The Equivalence Between the DHP and DLP for Elliptic Curves Used in Practical Applications, Revisited

K. Bentahar

Dept. Computer Science, University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
bentahar@cs.bris.ac.uk

Abstract. The theoretical equivalence between the DLP and DHP problems was shown by Maurer in 1994. His work was then reexamined by Muzereau *et al.* [11] for the special case of elliptic curves used in practical cryptographic applications. This paper improves on the latter and tries to get the tightest possible reduction in terms of computational equivalence, using Maurer's method.

Key words: DHP-DLP equivalence, Elliptic Curve Cryptosystems.

1 Introduction

Maurer and Wolf [6, 8, 7, 10] proved that, for every cyclic group G with prime order p, the DLP and DHP over G are equivalent if there exists an elliptic curve, called *auxiliary elliptic curve*, over $\mathbb{F}_p$ with smooth order.

Muzereau *et al.* [11] showed that such auxiliary elliptic curves are highly likely to exist for almost all elliptic curve groups. It is however remarked that it gets extremely hard to construct them as the order of G increases. Auxiliary elliptic curves with smooth orders were built and explicitly presented for most of the curves in the SECG standard, hence making Maurer's proof applicable to most of the groups used in practical elliptic curve cryptography.

The idea behind the method introduced by Maurer [6] rests on the concept of *implicit representation*: The *implicit representation* of an integer a (modulo p) is defined to be $g^a \in G$. The algorithm proceeds by doing computations in the implicit representation instead of the usual explicit representation. For example, to compute $a + b$ in implicit form, $g^a \cdot g^b$ is computed instead which costs one multiplication. For $a - b$, we compute

$g^a \cdot (g^b)^{-1}$ costing one inversion and one multiplication. To compute $a \cdot b$ in implicit form, one call to an *DH-oracle*, that computes g^{ab} given g^a and g^b, is needed. For the implicit form of a^{-1}, one uses the fact that $a^{p-1} = 1$, so $a^{p-2} = a^{-1}$, which would cost $\mathcal{O}(\lg p)$ calls to the DH-oracle. Hence, granted access to a DH-oracle for the group G, all algebraic algorithms can be converted to work in the implicit representation.

This paper builds on [11] by tightening the reduction and trying to extend the result to the remaining curves. Our goal is to show that, for the elliptic curve cryptosystems described in the various standards, the number of group operations and DH-oracle calls required to reduce the DLP to the DHP is reasonably "small." Say for example that this number is less than 2^r then, if we believe that the much more extensively studied DLP over the same group takes *at least* 2^ℓ operations to solve then an algorithm for solving the DHP, and thus breaking the DHP protocol, would require a minimum of $2^{\ell-r}$ group operations. Our target is therefore to minimise the value of r, in order to get the tightest possible security reduction.

Affine coordinates were used in [11] which requires division and hence a DH-inversion oracle was needed. This was implemented at the cost of $\mathcal{O}(\lg p)$ calls to a DH-oracle which is clearly an expensive choice as it leads to a large increase in the number of DH-oracle calls. We use *projective coordinates* instead to avoid this problem. As a further optimisation we use an optimised square root extraction algorithm.

One would also think that using *addition chains* may reduce the cost of exponentiation but it turns out that this saves very little and only adds complications. So it was decided to use traditional methods of exponentiation and concentrate on the more critical areas of the algorithm. Section 6 expands on this and justifies this decision.

A list of auxiliary elliptic curves giving almost the tightest possible reduction, using the Maurer method, is presented in Appendix C.

2   Notation and Definitions

Throughout the paper, we let G be a cyclic group with generator g and prime order $p > 3$. We begin by defining the problems DLP and DHP.

Definition 1 (DLP and DHP)

- *Given $h \in G$, the problem of computing an integer $\alpha \in [0, |G|)$ such that $g^\alpha = h$ is called the* Discrete Logarithm Problem *(DLP) with respect to g.*
- *Given two elements $g^a, g^b \in G$, we call the problem of computing g^{ab} the* Diffie-Hellman Problem *(DHP) with respect to g. (a, b are unknown)*

We also need to formalise the notion of a DH and DL oracles.

Definition 2 (DL and DH oracles)

- *A* DH-oracle *takes as input two elements $g^a, g^b \in G$ and returns g^{ab}. We write $\mathcal{DH}(g^a, g^b) = g^{ab}$.*
- *A* DL-oracle *takes as input an element $h = g^a \in G$ and returns $a \bmod |G|$. We write $\mathcal{DL}(h) = \mathcal{DL}(g^a) = a$.*

Both oracles return answers in unit time. (By definition of Oracles)

The equivalence between the two problems was theoretically established by Maurer and Wolf in the nineties [6, 8, 7, 10], but it relies on the existence of some auxiliary elliptic curves whose orders must be smooth. These auxiliary elliptic curves are not necessarily easy to build and it seems they are exceptionally hard to find in general. Hence, a more concrete treatment for the elliptic curve groups used in practice proved necessary and this was done in [11]. The paper discussed the *computational equivalence* between the DLP and DHP, and it also presented an explicit list auxiliary elliptic curves needed for the reduction.

Note that, since solving any instance of the DHP given access to a DL-oracle is trivial[1], we only concentrate on the reverse implication for the equivalence to hold: If we suppose the DHP turns out to be easy, we wish to know if this implies that the DLP is easy as well.

The base 2 logarithm will be denoted by $\lg x$ (instead of $\log_2$). We will also use $\mathfrak{M}$ and $\mathfrak{I}$ to denote multiplications and inversions in G, respectively, and $\mathfrak{DH}$ for DH-oracle calls. Formulae of the form

$$x\mathfrak{DH} + y\mathfrak{I} + z\mathfrak{M}$$

mean: Cost is x DH-oracle calls, y inversions and z multiplications in G.

---

[1] Given $g^a, g^b \in G$, we compute $a = \mathcal{DL}(g^a)$ and then compute $g^{ab} = (g^b)^a$.

3 The algorithm

Given $h \in G$, we want to find the unique α modulo p such that $h = g^{\alpha}$. We assume an elliptic curve E over $\mathbb{F}_p$ is given by the Weierstrass equation $y^2 = x^3 - 3x + b$, with smooth order given as a product of *coprime integers*

$$|E| = \prod_{j=1}^{s} q_j, \tag{1}$$

with $q_j < B$ of roughly the same size, where B is some smoothness bound. This choice of the defining equation of E saves $1\mathfrak{DH}$ while adding points on it. The point at infinity on E is denoted by $\mathcal{O}$.

To solve a DLP in G, Maurer's approach is to use a DH-oracle and solve the problem in the implicit representation over E, which is supposed to have a smooth order. So, given $h = g^{\alpha} \in G$ and the elliptic curve E, as above, we check whether $g^{y^2} = g^{\alpha^3 - 3\alpha + b}$ can be solved for y. If so then we have found a point Q on E in its implicit form, otherwise we replace α by $\alpha + d$ for some random, small, integer d and do the checking again until we get a point Q on E.

Note that, at this stage, we know Q in its implicit representation only. The idea now is to solve $Q = kP$ over E, where P is a generator of E. Upon finding the value of k, we then compute kP in the explicit representation and hence recover the value of α, from the explicit first coordinate of Q. Given that E has a smooth order, we simply use the naive Pohlig-Hellman method of first solving the problem in the subgroups of E of prime power order, and then recovering k using the Chinese Remainder Theorem (CRT). The reader is referred to Algorithm 1 for the detailed description of the algorithm.

The crucial point to note is that we have a wide choice of curves over $\mathbb{F}_p$ that have sizes distributed in the Hasse interval $[p + 1 - \sqrt{p}, p + 1 + \sqrt{p}]$. So, with a bit of luck, one hopes that one of these sizes is smooth enough and hence the corresponding auxiliary elliptic curve would make solving our DLP easy. We draw the reader's attention to the fact that this is the same reason that makes the ECM factoring method so successful.

In the description of Algorithm 1, note that for the comparison step (12) to test whether a point $(X : Y : Z)$, in projective coordinates, is equal to a point (x, y), in affine coordinates, we simply check whether $xZ^2 = X$ and $yZ^3 = Y$. In implicit representation this becomes

$$(g^{Z^2})^x = g^X \quad \text{and} \quad (g^{Z^3})^y = g^Y.$$

This use of projective coordinates gives our greatest improvement over [11]. We also make some savings by storing precomputed values and using them throughtout the algorithm. The next two subsections will describe the improvements made.

Algorithm 1 Solve a DLP in a group G given access to a DH-oracle for G.
Input: A cyclic group $G = \langle g \rangle$ of prime order p, an elliptic curve $E/\mathbb{F}_p\colon y^2 = x^3 - 3x + b$, generated by P, $|E| = \prod_{j=1}^{s} q_j$ and $h = g^\alpha \in G$
Output: $\alpha = \mathcal{DL}(h)$

    Step 1. Compute a valid implicit x-coordinate related to the DL α
1: **repeat**
2:    Choose d randomly, and set $g^x \leftarrow hg^d$ $\langle g^x \leftarrow g^{\alpha+d} \rangle$
3:    $g^z \leftarrow g^{x^3-3x+b}$.
4: **until** $g^{z^{(p-1)/2}} = g$ $\langle$ *Test quadratic-residuosity of $z \pmod p$* $\rangle$
    Step 2. Compute g^y from $g^z = g^{y^2}$:
5: Extract the square root of z in implicit representation, to obtain g^y.
    Now, $Q = (x, y)$ is a point on E known in the implicit representation only (g^x, g^y).
    Step 3. Compute $k\colon Q = kP$ in $E(\mathbb{F}_p)$: $\langle$ *Use the Pohlig-Hellman simplification* $\rangle$
6: **for** $j = 1, \ldots, s$ **do**
7:    Compute $Q_j = (g^{u_j}, g^{v_j}, g^{w_j})$, where $(u_j, v_j, w_j) = \frac{|E|}{q_j} Q$ $\langle$ *Projective coordinates* $\rangle$
8:    Set $i \leftarrow 0$, $(u, v) \leftarrow \mathcal{O}$, $P_j \leftarrow \frac{|E|}{q_j} P$ $\langle$ *Affine coordinates* $\rangle$
9:    **repeat** $\langle$ *Solve $Q_j = k_j P_j$ in the subgroup of $E(\mathbb{F}_p)$ of order q_j* $\rangle$
10:      $i \leftarrow i + 1$.
11:      $(u, v) \leftarrow (u, v) + P_j$. $\langle (u, v) \leftarrow iP_j = i\frac{|E|}{q_j} P \rangle$
12:    **until** $(g^{w_j^2})^u = g^{u_j}$ and $(g^{w_j^3})^v = g^{v_j}$ $\langle$ *Test if (g^u, g^v) equals $(g^{u_j}, g^{v_j}, g^{w_j})$* $\rangle$
13:    $k_j \leftarrow i$.
14: **end for**
    Step 4. Construct α
15: Compute $k \pmod{|E|}$ such that $\forall j \in \{1, \ldots, s\}\colon k \equiv k_j \pmod{q_j}$. $\langle$ *Use CRT* $\rangle$
16: Compute $kP = Q$ in affine coordinates.
17: Then $x \pmod p$ is the abscissa of Q, and $\alpha = x - d$.

3.1 Square root extraction

We describe the special cases in the explicit notation. This algorithm is used by Algorithm 1, in the implicit representation, to compute g^y from $g^z = g^{y^2} = g^{x^3-3x+b}$, see Algorithm 2.

Suppose a is known to be a quadratic residue modulo p and we want to compute $x \in \mathbb{F}_p$ such that $x^2 \equiv a \pmod p$. Then, besides the general Tonelli and Shanks algorithm used in [11], we also treat two special cases:

1. If $p \equiv 3 \pmod 4$ then $x \equiv a^{(p+1)/4} \pmod p$,
2. If $p \equiv 5 \pmod 8$ then do the following: Compute $s = a^{(p-5)/8}$, $u = a \cdot s$, $t = s \cdot u$. If $t = 1$ then $x = u$ otherwise $x = 2^{(p-1)/4} \cdot u$.

Treating these special cases is worthwhile since half the primes are congruent to 3 modulo 4, and half of the remaining primes are congruent to 5 modulo 8. The only remaining primes are all congruent to 1 modulo 8. We gain no advantage by using similar methods for this case, so we simply use the Tonelli-Shanks algorithm for the remaining primes, see [3, p. 32].

Algorithm 2 Implicit square roots in a group G using a DH-oracle for G.
Input: A cyclic group $G = \langle g \rangle$ of odd prime order p, and $g^z = g^{y^2} \in G$.
Output: g^y.

1: **if** $p \equiv 3 \pmod 4$ **then**
2:     $g^y \leftarrow g^{z^{(p+1)/4}}$.   ⟨*First case:* $p \equiv 3 \pmod 4$⟩
3: **else if** $p \equiv 5 \pmod 8$ **then**
4:     $g^s \leftarrow g^{z^{(p-5)/8}}, g^u \leftarrow g^{zs}, g^t \leftarrow g^{su}$.   ⟨*Second case:* $p \equiv 5 \pmod 8$⟩
5:     **if** $g^t = g$ **then**
6:        $g^y \leftarrow g^u$.
7:     **else**
8:        $g^y \leftarrow g^{u \cdot 2^{(p-1)/4}}$.
9:     **end if**
10: **else**
11:     Write $p - 1 = 2^e \cdot w$, w odd.   ⟨*Tonelli and Shanks algorithm for* $p \equiv 1 \pmod 8$⟩
12:     Set $g^s \leftarrow g$, $r \leftarrow e$, $g^y \leftarrow g^{z^{(w-1)/2}}$, $g^b \leftarrow g^{zy^2}$, $g^y \leftarrow g^{zy}$.   ⟨*Initialise*⟩
13:     **while** $g^b \not\equiv 1 \bmod p$. **do**
14:        Find the smallest $m \geq 1$ such that $g^{(b^{2^m})} \equiv 1 \bmod p$.   ⟨*Find exponent*⟩
15:        Set $g^t \leftarrow g^{(s^{2^{r-m-1}})}$, $g^s \leftarrow g^{t^2}$, $r \leftarrow m$, $g^y \leftarrow g^{yt}$, $g^b \leftarrow g^{bs}$.   ⟨*Reduction*⟩
16:     **end while**
17: **end if**

3.2 Explicit and implicit point multiplication

As already remarked, we use the projective coordinate system in step 3 of Algorithm 1 instead of the affine coordinate system. The formulae for addition and doubling[2] in the implicit representation easily follow from their standard explicit counterparts, see Appendix B. The cost of each operation is given in the following table.

---

[2] Doubling is the operation of adding a point to itself: $2P = P + P$.

	Doubling		Addition	
	Explicit	Implicit	Explicit	Implicit
$\mathfrak{D}\mathfrak{H}$		8		16
$\mathfrak{J}$		4		5
$\mathfrak{M}$	8	14	16	$\frac{3}{2}\lg p + \frac{13}{2}$

For the affine coordinates, note that we only need the explicit case (In the j-loop). The costs are:

$$1\mathfrak{J} + 4\mathfrak{M} \text{ for doubling and } 1\mathfrak{J} + 3\mathfrak{M} \text{ for addition.}$$

Since we will need to compute kP for different values of k but a fixed P, pre-computing the values $2^1 P, 2^2 P, \ldots, 2^{\lfloor \lg k \rfloor} P$ will save us some computation. Then, using the right-to-left binary method, we expect only $\frac{1}{2}\lg k$ elliptic curve additions. We now summarise the costs of exponentiation.

Implicit exponentiation in Projective coordinates: The cost of the precomputation is about

$$(8\mathfrak{D}\mathfrak{H} + 4\mathfrak{J} + 14\mathfrak{M})\lg k \tag{2}$$

and then each exponentiation would cost about

$$\left(8\mathfrak{D}\mathfrak{H} + \frac{5}{2}\mathfrak{J} + \frac{1}{4}(3\lg p + 13)\mathfrak{M}\right)\lg k. \tag{3}$$

Explicit exponentiation in Affine coordinates: The precomputation cost is

$$(1\mathfrak{J} + 4\mathfrak{M})\lg k \tag{4}$$

and then each exponentiation would cost

$$\frac{1}{2}(1\mathfrak{J} + 3\mathfrak{M})\lg k. \tag{5}$$

3.3 Complexity of the algorithm

The complexity analysis of Algorithm 1, presented in Appendix A, yields the following theorem

Theorem 1. *Let G be a cyclic finite group of prime order p. Assume an elliptic curve E over $\mathbb{F}_p$ has been found, whose B-smooth order is*

$$|E| = \prod_{j=1}^{s} q_j,$$

where q_j are not necessarily prime but are coprime of roughly the same size. Then, solving a given instance of the DLP in G requires on average about

$$\mathcal{O}\left(\frac{\log^2 p}{\log B}\right) \mathfrak{DH} + \mathcal{O}\left(\frac{B \log^2 p}{\log B}\right) \mathfrak{M}.$$

For comparison, we quote below the asymptotic costs obtained by [11]

$$\mathcal{O}\left(\frac{\log^3 p}{\log B}\right) \mathfrak{DH} + \mathcal{O}\left(\frac{B \log^2 p}{\log B}\right) \mathfrak{M}.$$

While the number of multiplications has remained the same, the number of DH-oracle calls has now become quadratic in the size of the group G instead of cubic.

Note that, in order to get a lower bound on the cost of solving a DHP instance, we no longer require the auxiliary elliptic curves' orders to be smooth. This is because as long as we assume that the DLP is an exponentially hard problem then we do not mind if the reduction from the DHP to the DLP is exponential too. This remark will allow us to choose $s = 3$ later, and then the task will be to find smooth elliptic curves whose orders are product of three coprime numbers. This is a significant relaxation of the smoothness condition.

4 Implications on the security of the DHP

The implications of this reduction on the security of the DLP was treated in [11]. We only comment on its implications on the security of the DHP, as it is here where the work done in this paper matters most.

Let C_{DLP}, C_{DHP} denote the costs of solving the DLP and DHP on an elliptic curve of size p, respectively. By Maurer's reduction, we have $C_{DLP} = N_{\mathfrak{DH}} \cdot C_{DHP} + N_{\mathfrak{M}}$, where $N_{\mathfrak{DH}}, N_{\mathfrak{M}}$ are respectively the number of calls to the DH-oracle and number of multiplications in G. Hence, for $N_{\mathfrak{M}} \ll C_{DLP}$ we get

$$C_{DHP} = \frac{C_{DLP} - N_{\mathfrak{M}}}{N_{\mathfrak{DH}}} \sim \frac{C_{DLP}}{N_{\mathfrak{DH}}}.$$

Since solving the DLP on an elliptic curve E is believed to take at least $\sqrt{|E|}$ steps [2], in general, then setting

$$T_{DH} = \frac{\sqrt{|E|}}{N_{\mathfrak{D}\mathfrak{H}}},$$

we see that T_{DH} gives us a lower bound on the number of operations required to break the DHP, assuming $N_{\mathfrak{M}} \ll C_{DLP}$. Hence, it is the value of T_{DH} that gives the exact security result, given the best auxiliary elliptic curves we found.

The tightness of the security reduction is controlled by two values. The first being the number of field multiplications $N_{\mathfrak{M}}$, and second and most important is the value T_{DH} for the reason put forth earlier.

Tables 1 and 2 give the logarithms of these key values, $\lg N_{\mathfrak{M}}$ and $\lg N_{\mathfrak{D}\mathfrak{H}}$, for the curves in the SECG standard [16]. They also give $\lg \sqrt{|E|}$, the logarithm of the (believed) minimum cost of solving an instance of the DLP. The column headed adv gives the number of security bits gained on the previous results from [11]. The last rows of the tables are detached to indicate that the values are theoretical and that no auxiliary elliptic curves could be generated for them, mainly due to the sheer size of the numbers that needed to be factored.

Table 1. Summary of results for curves of large prime characteristic

secp curve	$\lg \sqrt{|E|}$	$\lg N_{\mathfrak{M}}$	$\lg N_{\mathfrak{D}\mathfrak{H}}$	$\lg T_{DH}$	adv
secp112r1	55.9	46.3	11.4	44.4	6.4
secp112r2	54.9	45.6	11.4	43.5	5.5
secp128r1	64.0	51.9	11.6	52.4	6.4
secp128r2	63.0	51.2	11.6	51.4	5.4
secp160k1	80.0	62.9	12.0	68.0	8.0
secp160r1	80.0	62.9	12.0	68.0	6.0
secp160r2	80.0	62.9	12.0	68.0	7.0
secp192k1	96.0	73.8	12.2	83.8	7.8
secp192r1	96.0	73.8	12.2	83.8	6.8
secp224k1	112.0	84.7	12.4	99.6	6.6
secp224r1	112.0	84.7	12.4	99.6	7.6
secp256k1	128.0	95.5	12.6	115.4	7.4
secp256r1	128.0	95.5	12.6	115.4	7.4
secp384r1	192.0	138.8	13.2	178.8	8.8
secp521r1	260.5	184.9	13.7	246.8	-

Table 2. Summary of results for curves of even characteristic

sect curve	$\lg\sqrt{|E|}$	$\lg N_{\mathfrak{M}}$	$\lg N_{\mathfrak{DH}}$	$\lg T_{DH}$	adv
sect113r1	56.0	46.4	11.4	44.6	6.6
sect113r2	56.0	46.4	11.4	44.6	6.6
sect131r1	65.0	52.6	11.7	53.3	6.3
sect131r2	65.0	52.6	11.7	53.3	6.3
sect163k1	81.0	63.5	12.0	69.0	7.0
sect163r1	81.0	63.5	12.0	69.0	7.0
sect163r2	81.0	63.5	12.0	69.0	7.0
sect193r1	96.0	73.8	12.2	83.8	6.8
sect193r2	96.0	73.8	12.2	83.8	6.8
sect233k1	115.5	87.0	12.5	103.0	7.0
sect233r1	116.0	87.4	12.5	103.5	7.5
sect239k1	118.5	89.1	12.5	106.0	8.0
sect283k1	140.5	104.0	12.8	127.7	8.7
sect283r1	141.0	104.3	12.8	128.2	7.2
sect409k1	203.5	146.5	13.3	190.2	8.2
sect409r1	204.0	146.9	13.3	190.7	-
sect571k1	284.5	201.0	13.8	270.7	-
sect571r1	285.0	201.3	13.8	271.2	-

Now, given our estimates for the number of group operations and DH-oracle calls, we see that the smallest s for which $N_{\mathfrak{M}} \ll \sqrt{|E|}$ is $s = 3$. The reduction cost is then (see Appendix A for general s)

$$\left(\frac{149}{6}\lg p + \frac{55}{8}\right)\mathfrak{DH} + \left((\frac{3}{2}\lg p + \frac{13}{2})(3p^{1/3}) + (\frac{3}{2}\lg p + \frac{511}{4})\lg p\right)\mathfrak{M}.$$

As an illustration of the advantage gained over the previous results presented in [11], we consider the security of DHP for `secp256r1`: The DLP on this curve requires about 2^{128} computational steps, employing the currently known methods. Using our auxiliary elliptic curve, we deduce that the DHP cannot be solved in less than $2^{115.3}$ computational steps, as opposed to 2^{108} from the previous paper. That is a gain factor of about $2^{7.3}$ over the previously reported value in [11], see Table 1.

Since an amount of computation of about $2^{115.3} \approx 5 \cdot 10^{34}$ group operations is infeasible with today's computational power, one can draw the conclusion that a *secure implementation* of a protocol whose security depends on the intractability of the DHP on the curve `secp256r1` can safely be used, provided the DLP is really of the conjectured complexity.

Note that the SECG standard [16] includes all the curves in the NIST [12] and the most used ones in the ANSI [1] standards, covering the most commonly used elliptic curves in practice.

5 Building the auxiliary elliptic curves

By the argument presented in the previous section, we need to construct elliptic curves whose order is a product of three coprime numbers of roughly the same size. That is $q_i \approx p^{1/3}$. Muzereau *et el.* [11] used the Complex Multiplication (CM) technique to build auxiliary elliptic curves with smooth orders but this does not perform very well as p gets larger, due to the prohibitive precision then needed for the calculations. In our case, it proved to be computationally more efficient to generate random elliptic curves and then test if their sizes are of the required form.

Let us estimate the probability that a number in a large interval centred around p is a product of three co-primes of roughly the same size.

Given three randomly chosen (positive) integers, we first want to compute the probability that they are *pairwise* coprime. Let p be prime. The probability that p divides two of these integers but not the third is $3/p^2 \cdot (1 - 1/p)$ and the probability that p divides all of them at once is $1/p^3$. So, the probability that p is not a common divisor of any two of these integers is

$$1 - \frac{3}{p^2}\left(1 - \frac{1}{p}\right) - \frac{1}{p^3} = 1 - \frac{3}{p^2} + \frac{2}{p^3}.$$

Hence, the probability that three randomly chosen integers are pairwise coprime is

$$\prod_{p \text{ prime}} \left(1 - \frac{3}{p^2} + \frac{2}{p^3}\right) \approx 0.2867474.$$

The infinite product is clearly convergent but a closed form of its value could not be obtained by the author. The numerical approximation 0.2867474 was obtained using `PARI`, [13].

For a large interval (m, n), the product should be taken only for $p \leq m-n$. Now, since $1-3/p^2+2/p^3$ is positive, strictly increasing approaching 1 from below, we deduce that the above estimate is a *lower bound* to the actual probability we want.

In practice, for large p and corresponding Hasse intervals, the above value proved to be a good estimate and it matched nicely with a Monte Carlo simulation to estimate this probability over large Hasse intervals.

For most cryptographic groups G from the SECG standard, auxiliary elliptic curve E of the form $y^2 = x^3 - 3x + b$ were successfully generated by finding a suitable value of b. Appendix C specifies these values together with the (prime) size of the group G, which is the characteristic of the prime finite-field over which E is defined. The size of the elliptic curve group $|E|$ is given as a product of three coprime numbers of roughly equal size.

When trying to generate the auxiliary elliptic curves, the main difficulty was to actually factor $|E|$. For large $|G|$, factorisation fails most of the time and another random value of b is tried without any success. This is the main reason for failing to produce the necessary data for the three curves `secp521r1`, `sect571r1` and `sect571k1`. However, two missing auxiliary elliptic curves from [11], viz. `secp224k1` and `sect409r1` were successfully found. While first appears to have been just forgotten, the second was due to the difficulty of generating the auxiliary elliptic curves using the CM method.

6   Can we do better using Maurer's approach

Here, it is argued that not much improvement can be made using Maurer's reduction, as described in Algorithm 1.

Just computing g^{x^3} and (twice) checking the quadratic residuosity of $g^{x^3 - 3x + b}$ will cost at least

$$(2 + 2 \times \lg(p/2))\mathfrak{DH}.$$

For $s = 3$ we find that the ratio of the estimated cost of this paper to this bound is

$$\frac{\frac{149}{6}\lg p + \frac{55}{8}}{2\lg p} \sim \frac{149}{12} \approx 2^{3.6}.$$

Step 2 is not independent from the first so its cost can be reduced even further, but the third step does not seem to have any corelation with the previous steps. If we say that step 3 costs at least one exponentiation, to compute one of the $(|E|/q_j)Q$, then the ratio drops to

$$\frac{\frac{149}{6}\lg p + \frac{55}{8}}{(2 + 2/3)\lg p} \sim \frac{149}{16} \approx 2^{3.2}.$$

Hence, it turns out that about 3 bits of security is all that can be hoped for above the current work.

7   Conclusion

Assuming the DLP is an exponentially hard problem, we have shown that the Maurer-Wolf reduction with naive search yields a concrete security assurance for the elliptic curves recommended by the current standards, for which we could generate the auxiliary elliptic curves.

We have found two new auxiliary elliptic curves, missing from [11], viz. `secp224k1` and `sect409r1`. It remains open to find auxiliary elliptic curves for the curves `secp521r1`, `sect571r1` and `sect571k1`. These will have sizes larger than 500 bits, which presents the current factoring algorithms with a big challenge.

Acknowledgement

References

1. ANSI. *X9.62 – Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*. 1999.
2. I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic curves in cryptography*, Cambridge University Press, 1999.
3. H. Cohen. *A Course In Computational Algebraic Number Theory*. Springer-Verlag, GTM 138, 1993.
4. D. Hankerson, A. Menezes, S. Vanstone. *Guide to elliptic curve cryptography* Springer-Verlag, 2003.
5. D. E. Knuth. *The art of computer programming, vol. 2*. Addison Wesley Longman, 1998.
6. U. M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology – CRYPTO '94*, Springer LNCS 839, 271-281, 1994.
7. U. M. Maurer and S. Wolf. On the difficulty of breaking the DH protocol. *Technical Report #24*, Department of Computer Science, ETH Zurich, 1996.
8. U. M. Maurer and S. Wolf. Diffie-Hellman Oracles. In *Advances in Cryptology - CRYPTO '96*, Springer LNCS 1109, 268-282, 1996.
9. U. M. Maurer and S. Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, **28**, 1689–1721, 1999.
10. U. M. Maurer and S. Wolf. The Diffie-Hellman protocol. *Designs, Codes, and Cryptography*, **19**, 147–171, 2000.
11. A. Muzereau, N.P. Smart and F. Vrecauteren *The equivalence between the DHP and DLP for elliptic curves used in practical applications*, LMS J. Comput. Math. 7(2004) 50-72, 2004.

12. NIST. *FIPS 186.2 Digital Signature Standard (DSS)*. 2000.
13. PARI/GP, version `2.1.3`, Bordeaux, 2000, `http://pari.math.u-bordeaux.fr/`.
14. H.-G. Rück. A note on elliptic curves over finite fields. *Math. Comp.*, **49**, 301–304, 1987.
15. R. Schoof. Elliptic curves over finite fields and the computation of square roots mod *p*. *Math. Comp.*, **44**, 483–494, 1985.
16. SECG. *SEC2 : Recommended Elliptic Curve Domain Parameters.* See `http://www.secg.org/`, 2000.
17. W.C. Waterhouse. Abelian varieties over finite fields. *Ann. Sci. École Norm. Sup.*, **2**, 521–560, 1969.

A   Complexity analysis of Algorithm 1

To simplify this task, each step of Algorithm 1 will be studied separately and then the results will be added up to obtain the total average cost.

Step 1:

We first precompute g^{2^i} for $i = 1, \dots, \lfloor \lg p \rfloor$. This will allow us to compute any power g^k with an average cost of $\frac{1}{2} \lg k \mathfrak{M}$, using the double-and-add algorithm of exponentiation. The precomputation requires $\lfloor \lg p \rfloor$ squarings, which costs

$$\lg p \, \mathfrak{M}.$$

Without loss of generality, we set $d = 0$ at the start of this step. Then, evaluating $g^z \leftarrow g^{x^3 - 3x + b} = g^{x^3} \cdot ((g^x)^3)^{-1} \cdot g^b$ requires

$$2\mathfrak{DH} + +1\mathfrak{I} + (4 + \frac{1}{2} \lg b)\mathfrak{M}.$$

Note that

$$g^{(x+d)^3 - 3(x+d) + b} = g^{x^3 - 3x + b} \cdot (g^{x^2})^{3d} \cdot (g^x)^{3d^2} \cdot g^{d^3 - 3d}.$$

So for a second evaluation, we only need an extra

$$\left(3 + \frac{3}{2} \lg(3d) + \frac{3}{2} \lg(3d^2) + \frac{1}{2} \lg(d^3 - 3d)\right)\mathfrak{M} \sim (3 + 3\lg 3 + 6\lg d)\mathfrak{M}.$$

For the quadratic residuosity check we need to compute $g^{z^{(p-1)/2}}$. First pre-compute $g^{z^{2^i}}$ for $i = 1, \dots, \lfloor \lg \frac{p}{2} \rfloor$, then the total cost is

$$\left(\lg \frac{p}{2} + \frac{1}{2} \lg \frac{p-1}{2}\right)\mathfrak{DH} \sim \left(\frac{3}{2} \lg p - \frac{3}{2}\right)\mathfrak{DH}.$$

Now, let ν be the number of iterations for step 1. Since $\mathbb{F}_p$ has $(p-1)/2$ quadratic non-residues, the probability for having $\nu = k$ iterations is

$$\Pr[\nu = k] = \left(\frac{p-1}{2p}\right)^{k-1} \cdot \frac{p+1}{2p}.$$

Hence, the expected number $\bar{\nu}$ of iterations for step 1 is

$$\bar{\nu} = \sum_{k=1}^{\infty} k \cdot \Pr[\nu = k] = \frac{p+1}{2p} \sum_{k=1}^{\infty} k \left(\frac{p-1}{2p}\right)^{k-1} = \frac{2p}{p+1} \approx 2.$$

Thus the total average cost of this first step is $\lg p \mathfrak{M} + [2 \mathfrak{D}\mathfrak{H} + 1\mathfrak{J} + (4 + \frac{1}{2}\lg b)\mathfrak{M}] + [(3 + 3\lg 3 + 6\lg d)\mathfrak{M}] + 2 \times (\frac{3}{2}\lg p - \frac{3}{2})\mathfrak{D}\mathfrak{H}$. That is

$$(3\lg p - 1)\mathfrak{D}\mathfrak{H} + 1\mathfrak{J} + (\lg p + \frac{1}{2}\lg b + 6\lg d + 7 + 3\lg 3)\mathfrak{M}. \qquad (6)$$

Step 2: Following Algorithm 2, we treat three cases:

1. If $p \equiv 3 \pmod 4$ then, using the precomputations from the previous step, we can compute $g^{z^{(p+1)/4}}$ in an average of

$$\frac{1}{2}\lg\frac{p+1}{4}\mathfrak{D}\mathfrak{H} \sim (\frac{1}{2}\lg p - 1)\mathfrak{D}\mathfrak{H}.$$

2. If $p \equiv 5 \pmod 8$ then the computation of $g^{z^{(p-5)/8}}, g^{zs}$ and g^{su} costs $(2 + \frac{1}{2}\lg\frac{p-5}{8})\mathfrak{D}\mathfrak{H} \sim (\frac{1}{2}\lg p + \frac{1}{2})\mathfrak{D}\mathfrak{H}$ on average.
   If $t = 1$ then no further computation is needed and the total cost is $(\frac{1}{2}\lg p + \frac{1}{2})\mathfrak{D}\mathfrak{H}$. Otherwise, $t \neq 1$ and then computing

$$g^{u \cdot 2^{(p-1)/4}} = \mathcal{DH}(g^u, g^{(2^{(p-1)/4})} \bmod p)$$

   will cost an extra $1\mathfrak{D}\mathfrak{H} + (\frac{3}{2}\lg\frac{p-1}{4} + \frac{1}{2}\lg p)\mathfrak{M}$.
   Since t behaves like a random variable, the average cost for this case is then
$$(\frac{1}{2}\lg p + \frac{1}{2})\mathfrak{D}\mathfrak{H} + \frac{1}{2}\Big(1\mathfrak{D}\mathfrak{H} + (2\lg p - 3)\mathfrak{M}\Big).$$

3. Otherwise, we use the general (implicit) Tonelli and Shanks algorithm. We first write $p - 1 = 2^e \cdot w$, where w is odd.
   The initialisation step requires roughly $(\frac{1}{2}\lg\frac{w-1}{2} + 2)\mathfrak{D}\mathfrak{H}$. Finding the exponent and reducing it requires $(r+2)\mathfrak{D}\mathfrak{H}$ per iteration, and at most e iterations are expected. Since $r \leq e$, we will need $e \cdot (r+2) \leq e \cdot (e+2)$ calls to the DH-oracle. Hence, the total number of the DH-oracle calls is about
$$\left(\frac{1}{2}\lg\frac{w-1}{2} + 2 + (e+2)e\right)\mathfrak{D}\mathfrak{H}.$$
   Since p is odd, we can easily see that the expected value of e is
$$\sum_{k=1}^{\infty} k \cdot \Pr[e = k] = \sum_{k=1}^{\infty} k \cdot (1/2)^k = 2.$$

15

Bearing this in mind, we get $w = p/2^e = p/4$ and the total cost is then estimated to be

$$\left(\frac{1}{2}\lg p + \frac{17}{2}\right)\mathfrak{D}\mathfrak{H}.$$

Note: When concluding, we will use the weighted average of the costs above, which is

$$\left(\frac{1}{2}\lg p + \frac{15}{8}\right)\mathfrak{D}\mathfrak{H} + \frac{1}{8}(2\lg p - 3)\mathfrak{M}. \tag{7}$$

Step 3: Before entering the j-loop, we first pre-compute $2^i Q$ for $i = 1, \ldots, \lfloor \lg |E|^{1-1/s} \rfloor$. This is enough since q_j are of roughly the same size, so $q_j \approx |E|^{1/s}$ and then $\frac{|E|}{q_j} \approx |E|^{1-1/s}$.

Using equation (2), the cost of precomputation is found to be about

$$(8\mathfrak{D}\mathfrak{H} + 4\mathfrak{J} + 14\mathfrak{M})\left(1 - \frac{1}{s}\right)\lg |E|.$$

We also pre-compute $2^i P$ for $i = 1, \ldots, \lfloor \lg |E| \rfloor$ in affine coordinates[3]. According to equation (4), this costs about

$$(1\mathfrak{J} + 4\mathfrak{M})\lg |E|.$$

Now, let j be fixed (We want to analyse the cost of one j-loop). The cost for computing $Q_j = (g^{u_j}, g^{v_j}, g^{w_j})$ such that $(u_j, v_j, w_j) = \frac{|E|}{q_j}Q$, given by equation (3), is about

$$\left(8\mathfrak{D}\mathfrak{H} + \frac{5}{2}\mathfrak{J} + \frac{1}{4}(3\lg p + 13)\mathfrak{M}\right)\gamma_j,$$

where we have set $\gamma_j = \lg(|E|/q_j)$. For the evaluation of $P_j = \frac{|E|}{q_j}P$, in affine coordinates, equation (5) gives

$$\left(\frac{1}{2}\mathfrak{J} + \frac{3}{2}\mathfrak{M}\right)\gamma_j.$$

For the i-loop, we note that $g^{w_j^2}$ and $g^{w_j^3}$ need to be computed only once for each j-loop, which costs $2\mathfrak{D}\mathfrak{H}$.

Now fix i. Computing $iR = (i-1)R + R$, in affine coordinates, can be achieved with one elliptic curve addition costing $1\mathfrak{J} + 3\mathfrak{M}$, since $(i-1)R$ has been computed and $1R = R$ is trivial.

---

[3] We need i up to $\lg |E|$ as we will use these precomputed values in step 4 too.

The cost of comparison is about $2 \times \frac{3}{2} \lg p \mathfrak{M} = 3 \lg p \mathfrak{M}$.

On average there will be $q_j/2$ i-loops for each j-loop, and therefore the average cost of the i-loop is

$$\frac{q_j}{2} \left(1\mathfrak{I} + 3(\lg p + 1)\mathfrak{M}\right).$$

Hence, the cost per one j-loop is

$$(8\gamma_j + 2)\mathfrak{D}\mathfrak{H} + (\frac{1}{2}q_j + 3\gamma_j)\mathfrak{I} + \left(\frac{3}{2}(\lg p + 1)q_j + \frac{1}{4}(3\lg p + 19)\gamma_j\right)\mathfrak{M}.$$

Noting that

$$\sum_{j=1}^{s} \gamma_j = \sum_{j=1}^{s} \lg \frac{|E|}{q_j} = (s-1)\lg|E|,$$

we find that the total cost for step 3, without the precomputation costs, is on average

$$(8(s-1)\lg|E| + 2s)\mathfrak{D}\mathfrak{H} + \left(\frac{1}{2}\sum_{i=1}^{s} q_j + 3(s-1)\lg|E|\right)\mathfrak{I} +$$

$$+ \left(\frac{3}{2}(\lg p + 1)\sum_{i=1}^{s} q_j + \frac{1}{4}(3\lg p + 19)(s-1)\lg|E|\right)\mathfrak{M}.$$

Adding the precomputation costs, we finally get the total cost of step 3

$$(8(s-1/s)\lg|E| + 2s)\mathfrak{D}\mathfrak{H} + \left(\frac{1}{2}\sum_{i=1}^{s} q_j + (3s + 2 - \frac{4}{s})\lg|E|\right)\mathfrak{I} +$$

$$\left(\frac{3}{2}(\lg p + 1)\sum_{i=1}^{s} q_j + (\frac{1}{4}(3\lg p + 19)(s-1) + 18 - \frac{14}{s})\lg|E|\right)\mathfrak{M}. \tag{8}$$

Step 4: We use the Chinese Remainder Theorem to reconstruct $k \bmod |E|$ from $k \equiv k_j \pmod{q_j}$, $j = 1, \ldots, s$. We compute

$$k = \sum_{j=1}^{s} k_j \cdot \frac{|E|}{q_j} \cdot \hat{q}_j \pmod{|E|},$$

where $\hat{q}_j = \left(\frac{|E|}{q_j}\right)^{-1} \bmod q_j$. This requires $s\mathfrak{I} + 2s\mathfrak{M}$ operations. Note that inversions are computed in $\mathbb{F}_{q_1}, \ldots, \mathbb{F}_{q_s}$.

For computing kP, in affine coordinates, we use the previously pre-computed values of $2^i P$. So this exponentiation would cost only $(1\mathfrak{I} +$

$3\mathfrak{M})\frac{1}{2}\lg k$. Taking $k \bmod |E|$ to be $\frac{|E|}{2}$ on average, we find the average cost of step 4 to be

$$\frac{1}{2}(\lg|E| - 1)\mathfrak{I} + \frac{3}{2}(\lg|E| - 1)\mathfrak{M}. \tag{9}$$

Conclusion : We conclude that the total cost for Algorithm 1 is

$$\left(8\left(s - \frac{1}{s}\right)\lg|E| + \frac{7}{2}\lg p + 2s + \frac{7}{8}\right)\mathfrak{D}\mathfrak{H}$$

$$+ \left(\frac{1}{2}\sum_{i=1}^{s} q_j + (3s + \frac{5}{2} - \frac{4}{s})\lg|E| + \frac{1}{2}\right)\mathfrak{I}$$

$$+ \left(\begin{array}{c}\frac{3}{2}(\lg p + 1)\sum_{i=1}^{s} q_j + \left(\frac{1}{4}(3\lg p + 19)(s-1) + \frac{39}{2} - \frac{14}{s}\right)\lg|E| + \\ + \frac{5}{4}\lg p + \frac{1}{2}\lg b + 6\lg d + 3\lg 3 + \frac{41}{8}\end{array}\right)\mathfrak{M}.$$

Neglecting small terms and making the approximation[4] $|E| \approx p$ and $b \approx p/2$, the average cost of Algorithm 1 is then found to be

$$\left\{\left(8s - \frac{8}{s} + \frac{7}{2}\right)\lg p + 2s + \frac{7}{8}\right\}\mathfrak{D}\mathfrak{H} + \left(\frac{1}{2}\sum_{i=1}^{s} q_j + (3s + \frac{5}{2} - \frac{4}{s})\lg p\right)\mathfrak{I}+$$

$$+ \left\{\frac{3}{2}(\lg p + 1)\sum_{i=1}^{s} q_j + \left(\frac{1}{4}(3\lg p + 19)(s-1) + \frac{85}{4} - \frac{14}{s}\right)\lg p\right\}\mathfrak{M}.$$

Note that if we take q_j to be of roughly the same size and fix B to be of this size then

$$s \approx \frac{\log|E|}{\log B} \approx \frac{\log p}{\log B}$$

and then

$$\sum_{j=1}^{s} q_j \approx \sum_{j=1}^{s} B = sB \approx \frac{B\log p}{\log B} = \frac{B\lg p}{\lg B}.$$

In practice, the cost of an inversion is at most $10\mathfrak{M}$, see [2, p. 37]. Using this fact we have now established Theorem 1, stated on page 8.

---

[4] $|E| = p+1-t$ where $t \in [-\sqrt{p}, \sqrt{p}]$ is the Frobenius trace, so $E = p(1+(1-t)/p) \approx p$, $b \approx p/2$ is the average value of b, and d is small.

B Explicit and implicit point multiplication

We use projective coordinates in step 3 of Algorithm 1. Below, we give the formulae for implicit point multiplications on elliptic curves defined over a field of prime characteristic greater than 3. The formulae for the explicit case and their costs can be found in [2].

Doubling a point on an elliptic curve Let $P = (X : Y : Z)$ and $Q = 2P = (X' : Y' : Z')$. (Recall that we chose $a = -3$)

Explicit doubling costs $8\mathfrak{M}$, and the formula for implicit doubling is

$$
\begin{aligned}
g^{Y^2} &= \mathcal{DH}(g^Y, g^Y) \\
g^{Z^2} &= \mathcal{DH}(g^Z, g^Z) \\
g^{Y^4} &= \mathcal{DH}(g^{Y^2}, g^{Y^2}) \\
g^{\lambda_1} &= \{\mathcal{DH}(g^X \cdot (g^{Z^2})^{-1}, g^X \cdot g^{Z^2})\}^3 && 4m, 1i \\
g^{\lambda_2} &= (\mathcal{DH}(g^X, g^{Y^2}))^4 && 2m \\
g^{\lambda_3} &= (g^{Y^4})^8 && 3m \\
g^{X'} &= \mathcal{DH}(g^{\lambda_1}, g^{\lambda_1}) \cdot (g^{\lambda_2})^{-2} && 2m, 1i \\
g^{Y'} &= \mathcal{DH}(g^{\lambda_1}, g^{\lambda_2} \cdot (g^{X'})^{-1}) \cdot (g^{\lambda_3})^{-1} && 2m, 2i \\
g^{Z'} &= \mathcal{DH}((g^Y)^2, g^Z) && 1m
\end{aligned}
$$

The cost of implicit doubling is therefore $8\mathcal{DH} + 4\mathfrak{I} + 14\mathfrak{M}$.

Adding two distinct points on an elliptic curve Let $P = (X_1 : Y_1 : Z_1)$, $Q = (X_2 : Y_2 : Z_2)$ and $R = P + Q = (X_3 : Y_3 : Z_3)$.

Explicit addition costs $16\mathfrak{M}$, and implicit addition is done as follows

$$
\begin{aligned}
g^{Z_1^2} &= \mathcal{DH}(g^{Z_1}, g^{Z_1}) && \\
g^{Z_1^3} &= \mathcal{DH}(g^{Z_1^2}, g^{Z_1}) && \\
g^{Z_2^2} &= \mathcal{DH}(g^{Z_2}, g^{Z_2}) && \\
g^{Z_2^3} &= \mathcal{DH}(g^{Z_2^2}, g^{Z_2}) && \\
g^{\lambda_1} &= \mathcal{DH}(g^{X_1}, g^{Z_2^2}) && \\
g^{\lambda_2} &= \mathcal{DH}(g^{X_2}, g^{Z_1^2}) && \\
g^{\lambda_3} &= g^{\lambda_1} \cdot (g^{\lambda_2})^{-1} && 1m, 1i \\
g^{\lambda_4} &= \mathcal{DH}(g^{Y_1}, g^{Z_2^3}) && \\
g^{\lambda_5} &= \mathcal{DH}(g^{Y_2}, g^{Z_1^3}) && \\
g^{\lambda_6} &= g^{\lambda_4} \cdot (g^{\lambda_5})^{-1} && 1m, 1i \\
g^{\lambda_7} &= g^{\lambda_1} \cdot g^{\lambda_2} && 1m \\
g^{\lambda_8} &= g^{\lambda_4} \cdot g^{\lambda_5} && 1m \\
g^{\lambda_3^2} &= \mathcal{DH}(g^{\lambda_3}, g^{\lambda_3}) && \\
g^{\lambda_3^3} &= \mathcal{DH}(g^{\lambda_3^2}, g^{\lambda_3}) && \\
g^{\lambda_7 \lambda_3^2} &= \mathcal{DH}(g^{\lambda_7}, g^{\lambda_3^2}) && \\
g^{X_3} &= \mathcal{DH}(g^{\lambda_6}, g^{\lambda_6}) \cdot (g^{\lambda_7 \lambda_3^2})^{-1} && 1m, 1i \\
g^{\lambda_9} &= g^{\lambda_7 \lambda_3^2} \cdot (g^{X_3})^{-2} && 2m, 1i \\
g^{Y_3} &= \{\mathcal{DH}(g^{\lambda_9}, g^{\lambda_6}) \cdot (\mathcal{DH}(g^{\lambda_8}, g^{\lambda_3^3}))^{-1}\}^{1/2} && 1m, 1i, 1sqrt \\
g^{Z_3} &= \mathcal{DH}(\mathcal{DH}(g^{Z_1}, g^{Z_2}), g^{\lambda_3}) &&
\end{aligned}
$$

The cost of implicit addition is therefore $16\mathcal{DH} + 5\mathfrak{I} + 8\mathfrak{M}$ plus one explicit square root extraction (i.e. in G).

For the square root extraction, we can either use general purpose algorithms or simply raise to the power $(p+1)/2 \equiv 2^{-1} \pmod{p}$ in G, which costs about $\frac{3}{2}(\lg p - 1)\mathfrak{M}$. Employing the latter approach, the cost of addition becomes

$$16\mathcal{DH} + 5\mathfrak{I} + (\frac{3}{2}\lg p + \frac{13}{2})\mathfrak{M}.$$

These results are summarised in the table on page 6.

C   The auxiliary elliptic curve groups

C.1   Elliptic curve domain parameters over $\mathbb{F}_p$

secp112r1
$|G| = 4451685225093714776491891542548933$
$b = 2281028298640880380471050241629229$

$|E| = 161721374756 \cdot 170510910317 \cdot 161437658771$

secp112r2

$|G| = 1112921306273428674967732714786891$

$b = 2061835755930385486536405010948 54$

$|E| = 105310592296 \cdot 103373879227 \cdot 102230759539$

secp128r1

$|G| = 340282366762482138443322565580356624661$

$b = 2963822166721051279484480956810440 76642$

$|E| = 7551279841752 \cdot 6513487018025 \cdot 6918394582717$

secp128r2

$|G| = 850705916906205346039557219268 13660579$

$b = 730195426182061735823013771465 48133543$

$|E| = 4222485329260 \cdot 4376586107537 \cdot 4603369401979$

secp160k1

$|G| = 1461501637330902918203686915170869725397159163571$

$b = 1014269469389219214184903107646149695236127481640$

$|E| = 11130827212809215 \cdot 11394976247906837 \cdot 11522811061606267$

secp160r1

$|G| = 1461501637330902918203687197606826779884643492439$

$b = 1231565154230325865757423073063591837019188457168$

$|E| = 11174885494467645 \cdot 11008949181540889 \cdot 11879833598755579$

secp160r2

$|G| = 1461501637330902918203685083571792140653176136043$

$b = 1987871000780349598609964130362172069 2363507758$

$|E| = 10573725526879272 \cdot 11520572597065679 \cdot 11997678180434227$

secp192k1

$|G| = 6277101735386680763835789423061264271957123915200845512077$

$b = 1094708638413029664629646177364452405008715587623144058105$

$|E| = 16352962116221436126 \cdot 17705499411507224387 \cdot$
$\qquad 21679764265977655387$

secp192r1

$|G| = 6277101735386680763835789423176059013767194773182842284081$

$b = 7339867319969617520190619107777595180087882698 5233013574$

$|E| = 17294274520438999164 \cdot 19491494149529285201 \cdot$
$\qquad 18621372472744345117$

secp224k1

$|G| = 26959946667150639794667015087019640346510327083120074548994958668279$

$b = 24618590432167307909930264143550961204039679464315847760586750945971$

$|E| = 2599695970501167945066 \cdot 33448358726421720956541 \cdot$
$\qquad 31004280361955770972381$

secp224r1

$|G| = 269599466671506397946670150870196259404578077144243917 2\backslash$
1682722368061

$b = 861814932527596025116148711861115855634130668475173705465\backslash$
8821880904

$|E| = 29343613141744570024644 \cdot 3179841463232218870759 3 \cdot$
$\qquad 2889348797541489042015 1$

secp256k1

$|G| = 11579208923731619542357098500868790785283756427907490 4\backslash$
$38260516314151816149433 7$

$b = 58603723116421395918689089913861383686268511262358322 04\backslash$
6880666663466737354099

$|E| = 4749438323999976741932074 5 \cdot 4517522893992561768821156 9 \cdot$
$\qquad 53967993991985944506666061$

secp256r1

$|G| = 11579208921035624876269744694940757352999695522413576 0\backslash$
$3424222590610685120443 69$

$b = 47655894101463316762236526132016393253057270840001423 83\backslash$
9782911257030924437529

$|E| = 50851524730203743853228640 \cdot 5549703769234338652615688 1 \cdot$
$\qquad 41030339309908399787973083$

secp384r1

$|G| = 39402006196394479212279040100143613805079739270465446 6\backslash$
$67946905279627659399113263569398956308152294913554433653942643$

$b = 89890103691693584367418476819795701055812436905742082 63\backslash$
$269556059650466158270056995485882025406947986682587367889624$

$|E| = 33986987048189154740054658522517921329 0 \cdot$
$\qquad 34957975980158220309922293105381374555 3 \cdot$
$\qquad 331634259739663319085318305031105092059$

secp521r1

Not available due to hardness of factoring.

C.2   Elliptic curve domain parameters over $\mathbb{F}_{2^m}$

sect113r1

$|G| = 5192296858534827689835882578830703$

$b = 9876370995430137570295458100160 98$

$|E| = 178524038025 \cdot 170996556499 \cdot 170088694619$

sect113r2

$|G| = 5192296858534827702972497909952403$

$b = 4583769363017101608245187708458901$

$|E| = 173146840968 \cdot 166401825973 \cdot 180213306239$

sect131r1

$|G| = 1361129467683753853893932755685365560653$

$b = 1258328605209306875070716696495675196119$

$|E| = 11939631029912 \cdot 10689621208893 \cdot 10664640354101$

sect131r2

$|G| = 1361129467683753853879535043412812867983$

$b = 3582323423441193920584042308064535 94114$

$|E| = 11466564749342 \cdot 10619089660293 \cdot 11178378760169$

sect163k1

$|G| = 5846006549323611672814741753598448348329118574063$

$b = 17767337697332384777035473627178295668 99983248537$

$|E| = 18247804538816661 \cdot 19436468698941551 \cdot 16482812332852169$

sect163r1

$|G| = 5846006549323611672814738465098798981304420411291$

$b = 15874048673063598988481933915408278165358520 9324$

$|E| = 17869920899977912 \cdot 17551363444944923 \cdot 18639137321795381$

sect163r2

$|G| = 5846006549323611672814742442876390689256843201587$

$b = 29562833239804228892914784773703209533557315 76940$

$|E| = 18200719603559559 \cdot 17568086274440101 \cdot 18282950480080931$

sect193r1

$|G| = 6277101735386680763835789423269548053691575186051040197193$

$b = 353388959879161638324511889829153537 67627436600288649159$

$|E| = 16547960255111188472 \cdot 19478515037898861263 \cdot$
$ 19474165359321867611$

sect193r2

$|G| = 6277101735386680763835789423314955362437298222279840143829$

$b = 441755957568112116066633401133360511847396492629731764429$

$|E| = 19387762096509288342 \cdot 18577800791543661067 \cdot$
$ 17427583967788534019$

sect233k1

$|G| = 3450873173395281893717377931138512760570940988862252126\backslash$
328087024741343

$b = 25122149205491735595137688390486707351370368980297988538\backslash$
30832766551245

$|E| = 15540300934427811855423 \cdot 153385740717714666739125 \cdot$
$\qquad 144772003913287824778231$

sect233r1

$|G| = 6901746346790563787434755862277025555839812737345013555\backslash$
379383634485463

$b = 7040938164755706341740819287052251842563468263172882818\backslash$
1773151878529

$|E| = 20679961703033668255416 \cdot 19518549023892523058089 \cdot$
$\qquad 170986465134593155152949$

sect239k1

$|G| = 2208558830972980411979121875928648149482165613217098488\backslash$
87480219215362213

$b = 2765023524422850785335545043505729301441208234122605903\backslash 8$
361077531582683

$|E| = 54381492548936524083768 \cdot 61057636259914416948097 \cdot$
$\qquad 665147345183743261991485$

sect283k1

$|G| = 3885337784451458141838923813647037813284811733793061324\backslash$
$2958749975298158297044226003873$

$b = 2818355229865436714527343713677198960370730199306046248\backslash 1$
$46777682199067799961811453900$

$|E| = 16292450803352497273678817784 \cdot$
$\qquad 1520136195235055781268409704 9 \cdot 15687721231974421411325545219$

sect283r1

$|G| = 7770675568902916283677847627294075626569625924376904889\backslash$
$10919652677004427778737 8692871$

$b = 7176744548618087685180510964632152605218899792685130465 5\backslash$
$959654362505529324586 37035413$

$|E| = 16932408152570400028840713015 \cdot$
$\qquad 19857620455536755941661666843 \cdot 23110686327095779427460999989$

sect409k1

$|G| = 3305279843951242994759576540163855199142023414821406096\backslash$
$4232439502288071128924919105067325845777745801409636659061773\backslash$
1358671

$b = 1387707401997092358107730246620422497696426410234477082 7\backslash$
$4370480173588453714079223650928941369852833083698503107547969\backslash$
459853

$|E| = 5492362860323245533411367863112936041418 4 \cdot$
$\qquad 6203098894060615206452999702957741059657 3 \cdot$
$\qquad 9701531730246750593797337668903305267180 1$

24

sect409r1

$|G| = 6610559687902485989519153080327710398284046829642812192\backslash$
$846487983041577748273748052081437237621791109659798672883656\backslash$
7526771

$b = 1381771144636272836014530111143648653092550550740277 0142\backslash$
$258673028256246338430064054266470642072686 2665470461 34340903\backslash$
3831354

$|E| = 87268656040437200019781889318456334448900 \cdot$
$8106300327891523007433555254221935468 5229 \cdot$
$93445254974986510684197220630040488205129$

sect571k1

Not available due to hardness of factoring.

sect571r1

Not available due to hardness of factoring.