by rigidity. They also define some a particular rarity sets, which are perhaps part of their definition of rigidity.

5.3.6 Exhaustion Argument

An exhaustion argument depends on a further assumption about the attack, the adversary, and the set R.

We say that the attack set A is R-IID if the best way that the adversary knows how to find an element of $A \cap R$ is trial-and-error: to try some random $C \in R$, and apply a test if $C \in A$. (The test should have non-negligible cost, or at least finding a curve in R should have non-negligible cost.) In other words, the adversary has no special knowledge about A that allows a quick shortcut to finding elements in $A \cap R$.

If we were to formalize this lack of information that the adversary has about the answer to the question $C \in A$, we could define the boolean value of $C \in A$ to be random variable from the adversary's perspective. Each C defines one random variable. Then A is R-IID if each of the random variable $C \in A$ has an identical distribution, and the random variables are independent.

A user might believe that A is R-IID for two reasons: firstly, some existing attacks on elliptic curves—including the MOV, SASS, Pollard rho, and Pohlig–Hellman attack—can be characterized by a simple test that can be conducted for any curve; and secondly, the simple description of R, together with the adversary's goal not to leak information about the attack, makes it unlikely that something about R would make it possible to shortcut this trial-and-error process of testing curves one-by-one.

Lemma 5.2. *If A is R-IID and (P, R)-correlated, and the adversary expended at most t of the trials above, then the probability that the adversary finds a $C \in A \cap R$ is at most*

$$1 - (1 - q_{A,P})^t \approx t q_{A,P} \tag{28}$$

Proof. The adversary tries curves $C_1, \ldots, C_t$. Each C_i has probability $1 - q_{A,R} = 1 - q_{A,P}$ that $C_i \notin A$. The probability that all of the $C_i \notin A$ is thus $(1 - q_{A,P})^t$. The probability of the opposite, namely that at least one of the $C_i \in A$, is one minus that. The approximation on the right hand side follows if $q_{A,P}$ is small enough. □

Of course, the exhaustion argument does not depend on the cardinality $|R|$ being small, but does intuitively depend on the description of R being small in whatever sense is necessary to make R-IID condition believable.

If one defines R to be the output of a pseudorandom function, then the condition that a secret attack A is R-IID and (P, R)-correlated becomes more plausible. We will not formalize here how to rigorously deduce these properties from some kind of formal pseudorandomness property, because that would take us to too far from the focus of this paper which is the CM55 curve. Nevertheless, for some common pseudorandom function, the trial-and-error method is the best known way, is the R-IID condition may be plausible.

An additional reason to state Lemma 5.2 and pseudorandom curves is to provide a lower bound on $q_{A,P}$ on the hypothesis that some other curves, especially the pseudorandom curves the NIST or Brainpool, are vulnerable to a secret attack. If the user worries that other curves, especially special curves such as CM55 or Curve25519, might also be intentionally selected as vulnerable to this attack, then the user might use this lower bound on $q_{A,P}$ in Lemma 5.1 and to limit the usefulness of the rarity argument relative to the hypothesis of an attack on a pseudorandom curve.

5.4 Rarity Sets for CM55

Per our informal definition of rarity, we now describe a simple set. Since CM55 is really a set of curve parameters, we specify that CM55 is a field size p and one of a number of j-invariants, and finally, the curve cofactor, and the prime order subgroup n.

5.4.1 A First Rarity Set

First, here is a rather vague description of our first rarity set:

- The known attacks must be resisted. In particular, the curve size must be at least about 2^{256}, perhaps tolerating something slightly smaller.
- The den Boer reduction must be almost optimal, in the sense that $n - 1$ must be a power of two times a small number.
- The field size must be amenable to more efficient arithmetic than average.

The description above is too loose to assess the rarity, so the following make some more specific decisions:

- The known attacks must be resisted. In particular, the prime subgroup has order $n > 2^{256}$.
- The den Boer reduction must be almost optimal, in the sense that $n - 1 = 2^t d$ for $d < 1000000$.
- The field size p must have a binary expansion with at most 20 bit changes between consecutive bits, and maximum length 1000.
- The cofactor $h = 4$.

Note that some of criteria above are are simplistic proxies for more nuanced security and efficiency goals. The purpose of using such proxies is to ease the task of enumerating (sizing) the rarity set.

Now let's loosely estimate the size of this set. Note that the way we estimate the size is not necessarily related to the way an adversary might search the set.

Start with n. There are about a 2^{20} choices for d and 2^{10} choices for t. To account for the primality requirement, reduce this to 2^{23} choices for n, by invoking the prime number theorem estimate for the density of primes.

Next consider p. For each n, there exists many p in the interval Hasse interval for $4n$, but only at most $\binom{500}{20} \approx 2^{118}$ or less would meet the requirements binary expansion requirement. Reduce this estimate to 2^{112} for the primality requirements.

This estimate gives a set of size 2^{135}. If I knew of a weak class of curves independent of this rare set but sufficiently common, say affecting the fraction 2^{-40} of j-invariants, then I certainly could have searched through this set to find CM55 falling into the weak class.

But for a given pair (p, n) in the set above, it seems to be an open problem to find a matching j-invariant unless the pair is a CM pair, having low discriminant.

A heuristic estimate for the chance of a single pair (p, n) in the set of above having discriminant D is about $\frac{1}{\sqrt{-pD}}$. Summing this over all discriminants of magnitude less than D gives an estimate $\sqrt{\frac{D^3}{p}}$. An adversary searching curves incurs greater cost for larger magnitude discriminants.

As the discriminant grows, the adversary's cost search cost becomes too large. For sake of an example, suppose that the CM method is only feasible for the purpose of the attacker searching for a weak curve, for discriminant of absolute value at most 2^{10}. Subbing discriminant magnitude upper bound 2^{10} into this gives a probability of 2^{-113}, for a rarity set of expected size $2^{-113} \times 2^{135} = 2^{22}$.

Now if the density of the weak class is 2^{-30} or less, then we should not expect an intersection between the weak class and the rarity class. If the density of the weak class is 2^{-20}, then one would have a good chance of a weak curves in the rare class, and perhaps CM55 would be such a curve.

5.4.2 A Low Discriminant Rarity Set

I could instead tailor my rarity class more narrowly, so that it just barely holds CM55, using the following criteria:

- The known attacks must be resisted. In particular, the prime subgroup order $n > 2^{256}$.
- The den Boer reduction must be almost optimal, in the sense that $n - 1 = 2^t d$ for $d < 1000$.
- The field size p must have a binary expansion with at most 10 bit changes between consecutive bits, and maximum length 300.
- The discriminant is at most 1000.
- The cofactor $h = 4$.

These are all just quantitative adjustments of the previous criteria except the maximum discriminant criterion, which is a new criterion, to be discussed further below.

In this case, there are: about 2^{16} possible n; about $\binom{150}{8} \approx 2^{42}$ choices of p per n; so about 2^{58} pairs (p, n). If the previous heuristic arguments about discriminant sizes are correct, then the expected size of this set would be 2^{-73}. If the heuristic used to estimate the size are close to correct, then the fact that the rarity set has any members at all, namely CM55, is surprising. The low expected size of this non-empty set seems remarkable enough to warrant calling CM55 *unusual*.

The criterion for low discriminant would not be on the grounds for security. Indeed, many experts in elliptic curve theory suspect that low discriminant might make the DLP easier. Rather, it would be for efficiency. The first efficiency aspect is to make the curve more efficiently constructible. The second efficiency aspect is to make the GLV method more feasible. If the GLV method can be shown to be practical for the CM55 parameters, then this criterion would be similar to the field size criteria, and could be fully justified as an efficiency criterion.

If the GLV method does not prove practical for CM55, then the efficiency gain is merely in constructing the curve. But one could argue that this efficiency gain occurs before generation of the curve, and therefore confers to no benefit to the users of the curve. Under this argument, the low discriminant criterion would be an artificial rarity criterion. In fact, if the suspected risk for the low magnitude discriminant curves graduates into a real attack, then this criterion would be a harmful rarity criterion.

5.4.3 Loose Number-Theoretic Rarity Set

Taking a step further away in cryptographic relevance, adopt an elementary number-theoretic perspective. Observe that $(4n, p) = ((x-1)^2 + dy^2, x^2 + dy^2)$ for $(d, x, y) = (55, 3, 2^{286})$. If we insist that $y = 2^k$ with $k \geq 256$, and that, somewhat arbitrarily, $d, x \leq 100$, then under a heuristic for primality based on the prime number theorem, we expect the number of triples (d, x, y) yielding a pair $((x-1)^2 + dy^2, x^2 + dy^2)$ in which one element is prime, and the other element is four times a prime, to be at most about

$$\sum_{k \geq 256} \frac{10000}{k^2(\log(2))^2} \tag{29}$$

This is a finite number, rounded to nearest order of magnitude about 100. Of course, the heuristic used might break down the special numbers being considered. Anyway, under this analysis, the existence of this the pair $(4n, p)$ is not surprising, but is still among an informally rare class.

5.4.4 A Rarity Set Based on Special Field and Special Curve Size

Given that a rarity argument is mainly about avoiding an attack, one can view efficiency criteria in a rarity set as artifice. In other words, the argument that a curve is secure because it is efficient makes no sense. Smaller curves are more efficient, and less secure, for example.

In this section, we informally define a rarity set as follows:

- The curve resists the known attacks.
- For the curve order u, we insist that it has a prime factor n such that $n-1$ factors almost entirely into a power of two, just as required in our previous rarity sets. This security rationale is to make the den Boer reduction nearly as tight as possible.
- For the field size p, we adopt an artificial criterion, that it is very compressible.

The field size criterion is artificial, but field size is a security characteristic of the curve. Recall that Shoup's results assures us of such security against generic attacks. The known specific attacks seem to depend on the both the curve size and field size. So, the artificial criterion on the field size is an effort to reduce the possibility that the p has been selected maliciously. In other words, this rarity set helps to avoids secret attacks in which the attack's effectiveness is not correlated to the compressibility of the field size.

The CM55 parameters meets the first two conditions. For the last condition, the CM55 parameters have a field size p that can be specified in about 10 conventional mathematical symbols:

$$\texttt{9+55*2\^{}288} \tag{30}$$

It would be interesting to formally define this rarity set, and then to estimate its size.

By contrast, now consider some other curve in which either the field size or curve size is random-looking. The randomness may be disguising the secret attack. In other words, a malicious curve promoter cooked up the other parameters, with whatever flexibility was available under the attack, until the random-looking parameter (either the field size or the curve size) was amenable to the secret attack. The random-looking nature of affected parameter may somehow hide the nature of the attack, preventing users from reverse engineering the attack from the curve definition. More generally, anything random-looking in an algorithm could be viewed with suspicion.

Such an attacker could also argue that any randomness in the field size or curve size is merely a harmless and natural by-product of the elliptic curve generation process. The CM55 parameters at least show that such randomness is not a necessary by-product of curve generation.

That all said, the CM55 curve coefficients look random too. They can be also be specified in a slightly more compact form as the root of a Hilbert or Weber polynomial. Of course, such compactifications can sometimes be made for randomly-sized field or curves too (where, for example, the de-compactification step involves some form or point-counting). The distinction is that the known attacks are most directly characterized in terms of the curve and field size, and any more indirect compressibility of these parameters can only provide indirect protection.

5.4.5 Heuristics in the Rarity Sets Above

All of the arguments above are merely heuristic. A concrete counterexample to these rarity arguments would be to find one or more curve parameters that are superior to CM55 in all the criteria above (either the proxy criteria, or the security benefit criteria which the proxy criteria model). Such curves might not only refute the rarity argument for CM55, but might also improve on the benefits of CM55 for tightness of the reduction or for efficiency, in addition to improving the rarity argument, so could be more worthwhile than just a critique of CM55.

5.5 Bad and Rare Curves

The benefit of rarity should not be overstated. Abstractly, the limitation to the benefit of rarity is that it does not protect against attacks correlated with the rarity set, whether or not the curve promoter is aware of the attacks or not. The benefit of rarity is to resist hypothetical attacks, which forces one to adopt this abstraction in describing this limitation. But under such a high level of abstraction, it is very hard to distinguish how much of a limitation this is to the argument.

This limitation to the rarity argument is very severe. In other words, rarity conveys very little protection. The real protection is the extensive understanding and effort put into studying the elliptic curve discrete logarithm problem. Nonetheless, new users of elliptic curve cryptography might not appreciate the extent of this effort spent on studying the ECDLP. Furthermore, a less mathematically inclined user might find the narrative of a malicious organization subverting cryptographic standards much more appealing, and comprehensible, than an analysis of the xedni calculus attack, than a Weil pairing, than a p-adic map. Such a non-mathematical user may be more persuaded by a rarity argument, and think it provides equal, or even greater, protection, than the protection that a random curve provides against potential sparse secret attacks.

For example, this non-mathematical user may have little appreciation for the following mathematical facts. The four main attacks affecting prime-field elliptic curve DLP are Pohlig–Hellman, Pollard rho, MOV and SASS. The first two attacks are generic in the sense of applying to any Diffie–Hellman (or discrete log) group. These attacks depend only on the order of the group, and do not depend on the underlying field, the curve equation and so on. Furthermore, under Shoup's generic group model, [Sho97], these attacks are the best possible, so we can account for them perfectly up to the accuracy of the generic group model. So, there is not going to be some new generic attack on the ECDLP or ECDHP. If there is some new or secret attack, it must be non-generic. By contrast, the MOV and SASS attacks on the ECDLP are not generic attacks. In particular, they do not just depend on the group order, but also upon the underlying field size. These attacks are not ruled by Shoup's result [Sho97] because they are non-generic attacks.

So, the main risk against the ECDLP and ECDHP is another non-generic attack, akin to the MOV or SASS attack. It is reasonable to presume that if there is such an attack, that it is more complicated than the MOV and SASS attack, on the principle of simpler attack are likely to be found first. In other words, a simpler attack would have been unlikely to escape detection. Indeed, the SASS attack is arguably more complicated than the MOV attack, confirming that the trend that simpler attacks are found earlier.

Next, it is plausible to assume that a more complicated attack affects a smaller fraction of curves than a simpler attacks, because the conditions under which the complicated works are more complicated and thus less commonly attained. The MOV and SASS both affect a negligible fraction of curves, and are easily avoided by selecting a pseudorandom curve, even with a seed freely chosen by the adversary, provided that the pseudorandom function ensures the best way to control the output is exhaustively trying different input values.

Remark 5.8. The argument above omits the case of singular curves. These curves are cubic curves that fail to be elliptic curves, but still define a group, and have an easy to solve discrete logarithm problem. This can be viewed a third class, again of negligible fraction of curves, that is weak. This bolsters the arguments here further.

If we drop the latter assumption, and allow for another non-generic attack that is more frequent than MOV and SASS, and enough so to allow a pseudorandom seed to be searched, then we should insist on some kind of trustworthy seed, which can be a rather difficult task. Establishing trustworthiness of a seed may involve some kind of artificial rarity argument.

Clearly, the protection provided by a pseudorandom curve, even with a trustworthy seed, has its limitations. The non-mathematical user may still not realize how implausible the hypothesizes above are. In particular, this user's natural skepticism might lead the user to think that a rare curve provides just as much protection.

Where the rare curve fails to provide protection is against sparse attacks, like MOV and SASS. These real world attacks serve as evidence plausibility against any more dense non-generic attacks on prime-field elliptic curves. Yet, the user may intuit that the chance that a sparse attack could affect a rare curve is negligible. After all, sparse times rare ought to be small. This intuition presumes independence of the rarity of the attack from the rarity of the curve.

The non-mathematical user might reason that such an independence is reasonable, because if one assumes nothing about the attack, then we have zero information about it, which by inverting Shannon's theorem means it is random with respect to everything we know, including the rarity set of the curve we considering. This reasoning may be correct in the natural sciences but it is too optimistic for cryptology.

So, to illustrate this less abstractly, I next provides some examples of rare curves that are affected by published attacks. If the published attacks had remained secret, then rarity of the curves would have provided no protection. This is partly because the rarity of the curves correlates with the attacks. In particular, the independence argument is incorrect.

Consider the elliptic curve equation $E : y^2 = x^3 + x$. This in short Weierstrass form, but also in Montgomery form, which appears to admit more efficient implementation of the Montgomery ladder. It also has the smallest curve coefficients of a non-singular cubic curve that has both of these forms. In other words, it seems rare.

Lemma 5.3. *If p is a prime with $p \equiv 3 \bmod 4$, then the curve $E(\mathbb{F}_p)$ has $p + 1$ points.*

Proof. Note $4 \nmid p - 1$, so $\mathbb{F}_p^*$ has no points of order four. Therefore, -1 is a quadratic non-residue modulo p, because it has order two in $\mathbb{F}_p^*$ and if it had a square root, that square root would have order four.

Now group the elements of $\mathbb{F}_p^*$ into cosets $\{x, -x\}$ of the subgroup $\{1, -1\}$. Since -1 is quadratic non-residue, exactly one member of each coset has a square root. There are $(p-1)/2$ of these cosets. For each such coset $\{x, -x\}$, consider the set $\{x^3 + x, -(x^3 + x)\}$. Note that x^3+x is not zero, because x is not zero, and $x^2+1 = 0$ has no solution. Therefore, the set $\{x^3 + x, -(x^3 + x)\}$ is also one of the cosets of $\{1, -1\}$, and exactly one element of its elements has a square root. Suppose, without loss of generality, the quadratic residue is $x^3 + x$, and it has square roots y and $-y$. This gives two points on the curve for each of the $(p - 1)/2$ cosets $\{x, -x\}$ which makes for $p - 1$ points.

There is only one point with $x = 0$, namely $(0, 0)$, and one point at infinity on a Weierstrass curve, so the curve has $p + 1$ points. □

The proof above gives an elementary way to count the number of points on a curve. For almost all elliptic curve equations define over prime fields, the best way to determine the curve order is either the Schoof–Elkies–Atkin algorithm or the complex multiplication method, both of which of involve extensive calculations and cannot be considered elementary. Therefore, the ease with one can count the points of on such curves can be considered a rarity criterion, albeit perhaps an artificial one.

These curves are vulnerable to the MOV attack, because they have embedding degree two. Therefore, anybody who knew the MOV attack before it was published, could have feasibly solved the discrete logarithm in these despite their having some rarity property. Furthermore this adversary could maliciously promote this curve to users, merely trying to arrange that prime $p \equiv 3 \bmod 4$. The adversary could select p at random until p met this condition, devise some rare p with this property, or, more directly, argue that $p \equiv 3 \bmod 4$ allows square roots to calculated more efficiently.

Remark 5.9. For sake of concreteness, we give an example. Let $p = 2^{256} - 39253$, which is probable prime. Then the curve above has order $4n$ for probable prime $n = 2^{254} - 9563$.

The last example has a prime field which can be represented in 11 conventional mathematical symbols. In other words, its field size compressibility is similar to that of CM55. This shows how field size compressibility, an artificial rarity criterion, does little to prevent a special attack like MOV.

Remark 5.10. For a similar but slightly smaller example, consider $p = 2^{225} - 49$, which is the largest 225-bit prime. The curve $y^2 = x^3 + x$ has $p + 1 = 16n$ points where n is the prime $2^{221} - 3$, which is the largest 221-bit prime. If the MOV attack were secret, then this curve, which one might call Curve22549, might seem a viable alternative to NIST curve P-224.

Unlike curve CM55, there does not seem to be any direct security gain from the special form $n = 2^{221} - 3$ of the large prime order of the Diffie–Hellman subgroup, at least compared to orders for other small-cofactor curves. In other words, this special form of n is an artificial rarity condition.

Remark 5.11. The name Curve22549 from the previous remark is chosen to point out some similarities to Curve25519, specifically that: the field size is the largest prime of a certain bit length; and the Montgomery form of the equation $y^2 = x^3 + ax + x$, with the value of a minimized. Of course, for Curve25519, the minimization occurs under the constraint that the MOV attack is avoided. (Question: is there a smaller-a variant of Curve25519 which is vulnerable to the MOV attack?) To the extent of this similarity, Curve22549 demonstrates the obvious principle that certain special properties of Curve25519 provides no essential protection against secret attacks, including sparse attacks.

Not all the maximal primes of a given bit length yield a weak curve with a story as plausible as Curve22549. In other words, I had to do a little search to find Curve22549. Specifically, I had to expand my maximum cofactor from 4 to to 16, and lower my minimimum field size. In other words, I needed some wiggle room, to make the MOV attack viable. In yet other words, I manipulated the choice of curve to arrange that MOV was viable. So, it would seem unreasonable to call Curve22549 “rigid”, since rigid seems to imply lack of manipulation.

Tangent: probably some additional cover story would be needed for justification of Curve22549. For example, to justify a drop below the usual but arbitrary convention of a group size of at least $2^{256} - 2^{130}$, perhaps some argument that 2^{112} security is good enough for most applications could serve to persuade users that that manipluation above is merely an engineering practicality. Similarly, the field size and cofactor 4 in the Curve25519 results in a group of size of approximately 2^{253}, which is smaller than the usual size of 2^{256}, and therefore requires some kind of justification story. (One such justification might be saturated security, per §B.4.)

Next, suppose that the MOV attack was never discovered. Indeed, if we are trying to remedy the possibility the NIST curves are malicious, then negligibly sparse attacks like the

MOV and SASS are not the risk. As discussed earlier, the main concern with NIST curves is that there exists some spectral weakness in a class of curves, which occurs frequently enough to find "verifiably random" curves with this weakness.

By definition, we do not know what the spectral weakness is, since it is a secret attack. Therefore, we have to struggle with abstractness of it all. So, to illustrate the principle, we again model a secret weak by a real attack, in this case the Pohlig–Hellman (PH) attack. Like the hypothetical spectral weakness, it should feasible to search through pseudorandom curves until one finds one a curve in which the Pohlig–Hellman attack is feasible. In other words, the PH attack serves well as an illustrative model for the spectral weakness that the NIST curve P256 might possess.

Remark 5.12. In an email to the CFRG list, Bernstein cited the PH attack to argue for the plausibility of a "one-in-a-million" attack (which I am calling a spectral attack in the case of the NIST curves). In particular, I do not seek any credit for modelling the characteristics of a secret attack by using the characteristics of a real attack. (Kobliz, Koblitz and Menezes [KKM08] also model a secret attack by a real attack.)

On the other hand, I do not condone Bernstein's argument the plausibility of a spectral weakness, because the PH attack is a generic attack, which we know how to resist due to Shoup's result. Here, I merely use the PH attack as a model for the implausibly high frequency of hypothetical spectral weakness. In particular, the fact below that the PH attack can be arranged without using exhaustive trial-and-error makes plausible that the spectral weakness has a similar property.

Consider a field size $p = 2^m - 1$ where $m \in \{127, 521\}$. These field sizes are Mersenne primes, a fact which seems to enable quite efficient arithmetic, especially if one can successfully use radix size less than the machine word size. Also, there are very few Mersenne primes, and no others between these two values. In other words, these are rare fields.

Corollary 5.4. *If $p = 2^m - 1$ is a Mersenne prime, then $E(\mathbb{F}_p)$ has 2^m points.*

Proof. Because $m \geq 2$, we have $p \equiv 3 \bmod 4$, so Lemma 5.3 applies, to give size $p + 1 = 2^m$. □

These curves are optimally vulnerable to the PH attack. Therefore, the rarity properties of the field size have somehow correlated with the effectiveness of the secret attack.

Remark 5.13. In the rarity story above, the restriction to Mersenne primes and the efficiency thereof, does not seem to directly leak any information about the Pohlig–Hellman attack. Of course, the unusually smooth curve order might suggest a problem, but that is not part of the rarity set definition, but rather a by-product, which in hindsight looks like an obvious sign of trouble.

Finally, if the secret attack is a scorched earth attack that affects all curves equally, like the Pollard rho attack, then the only possible protection is a margin of error, by using a curve larger than seems necessary, assuming the attack's efficacy decreases with curve size.

Remark 5.14. In metaphor, the protection provided by rarity arguments is newspaper thin. The thinness is fair in the sense that rarity would not have protected any user against any of the published attacks. The reference to newspaper quantifies this thinness, and further reflects the strong appeal of the rarity argument to users whose main source of information is from journalists.

Remark 5.15. An ECC user's real protection arises primarily due to the diligence of cryptanalysts' effort to find attacks, and secondarily due to the work of cryptographers like Shoup and den Boer whose provable security work deduces a reduced set of possible attacks.

Remark 5.16. Although it is difficult to assess the possibility of secret attacks, one might try to infer something about this risk by consulting with the discoverers of published attacks what they think the chance of secret attacks are, rather than, say, just somebody who has written a textbook or is an eloquent writer or proficient implementer.

Remark 5.17. The reason that I present a rarity argument at all—despite its limitations—is that a rarity argument can still provide some assurances to the user, provided that the user believes that no secret attacks correlated with the rarity set exist. The reason that I presented this long counterargument—including examples—against rarity arguments is to avoid misleading the reader by overstating the value of rarity.

5.6 How the CM55 Parameters were Found

The rarity arguments above attempt to address the possibility of the CM55 parameters being generated maliciously. The CM55 were not generated with a rarity argument in mind. The rarity arguments above are essentially afterthoughts. This section describes how the CM55 were actually generated, if the reader is willing to trust me on that point.

Recall, that we sought a curve with a point of prime order n such that $n-1$ a small multiple of a power of two. The only way I knew how to generate ordinary (non-supersingular) with this property was with the CM method, which I understood to work in such a way that the curve order u and the a number called a discriminant D could be used to determine a prime p for the field size, and then to determine the curve. I had implemented a version of the CM method described in the standard IEEE 1363 [IEE00], and was not working from any deeper understanding of the CM theory.

The CM method, as I understood it, generates pairs $p = (x\pm1)^2+Dy^2$ and $u = x^2+Dy^2$ for integers x and y (not point coordinates). Since, I did not expect to have much control over p, I did not expect to be able to choose an even value for p, so that required p to be odd. This requires u to be even, and actually a multiple of four. So, I sought a minimal cofactor of four. This makes x and y even, and dividing by four gives $n = (x/2)^2+D(y/2)^2$. The simplest way to make $n-1$ divisible by a large power of two was to make $x/2 = 1$ and y divisible by a large power of two. Usually, the CM method would try various $n-1$ of the desired form, and then use Cornacchia's algorithm for various values of D. At the time, I was not contemplating rarity argument, so took the simpler strategy, putting $x = 2$ and trying various pairs (D, y) with D small and y divisible by a large power of two. Each pair (D, y) yields candidate pairs (p, n) which can be tested for primality. This led quickly to the pair $(55, 2^{288})$.

I would need to dig through some archives to see if I have records of the search criteria for the (D, y) and the set of results found. Off-hand, I remember thinking that the pair $(55, 2^{288})$ seemed nice enough to be noteworthy.

5.7 Special Curves: Serpentine Readiness and Diversity

Koblitz, Koblitz and Menezes [KKM08] argue that special curves might survive significant attacks that random curves do not. For convenience, we may call this a serpentine attack. A serpentine attack seems very unlikely, but nonetheless preparing for such a possibility may be worthwhile. Indeed, the whole point of the discipline of provable security is to rule out the effects of unanticipated attacks. In that spirit, we can see what can be done.

Note that a reasonable model for serpentine attack would be the Pohlig–Hellman (PH) attack, if it had been a secret. So, to make this discussion less abstract, we may occasionally refer to the PH attack. I credit Bernstein, in emails to the CFRG list, with using PH as a model for such a serpentine attack, but please do not assume this discussion represents Bernstein's position.

It seems preposterous that a serpentine attack would exempt every single curve that seems special, peculiar, or rare in any way at all. Rather it seems much more plausible, that only specific characteristics of the curve will ensure resistance to the serpentine attack.

For example, if the PH attack had been a serpentine attack, then the safe special characteristic would be having a large prime order subgroup. For random groups chosen with size sufficiently larger, on the basis of having a margin of error, a serpentine PH attack would reduce the security level but still leave the security tolerable. For smaller curves, only the curves special in the sense of having nearly prime order would serve. Continuing with this metaphor, we can say that nearly prime order grants immunity from the serpentine attack.

A CM55 curve might survive a serpentine attack in which low discriminant or low trace grants immunity. Taking the metaphor further, in our population of curves, having some diversity of curves may permit better recovery from a serpentine attack, because one of the curves in the diverse population may have immunity to the serpentine attack.

Remark 5.18. A notion of being ready to switch algorithms in case of an attack is sometimes called algorithm agility. The notion of using diversity aiding recovery by increasing the chances of immunity would therefore be an instance of algorithm agility.

Of course, this recovery against serpentine attack would not be retroactive. In other words, having a CM55 ready to use, but not actually used, will not provide forward secrecy in the face of serpentine attack.

A costly remedy would be the use a diversity of curves in a single ECDH key exchange. In other words, Alice and Bob do ECDH with multiple different curves, and then compute a session key derived from the resulting set of multiple ECDH shared secrets. As some user devices become more proficient, the real-world cost of this remedy may go down. In metaphor, this extra cost would represent an effort to obtain immunity from a serpentine attack, a kind of vaccination.

If one does apply multiple algorithms, it makes sense not only to have diversity, but to try to have substantive diversity. For example, using two curves one of whose j-invariant has a natural decimal representation contains fifteen consecutive digits of value five, and another curve that does not have the same property, would represent an artificial diversity. There is hardly any reason to believe that a serpentine attack would treat these curves differently. A better form of diversity is a difference that is already widely suspected to affect security, or perhaps just efficiency.

So, according to conventional wisdom, the low discriminant of CM55 makes it suspicious. If the conventional wisdom is exactly wrong, then perhaps low discriminant curves are a good vaccine against a serpentine attack. Of course, if using a single curve, one should probably follow one's suspicion, rather than the opposite. A second aspect, in which CM55 differs from average, is the near-optimal tightness of the den Boer reduction. Under the analysis of this report, this feature appears to be only mild security advantage over a random similar-sized curve. But perhaps, under a serpentine attack, this mild difference will become a large difference.

To make this less abstract, return to the example of the PH attack as a serpentine attack. Keeping in mind a hypothetical ignorance of the PH attack, consider three classes of curves: prime order curves, random order curves, and smooth order curves. Suppose further that the prevailing wisdom was to use a large margin or error and use 512-bit curves and for everybody to use their own curves. So, against the serpentine PH attack, the prime, random and smooth curve orders would offer on average about 256-bit, 84-bit and say 10-bit security for the DLP. In this setting, one might think that random order curves offer better security since factorizing the curve order is harder (though nowadays might be quite feasible), and as such are equipped with an additional hard problem, which may have potential benefits. In other words, the conventional wisdom may have slightly favored random order curves. Yet, under the PH attack, prime-order curves should be strongly favored over random curves of equal size. On the other hand, a dissenting view might have been that prime order curves somehow provided some other benefit, such as simplicity.

6 Conclusions and Questions

As a result of writing this report, I consider the CM55 parameters as experimental. In other words, I encourage: implementers to determine how efficient the CM55 parameters can be made; cryptanalysts to further investigate the risks of low discriminant and low trace curves; cryptographers to consider the theoretical security benefits of CM55; elliptic curve specialists to find curves that improve upon CM55's properties (especially curves with a larger margin of error beyond the 128-bit security level).

For those who can afford the extra cost, such as end-to-end email users, I recommend using CM55 with some form of multiple-curve-based ECDH-based key agreement. The other curves should complement CM55 in the sense of avoiding the potential risks of CM55, such as low magnitudes of trace and discriminant, and non-pseudorandomness. In other words, some form a random or pseudorandom curve should be used to complement CM55's deficits, with the CM55 contributing a benefit from its tighter reduction between the ECDH and ECDLP.

A Actual CM55 Curves

In 2005, I applied the CM method not only to find the CM55 parameters, but also to find an actual CM55 curve, with short Weierstrass form coefficients:

$$
\begin{aligned}
a &= 27228472517654392839423150699310359352727681\backslash\\
&\quad 28884604946954788278061429824679470260583 9963,\\
b &= 19862846360388611577546848247471636335510997\backslash\\
&\quad 19007654476288800239255609922781739015146 0715,
\end{aligned}
$$

in decimal radix expansions, each integer broken into two lines with a backslash terminating the unfinished expansion on the first line (one of the conventions in some programming languages to break long lines).

Obviously, these can be scaled to other values, and the curve equation can also be transformed to other shapes such as Montgomery or Edwards. Also, since $h(-55) = 4$, I think there would be four possible j-invariants.

I am not sure exactly which endomorphisms this particular curve admits, because I did not pursue that question at the time I generated it. But I now suspect that it would have conductor-1 order for its endomorphism ring and thus admit all possible algebraic integers in the quadratic imaginary number field $\mathbb{Q}(\sqrt{-55})$.

Note that by taking $D = -55f^2$ where the conductor f is a power of two, and computing the Hilbert polynomial $H_D(x)$ one might be able to get yet more j-invariants matching the CM55 parameters. The larger conductors would likely result in larger norm endomorphisms, which would likely be less efficient. Nevertheless, efficient polynomials can sometimes have high degree, so it might worth investigating such curves too. This might also alleviate some of the suspicion attached to low magnitude discriminants (though the discriminant would still be very smooth).

If I understand the CM theory correctly, all curves with the CM55 parameters have a discriminant of the form $D = -55f^2$ for some f which is a power of two, so the paragraph above describes all CM55 curves, of which one can expect about $\sqrt{p} \approx 2^{147}$. Furthermore, finding CM55 curves with large f would be infeasible.

B Comparing Curves of Given Security and Efficiency

Given a choice between two curves, one can compare them on the basis of security and efficiency. This section tries to more formally paraphrase some of the usual ways people propose making these decisions. Furthermore, it makes some recommendations which might be novel.

Remark B.1. This report does not consider a specific curve, or set of curves in mind, with which to compare CM55 curves. The interested reader should be able to apply the comparison methods in section to compare another curve (or curves) to CM55, or to compare any two (or more) other curves.

B.1 Absolute Metrics

Obviously, one's first task to is to establish some metrics[2] of security and efficiency.

For the purposes of comparing algorithms, I recommend that the metrics chosen to be absolute in two senses:

- not being being defined relative to some other algorithm-dependent parameter,
- expressed in quantifiable units of algorithm cost to permit trade-off between metrics.

For a non-absolute metric example, if the hardness of the discrete logarithm problem is measured in terms of group operations, then that security metric would be a relative metric not an absolute metric. For another non-absolute metric, consider an efficiency metric that

[2]In math terminology, metric often refers to a distance metric, but here we take a more general meaning of a method of measurement.

simply ranked algorithms, or provided asymptotic running times: this metric would be not be quantifiable enough to enable trade-off between different metrics.

An absolute efficiency metric might be the number of Diffie–Hellman operations per bit operation. An absolute security might be the minimum expected number bit operations to solve a random discrete logarithm problem in the group. Note that these units of the efficiency and security metrics are inverses of each other: the efficiency metric units are inverse bit operations, and the security metric units are bit operations.

The unit of a bit operation in the example metrics above is meant to be a platform-agnostic choice of unit, but it is not very pragmatic. For a more pragmatic metric, one might use clock cycles on some kind of reference platform using some kind of best effort implementations of the algorithms (perhaps with some heuristic estimates for the attack algorithms).

For many natural choices of units, the efficiency metric will be a number smaller than one, while a desirable security metric will be a large number. Regardless, users want both metrics to be as large as possible.

For notation, suppose that one has established some absolute efficiency metric and security metrics, e and s, and one wants to compare two curves C and D. We may write s_C and s_D for the value of the security metric for C and D, respectively, and e_C and e_D for the respective efficiency metric values.

Generally, we will simplify matters by assuming that the security and efficiency have been reduced to a single quantity. A more complicated analysis might encode these metrics as vectors. In that case, some of the discussions below will be further complicated.

Some security metrics involve a success rate for the adversary, and also an assessed risk of the existence of an attack algorithm of a certain cost. This can complicate matters. For simplicity, these multiple quantities might be rolled into a single quantity.

B.2 Excluding Defective Algorithms

Defective security s_0 is a level of security below which algorithm is breakable and just not acceptable. For example, attacks could be practical. One might set s_0 according to one's adversary. One usually aims for a security level well above s_0, as we shall soon see. Throughout, we exclude from all consideration any algorithms we know to have a defective level of security.

Similarly, a defective efficiency e_0 is an efficiency below which the algorithm is just unusable. The level of e_0 may vary between devices.

Over time, both s_0 and e_0 have a tendency to decrease, as devices become more proficient. But for deciding what algorithms to use now, we exclude any algorithms with defective security or efficiency. In the case of s_0, we might to anticipate the time depreciation.

B.3 Dominance: A Partial Ordering via Metrics

Suppose that the security metric allows comparisons, that is, it is rankable, and likewise assume the same for the efficiency metric.

This allows us to define a dominance ranking betweem curves C and D, if $e_D > e_C$ and $s_D > s_C$, then we say curve D dominates curve C, and D is clearly superior to C. See Figure 1, in which the curves C and D are indicated by circles in security–efficiency plane.

(The security axis is shown logarithmicly for consistency with the following figures.) The circle for D contains a $+$, and that for C contains a $-$ sign, indicating the superiority of D over C.

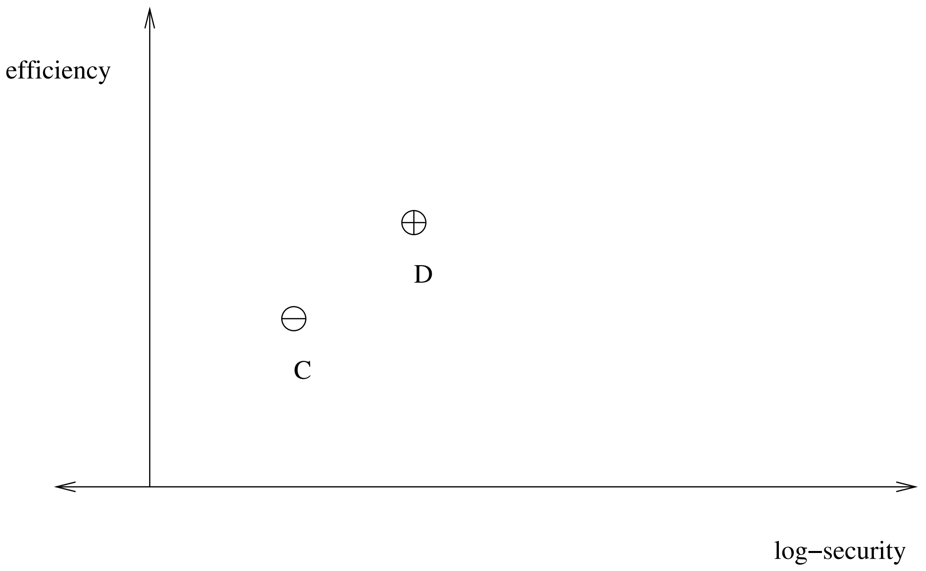

Figure 1: Security–Efficiency Dominance of Curves (Algorithms)

If one of these inequalities is replaced by the corresponding approximation, that is, if $e_C \approx e_D$ or $s_C \approx s_D$, then one can say that C weakly dominates D, and one may still prefer curve C, though the choice would be less clear.

If one of these inequalities is replaced by the opposite direction inequality, say $e_C > e_D$ with $s_D > s_C$, then the curves are mutually non-dominating, and the choice of curve (or more generally, algorithm) will have to be determined by other means. See Figure 2, in which the circles indicating the curves are left empty here, because at this point, no reason has been proffered to prefer one over the other. The following sections describe some ways of deciding between mutually non-dominating algorithm candidates.

B.4 Saturated Security? Optimize Efficiency!

Suppose that there is threshold amount s_1 of computation which is infeasible. Any larger amount of computation is also infeasible, so any security level $s > s_1$ is a saturated security level, and adds no value.

The justifications of the existence of such a threshold s_1 are quite reasonable. For example, most people believe that the Moore's law will stop, and perhaps has already stopped. It is also possible to argue along the lines of amount of computational power per time per mass of a modern-day CPU, multiply this by the mass of the earth, and the time in which one wants security, plus some corrective factor. One can delve even deeper, perhaps invoking some laws of physics that conserve information exchange in terms of energy transfer (and perhaps such a law would be independent of other unknowns in laws of physics, such as the characteristics of graviton, or what not).

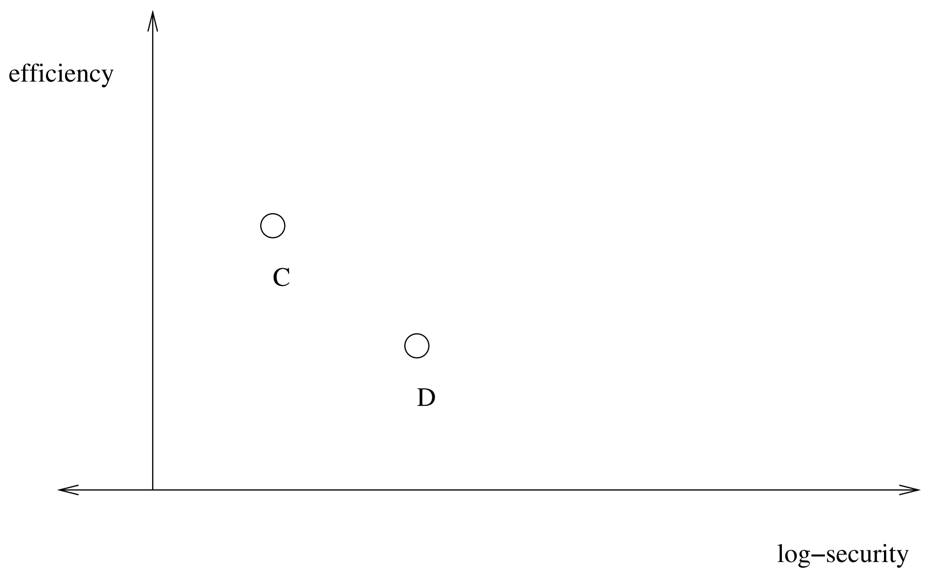

Figure 2: A Non-Dominating Pair of Curves (Algorithms)

One can reasonably set s_1 to be the infeasible amount of computation for today, or before some (near) future time.

It seems that the exact value of s_1 may vary drastically between methods to determine it. It may also depend on one's degree of optimism and pessimism, perhaps about the nature of computation.

Also, probably no sharp threshold s_1 actually exists either. If s_1 represents an infeasible amount of computation, then likely so does $s_1/2$. Perhaps algorithms that require $s_1/2$ effort to break should be considered. Then perhaps $s_1/4$ and so on. For simplicity, one should just set a value of s_1 and stick to it.

If one has has the luxury of choosing between two curves C and D that one believes to have saturated security, then one can choose the one with better efficiency. For example, just choose the one with greater efficiency, such as C if $e_C > e_D$. See Figure 3.

Remark B.2. The notions of defective security and saturated security squash the absolute metric by making its extreme values unrankable, so that it is no longer an absolute metric. It is my view that this outcome should not deter us from trying to develop an absolute metric first, and then determining the defective and saturated ranges.

Remark B.3. The log-security axis contains negative values, because the security level represents any non-negative quantity of work for an adversary. In the quantities of work are measured in discrete units, such as bit operations, then zero can be taken as smallest value of the log-security, and one can ignore the negative values of log-security. If some continuous measure of security is used, such as energy, or if some more complicated units intermixing a discrete cost and continuous probability of success, then perhaps negative values of log-security will arise.

The figures will generally adhere to the convention that defective log-security is at least zero, so negative values of log-security will always be out of consideration.

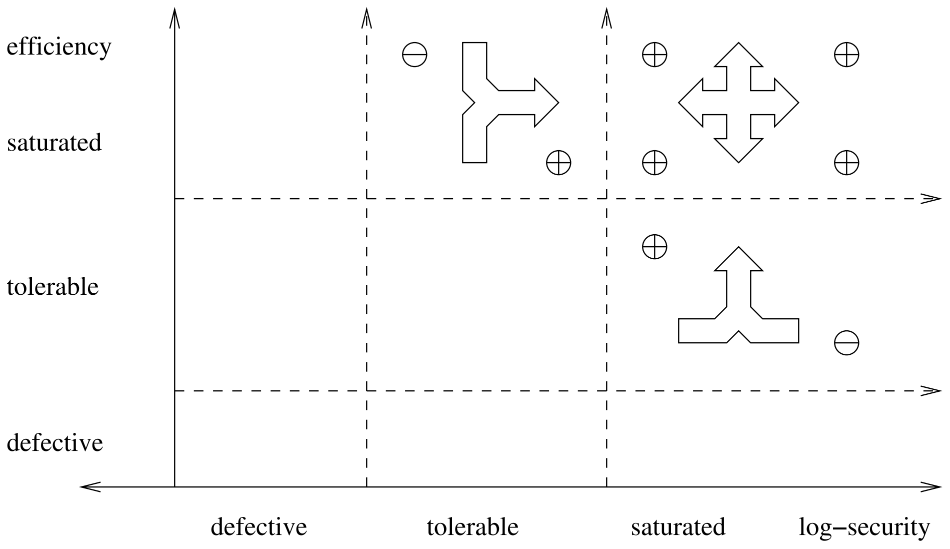

Figure 3: Saturated Security and Saturated Efficiency

B.5 Saturated Efficiency? Optimize Security!

Cryptography is used by people, and implemented by computing devices. For a specific user application on a personal device, one can probably establish a level of efficiency e_1, which is fully sufficient. For example, if Alice wants to send Bob an encrypted personal email, she may need to do one Diffie–Hellman operation. Alice will not need to send personal emails beyond a given a rate, say one per second, so it may fully suffice, for this application, for a Diffie–Hellman operation to take a tenth of a second (of real-time, assuming a normal load for other activities) on Alice's device.

So, there may be some level e_1 of saturated efficiency, beyond which any more efficiency adds no value.

Different applications and entities may have different levels of e_1. For example, a web site with large amount of traffic may value efficiency much more than a single user. Even so, a busy web site may still have some level of sufficient efficiency, perhaps determined by its existing equipment, existing budget, expected number of user traffic, and predicted profit per customer, and so on.

To accommodate for multiple applications and entities using a single algorithm or curve, one can take the maximum of all the individual saturated efficiencies as the system-wide saturated efficiency.

Now supposes one has two curves with efficiency better than one's saturated efficiency: $e_C, e_D > e_1$. At this point, it makes sense to choose between C and D on the basis of maximum security, such as choosing D if $s_D > s_C$. See Figure 3.

So, when saturated efficiency is available, we have the luxury to optimize security.

Note that e_1 tends to decrease over time as computing devices become more proficient, because we have assume an absolute metric for efficiency. Consequently, even if an algorithm does not have saturated efficiency today, it have saturated efficiency the future.

If one has the double fortune of curves with both security and efficiency saturated, then one is really free to choose anything saturated. See Figure 3, in which the top right corner illustrates this situation.

Remark B.4. If one thinks that optimizing efficiency is a good option here, then really efficiency has not been saturated.

One might also have a pair of non-dominating curves C and D, where: C has saturated efficiency with tolerable security; and D has saturated security with tolerable efficiency. In this case,

In the remaining sections, we treat the case of non-dominated pairs of curves (or algorithms) C and D that are not both saturated in the same metric (either security or efficiency). This includes the middle empty box in Figure 3, but also the case of one curve in the top-middle box and the other curve in the right-middle box.

B.6 Trading Efficiency for Security, and Vice Versa

The opposite of saturated efficiency is defective efficiency e_0. Any efficiency below e_0 is just unusable. Anything between e_0 and e_1 is tolerable efficiency.

Similarly, the opposite to saturated security is defective security s_0. Below this security level, all attacks are equally feasible for the purposes of one's security. Any security value between s_0 and s_1 is tolerable security.

Suppose that one has a pair of non-dominating algorithm candidates, one with tolerable efficiency, and the with other tolerable security.

- trade efficiency-for-security (E4S), or
- trade security-for-efficiency (S4E).

So, suppose that curve C has tolerable security and curve D has tolerable efficiency, and

$$s_1, s_D > s_C > s_0, \tag{31}$$

$$e_1, e_C > e_D > e_0. \tag{32}$$

In the trade E4S option, choose D, and in the trade S4E option choose C.

These trading strategies are rather simplistic because they work solely by ranking of the metrics. They do not try to take advantage of the absolute nature of our metrics. Recall that we sought metrics in which security and efficiency are measured in similar units, allowing meaningful exchange between the two.

Also, if there is a large array of algorithm choices, which is the case for elliptic curves, following these simplistic strategies to their fullest extent may take one to near the defective level of the metric that is one trading away. In other words, the use of simple ranking in the E4S and S4E trading strategies completely ignores the finer details of security and efficiency, and cannot detect if a large loss in efficiency is made for small gain in security, or vice versa.

Furthermore, there is not sharp cutoff of the defective levels, just as there is no sharp cutoff at the saturated levels. Therefore, the simplistic trading strategies E4S and S4E risk making our algorithm precariously close to defective. We may want a better strategy.

B.7 Work Ratio: Security times Efficiency

If one's bound for saturated security and saturated efficiency are higher than all one's known algorithms, or even if one has no doubly-saturated, algorithms, then rather than trying to optimize only one of security and efficiency, as above in §B.6, one can instead attempt to unify the metrics into a single objective to optimize.

One reasonable unified metric is the work ratio w, which is security times efficiency $w = se$. This metric can be called the work ratio because efficiency is an inverse of the user's work, so the work ratio is the adversary's work divided by the user's work.

Note that for work ratio to make any sense, it is important to use absolute metrics—not metrics that are artificially relative to something like curve size.

In the security-efficiency plane, the constant work ratio contours are hyperbolas. But if we plot these contours using the logarithm of security as the x-axis, then the contours become reverse exponential functions to the base two. See Figure 4. In this plane, if the defective efficiency level is non-negligible, the nearly vertical slope of these contours makes work ratio appear quite similar to log-security. Note that this is vertical appearance is preserved under scaling of efficiency and translation of log-security. These exponential functions are of course what happens to the hyperbola contour as the security compressed into the log-security axis.

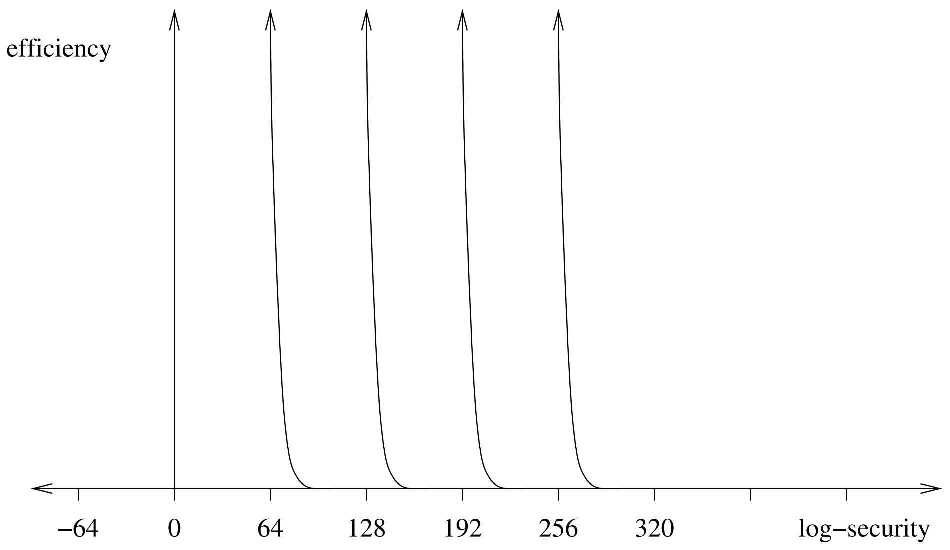

Figure 4: Work Ratio: Constant Value Contour Curves Nearly Vertical in the Log-Security-Efficiency Plane

Remark B.5. Figures can be misleading, but so can numbers. It is common to refer to security in bits, or in powers of two. Hence this report's use of log-security in the figures. Also, it is common to consider a range of security levels that cannot fit into any meaningful plot.

On the one hand, one might think the steep slopes of work ratio contours as a misleading image. The steepness is partially an artifact of the tremendous compression that happens when viewing log-security instead of security. We will see later another intermediate metric with less steep slopes. On the other hand, using log-security numbers is misleading in the sense of under-representing larger security levels, e.g. 2^{256} or 256-bit, security is substantially more than than 2^{128} or 128-bit security. As much as a person tries to remember that one is thinking about log-security, not security, it is so easy to slip into the thought that the gain is small.

If the figures in this report used security on the horizontal axis instead of log-security, and showed a 2^{256} security level, the security levels 2^{128} and 0 would be closer together than a proton.

More specifically, the work ratio in elliptic curve cryptography usually depends mainly on the curve size n. Suppose that curves C and D have nearly the same orders $n_C \approx n_D \approx n$, but clearly different efficiencies $e_C > e_D$. A DH-based efficiency metric usually means that the group operation will be proportional to the DH operation. The Pollard rho algorithm will take a time of $C\sqrt{n}$ group operations (but perhaps with an extra factor depending on n). In this case, the ratio w_C will be proportional to about $\frac{\sqrt{n}}{\log(n)}$ and depend only on n. So, $w_C \approx w_D$. In particular, the effect of the curve efficiency e_C will be cancelled out of the the work ratio. (This may be counter-intuitive, but see the next section on progressive ratio.)

So, using work ratio for comparing curves will usually favor larger curves, maximizing the curve size n_C, always assuming prime order curve sizes. (This may change for different cofactor values.)

Curves can be arbitrarily large, as the curve size increases, its size will reach the point where either:

- efficiency $e_C \approx e_0$ becomes almost defective, and thus barely tolerable, overriding the benefit of work ratio; or

- security $s_C \approx s_1$ becomes almost saturated, and thus one can safely optimize efficiency without worrying any more about work ratio;

or perhaps both. The latter case in which security continues to grow with size hinges on the widely accepted belief that generic group algorithms represent the main way in security depends on the elliptic curve choice.

Given a choice between two or more curves (algorithm) with the equally good work ratio, they will necessarily be mutually non-dominating, because of the negative slope of the work ratio contours. If none of the curve choices have both security and efficiency saturated, and if not all have the saturated security, and if not all have saturated security, then the trading methods above can be used, but these are not ideal. The next intermediate metric might be useful in this context.

Remark B.6. The near vertical slope of the work ratio contours creates a misleading impression that climbing up the slope leads to much greater gain than descending it. This is an artifact of the compression used in drawing log-security. One should not base a curve choice on such an artificial sketch, I think.

Remark B.7. Given the previous observation that work ratio depends mainly on curve size. It seems that work ratio ties will arise mainly between of very similar sizes.

It seems to be a tradition in ECC to favor efficiency in this situations. Consider the NIST curve P256 which uses a special Solinas prime to be more efficient.

This would be perfectly justifiable if one thinks security saturated, but another tradition in ECC is to consider two or more different security levels, which indicates that at least the lower levels among the list are not saturated. For example, the NIST recommended curves and the Suite B subset of these.

Because smaller-sized curves are considered in these lists, it seems that work ratio is not being optimized. Perhaps these multiple security levels represent different estimates on what constitutes saturated security. But that seems unlikely, since the estimates for saturated involve simple powers of two for the approximate curve sizes, rather than some kind of assumptions about computing power and so on.

Therefore, the current tradition seems not to follow the strategy of work ratio. This seems to be mistake.

Remark B.8. Another example of curves with similar work ratios are the ten NIST binary curves. They group in five pairs, a random curve and a Koblitz with nearly the same size. It suspect that the Koblitz has a smaller work ratio, because the speed-up of Pollard rho due to the Frobenius endomorphism is greater than the speed-up of scalar multiplication. If this is true, then one should prefer the random curves, unless the random curve has defective efficiency, or the corresponding Koblitz curve has saturated security.

B.8 Progressive Ratio: Log-Security times Efficiency

The progressive ratio, defined below, provides a second intermediate metric that unifies security and efficiency metric. For the reasons stated below, this metric should be optimized only after optimizing the work ratio metric. That said, it is probably less arbitrary to optimize this metric after work ratio, than to merely pick one of efficiency or security as a secondary objective.

First, extrapolating Moore's law in a simplistic form, define the log-security value $t = v\log(s)$, where v is some constant, as representing the amount of time t the algorithm can resist an attack. Log-security essentially grants the adversary, and actually anybody, effectively exponential-time computing power, which is bizarrely contrary to the usual notions of computational complexity.

Log-security is monotonic function of security, so preserves any ranking and thresholds. Therefore, one can talk about defective, tolerable, and saturated log-security, and one can trade between log-security and efficiency, without making any difference to one's final decision of curve. So, using log-security instead of security in these decision strategies serves merely as a convenient way to view and think about the security metric, such as for displaying in a table or a graph or customer explanation.

The progressive ratio goes one substantial step further with log-security, by multiplying it by efficiency; so the progressive ratio is $u = ve$. Doubling the efficiency of an algorithm doubles its progressive ratio, squaring its security doubles its Moore value, because the progressive ratio only presumes that squaring the security doubles the lifetime. The progressive ratio places a bet on the fact that Moore's law will progress indefinitely, and as such, is a rather pessimistic bet since it favors the adversary of the future.

One technical difficulty with the progressive ratio is its lack of scale-invariance: if we change our units of costs in the absolute metrics, this can change the progressive ratio. By contrast, in the work ratio, the cost units cancel, so work ratio is more a dimensionless parameter. By contrast, progressive ratio has some units: log-cost divided by cost. Nevertheless, when comparing the progressive ratios of two different algorithms, this is not a problem if one uses the same cost units for each algorithm.

Again, I recommend optimizing work ratio before optimizing progressive ratio, because the progressive ratio depends on the questionable assumption of Moore's law continuing indefinitely, whereas the work ratio does not. However, even after optimizing work ratio, (or trading security or efficiency), there may still remain a degree of freedom in the choice. At this point, it appears reasonable to optimize the progressive ratio.

The progressive ratio has one potential advantage over the work ratio, in that it perhaps models an adversary in the future, which may be useful for things like confidentiality. For security goals like real-time authentication, future adversaries are not a concern, so the progressive ratio appears irrelevant. Beware: even signature algorithms can also provide stale authentication, which needs future security, so the progressive ratio may be relevant to signatures.

In the case of elliptic curves, unlike work ratio, the progressive ratio depends more strongly upon the curve efficiency, rather than just the curve size. One might expect that, for a fixed curve size, the progressive ratio generally increases with efficiency of the curve. But one should calculate this to be sure.

Graphically, the progressive ratio is just a hyperbola on the log-security-efficiency plane as in Figure 5. This figure, if overlaid with Figure 4 suggests that, for any algorithm

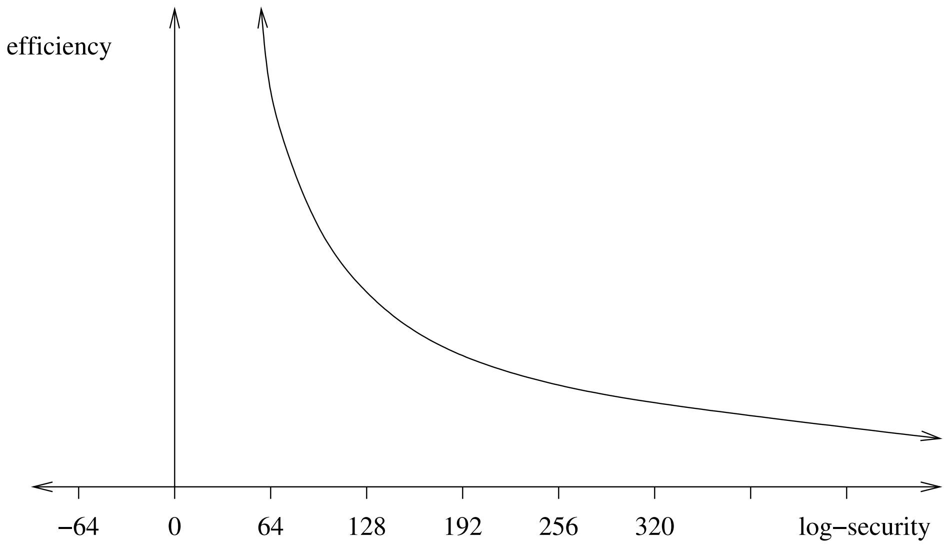

Figure 5: Progressive Ratio: Constant Value Contours are Hyperbolas in the Log-Security-Efficiency Plane

whatsoever, when making a choice between two algorithms with the same work ratio, one should choose more efficient one will have a higher value of progressive ratio. This seems to follow because the work ratio contours appear always steeper than the progressive ratio contours where the contours cross.

Remark B.9. Actually, this appearance may not quite be correct, because Figure 5 shows only one contour. There is actually a divide in the plane, whose graph is reverse exponential similar to the work ratio contours, below which the progressive ratio contours are steeper. The dividing curve is very close the horizontal and vertical axes (log-security of zero).

Anyway, this seems to better justify choosing the the most efficient curve of a given size. In other words, the benefit of efficiency to user is greater than to the adversary, because under progressive ratio, a gain in efficiency buys the user more time in the present than the secure lifetime is lost in the future under Moore's law progressing.

Conversely, if one makes progressive the primary objective, and work ratio the second, then one would choose resolve a tie of optimal progressive ratio by choosing the highest security, because that should optimize work ratio.

B.9 Curve Availability

Given the comparison strategy above, a natural next step for choosing an elliptic curve is to plot out in the log-security-efficiency plane a graph for all available elliptic curves. For completeness, one might include in the plot fairly large and small curves. Obviously, there is an infeasible number of curves to plot each individually, but one can take some class of curves with a single parameter, and plot a curve based on this parameter.

For example, consider the class of curves with a prime order n, so cofactor one, where the field size has been selected randomly from some interval, and the j-invariant has been selected randomly, subject to the condition that the curve order n is prime.

This class has a few security outliers due to the MOV and SASS attacks, but otherwise the security and efficiency can be heuristically estimated based on the parameter n.

We will call this class the main stream. A number of possible heuristics can be used to estimate the efficiency and security of such curves. Let us estimate the field arithmetic efficiency by:

$$e_f(n) = \frac{C_f}{\log_2(n)^{k(n)}} \tag{33}$$

where the Karatsuba exponent $k(n)$ starts from 2 and decreases erratically to 1. For simplicity, we may consider $k(n) = 2$ for all n in the range of interest. Here C_f is a constant for all curves, and depends on the units, and other non-curve dependent details. From this, we can the ECDH efficiency is given by approximately:

$$e(n) = \frac{C_f C_c}{\log_2(n)^{1+k(n)}} \tag{34}$$

because scalar multiplication takes a linear number of field operations. The constant C_c allows us to choose the best generic curve formula, windowing techniques, and so on, as long as they are applicable to random field and random j-invariant for all of our mainstream curves.

Next, we might want to consider a security parameter. For example, we can take the expected cost of the Pollard rho algorithm (with success rate one-half, say), because this is the best way to the solve the ECDLP and ECDHP.

Remark B.10. If one instead considers the known provable lower bounds on the ECDHP, relative to some conjecture on the hardness of the ECDLP, then one would have a different security metric. In this metric CM55 would fare better than average for its size: it would be especially secure. More generally, the security metric would depend on the factorization on the factorization of $n - 1$, so would not appear very continuous in a figure.

Of course, under some security metrics, such as those that incorporate a penalty for the perceived risk of attack from low magnitude discriminant, the CM55 could would fare worse than a mainstream curve.

In this case, the security metric would have an approximate formula:

$$s(n) = C_s \sqrt{n} \log_2(n)^{k(n)}. \tag{35}$$

where the constant C_2 accounts for the units of cost, the expected run-time of Pollard rho (with any generic speedups due to negation operator, and so on).

Remark B.11. This cost might have an additional factor depending on n to account for quality of the pseudorandom function. For now, I presume not.

If we accept the heuristic approximations above, and force $k(n) = 2$ for simplicity, we can derive a relationship between $s(n)$ and $e(n)$ and then plot the curve. First solve for $\log_2(n) = C_e e(n)^{-1/3}$. Then solve for log-security:

$$\log_2(s(n)) \approx \tfrac{1}{2} e(n)^{-1/3} - \tfrac{2}{3} \log_2 e(n) + \log_2(C_s) + \tfrac{1}{2} C_e \tag{36}$$

The exact nature of this curve depends on many details, including the constants and units. But where the first term dominates the shape of the curve, we will have $e(n)$ proportional to $s(n)^{-3}$. At these points in the mainstream curve, it seems plausible that the mainstream curve is steeper than the contours of progressive ratio, but less steep than work ratio contours. If so, optimizing for progressive ratio before work ratio results in a smaller curve, and more efficient curve among the mainstream curves.

One is also interested in special curves outside of the mainstream. For example, curves with efficient field arithmetic, extra security properties, or large cofactor. This is illustrated in Figure 6.

Warning: the positions of of special curves in Figure 6 is not to scale.

- A curve with extra-efficient field arithmetic, has slightly less log-security, because efficiency helps the attacker. This does not affect the work ratio at all, but can potentially improve the progressive ratio. Such a curve probably dominates smaller curves, but not curves of the same size. In the figure, the drop in log-security, though small, is actually exaggerated.

- A curve that is especially secure (under some other security metric), that is, more secure than a mainstream curve of the same size, would dominate curves of the same size, and slightly larger size, and would improve work ratio, and progressive ratio. In the figure, a very dramatic improvement for log-security is shown, but this is unlikely at a quantitative level. Nevertheless, this is not totally misleading, because log-security has been used for the horizontal axis, which severely under-represents the role of security. So, an especially secure curve improves both work ratio and progressive ratio, which is a good improvement of nearby mainstream curves, unless efficiency is worse than average.

- Enlarging a curve's cofactor, but not size, allows more efficient ECDH operations because private keys are shorter. But it is also less secure than a curve with the

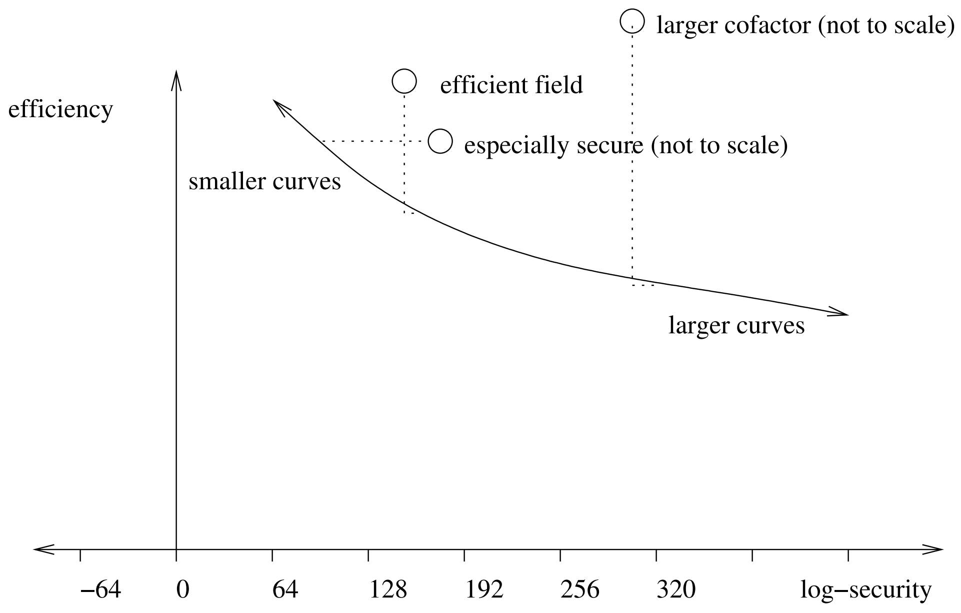

Figure 6: Special Curves Compared to Mainstream Curves (Not to Scale)

same size but with smaller factor (including a mainstream curve), because the group used is smaller (or due to the Pohlig–Hellman attack if the larger composite order group is used). In general, if the cofactor is h, then the efficiency of ECDH can be about $\left(1 - \frac{\log(h)}{\log(n)}\right)^{-1}$ times better than for a mainstream curve of the same field size, because the private keys can be chosen to be smaller.[3] (This efficiency gain is absent in cofactor-multiplicatio modes of ECDH.) The security falls by a factor of about $\sqrt{h}$ in the sense that the Pollard rho becomes about $\sqrt{h}$ times faster. So, the overall work ratio decreases by a factor quite nearly $\sqrt{h}$, for small h. So, generally, increasing cofactor decreases the work ratio. In other words, when optimizing work ratio, a small cofactor is ideal.

That said, more refined analyses of security and efficiency can arise. Regarding efficiency, it has long been known that, when the cofactor is divisible by two, prime-field curves can be transformed into Montgomery form, which offers a more substantial efficiency gain than the one above due to smaller keys. More recent work of Bernstein and Lange, has showen that when the cofactor is divisible by four, the curve can be transformed into various kinds of Edwards forms, which further improve efficiency. Regarding security, both the Montgomery and Edwards forms of curves are amenable to constant-time scalar multiplication algorithms which mat help reduce the risk of side channel attacks. In other words, a cofactor of two or four may boost both the security and efficiency. These kinds of observations are due to Bernstein. It seems a bit peculiar that cofactor 1 one curves are not amenable to these security and effi-

[3]In an earlier version of this report, I incorrectly claimed an efficiency improvement factor of h, which is drastically flawed, and leads to non-sensical conclusions. I thank Mike Hamburg for responding to my message on the CFRG list to help me realize my error.

ciency techniques, but that quirky situtation seems to be what the current research is suggesting.

C Implementation Sketch

Two important user implementation issues are efficiency and side channel resistance.

To concretely measure the efficiency of an algorithm, one can implement it across a set of target user equipment. This measure would be specific to the equipment on which it is implemented.

This section provides a more abstract and incomplete approach. I apply and adapt some of the efficient implementations from Hankerson, Menezes and Vanstone [HMV04] to the CM55 curves. More specifically, I find mathematical formula merge the idea of Montgomery multiplication with the Solinas reduction algorithm for special primes.

I have not tested whether these formula are the most efficient way to implement CM55 curves, let alone how they compare to efficient elements of other curves. Rather, I hope that because I have adapted some existing efficient techniques that they will also prove efficient for CM55.

C.1 General Expectations

A large part of the efficiency of an elliptic curve depends on the bit length of the field elements and the efficiency of the modular reduction.

Curve CM55 uses a 294-bit field, which is 38 bits larger than the usual size of 256 bits considered the approximate minimum for acceptable security. As such, we can expect CM55 to be significantly slower than the usual 256-bit curves. If we are aiming for better than the usual 128-bit security, then CM55 might be closer in efficiency to the fastest curves of similar size.

In my non-expert understanding, the most efficient modular reduction occurs when the modulus is just below of a power of two. Examples are NIST curve P521 and Bernstein's curve in Curve25519. Another type of efficiency modular reduction occurs when the field size has a signed binary expansion of low Hamming weight, with the set bits being multiples of a typical computer's word size: the Solinas primes. An example curve with a Solinas prime field is NIST P256.

The primary goal of the CM55 parameters was to optimize the provable security of the ECDHP, for curves of the given size. A secondary goal was efficiency of the field arithmetic. More precisely, the hope was that a CM55 curve is more efficient than a similar-sized curve with a half-random prime. If the implementation ideas in this section can achieve that, then the secondary goal of the CM55 parameters has been met.

C.2 A Naive Field Implementation Outline

This section provides an outline for a naive implementation of prime field arithmetic of CM55. It separates into two separate tasks: implementing big integer arithmetic and implementing modular arithmetic.

I hear that other curve implementations, in particular for Curve25519, aim for a constant-time field arithmetic, as security goal for the user. My outlined algorithm also aims for the same goal. Nonetheless, I have two reservations about that aim, expressed in the form of the following questions:

- Are any known practical side channel attacks due to non-constant field arithmetic?
- Does the noise from non-constant-time field arithmetic help security in hiding the side channel signal from operations that more directly depend on the elliptic private key?

In other words: is there a security gain that warrants the efficiency loss due to constant-time field operations (especially modular reductions)?

Absolute metrics for security try to account to for the computational cost of the best attack algorithms. The field algorithms try to use constant-time, but that is not directly of concern to the adversary. However, constant-time algorithms may benefit the adversary if the adversary is using single-instruction-multiple-data (SIMD) hardware, because many attack algorithms benefit from large parallelism. Often SIMD hardware is available in standard hardware equipment for the purposes of doing graphics processing. If an attacker is designing custom hardware, it may still be possible that custom SIMD hardware will help the attacker in some way, and in this case, the constant-time field arithmetic may help the attacker too.

C.2.1 Big Integer Arithmetic

This report considers a big integer arithmetic system that can represent and operate upon non-negative integers up to a maximum value. More sophisticated implementations may permit negative integers, in order to make various operations such as subtraction faster.

Big integer multiplication is usually implemented using some kind of radix representation of the integers. Given that, $288 = 9 \times 32$, so a radix of 2^{32} might be convenient. If the hardware supports multiplication of 32 bits into a 64 bit word, then this radix could be implemented using the classical algorithms [Knu98]. Barret [?] and Comba [?] suggested using a smaller radix value in this setting in order to reduce the number of carry operations in big integer multiplication. For example, with the CM55 field, a radix of size 2^{31} would require the same number of words, 10, to represent a 304-bit integer as a radix 2^{32}, but could nearly halve the number of carry operations. The advantage seems to be on hardware in which word multiplications have cost similar to word additions, even though the multiplications should require more bit operations. In other words, more hardware area has been devoted to multiplication. A yet smaller radix would require more multiplications, but fewer additions.

Unfortunately, a bit size of 31 may require more non-byte-aligned shifting, however, so it is unclear to me if it would be any better for the multiplication (and further shifting would be required for the modular reductions).

I have heard that more recent implementations, such as that for Curve25519, use such a reduced radix, and greatly benefit from this. If so, then even smaller radix sizes may benefit CM55 field arithmetic. In the case of CM55, the bit size of 288 figures quite often in some of the formulas in the next section. More specifically, some of the integers need to be shifted by 288 bits, or reduced by 2^{288}. So it may help to use a radix whose bit size divides 288, with one of the factors:

$$4, 6, 8, 9, 12, 16, 18, 24, 32, 36, 48, 72, 96. \tag{37}$$

Again, the reason that the radix bit-size dividing 288 may help is that it may reduce the need to bit-shift the words.

C.2.2 Modular Representation and Reduction

This section combines and adapts some of the algorithms from Hankerson, Menezes and Vanstone's book [HMV04] in an effort to provide an efficient representation and reduction algorithm.

It is unclear to me if the algorithm outlined below will be faster than the known algorithms of Barrett reduction or Montgomery reduction, which work for any modulus. I would hazard a guess that it is better than these generic methods, because I have tried to take advantage of the special form of the form of the prime p, mimicking the techniques that have been applied to other special primes, such as the Solinas primes. My doubt on this guess stems from the fact the the CM55 field size p is not as special as these other primes, and perhaps the specialized reduction methods will not compete with the generic reduction methods.

Representation Represent a field element x by any integer x' such that:

$$3025x' \equiv x \bmod p \tag{38}$$

$$0 \le x' < 2^{304} \tag{39}$$

Essentially, this is a variant of a Montgomery's representation. Of course, the value x' will be represented as a big integer, as described in the previous section.

Throughout the following text, we may assume for simplicity a radix size of 2^{32}, so that 304-bit integer might be stored as a 320-bit integer. We will of course need to compute integers larger than 2^{304}, when we add or multiply two such integers, so we will generally need to be able to store and operate integers of bit size greater than 304.

Operations on Representatives Computing a representative $(x + y)'$ can be done by computing $x' + y'$. The result can exceed 2^{304}, making the value land out of range. To keep correct the range issue in the overflow setting, a direct partial reduction, defined later below, can be applied. To help achieve constant-time implementation, the direct partial reduction should always be applied, regardless of whether overflow occurs.

Negation can be implemented $(-x)' = mp - x'$ for some fixed, but suitable choice m, followed by a direct partial reduction. One choice for m would be $\lceil 2^{16}/55 \rceil$.

Computing a representative $(xy)'$ is achieved by first computing the non-negative integer $z' = x'y' < 2^{608}$, and then applying indirect partial reduction which is to a representative $(xy)'$ such that $(xy)' \equiv 3025z' \bmod p$ and $0 \le (xy)' < 2^{304}$.

Direct Partial Reduction The input is an integer j with $0 \leq j < 2^{305}$.

We can write $j = a2^{288} + b$ for non-negative integers a, b such that $b < 2^{288}$ and $a < 2^{17}$.

Compute non-negative integers q and r such that $a = 55q + r$ for $0 \leq r < 55$. This requires a division and remainder operation, but the integer operands are small and non-negative.

Compute the big integer:

$$z = (550 + r)2^{288} + (b + 90) - 9q \tag{40}$$

using big integer addition and subtraction.

Lemma C.1. *With the assumptions and notation above:* $z \equiv j \bmod p$, *and* $0 \leq z < 2^{304}$.

Proof. To show $z \equiv j \bmod p$, one can argue as follows:

$$\begin{aligned} z &= (550 + r)2^{288} + (b + 90) - 9q \\ &= 10p - 9q + r2^{288} + b \\ &\equiv (p - 9)q + r2^{288} + b \\ &= (55q + r)2^{288} + b \\ &= j \end{aligned} \tag{41}$$

To show that $z \geq 0$, it suffices to show that $550 \times 2^{288} > 9q$, which follows easily because $q < 2^{17}/55$.

To show that $z < 2^{304}$, just observe that $z < (550 + 54 + 2)2^{288}$. $\square$

Just to be clear, even if $j < 2^{304}$, that is, if $a < 2^{16}$, one should still apply the same operations above to help towards having a constant-time implementation.

Note that instead of computing the single-word quotient and remainder, one should be able to do a few careful comparisons with numbers of size comparable 2^{304} and a subtraction of an appropriate multiple of p. If done carefully enough, this can be done in constant time.

Indirect Partial Reduction The input is an non-negative integer j such that $j < 2^{608}$. Write $j = a2^{576} + b2^{288} + c$ with non-negative integers a, b, c such that $a, b, c < 2^{288}$.

Compute the big integer:

$$z = (81a) + (10p - 495b) + 3025c \tag{42}$$

using three big integer multiplications in which one operand is small, and two big integers additions and a bit integer subtraction.

Lemma C.2. *With the assumptions and notation above:* $z \equiv 3025j \bmod p$, *and* $0 \leq z < 2^{304}$.

Proof. To show $z \equiv 3025j \bmod p$, one can argue as follows:

$$\begin{aligned} z &= 81a + (10p - 495b) + 3025c \\ &\equiv 9^2 a - (9 \times 55)b + 3025c \\ &\equiv (p-9)^2 a + (p-9)55b + 3025c \\ &= 3025a2^{576} + 3025b2^{288} + 3025c \\ &= 3025j \end{aligned} \tag{43}$$

To show that $z \geq 0$, it suffices to show that $10p \geq 495b$, which follows easily because $10p > 550 \times 2^{288} > 495 \times 2^{288} > 495b$.

To show that $z < 2^{304}$: note that $z < (81 + 551 + 3025)2^{288} < 2^{304}$. □

Conversions Most of the field arithmetic in an entire elliptic curve operation, can be implemented using the Montgomery representation form above. Only the final step will convert back to the normal representation.

The final form requires a full reduction, not just the partial reductions described above. To be completed.

Remark C.1. A full reduction may be necessary in order to achieve interoperability between two parties who may be using different implementations, or who may have different starting points, and may therefore obtain different representations of the same value. Furthermore, the internal representation of the value may contain some additional information about the value. If this extra information is leaked, then the security may be compromised, as in Smart et al, "projective coordinates leak".

C.2.3 Lengthier Field Operations

Elliptic curve cryptography sometimes requires computation of inverses and of squared roots in the field.

Inverses Inverses are usually required to convert between various types of coordinates. In particular, the projective coordinates can be used for the intermediate calculations in a elliptic curve scalar multiplication to calculate a DH shared secret, but an elliptic curve point has multiple projective coordinate representations. If Alice and Bob are to get the same values, they need a unique representations, such as an affine coordinate representation. Conversion from the projective to affine usually requires an inversion.

If Alice and Bob are using an inversion to convert a shared secret to an interoperable form, then they may want to use a constant-time inversion algorithm, in order to avoid side channel attacks. This rules out algorithms derived from the extended Euclidean algorithm.

It appears that Alice and Bob are safest if using an exponentiation algorithm based on Fermat's little theorem:

$$z^{-1} \equiv z^{p-2} \equiv z^{7+55\times 2^{288}} \bmod p. \tag{44}$$

One can compute this with at most 298 modular exponentiations, one for each value in the sequence

$$z^2, z^4, z^5, z^7, z^{10}, z^{20}, z^{40}, z^{50}, z^{55}, z^{55\times 2}, \ldots, z^{55\times 2^{288}}, z^{7+55\times 2^{288}} \tag{45}$$

Again, using some variant of the extended Euclidean algorithm should be faster. If Alice and Bob are likely to compute a given shared secret only once, but still need to compute lots of shared secrets, then maybe they could safely risk the side channels and use the faster algorithm.

Square Roots In an ECC implementation, square roots are mainly needed during a decompression. Compression is mainly used to keep public keys small, for reduced data transmission, so there is usually little risk of a side channel when computing square roots, because the operation affects public information.

Because $2^3 | p-1$, a square root algorithm is a little more complicated to describe than for primes p such that $2^2 \nmid p-1$. Fortunately, it does not require many more modular reductions. The square root algorithm that we now describe is essentially a special case of the Tonelli–Shanks algorithm.

Remark C.2. The Tonelli–Shanks square root algorithm works modulo any prime. For CM55, we need the algorithm for just a fixed prime, and therefore, we can fix certain steps in the more general Tonelli–Shanks algorithm, thereby slightly simplifying the algorithm.

Let:

$$q = 1 + 55 \times 2^{285}, \tag{46}$$

so that $p-1 = 8q$. Suppose that $y \equiv x^2 \bmod p$, and we are trying to find x. If $y \equiv 0 \bmod p$, then output $x = 0$, and we are done. Otherwise, we proceed as follows.

For convenience, as we describe the algorithm, we also describe its proof, which involves introducing some extra variables, g and z. Let g be a generator for the multiplicative group of integers modulo p. The algorithm does not need to use a specific g, although a pre-computation phase of the algorithm is described easily in terms a specific generator g. Let z be such that $x \equiv g^z \bmod p$. Again, g and z are just for the proof, and are not used in the algorithm.

Now, the value of z only matters modulo $p-1 = 8q$. Since 8 and q are relatively prime, we can use CRT coordinates, and write $z \equiv (u, v) \bmod (8, q)$. Note that the square root algorithm will try to determine $2u \bmod 8$, but will not determine v, so v is only used for the proof.

Now, $y \equiv x^2 \equiv g^{(2u,2v)} \bmod p$. Let

$$h = 1 + 55 \times 2^{284}. \tag{47}$$

Note that $2h = q+1 \equiv 1 \bmod q$, so we may write $h = (1, \frac{1}{2}) \bmod (8, q)$ in CRT coordinates. Compute

$$w_{h-1} \equiv y^{h-1} \equiv y^{55 \times 2^{284}} \bmod p \tag{48}$$

which can be done with 292 modular multiplications. Next, compute

$$w_h \equiv y w_{h-1} \equiv y^h \equiv g^{(u,v)(1,\frac{1}{2})} \equiv g^{(2u,v)} \bmod p \tag{49}$$

using one more modular multiplication. Then, compute

$$w_q \equiv y w_{h-1} w_{h-1} \equiv y^q \equiv g^{(2u,0)} \bmod p \tag{50}$$

using two more modular multiplications.

At this point, the algorithm does a table look-up. Define a table of four pairs (g_t, s_t) for $t \in \{0, 1, 2, 3\}$, with values $g_t \equiv g^{(t,0)} \bmod p$ for and $h_t \equiv g_t^2 \bmod p$. This table is fixed and does not depend on y, so it can be pre-computed. (Alternatively, to reduce the storage needed for the table, just store g_1 and compute the rest of the table using five modular multiplications at run-time.)

Let $s = s_t$ for the value of t such that $w_q = h_t$. Note that $u \equiv t \bmod 4$. Then compute

$$sw_h w_q w_q w_q \equiv g^{(t,0)+(2u,v)+(6u,0)} \equiv g^{(t,v)} \equiv \pm x \bmod p \tag{51}$$

using four modular multiplications. The total number of modular multiplications with inputs depending on y is 299.

Remark C.3. If y is not a square, then the table look-up will fail. In other words, unlike some other square root algorithms, this algorithm will not produce false square roots for non-squares. In particular, if one is decompressing a compressed point that lies on the twist of the curve, this algorithm will reject the point. Hence there is little need for the twist curve to have a large prime factor.

Remark C.4. As written, finding the g_1 in the table seems to require knowing the generator g. This can be avoided as follows. Pick any quadratic non-residue r modulo p, and then put $g_1 \equiv r^q \bmod n$. This effectively defines g implicitly. To test if any element r is a quadratic non-residue, one can either compute the Legendre symbol, or just check that $r^{4q} \not\equiv 1 \bmod p$. For example, one can set $r = 23$.

C.2.4 Comparison to Other Fields

The techniques above merely describe ideas for an efficient implementation. They are insufficient to determine an absolute measure of efficiency (if such a measure is possible).

The main aim was that CM55 should more efficient than half-random primes that would arise naturally when using the CM method to optimize the den Boer reduction. For simplicity, and due lack of detail, I will just loosely estimate the modular reduction algorithm compared to the generic reduction algorithms.

Let t be the number of words used to represent field elements. For random field sizes the generic method use about t^2 word multiplications. By contrast CM55 seems to use about t word multiplications. This suggests that the CM55 modular reductions would be faster than the reductions for a random modulus of similar size.

Of course, other field sizes, such as the Solinas primes used in NIST curve P256, entirely avoid field multiplication in the reduction steps. But considering the that big integer multiplication already cost t^2 multiplications, most of the relative savings would from the modular reduction would have already been achieved by using the CM55 field.

C.3 GLV and Endomorphism

A CM55 curve has a feasible-to-compute endomorphisms because of it has a low magnitude discriminant. If an endomorphism is substantially more efficient than an scalar multiplication in the group, then perhaps the GLV method could be used.

Remark C.5. Gallant, Lambert and Vanstone [GLV01] suggest that the endomorphism should be at least as efficient as five doubling operations: but perhaps this threshold is applicable to a general purpose scalar multiplication, but could be increase for a dedicated ECDH implementation.

An endomorphism will correspond to multiplication by a complex number α, which will be a quadratic imaginary algebraic integer. In our case, we will have $\alpha \in K = \mathbb{Q}(\sqrt{-55})$. Some theorems from complex multiplication say something like this. In short Weierstrass coordinates, the complex endomorphism has the form:

$$(x, y) \mapsto (R(x), \alpha^{-1}R'(x)y) \tag{52}$$

where $R(x)$ is a rational function $R(x) = A(x)/B(x)$ and $A(x)$ and $B(x)$ are polynomials of degree $N(\alpha)$ and $N(\alpha) - 1$ where $N(\alpha)$ is the algebraic norm of α. Generally, we expect that $R(x)$ can be reduced into the field $\mathbb{F}_p(x)$.

Remark C.6. The exact form of the endomorphism, and in particular the rational function $R(x)$ is probably determinable from a deeper understanding of CM theory. Although I have not fully learned that theory I would hazard the following guess at the approximate form of $R(x)$.

Consider the point $H = (0, \sqrt{b})$ on the curve. Find a point T such that $2T = H$. This may perhaps involving using coordinates in an extension field. Next, consider an extension $\mathbb{F}_{p^3}$ which, my hand calculation suggest, may be such that $14 | \#E(F_{p^3})$. Then find a point Q of order 14. Then $R(x)$ is likely a constant times:

$$\frac{\prod_{u=0}^{13}(x - x(T + uQ))}{\prod_{u=1}^{13}(x - x(uQ))} \tag{53}$$

where $x(P)$ for any point P indicates the x-coordinate of P. (Everything above assumes short Weierstrass form of the curve). The constant can be determined by reducing α modulo p to some integer a, and scalar multiplying a random point P on the curve by this reduced value to get $a(P)$. Applying the rational function to $x(P)$ to get some value x'. Then multiply the rational function by $x(aP)/x'$.

The process above is guesswork, and may involve multiple choices for some of the values (Q, a, and H), which may require multiple trials to get the correct form of the endomorphism. The main advantage of the approach above is to bypass working in the characteristic zero fields, such as Hilbert class field of $\mathbb{Q}(\sqrt{-55})$.

We are mainly interested in the cost of computing $R(x)$, but the degree can serve as an approximate predictor of the cost. This suggests we might seek endomorphism corresponding to α of low norm, such as $\frac{1+\sqrt{-55}}{2}$ of norm 14, or $\sqrt{-55}$ of norm 55. Notice that modulo n, we have $\sqrt{-55} \equiv 2^{143} \bmod n$, which may be convenient from a GLV implementation perspective.

Recall that the basic idea of the GLV method is to halve the number of doublings, at the cost of computing the endomorphism, and also splitting the private key into two parts. Therefore, the endomorphism must be sufficiently efficiently for this to be useful.

If one is aiming for a constant-time scalar multiplication, then the GLV method might be combined with the Bernstein ladder, which is an extension of the Montgomery ladder algorithm, that computes linear combinations of two base points.

D Primality Proofs

A user should try to avoid using a discrete logarithm group vulnerable to the Pohlig–Hellman and Pollard rho attack. One way to do this is check that the group order is divisible by a large enough prime. Users can easily test this condition themselves with various primality tests. An alternative is for users to verify a pre-supplied primality proof. This section provides a short and computationally verifiable primality proof (sometimes called a certificate).

The number $n = 1 + 55 \times 2^{286}$ is a Proth prime, and its primality follows from Proth's theorem (and some computations). For convenience, we provide an additional proof, which generalizes to other numbers whose factorization of $n - 1$ is available.

Recall that one way to prove a positive integer n is prime, is to find another integer g that has order $n - 1$ modulo n, because if n is composite with $n = ab$ for $a, b > 1$ relatively prime, then the group $(\mathbb{Z}/n)^*$ has at most $n - 2$ elements, ensuring that the order of every element is the group is less than $n - 1$.

The immediately verifiable smoothness of $n - 1 = 2^{286} \times 5 \times 11$ makes proving an element to have order $n - 1$ quite easy. (For non-smooth $n - 1$, this approach requires a factoring $n - 1$ into primes, and then finding primality proofs for each prime factor.) We will check that 3 has order $n - 1$. To do this, compute

$$a \equiv 3^{2^{285}} \bmod n \tag{54}$$

Then check for each of the following:

$$a^{10} \not\equiv 1 \bmod n \tag{55}$$
$$a^{22} \not\equiv 1 \bmod n \tag{56}$$
$$a^{55} \not\equiv 1 \bmod n \tag{57}$$
$$a^{110} \equiv 1 \bmod n \tag{58}$$

The computations and checks above mean that, modulo n, the integer 3 has order dividing $n - 1 = 55 \times 2^{288}$ but not any smaller factor of $n - 1$, so it must have order $n - 1$. (Proth's theorem provides a proof that is more efficient in that one just needs to check that $a^{55} \equiv -1 \bmod n$, instead of the four other checks.)

To prove that p is prime, one can use the prime factorization of $p - 1$ given in the next section, and proceed similarly, except that one must also prove the factors in the alleged prime factorization are indeed primes.

Alternatively, one can use the well-known elliptic curve based primality proof for p. For completeness, we sketch such a proof here, although it is probably not an optimal proof. First, one finds a point (x, y) on a CM55 curve, and scalar multiplies it by $4n$ to get the point at infinity. If $4(x, y)$ is not the point at infinity, then the curve has a point of prime order n. This can only happen if one of the prime factors r of p is such that CM55 curve defined over $\mathbb{F}_r$ has a point of order n. Now $n = (p - 5)/4$, and since $r|p$, we can see that this curve over $\mathbb{F}_r$ must have size $u = hn$ for $h \in \{1, 2, 3, 4\}$, with larger sizes ruled out by the Hasse interval theorem that $|\sqrt{u} - \sqrt{r}| \leq 1$. Let $p = fr$, and divide the Hasse interval theorem by $\sqrt{p}$ to get:

$$\left|\sqrt{\frac{h(1-5/p)}{4}} - \sqrt{1/f}\right| \leq \frac{1}{\sqrt{p}}, \tag{59}$$

which implies that $f \leq 4$. Clearly, 2, 3, and 4 do not divide $p = 9 + 55 \times 2^{288}$, so we must have $f = 1$, and $p = r$ is prime.

The fact that the primality proofs for n and p are relatively short for curve parameters of the given size could perhaps be used as part of a rarity argument. However, the brevity of the proof above arises mainly because of the smoothness of $n - 1$ which was already a criterion in previous rarity argument. In other words, this property would not add any rarity.

E Miscellaneous Additional Data

The prime factorization of $p - 1$ appears to be:

$$\begin{aligned} p - 1 = 2^3 \times 3^2 \times 7 \times 53 \times 170532019843 \times 7997476149527 \times 4153194738960947 \\ \times 1807814435180469363001416834957255122186151 97 \end{aligned} \tag{60}$$

This factorization can be used to help find the generator of the multiplicative group of the finite field, but finding such a generator is not necessarily very helpful.

The twist of the curve has order $16 + 55 \times 2^{288}$ which appears to factorize as:

$$\begin{aligned} &2^4 \times 593 \times 5568161 \times 285802353517 \\ &\quad \times 18115452214802933994975811676644940383238523600177402766157160 98741 \end{aligned} \tag{61}$$

Because the natural square root algorithm for the field of size p does not yield false square roots, there is little need for the twist curve to avoid small subgroups, or to have a hard discrete logarithm problem. Anyway, if the factorization above is correct, then the twist curve provides at least 110-bit security against discrete logarithms.

References

[AM93] A. O. L. Atkin and F. Morain. *Elliptic curves and primality proving.* Mathematics of Computation, **61**:29–68, 1993.

[dB90] B. den Boer. *Diffie–Hellman is as strong as discrete log for certain primes.* In S. Goldwasser (ed.), *Advances in Cryptology — CRYPTO '88*, LNCS 403, pp. 530–539. IACR, Springer, 1990.

[Bro05] D. R. L. Brown. *Prompted user retrieval of secret entropy: The passmaze protocol.* ePrint 2005/434, IACR, 2005. http://eprint.iacr.org/2005/434.

[BB04] D. Boneh and X. Boyen. *Short signatures without random oracles.* In C. Cachin and J. Camenisch (eds.), *Advances in Cryptology — EUROCRYPT 2004*, LNCS 3027, pp. 56–73. IACR, Springer, May 2004.

[BG04] D. R. L. Brown and R. P. Gallant. *The static Diffie–Hellman problem.* ePrint 2004/306, IACR, 2004. http://eprint.iacr.org/2004/306.

[BGV07] D. R. L. Brown, R. P. Gallant and S. A. Vanstone. *Custom static Diffie–Hellman groups.* Patent 8588409, United States, 2007.

[BL96] D. Boneh and R. J. Lipton. *Algorithms for black-box fields and their application to cryptography.* In Koblitz [Kob96], pp. 283–297.

[BL14] D. J. Bernstein and T. Lange. *Rigidity.* http://safecurves.cr.yp.to/rigid.html, Sep. 2014.

[BSS99] I. Blake, G. Seroussi and N. Smart. *Elliptic Curves in Crytpography.* No. 265 in London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[Che10] J. H. Cheon. *Discrete logarithm problems with auxiliary inputs.* Journal of Cryptology, **23**(3):457–476, 2010.

[FK00] W. Ford and B. Kaliski. *Server-assisted generation of a strong secret from a password.* In *9th International Workshop on Enabling Technologies — WET ICE 2000*. IEEE Press, 2000.

[GJ03] E.-J. Goh and S. Jarecki. *A signature scheme as secure as the Diffie–Hellman problem.* In E. Biham (ed.), *Advances in Cryptology — EUROCRYPT 2003*, LNCS 2656, pp. 401–415. IACR, Springer, 2003.

[GLV00] R. P. Gallant, R. J. Lambert and S. A. Vanstone. *Improving the parallelized Pollard lambda search on anomalous binary curves.* Mathematics of Computation, **69**:1699–1705, 2000.

[GLV01] ———. *Faster point multiplication on elliptic curves with efficient endomorphisms.* In J. Kilian (ed.), *Advances in Cryptology — CRYPTO 2001*, LNCS 2139, pp. 190–200. IACR, Springer, 2001.

[HMV04] D. Hankerson, A. Menezes and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer, 2004.

[IEE00] IEEE P1363. *IEEE Std 1363-2000 Standard Specifications for Public-Key Cryptography.* Institute of Electrical and Electronics Engineers, Aug. 2000.

[JKS$^+$99] M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein and E. Teske. *Analysis of the xedni calculus attack.* Design, Codes and Cryptography, **20**:41–64, 1999.

[Knu98] D. E. Knuth. *Seminumerical Algorithms*, The Art of Computer Programming, vol. 2. Addison-Wesley, 3rd edn., 1998.

[Kob96] N. Koblitz (ed.). *Advances in Cryptology — CRYPTO '96*, LNCS 1109. IACR, Springer, 1996.

[KKM08] A. H. Koblitz, N. Koblitz and A. Menezes. *Elliptic curve cryptography: The serpentine course of a paradigm shift.* ePrint 2008/390, IACR, 2008. http://eprint.iacr.org/2008/390.

[LHA$^+$12] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung and C. Wachter. *Ron was wrong, Whit is right.* ePrint 2012/064, IACR, 2012. http://eprint.iacr.org/2012/064.

[MOV93] A. J. Menezes, T. Okamoto and S. A. Vanstone. *Reducing elliptic curve logarithms to logarithms in a finite field.* IEEE Transactions on Information Theory, **39**:1639–1646, 1993.

[MSV04] A. Muzereau, N. P. Smart and F. Vercauteren. *The equivalence between the DHP and DLP for elliptic curves used in practical applications.* LMS J. Comput. Math., **7**:50–72, Mar. 2004. http://www.lms.ac.uk.

[MW96] U. M. Maurer and S. Wolf. *Diffie–Hellman oracles.* In Koblitz [Kob96], pp. 268–282.

[NIS99] NIST. *Recommended Elliptic Curves for Federal Government Use.* National Institute for Standards and Technology, Jul. 1999.

[NIS05] ———. *FIPS 186-3: Digital Signature Standard (Change Notice).* National Institute for Standards and Technology, 2005.

[Sho97] V. Shoup. *Lower bounds for discrete logarithms and related problems.* In W. Fumy (ed.), *Advances in Cryptology — EUROCRYPT 1997*, LNCS 1233, pp. 256–266. IACR, Springer, Apr. 1997.

[Tes01] E. Teske. *On random walks for Pollard's rho method.* Mathematics of Computation, **70**:809–825, 2001.