Research Article

Prabhat Kushwaha*

Improved lower bound for Diffie–Hellman problem using multiplicative group of a finite field as auxiliary group

<https://doi.org/10.1515/jmc-2017-0053>

Received November 28, 2016; revised October 25, 2017; accepted February 12, 2018

Abstract: In 2004, Muzereau, Smart and Vercauteren [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 (2004), 50–72] showed how to use a reduction algorithm of the discrete logarithm problem to Diffie–Hellman problem in order to estimate lower bound for the Diffie–Hellman problem on elliptic curves. They presented their estimates on various elliptic curves that are used in practical applications. In this paper, we show that a much tighter lower bound for the Diffie–Hellman problem on those curves can be achieved if one uses the multiplicative group of a finite field as auxiliary group. The improved lower bound estimates of the Diffie–Hellman problem on those recommended curves are also presented. Moreover, we have also extended our idea by presenting similar estimates of DHP on some more recommended curves which were not covered before. These estimates of DHP on these curves are currently the *tightest* which lead us towards the equivalence of the Diffie–Hellman problem and the discrete logarithm problem on these recommended elliptic curves.

Keywords: Discrete logarithm problem, lower bound for the Diffie–Hellman problem, elliptic curves used in practical applications

MSC 2010: 94A60

Communicated by: Simon Blackburn

1 Introduction

It is well known that the discrete logarithm problem (DLP) is one of two primitives that are commonly used as a building block in public key protocols, other being integer factorization. Computational difficulty in solving DLP is a security necessity for the protocols based on it. However, an interesting thing about these DLP-based protocols is that security of many of such protocols does not exactly rely on the hardness of DLP. For example, the ElGamal public key cryptosystem is secure if and only if the Diffie–Hellman problem (DHP) is hard to solve [7, Proposition 2.10]. That means, it is enough for an attacker to solve DHP to break the ElGamal cryptosystem. The Diffie–Hellman key exchange, pairing-based cryptosystems, digital signature schemes and many more protocols are some other examples where the security of the protocol depends on hardness on DHP. This is why hardness of DHP is of utmost importance in public key cryptography.

If DLP is easy, DHP is easy because a solution of DLP immediately yields a solution of DHP. Therefore, the only meaningful scenario to study the hardness of DHP is when DLP is known to be hard. Barring some weak elliptic curves over finite fields, there is no efficient algorithm to solve the discrete logarithm problem on the

*Corresponding author: Prabhat Kushwaha, SEAL Lab, Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India, e-mail: prabkush@gmail.com

group of points of an elliptic curve over finite field (ECDLP) and thus, those elliptic curves are widely used for practical purposes. Thus, it is of paramount importance to study about the hardness of the elliptic curve Diffie–Hellman problem (ECDHP) from the point of view of practical cryptography. The *central theme* of this paper is to study the hardness of ECDHP on curves used in practice since a number of public key protocols are designed on such curves and their security depends only on the hardness of ECDHP.

1.1 Summary of existing work

To study the hardness of DHP the traditional method, and the only method known so far, involves reduction arguments from DLP to DHP. In this subsection, we will summarize those reduction arguments. In such reductions, one tries to solve DLP efficiently (in polynomial time of the input bit), using the existence of the a solution of DHP as sub-routine. If there exists such an algorithm, we say that *DLP reduces to DHP* in polynomial time, denoted by $DLP \leq_p DHP$. Informally, if $DLP \leq_p DHP$, it implies that DHP is at least as hard as DLP, or equivalently, DLP is no harder than DHP. Clearly, existence of any such reduction algorithm in case of elliptic curve groups would imply that ECDHP is hard, since ECDLP is hard to solve.

The first published result regarding the equivalence between DLP and DHP was due to den Boer [6]. He proved the equivalence in the group $\mathbb{F}_p^\times$ if $\phi(p-1)$ (not $p-1$) is smooth, where ϕ is Euler's totient function. Later on, Maurer attempted to generalize the idea of den Boer to any generic group G by introducing the technique of implicit representation of elements of a finite field [11]. Soon after, Maurer and Wolf extended the idea of den Boer in another sense by using the concept of auxiliary groups in constructing such reduction algorithms: den Boer was using $\mathbb{F}_p^\times$ as auxiliary group but Maurer and Wolf proposed to also use the elliptic curve over $\mathbb{F}_p$ as auxiliary group. They showed that $DLP \leq_p DHP$ for any group G of prime order p if we are able to find an elliptic curve over $\mathbb{F}_p$ with smooth order [12–15]. Smooth order of the auxiliary elliptic curve was the main reason behind the polynomial time reduction of DLP to DHP in their algorithm because it ensures that the total number of group operations as well as the total number of calls to the DH-oracle required in their algorithm remain polynomial in the input size. However, it is exceptionally hard, in general, to construct an elliptic curve over $\mathbb{F}_p$ of smooth order for large p , resulting in the failure of above theorem. Therefore, some alternate method was needed to study the hardness of DHP in general.

In 2004, Muzereau, Smart and Vercauteren [18] re-visited the Maurer and Wolf reduction algorithm of DLP to DHP for special case of elliptic curve groups over finite field. They explicitly constructed auxiliary elliptic curves, which were required in the reduction algorithm, for a number of elliptic curves recommended for practical implementation in SEC 2 [20] by Standard for Efficient Cryptography Group (SECG) at Certicom Corporation. We will refer to those recommended elliptic curves in SEC 2 as *SECG curves* [20] throughout this paper. However, the orders of the auxiliary elliptic curves constructed were not smooth enough, making the cost of the reduction algorithm exponential. Therefore, their reduction algorithm with those auxiliary elliptic curves failed to prove the polynomial reduction of ECDLP to ECDHP for those recommended SECG curves. Nevertheless, all was not lost as it might seem, since they were the first to give precise estimate of the number of group operations needed in such a reduction algorithm and showed how to use such a reduction algorithm to estimate the minimum number of group operations required to solve ECDHP on those SECG curves [18, Table 1, Table 2].

Bentahar later applied the idea similar to Muzereau, Smart and Vercauteren [18] but constructed different auxiliary elliptic curves over $\mathbb{F}_p$ to improve those estimates of Muzereau, Smart and Vercauteren on the exact lower bounds for ECDHP on those important SECG curves [2, Table 1, Table 2]. Since those lower bounds for ECDHP are assumed to be beyond the reach of present computational power, it establishes the security of several protocols relying on the hardness of ECDHP on those recommended SECG curves. This shows the significance of this remarkable approach.

In another related work, Brown and Gallant [4, Theorem 1] presented a generic algorithm reducing DLP to *static Diffie–Hellman problem (SDHP)*. SDHP is a special case of DHP. They also gave the minimum number of group operations required to solve SDHP [4, Corollary 1] on any group of order p where the best algorithm to solve DLP requires at least $\sqrt{p}$, for example the group of points on an elliptic curve over a finite field.

1.2 Our contribution

The algorithms of Muzereau, Smart and Vercauteren [18] and Bentahar [2] both use the same reduction algorithm suggested by Maurer and Wolf. They used suitable elliptic curves over a finite field as auxiliary groups for estimating the lower bound for DHP. Our contribution in this paper is that we have presented a DLP to DHP reduction algorithm that, even though it uses the multiplicative group of a finite field as an auxiliary group like den Boer [6], does not require the *Chinese Remainder Theorem*. We want to emphasize that earlier work of den Boer [6] (with $\mathbb{F}_p^\times$ as the auxiliary group) or Maurer and Wolf [12–15] (with elliptic curve over $\mathbb{F}_p$ as the auxiliary group) both require the Chinese Remainder Theorem. This absence of the Chinese Remainder Theorem makes this reduction different from previous reductions. However, our work in estimating the lower bound for DHP may be seen as an application of the den Boer reduction algorithm with $\mathbb{F}_p^\times$ as auxiliary group which is in contrast to the fact that Muzereau, Smart and Vercauteren [18] as well as Bentahar [2] both applied the Maurer and Wolf reduction algorithm with elliptic curves as auxiliary groups. The application of $\mathbb{F}_p^\times$ as auxiliary group in context of estimating the lower bound for DHP is a new idea.

Owing to this difference between our algorithm and previous algorithms and the change in the auxiliary group from an elliptic curve over a finite field to the multiplicative group of a finite field, our reduction algorithm requires a very small number of DH-oracle calls. Thus, it results in increasing the lower bound for DHP because the lower bound for DHP is inversely proportional to the number of calls to the DH-oracle. When applied to SECG curves studied first by Muzereau, Smart and Vercauteren and then by Bentahar, our reduction algorithm improves the previous lower bounds for ECDHP.

More precisely, assuming that the best algorithm to solve DLP on an elliptic curve of order p takes at least $\sqrt{p}$ group operations, Muzereau, Smart and Vercauteren gave the following estimate on the lower bound for ECDHP [18, Theorem 4]:

Theorem 1. *Let p be a prime. Assuming that in the interval $[p + 1 - \sqrt{p}, p + 1 + \sqrt{p}]$ there is an integer which is the product of three primes of roughly equal size. Then there exists a string S which implies that the best algorithm to solve the ECDHP for an elliptic curve of order p takes time at least $O(\frac{\sqrt{p}}{(\log_2 p)^2})$ group operations.*

Under the same assumption as above that the best known algorithm to solve ECDLP on an elliptic curve of order p requires at least $\sqrt{p}$ group operations, our reduction algorithm will prove the following theorem which improves the lower bound for ECDHP given in Theorem 1:

Theorem 2. *For a prime p , assume that there exists a divisor d of $p - 1$ of the size roughly equal to $\sqrt[3]{p}$. Then any algorithm that solves ECDHP for an elliptic curve of order p requires at least $O(\frac{\sqrt{p}}{\log_2 d})$ group operations.*

The significance of the above result is that it applies to almost all the recommended SECG curves because such a divisor d exists for almost all of those curves where prime p is either the order of those elliptic curve groups or the largest prime divisor (with a very small co-factor of either 2 or 4). Therefore, the above reduction gives us the *tightest* lower bound for ECDHP on these SECG curves.

Moreover, for curves SECP521R1, SECT409R1, SECT571R1, SECT571K1, Bentahar was unable to construct the auxiliary elliptic curves. However, we had no problem applying our algorithm to these curves and the lower bound estimates of ECDHP on these curves are also given here. Moreover, we have also extended our idea beyond SECG curves and estimated the lower bounds for ECDHP on non-SECG curves such as Curve25519 and Curve448 (recently added by Crypto Forum Research Group (CFRG)) [10], Brainpool curves [17], pairing friendly Barreto–Naehrig curves (BN curves) [1], CFRG BN curves [8].

The rest of the paper is organized as follows. In Section 2, we mention some notations and definitions that will be required to understand the material of this paper. Our main contributions are described in Section 3 and Section 4. In Section 3, we present a reduction algorithm of DLP to DHP without using the Chinese Remainder Theorem and prove Theorem 2 regarding the improved lower bound for ECDHP as a result of that reduction algorithm. In Section 4, we present the improved lower bounds for ECDHP for various SECG curves (Table 1 and Table 2), along with the similar estimates of ECDHP on some additional non-SECG elliptic curves (Table 3). Finally, we conclude in Section 5.

2 Notations and definitions

Let $\langle \mathbb{G}, + \rangle$ be a cyclic (additive) group generated by P and order of P is a prime p .

Definition 1. Given $Q \in \mathbb{G}$, the problem of computing the integer x modulo p such that $Q = xP$ is called the *discrete logarithm problem (DLP)* with respect to P .

Definition 2. Given $Q = xP, R = yP \in \mathbb{G}$ (x, y are unknown integers), the problem of computing $S = xyP$ is called the *Diffie–Hellman problem (DHP)* with respect to P .

From the above definitions, it is clear that if one can compute x from $Q = xP$ and then one can compute $xR = xyP \in \mathbb{G}$. Thus, the solution of DLP readily yields the solution of DHP. However, as discussed earlier, we are interested in the reverse implication: does a solution of DHP solve DLP as well? To answer this question, reduction of DLP to DHP has been suggested and also given in some particular cases by Maurer and Wolf. As mentioned earlier, one tries to solve DLP assuming that a solution of DHP is known, or equivalently, one has access to a DH-oracle. We define it formally as follows:

Definition 3. A *DH-oracle* is a function that takes $xP, yP \in \mathbb{G}$ as inputs and returns $xyP \in \mathbb{G}$ as output. We write it as $\mathcal{DH}(xP, yP) = xyP$.

It was great insight of Maurer and Wolf who generalized den Boer's first such reduction algorithm by introducing the ideas of *implicit representation* and *auxiliary groups*.

2.1 Implicit representation of elements of $\mathbb{F}_p$

Let $\mathbb{G}$ be a cyclic group with generator P whose order is a prime number p . Let $y \in \mathbb{F}_p$. Then $yP \in \mathbb{G}$ is called the *implicit representation of $y \in \mathbb{F}_p$ (with respect to $\mathbb{G}$ and P)*. We denote this by $y \rightsquigarrow yP$.

Let $yP, zP \in \mathbb{G}$ be implicit representations of $y, z \in \mathbb{F}_p$, respectively. Then the following basic algebraic operations in $\mathbb{F}_p$ can also be realized in $\mathbb{G}$ as follows:

- *Equality testing:* $y = z$ if and only if $yP = zP$.
- *Addition:* $y + z \rightsquigarrow yP + zP$ (one group operation in $\mathbb{G}$).
- *Subtraction:* $y - z \rightsquigarrow yP - zP$ ($O(\log p)$ group operations in $\mathbb{G}$).
- *Multiplication:* $y \cdot z \rightsquigarrow yzP = \mathcal{DH}(yP, zP)$ (one call to DH-oracle).
- *Inversion:* $y^{-1} = y^{p-2} = y \cdots y \rightsquigarrow y^{p-2}P$ ($O(\log_2 p)$ DH-oracle calls by using binary expansion).

Observe that the DH-oracle is used only for multiplication and inversion in $\mathbb{F}_p$. Therefore, the number of DH-oracle calls required in the reduction algorithm increases with the increase of the number of multiplication and inversions in $\mathbb{F}_p$ required in the reduction algorithm. We will see the importance of this in later sections.

2.2 Auxiliary groups

As the name suggests, any group (other than the group $\mathbb{G}$) is called an auxiliary group if it can be used to achieve the targeted goal of an algorithm. In the present context of DLP to DHP reduction, the goal is to solve DLP using the DH-oracle calls and implicit representations. Therefore, two essential properties of a possible auxiliary group $\mathbb{H}$ are:

- Elements of $\mathbb{H}$ can be represented as m -tuples of elements of $\mathbb{F}_p$ for some $m \geq 1$.
- Group operation in this auxiliary group $\mathbb{H}$ can be defined from algebraic operations in $\mathbb{F}_p$.

Note that if $\mathbb{H}$ has these two properties, then any computation in $\mathbb{H}$ (for example, equality testing, exponentiation in $\mathbb{H}$) can also be performed on their implicitly represented elements of $\mathbb{G}$. Interestingly, den Boer was the first to use $\mathbb{F}_p^\times$ as the auxiliary group [6], even though there was no mention of the term “auxiliary group” in his work. Later on, Maurer and Wolf coined the term “auxiliary group” and extended den Boer's idea by

also using elliptic curve as auxiliary group. They called elliptic curves $\tilde{E}(\mathbb{F}_p)$ and subgroups of $\mathbb{F}_{p^n}^\times$ for some $n \geq 1$ as *applicable auxiliary groups over $\mathbb{F}_p$* . For more details, refer to [14].

2.3 General idea of solving DLP using auxiliary groups and implicit representation computation

Let $\mathbb{G}$ be a cyclic group generated by P and order of P is a prime p . To solve DLP for Q , one requires to find the integer x where $Q = xP$. Moreover, we also have access to a DH-oracle on $\mathbb{G}$ and we are allowed to make calls to the DH-oracle to solve DLP on $\mathbb{G}$. To this end, using auxiliary groups over $\mathbb{F}_p$ and computation on implicitly represented elements of $\mathbb{F}_p$, the general idea for a DLP to DHP reduction algorithm can be described as follows:

- (1) Choose a cyclic auxiliary group $\mathbb{H}$ over $\mathbb{F}_p$ generated by ζ_0 .
- (2) *Embed* the unknown x into an implicitly represented element c of $\mathbb{H}$.
- (3) Compute the discrete logarithm of c with respect to ζ_0 in $\mathbb{H}$ explicitly, using computation (in $\mathbb{G}$) of implicitly represented elements of $\mathbb{F}_p$. Observe that computing implicit representations of finite field elements is exactly the place where the DH-oracle is used.
- (4) *Extract* the unknown x from the discrete logarithm of c with respect to ζ_0 found in the last step.

It is interesting to note that all DLP to DHP reduction algorithms known so far are based on the above general idea. More intriguing is the fact that only elliptic curves over $\mathbb{F}_p$ of smooth order have been used as auxiliary groups and studied extensively, except den Boer who used $\mathbb{F}_p^\times$ as auxiliary group [6].

If we take $\mathbb{H} = \tilde{E}(\mathbb{F}_p)$ as the auxiliary group with smooth order N where the elliptic curve $\tilde{E}(\mathbb{F}_p)$ is given by $Y^2 = X^3 + AX + B$, $A, B \in \mathbb{F}_p$, and generated by $P_0 = (x_0, y_0) \in \mathbb{H}$, the reduction algorithm of Muzereau, Smart and Vercauteren [18] (which follows the above general idea) *embeds* the unknown x implicitly into $c = Q_0 = (x, y) \in \mathbb{H}$ for some $y \in \mathbb{F}_p$. After that, the discrete logarithm k of Q_0 with respect to P_0 is computed *explicitly* using computations on implicitly represented elements. The last step is to *extract* x from $kP_0 (= Q_0)$ which is the abscissa of the point Q_0 . Observe that $Q_0 = (x, y)$ was not *explicitly* known before the computation of k . However, once we have k , we can compute Q_0 *explicitly* using P_0 and k as $Q_0 = kP_0$. Muzereau, Smart and Vercauteren first computed k modulo each prime power of N by repeatedly applying the Pohlig–Hellman algorithm on implicitly represented elements along with exhaustive search to find a collision, then used the Chinese Remainder Theorem to find k . Bentahar also applied the same method. We call this several instances of Pohlig–Hellman algorithm and exhaustive search in their reduction algorithms collectively as *sub-algorithm A*. For more details on this reduction, see [18].

In the following section, we present the proof of Theorem 2 by using a reduction algorithm that uses $\mathbb{F}_p^\times$ as auxiliary group (instead of an elliptic curve over $\mathbb{F}_p$) but without using the Chinese Remainder Theorem. The absence of the Chinese Remainder Theorem is what makes this reduction algorithm different from existing ones.

3 Main results

3.1 DLP to DHP reduction using $\mathbb{F}_p^\times$ as auxiliary group

We first present two lemmas which will lead to the proof of Theorem 2. The reduction algorithm given in Lemma 2 is similar to den Boer’s reduction [6] in the sense that $\mathbb{F}_p^\times$ plays a major role in both reductions. The reduction is motivated by Gallant and Brown [4, Theorem 1] and of Cheon’s work [5, Theorem 1]. The central idea used in both of them is exactly the same once one *particular* element is known. However, their goals were different. While Gallant and Brown used *SDHP oracle* to compute that *particular* element, Cheon included the existence of that *particular* element in the assumption only. As a result, work of Gallant and Brown yields a reduction algorithm of *DLP to SDHP* while Cheon’s work leads towards solving *DLPwAI*.

In our case, Lemma 1 computes that *particular* element using small number of calls to a DH-oracle which gives the DLP to DHP reduction algorithm in Lemma 2. We also observed that the reduction algorithm fits perfectly well into the general idea of implicit representation formulated by Maurer and Wolf but with $\mathbb{F}_p^\times$ as auxiliary group. Thus, for the sake of consistency, we present our algorithm into that setting only.

Lemma 1. *Let $\mathbb{G}$ be a cyclic additive group of prime order p , generated by $P \in \mathbb{G}$. Suppose we are given another element $Q = xP \in \mathbb{G}$ (x unknown). Then $x^d P \in \mathbb{G}$ can be computed by making at most $2\lceil \log_2 d \rceil$ calls to a DH-oracle for any positive integer d , where $\lceil \cdot \rceil$ is the greatest integer function.*

Proof. We start by observing that if $d = 2^r$, it takes exactly r calls to the DH-oracle for computing $x^d P \in \mathbb{G}$:

$$x^2 P = \mathcal{DH}(xP, xP) \quad (1 \text{ call}),$$

$$x^4 P = x^{2^2} P = \mathcal{DH}(x^2 P, x^2 P) \quad (2 \text{ calls}),$$

$$x^8 P = x^{2^3} P = \mathcal{DH}(x^4 P, x^4 P) \quad (3 \text{ calls}),$$

$$x^{16} P = x^{2^4} P = \mathcal{DH}(x^8 P, x^8 P) \quad (4 \text{ calls}).$$

So, we see that continuing the same steps we can compute $x^d P = x^{2^r} P \in \mathbb{G}$ for $d = 2^r$ in exactly r calls to the DH-oracle.

Let d be any integer, not necessarily some power of 2. Let k be the unique integer such that $2^{k-1} \leq d < 2^k$. So, we can write the binary representation of d as follows:

$$d = a_0 + a_1 2^1 + a_2 2^2 + \cdots + a_{k-1} 2^{k-1},$$

where the a_i are 0 or 1 with $a_{k-1} = 1$.

The above remark shows that the elements $x^{2^1} P, x^{2^2} P, x^{2^3} P, \dots, x^{2^{k-1}} P \in \mathbb{G}$ can be computed by making at most $k - 1$ calls to the DH-oracle. By using these elements, it is easy to check that $x^d P$ can be computed by making at most $k - 1$ additional calls to the DH-oracle as the number of non-zero a_i can be at most $k - 1$ for $i = 0, 1, \dots, k - 2$. Therefore, we need to make at most $2(k - 1) = 2\lceil \log_2 d \rceil$ calls to the DH-oracle to compute $x^d P \in \mathbb{G}$ since $k - 1 = \lceil \log_2 d \rceil$. This completes the proof of the lemma. $\square$

Now, we present our DLP to DHP reduction algorithm using implicit representations with $\mathbb{F}_p^\times$ as auxiliary group in the following lemma:

Lemma 2. *Let $\mathbb{G}$ be a additive cyclic group generated by $P \in \mathbb{G}$ and the order of P is a prime number p . Let $Q = xP \in \mathbb{G}$. Then x can be computed using at most $2 \log_2 p (\lceil \sqrt{\frac{p-1}{d}} \rceil + \lceil \sqrt{d} \rceil)$ group operations and by making at most $2\lceil \log_2 d \rceil$ calls to the DH-oracle. Here d is a positive divisor of $p - 1$.*

Proof. As already discussed, the proof is based on implicit representation of elements of $\mathbb{F}_p$ using $\mathbb{H} = \mathbb{F}_p^\times$ as the auxiliary group. Recall that the unknown x will be implicitly represented by $Q = xP \in \mathbb{G}$. Furthermore, $\mathbb{F}_p^\times$ is a cyclic group with $\phi(p - 1)$ generators, where ϕ is the Euler totient function. Since a random element in $\mathbb{F}_p^\times$ is a generator with probability

$$\frac{\phi(p - 1)}{p - 1} > \frac{1}{6 \log(\log(p - 1))}$$

which is large enough (see [5]), it is easy to choose a generator of $\mathbb{F}_p^\times$.

Let ζ_0 be a generator of $\mathbb{H} = \mathbb{F}_p^\times$. Then

$$x = \zeta_0^{i_0} \pmod{p}$$

for some integer i_0 such that $1 \leq i_0 \leq p - 1$.

We want to compute i_0 explicitly and then x can be computed using the above equation. To this end, let $\zeta = \zeta_0^d \pmod{p}$. Since $d \mid (p - 1)$, there exists unique cyclic subgroup $\mathbb{K}$ of $\mathbb{H} = \mathbb{F}_p^\times$ of order $\frac{p-1}{d}$ generated by ζ . Now as $(x^d)^{(\frac{p-1}{d})} = 1$, it implies that $x^d \in \mathbb{K}$. Therefore, there exists unique non-negative integer j with $1 \leq j \leq (\frac{p-1}{d})$ such that

$$x^d = \zeta^j \pmod{p}. \quad (1)$$

Let $d_1 = \lceil \sqrt{\frac{p-1}{d}} \rceil$. Since j is between 1 and $\frac{p-1}{d}$, there exist two unique non-negative integers u_1 and v_1 with

$0 \leq u_1, v_1 \leq d_1$ such that $j = u_1 d_1 - v_1$. Plugging this value of j in equation (1), we get

$$x^d = \zeta^{u_1 d_1} \zeta^{-v_1} \pmod{p},$$

which implies

$$\zeta^{v_1} x^d = (\zeta^{d_1})^{u_1} \pmod{p}. \quad (2)$$

Recall that the equality of two field elements can also be checked on their implicitly represented elements as follows: $y = z$ (in $\mathbb{F}_p^\times$) is equivalent to $yP = zP$ in $\mathbb{G}$. Therefore, the above *implicit equation* in $\mathbb{F}_p^\times$ is equivalent to the following *explicit equation* in $\mathbb{G}$:

$$\zeta^{v_1} (x^d P) = (\zeta^{d_1})^{u_1} P. \quad (3)$$

By Lemma 1 above, we can compute the implicit representation $x^d P$ of x^d by making at most $2 \lceil \log_2 d \rceil$ calls to the DH-oracle. Looking at equation (3), it is clear that the elements on the left-hand side can be computed using $x^d P$ for any value of v_1 with $0 \leq v_1 \leq d_1$, by repeated addition of previous terms by ζ -times. Similarly, the elements on the right-hand side can be computed for any value of u_1 with $0 \leq u_1 \leq d_1$ using P by repeated addition of previous terms by ζ^{d_1} -times. So, we compute $\zeta^{v_1} (x^d P)$ for each v_1 with $0 \leq v_1 \leq d_1$ and store them. Then we compare them with each of the right-hand side terms (similar to Baby-Step Giant-Step (BSGS) algorithm [16]) to find a match and it yields the integer $j = u_1 d_1 - v_1$.

Note that the non-negative integer $j = u_1 d_1 - v_1$ in equation (1) is nothing but i_0 modulo $\frac{p-1}{d}$. Now to compute i_0 modulo $(p-1)$ from this integer j , we apply the division algorithm on i_0 with divisor $\frac{p-1}{d}$ to get a relation between i_0 and j and it gives us $i_0 = (\frac{p-1}{d})t + j$ for some non-negative integer t . Observe that $0 \leq t < d$, otherwise $i_0 \geq p-1$, a contradiction. Therefore, the integer t can be written uniquely as $t = u_2 \lfloor \sqrt{d} \rfloor - v_2$ for some $0 \leq u_2, v_2 \leq \lfloor \sqrt{d} \rfloor$, again by the division algorithm. Thus, we get the following *implicit equation* in $\mathbb{H} = \mathbb{F}_p^\times$:

$$x = \zeta_0^{i_0} = \zeta_0^{j+t(\frac{p-1}{d})} = \zeta_0^j \zeta_0^{(\frac{p-1}{d})(u_2 \lfloor \sqrt{d} \rfloor - v_2)},$$

which is equivalent to

$$(\zeta_0^{\frac{p-1}{d}})^{v_2} x = (\zeta_0^{(\frac{p-1}{d}) \lfloor \sqrt{d} \rfloor})^{u_2} \zeta_0^j.$$

The last *implicit equation* in $\mathbb{H} = \mathbb{F}_p^\times$ is equivalent to the following *explicit equation* in $\mathbb{G}$:

$$(\zeta_0^{\frac{p-1}{d}})^{v_2} (xP) = (\zeta_0^{(\frac{p-1}{d}) \lfloor \sqrt{d} \rfloor})^{u_2} (\zeta_0^j P). \quad (4)$$

As xP and $\zeta_0^j P$ are known, we can solve for u_2, v_2 by finding a match between two sides of equation (4) using the BSGS algorithm. This solution for u_2, v_2 would give us

$$i_0 = \left(\frac{p-1}{d} \right) (u_2 \lfloor \sqrt{d} \rfloor - v_2) + j.$$

Thus, we have *explicitly* computed i_0 . Lastly, we *extract* the original discrete logarithm x from this i_0 and the relation $x = \zeta_0^{i_0}$.

It is easy to see that it takes at most $2 \log_2 p (\lfloor \sqrt{\frac{p-1}{d}} \rfloor)$ group operations to find a match in equation (3) and at most $2 \log_2 p (\lfloor \sqrt{d} \rfloor)$ group operations to find a match in equation (4). Therefore, we have computed the discrete logarithm x using at most $2 \log_2 p (\lfloor \sqrt{\frac{p-1}{d}} \rfloor + \lfloor \sqrt{d} \rfloor)$ group operations and by making at most $2 \lceil \log_2 d \rceil$ calls to the DH-oracle. This completes the proof. $\square$

Remark 1. One can get rid of the factor $\log_2 p$ from the above time complexity using KKM improvement [9]. Then the above time complexity reduces to $2(\lfloor \sqrt{\frac{p-1}{d}} \rfloor + \lfloor \sqrt{d} \rfloor)$. Moreover, the algorithm follows the general model with $\mathbb{H} = \mathbb{F}_p^\times$ as auxiliary group, where the unknown x is *embedded implicitly* into itself, i.e. $c = x$.

Remark 2. Observe that x^d is unknown in equation (2) because x is unknown. This makes equation (2) an *implicit equation* in $\mathbb{H} = \mathbb{F}_p^\times$. This is exactly the place where implicit representation computation comes into play: to compute the implicit representation $x^d P$ of x^d . Note that the Chinese Remainder Theorem is *not needed* in the reduction. In all, we have used the Pohlig–Hellman algorithm once, the BSGS algorithm twice and did not use the Chinese Remainder Theorem in our reduction algorithm. We call this single occurrence of the Pohlig–Hellman algorithm and the use of the BSGS algorithm twice in our reduction algorithm collectively as *sub-algorithm B*.

3.2 Proof of Theorem 2

Since proving computational equivalence of DLP and DHP is very hard in general, the next best thing would be to somehow estimate the minimum number of group operations required to solve DHP. That is exactly what Muzereau, Smart and Vercauteren [18] did for the elliptic curves groups [20] recommended for practical implementation by SECG. Their idea was to construct a DLP to DHP reduction algorithm in which the total number of group operations needed in the reduction algorithm should be very small, thus *insignificant* when compared to the cost of solving DLP. Once we have such a reduction algorithm, they proposed that ratio of the cost of DLP and the number of calls to the DH-oracle needed in the algorithm gives the minimum number of group operations required by any algorithm to break DHP. This is how the above reduction algorithm can also be used to estimate the minimum cost of DHP. These estimates of the lower bound for DHP in view of the above reduction may also be seen as an application of the den Boer reduction algorithm with $\mathbb{F}_p^\times$ as auxiliary group which is in contrast to the fact that Muzereau, Smart and Vercauteren [18] and Bentahar [2] both applied the Maurer and Wolf reduction algorithm with elliptic curves as the auxiliary groups to get their estimates of DHP. The application of $\mathbb{F}_p^\times$ as auxiliary group in context of estimating the lower bound for DHP is not reported before and seems to be an interesting idea.

Proof. Since we are dealing with an elliptic curve of prime order p , we assume that the best algorithm to solve the elliptic curve discrete logarithm problem will take at least $\sqrt{p}$ group operations. First, we give the general set up of estimating the lower bound on DHP.

Let C_{DLP} and C_{DHP} denote the time complexity of solving DLP and DHP, respectively. Therefore, in view of a general DLP to DHP reduction algorithm, we get $C_{\text{DLP}} = n \cdot C_{\text{DHP}} + M$, where n is the number of calls to the DH-oracle and M is the number of group operations required in the reduction algorithm. Now, if we assume that $M \ll C_{\text{DLP}}$, then we have

$$C_{\text{DHP}} = \frac{C_{\text{DLP}} - M}{n} \approx \frac{C_{\text{DLP}}}{n}.$$

If we set $T_{\text{DH}} = \frac{C_{\text{DLP}}}{n}$, the number T_{DH} is exactly what gives the minimum number of group operations needed by any algorithm to solve DHP, assuming $M \ll C_{\text{DLP}}$. This is how Muzereau, Smart and Vercauteren [18] estimated the minimum number of group operations required by any algorithm that solves DHP. Of course, the aim would be to make n as small as possible to have the value of T_{DH} as large as possible.

Now, we prove the lower bound on ECDHP. In case of G being an elliptic curve group of prime order p , one can take $C_{\text{ECDLP}} \geq \sqrt{p}$ under our assumption. Since there is a divisor d of $p - 1$ such that $d \approx \sqrt[3]{p}$, then it is easy to check that $M \leq 2([\sqrt{\frac{p-1}{d}}] + [\sqrt{d}]) \approx \sqrt[3]{p}$, satisfying the condition $M \ll C_{\text{ECDLP}}$ because $\sqrt[3]{p} \ll \sqrt{p}$ and $\sqrt{p} \leq C_{\text{ECDLP}}$. Since $n \leq 2[\log_2 d]$, we finally get

$$T_{\text{DH}} = O\left(\frac{\sqrt{p}}{\log_2 d}\right).$$

This implies that the minimum number group operation to solve ECDHP on any elliptic curve group of prime order p by any algorithm is of the order of $O(\frac{\sqrt{p}}{\log_2 d})$ if there exists a divisor d of $p - 1$ of size approximately $\sqrt[3]{p}$. This completes the proof. $\square$

Remark 3. If we assume that a divisor d of $p - 1$ of size approximately $\sqrt[3]{p}$ exists, then the above result shows that the cost of ECDHP is getting closer to the cost ECDLP. Moreover, observe that the total number of group operations, M needed in the reduction algorithms of Muzereau, Smart and Vercauteren [18] and Bentahar [2] was also of the same order, i.e. $M \approx \sqrt[3]{p}$. This indicates the importance of such a divisor d of size approximately $\sqrt[3]{p}$ in our reduction algorithm.

Remark 4. It is clear that the *sub-algorithm B* used in our reduction algorithm (with $\mathbb{F}_p^\times$ as auxiliary group but without the Chinese Remainder Theorem) and *sub-algorithm A* used in the previous reduction algorithms (with $\bar{E}(\mathbb{F}_p)$ as an auxiliary group along with Chinese Remainder Theorem) are quite different. We will see the positive effect of this difference on the estimates of the lower bound for DHP in the next section.

Remark 5. It should be mentioned that the author came to know about the similar result of Brown [3, Section 3.3] only after the completion of our work and we maintain that our work is done independently. One

good enough reason to convince ourselves is that his work was motivated mainly by the search of one particular suitable elliptic curve for which den Boer's reduction of DLP to DHP is optimum and no numerical estimates of DHP were provided for any of the existing recommended curves. On the other hand, our work is motivated by improving the results of Muzereau, Smart and Vercauteren [18] and Bentahar [2] on the lower bound estimates of DHP on SECG curves. It is a mere coincidence that the use of $\mathbb{F}_p^\times$ as auxiliary group (as in the case of den Boer's reduction) happened to make us achieve our goal for already studied SECG curves. Moreover, we have also given the lower bound estimates of ECDHP on some more non-SECG curve parameters such as CFRG curves (Curve25519, Curve448) [10], Brainpool curves [17], pairing friendly Barreto–Naehrig curves (also known as BN curves) [1, 8] which were not covered in the previous work of Muzereau, Smart and Vercauteren or Bentahar.

Remark 6. The technique of Muzereau, Smart and Vercauteren [18] to estimate the lower bound for the ECDHP is based on the assumption that the best-known attack on ECDLP has running time of $\sqrt{p}$. Therefore, the previous estimates as well as our improved estimates of the cost for solving ECDHP would no longer be valid if much faster attacks on ECDLP, say of the order of $\sqrt[4]{p}$, are discovered.

4 Advantage of $\mathbb{F}_p^\times$ over $\bar{E}(\mathbb{F}_p)$ as auxiliary group

4.1 Improved value of T_{DH}

The difference between *sub-algorithm A* and *sub-algorithm B* as well as the change of auxiliary group from $\bar{E}(\mathbb{F}_p)$ to $\mathbb{F}_p^\times$ both have their implications on the number of DH-oracle calls, consequently affecting the value of T_{DH} . Since *sub-algorithm A* used in previous reduction algorithms required several iterations of the Pohlig–Hellman algorithm, one had to compute a large number of implicitly represented elements in those reduction algorithms. Therefore, a large number of DH-oracle calls were needed in the previous reduction algorithms. On the other hand, our reduction algorithm while using *sub-algorithm B* requires *only* one implicitly represented element $x^d P$ of $x^d \in \mathbb{F}_p^\times$. This element can be computed by using at most $n \leq 2[\log_2 d]$ DH-oracle calls which can further be made really small by taking small value of d .

Recall that the addition operation in $\bar{E}(\mathbb{F}_p)$ requires many multiplications in $\mathbb{F}_p$ (one multiplication in $\mathbb{F}_p$ means one DH-oracle call to compute the implicit representation) and many inversions in $\mathbb{F}_p$ (one inversion in $\mathbb{F}_p$ means on average $\frac{3}{2}[\log_2 p]$ calls to the DH-oracle to compute the implicit representation). Thus, in terms of DH-oracle calls, computing the sum of elements in $\bar{E}(\mathbb{F}_p)$ is much more expensive than multiplying elements in $\mathbb{F}_p^\times$.

Since our main aim through this reduction algorithm is to increase the value of T_{DH} which is inversely proportional to the number of DH-oracle calls n , it will be nice to reduce the number of DH-oracle calls as much as possible. That is exactly what our reduction algorithm does using *sub-algorithm B* and $\mathbb{F}_p^\times$ as auxiliary group. This shows that the advantage of our reduction algorithm in achieving the improved value of T_{DH} , over the previous reduction algorithms which used *sub-algorithm A* and $\mathbb{H} = \bar{E}(\mathbb{F}_p)$ as auxiliary group.

4.2 Tightest lower bound for ECDHP for SECG and non-SECG curves

In this subsection, we study the positive effect of our reduction on the lower bound for ECDHP for various important SECG curve parameters [20] and present the improved lower bound for ECDHP on those SECG curves (see Table 1 and Table 2). These SECG curves are recommended in SEC 2 by Standard of Efficient Cryptography Group (SECG) at Certicom Corporation to be used for practical purposes and we are calling those curves SECG curves. These SECG curves are divided into two sub-categories: curves over prime fields of large odd characteristic and curves over binary fields. The prime p denotes the order of those SECG curves defined over prime fields of odd characteristic. For the remaining SECG curves defined over binary fields, p denotes the prime divisor of the order of the curve, with a very small co-factor of either 2 or 4.

It should also be noted that SECG curves [20] include all curves recommended by NIST [19] and the most used ones in ANSI [21]. These covers the most commonly used curves in practice. Thus, these are important curves from the point of view of public key cryptography. Moreover, we have also extended our idea to previously uncovered non-SECG curves such as CFRG curves (Curve25519, Curve448) [10], pairing friendly Barreto–Naehrig curves (also known as BN curves) [1], some more pairing friendly curves given by CFRG [8], and Brainpool curves [17]. We have presented the similar estimates of ECDHP on those non-SECG curves (see Table 3). CFRG curves (Curve25519, Curve448) [10] are recently proposed by the Crypto Forum Research Group (CFRG) and they are already under consideration to be included in the NIST curves. Four pairing friendly curves (also known as BN curves) were given by Barreto and Naehrig in [1] and we are calling them BN160, BN192, BN224 and BN256 depending upon their field sizes. Some more pairing friendly BN curves were proposed by CFRG [8] which also include BN224 and BN256. Brainpool curves [17] are another set of curve parameters which are given by CFRG.

Muzereau, Smart and Vercauteren [18] used the value of T_{DH} as the lower bound on group operations to break DH-protocol and also gave the estimates for T_{DH} on various SECG curves. Thereafter, Bentahar [2] improved the previous values of T_{DH} given by Muzereau, Smart and Vercauteren and his estimates remain the best estimates till date. Now, in our algorithm, with $\mathbb{F}_p^\times$ as the auxiliary group, we have

$$n \leq 2\lceil \log_2 d \rceil \quad \text{and} \quad M \leq 2\left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \lceil \sqrt{d} \rceil\right),$$

where d is some divisor of $p-1$. As per the discussion above, to achieve a tighter (larger) value of T_{DH} using our reduction algorithm, one should try to make $n \leq 2\lceil \log_2 d \rceil$ as small as possible, which forces d to be small as well. On the other hand, we have to make sure that $M \approx \sqrt[3]{p}$, so that it does not violate $M \ll C_{ECDLP} = \sqrt{p}$. It is not hard to see that for really small value of d , M is inversely proportional to d . Therefore, too small value of d must not be used to avoid the violation of $M \ll C_{ECDLP} = \sqrt{p}$. Also note that $d \approx \sqrt[3]{p}$ yields $M \approx \sqrt[3]{p}$ in our reduction algorithm.

Keeping all these in mind, we factored $p-1$ and found that most of SECG curves contain divisors d which are between $\sqrt[3]{p}$ and $\sqrt{p}$ and we have taken the smallest such d in the range $\sqrt[3]{p}$ and $\sqrt{p}$ to compute the values in Tables 1 and 2 given below. For those curves where such a divisor d does not exist, we have chosen the largest d less than $\sqrt[3]{p}$ to compute the values in the tables. For exact values of the d in case of various curves, see the appendix.

For those choices of d , we calculated the *exact* number of the DH-oracle calls, $n \leq 2\lceil \log_2 d \rceil$, using binary expansion of d . The values of n thus achieved are significantly small as compared with the values of n shown by Bentahar [2] (and much smaller than those in the work of Muzereau, Smart and Vercauteren [18]). Consequently, these significantly small values of n resulted in much tighter (larger) values of T_{DH} for all SECG curves. Therefore, it implies that we have given the *tightest lower bound*, known so far, on ECDHP for all SECG curves [20] (except SECP224K1). In other words, our results show the gap between the cost of ECDHP and ECDLP to be the *least* (known so far) for these curves and it leads us one step closer towards the computational equivalence of ECDHP and ECDLP for these important curves.

One additional advantage of our algorithm is that the values of M in our algorithm are less than or of almost same order as the ones given by Bentahar [2] for most of SECG curves.

Table 1 and Table 2 present the key values, $\log_2 M$, $\log_2 n$ and $\log_2 T_{DH}$ for various SECG curves. The tables also have the value of $\log_2 \sqrt{|E|}$ which refers to the assumed minimum cost of solving DLP in that particular SECG curve E . The column under ADV in Table 1 and Table 2 shows the number of security bits gained by the values of T_{DH} in our algorithm over the previous best known values of T_{DH} given by Bentahar [2]. Moreover, the present algorithm works for the curves SECP521R1, SECT571R1, SECT571K1 as well which were out of reach in previous work due to inability to construct auxiliary elliptic curves, and Tables 1 and 2 give the key data for these curves as well. Table 3 presents the key values $\log_2 M$, $\log_2 n$ and $\log_2 T_{DH}$ for various non-SECG curves which were not covered in the previous works [2, 18].

It should also be remarked that the current algorithm fails for the SECG curve SECP224K1, and three non-SECG curves BRAINPOOL224R1, BRAINPOOL256R1 and BRAINPOOL320R1 as there does not exist any divisor of $p-1$ of appropriate size. Therefore, Bentahar's result still gives the tightest value of T_{DH} for the curve SECP224K1.

To understand the advantage gained by our result over the work of Bentahar [2], as an example we consider the security of ECDHP for SECP256R1. The best known algorithm at present to solve ECDLP on this curve takes on an average 2^{128} group operations. Now, our algorithm implies that ECDHP cannot be solved in less than $2^{121.00}$ group operations, in contrast to $2^{115.40}$ group operations from the work of Bentahar [2]. This shows that there is a gain factor of $2^{5.60}$ over the previous best known result given by Bentahar for the curve SECP256R1, see Table 1. If we assume that today's computational power is incapable of performing $2^{121.00}$ group operations (which is considered to be true by many), then ECDHP on the curve SECP256R1 is secure and any cryptography protocol which rely on DHP for its security can safely be implemented on the curve SECP256R1, under the assumption above.

SECP curve	$\log_2 \sqrt{ E }$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$	ADV	SECT curve	$\log_2 \sqrt{ E }$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$	ADV
SECP112R1	55.89	48.34	4.59	51.30	6.90	SECT113R1	56.00	38.06	5.67	50.33	5.73
SECP112R2	54.90	37.54	5.88	49.01	5.51	SECT113R2	56.00	38.17	5.76	50.25	5.65
SECP128R1	64.00	43.45	6.02	57.98	5.58	SECT131R1	65.00	58.75	4.46	60.54	7.24
SECP128R2	63.00	48.23	5.49	57.51	6.11	SECT131R2	65.00	51.57	5.43	59.57	6.27
SECP160K1	80.00	48.39	6.55	73.45	5.45	SECT163K1	81.00	54.56	6.36	74.64	5.64
SECP160R1	80.00	53.85	6.30	73.70	5.70	SECT163R1	81.00	54.69	6.36	74.64	5.64
SECP160R2	80.00	47.53	6.70	73.30	5.30	SECT163R2	81.00	67.16	5.56	75.45	6.45
SECP192K1	96.00	84.31	5.36	90.64	6.84	SECT193R1	96.00	61.74	6.76	89.25	5.45
SECP192R1	96.00	55.51	6.97	89.03	5.23	SECT193R2	96.00	56.08	6.99	89.01	5.21
SECP224R1	112.00	98.50	5.55	106.45	6.85	SECT233K1	115.50	79.89	6.77	108.73	5.73
SECP224K1	—	—	—	—	—	SECT233R1	116.00	77.72	6.92	109.08	5.58
SECP256K1	128.00	86.12	7.00	121.00	5.60	SECT239K1	116.00	79.70	6.87	111.63	5.63
SECP256R1	128.00	86.06	7.00	121.00	5.60	SECT283K1	140.50	94.51	7.15	133.35	5.65
SECP384R1	192.00	141.33	7.33	184.67	5.87	SECT283R1	141.00	94.61	7.18	133.82	5.62
SECP521R1	260.50	196.26	7.67	252.83	6.03	SECT409K1	203.50	150.09	7.44	196.07	5.87
						SECT409R1	204.00	136.70	7.66	196.34	5.64
						SECT571K1	284.50	190.46	8.08	276.41	5.71
						SECT571R1	285.00	190.77	8.15	276.85	5.65

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

Table 2: Summary of results for SECG curves of even characteristic.

Non-SECG curve	$\log_2 \sqrt{ E }$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$
Curve25519	126.00	71.69	7.75	118.25
Curves448	223.00	149.54	8.22	214.78
BN160	79.99	52.54	6.83	73.16
BN192	95.99	64.61	7.02	88.97
BN224	112.00	75.60	7.23	104.77
BN256	128.00	86.16	7.42	120.58
Fp254BNa	126.57	85.32	7.40	119.17
Fp254BNb	126.60	85.37	7.40	119.20
Fp224BN	112.00	75.60	7.23	104.77
Fp256BN	128.00	86.16	7.42	120.58
Fp384BN	192.00	128.75	8.01	183.99
Fp512BN	256.00	171.36	8.42	247.58
BRAINPOOL160R1	79.93	54.10	6.75	73.18
BRAINPOOL192R1	95.80	55.06	7.38	88.42
BRAINPOOL224R1	—	—	—	—
BRAINPOOL256R1	—	—	—	—
BRAINPOOL320R1	—	—	—	—
BRAINPOOL384R1	191.57	128.71	7.99	183.58
BRAINPOOL512R1	255.71	171.39	8.42	247.29

Table 3: Summary of results for non-SECG curves.

5 Conclusion

In this paper, we have presented a DLP to DHP reduction algorithm on any group G of prime order p using $\mathbb{F}_p^\times$ as auxiliary group in the implicit representation method but without using the Chinese Remainder Theorem. Earlier reduction of den Boer (with $\mathbb{F}_p^\times$ as auxiliary group) or Maurer and Wolf (with elliptic curve over $\mathbb{F}_p$) both require the Chinese Remainder Theorem. The application of $\mathbb{F}_p^\times$ as auxiliary group to estimate the lower bound for DHP is a new technique and we also established the advantage of our reduction algorithm over previously known reduction algorithms to achieve better (increased) lower bound for the number of group operations needed to solve DHP. As a consequence of our reduction algorithm, we have presented the *tightest lower bound* known so far on ECDHP for all recommended SECG curves [20] (except SECP224K1). Moreover, we have also extended our idea to non-SECG curves [1, 8, 10, 17] and also provided the similar estimates on those previously uncovered curve parameters. This work is of practical significance as it provides tighter security for protocols which depend on ECDHP for their security. Thus, it leads us towards the computational equivalence of DHP and DLP for these elliptic curves since the gap between the cost of DHP and DLP has been further reduced for these curves.

A Elliptic curve domain parameters over prime field

The following data present several SECG curves [20] which are defined over some prime field of characteristic not equal to 2 and are used for practical purposes. For these curves, the prime p denotes the order of the elliptic curve group and d is the suitable divisor of $p - 1$ which is used by us for various computations in Table 1.

- **SECP112R1**
 $p = 4451685225093714776491891542548933,$
 $d = 140876.$
- **SECP112R2**
 $p = 1112921306273428674967732714786891,$
 $d = 110852811870.$
- **SECP128R1**
 $p = 340282366762482138443322565580356624661,$
 $d = 9476076960994.$
- **SECP128R2**
 $p = 85070591690620534603955721926813660579,$
 $d = 3101689558.$
- **SECP160K1**
 $p = 1461501637330902918203686915170869725397159163571,$
 $d = 42918291593381467397.$
- **SECP160R1**
 $p = 1461501637330902918203687197606826779884643492439,$
 $d = 22167198845997443.$
- **SECP160R2**
 $p = 1461501637330902918203685083571792140653176136043,$
 $d = 142004808588765074419.$

- **SECP192K1**
 $p = 6277101735386680763835789423061264271957123915200845512077,$
 $d = 43818996.$
- **SECP192R1**
 $p = 6277101735386680763835789423176059013767194773182842284081,$
 $d = 9564682313913860059195669.$
- **SECP224K1**
 $p = 26959946667150639794667015087019640346510327083120074548994958668279.$
 Appropriate size of divisor d of $p - 1$ not available.
- **SECP224R1**
 $p = 26959946667150639794667015087019625940457807714424391721682722368061,$
 $d = 533642580.$
- **SECP256K1**
 $p = 11579208923731619542357098500868790785283756427907490438260516$
 $3141518161494337,$
 $d = 65709355417112419152054124.$
- **SECP256R1**
 $p = 1157920892103562487626974469494075735299969552241357603424$
 $22259061068512044369,$
 $d = 71482998987075857096374359.$
- **SECP384R1**
 $p = 3940200619639447921227904010014361380507973927046544666794690527962765939$
 $9113263569398956308152294913554433653942643,$
 $d = 12895580879789762060783039592702.$
- **SECP521R1**
 $p = 6864797660130609714981900799081393217269435300143305409394463459185543183$
 $3976553942450577463332171975329639963713633211138647686124403803403728088$
 $92707005449,$
 $d = 1898873518475180724503002533770555108536.$

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

The following data present several SECG curves [20] which are defined over a binary field and are used for practical purposes. For these curves, the prime p is the largest divisor of the order of that particular elliptic curve group (with a very small co-factor of either 2 or 4) and d is the appropriate divisor of $p - 1$ used by us for various computations in Table 2.

- **SECT113R1**
 $p = 5192296858534827689835882578830703,$
 $d = 253877289037.$

- **SECT113R2**

$p = 5192296858534827702972497909952403,$
 $d = 215851796187.$

- **SECT131R1**

$p = 1361129467683753853893932755685365560653,$
 $d = 23348.$

- **SECT131R2**

$p = 1361129467683753853879535043412812867983,$
 $d = 485524729.$

- **SECT163K1**

$p = 5846006549323611672814741753598448348329118574063,$
 $d = 33118034411893094.$

- **SECT163R1**

$p = 5846006549323611672814738465098798981304420411291,$
 $d = 27744064547201903.$

- **SECT163R2**

$p = 5846006549323611672814742442876390689256843201587,$
 $d = 859825042.$

- **SECT193R1**

$p = 6277101735386680763835789423269548053691575186051040197193,$
 $d = 1697589986603916123127.$

- **SECT193R2**

$p = 6277101735386680763835789423314955362437298222279840143829,$
 $d = 4345632155805272808276901.$

- **SECT233K1**

$p = 3450873173395281893717377931138512760570940988862252126328087024741343,$
 $d = 11064269030135607689238.$

- **SECT233R1**

$p = 6901746346790563787434755862277025555839812737345013555379383634485463,$
 $d = 443484653691663066996649.$

- **SECT239K1**

$p = 220855883097298041197912187592864814948216561321709848887480219215362213,$
 $d = 912013207122974008798076.$

- **SECT283K1**

$p = 388533778445145814183892381364703781328481173379306132429587499752981582$
 $9704422603873,$
 $d = 19578145037471479248182334822.$

- **SECT283R1**

$p = 777067556890291628367784762729407562656962592437690488910919652677004427$
 $7787378692871,$
 $d = 34107744933314238426752172695.$

- **SECT409K1**

$p = 330527984395124299475957654016385519914202341482140609642324395022880711$
 $289249191050673258457777458014096366590617731358671,$
 $d = 572443222870261113609193333057890.$

- **SECT409R1**

$p = 661055968790248598951915308032771039828404682964281219284648798304157774$
 $827374805208143723762179110965979867288366567526771,$
 $d = 133035142307481057108300314154446543724338.$

- **SECT571K1**

$p = 193226876150862917234767594546599367214946366485321749932861762572575957$
 $114478021226813397852270671183470671280082535146127367497406661731192968$
 $2421617092503555733685276673,$
 $d = 1650836032275210526255468059063336914554249497826676631916.$

- **SECT571R1**

$p = 386453752301725834469535189093198734429892732970643499865723525145151914$
 $228956042453614399938941577308313388112192694448624687246281681307023452$
 $8288303332411393191105285703,$
 $d = 2160677396588220552651437946338996605699043277407755096919.$

C CFRG curves

The following two curves are proposed by CFRG [10]. For these curves, p denotes the order of the curve subgroup with the cofactor of 8 and 4 for Curve25519 and Curve448, respectively, and d denotes the suitable divisor of $p - 1$.

- **Curve25519**

$p = 72370055773322622139731865630429942408571163593799076060019509382854$
 $54250989,$
 $d = 198211423230930754013084525763697.$

- **Curve448**

$p = 18170968107390172263733095197200113358841034017182951507037254979514$
 $6003961539585716195755291692375963310293709091662304773755859649779,$
 $d = 671171546397993813511248341018818572482169507.$

D BN curves

The following four prime order curves are given by Barreto and Naehrig [1], especially for the implementation of pairing-based protocols. We are calling them BN curves depending on the size of the field size. For example,

BN160 would mean that field size (as well as the prime order of the curve) is 160 bits. Here, p denotes the prime order of these curves and d denotes the suitable divisor of $p - 1$.

- **BN160**
 $p = 1461501624496790265145447380994971188499300027613,$
 $d = 135993458106516349.$
- **BN192**
 $p = 6277101719531269400517043709981664699904401744160036556389,$
 $d = 31884532810558979926.$
- **BN224**
 $p = 26959946667149205758383469736921690242718878200571531029749235996909,$
 $d = 32941888801002250879962.$
- **BN256**
 $p = 11579208923731493687268856124447174205803559598884026858448875799942,$
 $9535617037,$
 $d = 62315634532338720804070873.$

E More BN curves from CFRG

The following curves are some more pairing friendly prime order BN curves given in [8]. Note that curve Fp224BN is same curve as BN224 and Fp256BN is the same curve as BN256. Again, p denotes the prime order of these curves and d denotes the suitable divisor of $p - 1$.

- **Fp254BNa**
 $p = 16030569034403128277756688287498649515510226217719936227669524443298095$
 $169537,$
 $d = 27393617468666409313669099.$
- **Fp254NBb**
 $p = 16798108731015832284940804142231733909759579603404752749028378864165570$
 $215949,$
 $d = 26696234517031531215864644.$
- **Fp224BN**
 $p = 26959946667149205758383469736921690242718878200571531029749235996909,$
 $d = 32941888801002250879962.$
- **Fp256BN**
 $p = 11579208923731493687268856124447174205803559598884026858448875799942953$
 $5617037,$
 $d = 62315634532338720804070873.$
- **Fp384BN**
 $p = 39402006196394479212278605372068645601647601647604977711035712095477795$
 $040353349547661912204165862202002491141937013,$
 $d = 482712679794742648236472191811900110458.$

- **Fp512BN**

$p = 13407807929942597099574024998205830437246153344875111580494527427714590$
 $09988168005389192020040957072065474214644567793930640846175462664783326$
 $2056300743149,$
 $d = 3633375643565235648064095124859887114667385103908742.$

F Brainpool curves

Following curve parameters given by CFRG are known as Brainpool curves. They are also prime order curves. Again, p denotes the prime order of these curves and d denotes the suitable divisor of $p - 1$.

- **BRAINPOOL160R1**

$p = 1332297598440044874827085038830181364212942568457,$
 $d = 14296469036998443.$

- **BRAINPOOL192R1**

$p = 4781668983906166242955001894269038308119863659119834868929,$
 $d = 13609004849343556497893651.$

- **BRAINPOOL224R1**

$p = 227216229324543527875525379959109236125675463423307571913906560966559.$
 No suitable divisor d exists.

- **BRAINPOOL256R1**

$p = 768849563970453442208097466290016490927375317844145295387555190630635$
 $36359079.$
 No suitable divisor d exists.

- **BRAINPOOL320R1**

$p = 176359332223916635416190984244601952088951277271768606376068612401678$
 $4784845843468355685258203921.$
 No suitable divisor d exists.

- **BRAINPOOL384R1**

$p = 165927077011931617306923684233260497979611638701764860007564527482161$
 $1501358515537962695117368903252229601718723941,$
 $d = 281013799650838810245831691854138883369.$

- **BRAINPOOL512R1**

$p = 894896220765023255165660281515915342216260964409835451134459718720005$
 $701041341852837898173064352495985745139837002928058309421561388204397$
 $3354392115544169,$
 $d = 233393970660364590999999972172975707584270361234257.$

Acknowledgment: I wish to thank my advisor Dr. Ayan Mahalanobis for his continuous help and excellent guidance throughout this project. I also wish to thank anonymous referees for their valuable suggestions.

References

- [1] P. S. L. M. Barreto and M. Naehrig, Pairing-friendly elliptic curves of prime order, in: *Selected Areas in Cryptography*, Lecture Notes in Comput. Sci. 3897, Springer, Berlin (2006), 319–331.
- [2] K. Bentahar, The equivalence between the DHP and DLP for elliptic curves used in practical applications, revisited, in: *Cryptography and Coding*, Lecture Notes in Comput. Sci. 3796, Springer, Berlin (2005), 376–391.
- [3] D. R. L. Brown, CM55: Special prime-field elliptic curves almost optimizing den Boer’s reduction between Diffie–Hellman and discrete logs, preprint (2014), <https://eprint.iacr.org/2014/877>.
- [4] D. R. L. Brown and R. P. Gallant, The static Diffie–Hellman problem, preprint (2014), <https://eprint.iacr.org/2004/306>.
- [5] J. H. Cheon, Security analysis of the strong Diffie–Hellman problem, in: *Advances in Cryptology—EUROCRYPT 2006*, Lecture Notes in Comput. Sci. 4004, Springer, Berlin (2006), 1–11.
- [6] B. den Boer, Diffie–Hellman is as strong as discrete log for certain primes, in: *Advances in Cryptology—CRYPTO’88*, Lecture Notes in Comput. Sci. 403, Springer, Berlin (1990), 530–539.
- [7] J. Hoffstein, J. Pipher and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer Undergrad. Math. Ser., Springer, New York, 2008.
- [8] K. Kasamatsu, S. Kanno, T. Kobayashi and Y. Kawahara, Barreto–Naehrig curves, preprint (2014), <https://tools.ietf.org/html/draft-kasamatsu-bncurves-00>.
- [9] S. Kozaki, T. Kutsuma and K. Matsuo, Remarks on Cheon’s algorithms for pairing-related problems, in: *Pairing-Based Cryptography—Pairing 2007*, Lecture Notes in Comput. Sci. 4575, Springer, Berlin (2007), 302–316.
- [10] A. Langley, M. Hamburg and S. Turner, Elliptic curves for security, preprint (2016), <https://tools.ietf.org/html/rfc7748>.
- [11] U. M. Maurer, Towards the equivalence of breaking the Diffie–Hellman protocol and computing discrete logarithms, in: *Advances in Cryptology—CRYPTO’94*, Lecture Notes in Comput. Sci. 839, Springer, Berlin (1994), 271–281.
- [12] U. M. Maurer and S. Wolf, Diffie–Hellman oracles, in: *Advances in Cryptology—CRYPTO’96*, Lecture Notes in Comput. Sci. 1109, Springer, Berlin (1996), 268–282.
- [13] U. M. Maurer and S. Wolf, On the difficulty of breaking the Diffie–Hellman protocol, Technical Report 24, ETH Zurich, Zurich, 1996.
- [14] U. M. Maurer and S. Wolf, The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms, *SIAM J. Comput.* **28** (1999), no. 5, 1689–1721.
- [15] U. M. Maurer and S. Wolf, The Diffie–Hellman protocol, *Des. Codes Cryptogr.* **19** (2000), no. 2–3, 147–171.
- [16] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, Discrete Math. Appl. (Boca Raton), CRC Press, Boca Raton, 1997.
- [17] J. Merkle and M. Lochter, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, preprint (2010), <https://tools.ietf.org/html/rfc5639>.
- [18] 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** (2004), 50–72.
- [19] FIPS NIST. 186.2 Digital Signature Standard (DSS), National Institute of Standards and Technology (NIST), 2000.
- [20] SECG. SEC 2.: Recommended Elliptic Curve Domain Parameters, 2000, <http://www.secg.org/>.
- [21] X9 ANSI. 62: Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American National Standards Institut, 1999.