Int J Inf Secur (2004) 3: 86–98 / Digital Object Identifier (DOI) 10.1007/s10207-004-0045-9

The efficiency of solving multiple discrete logarithm problems and the implications for the security of fixed elliptic curves*

Yvonne Hitchcock[1,**], **Paul Montague**[2], **Gary Carter**[3], **Ed Dawson**[1]

[1] Information Security Research Centre, Queensland University of Technology, GPO Box 2434, Brisbane Q 4001, Australia
e-mail: {y.hitchcock,e.dawson}@qut.edu.au
[2] Motorola Australia Software Centre, 2 Second Ave, Mawson Lakes, SA 5095, Australia
e-mail: pmontagu@asc.corp.mot.com
[3] School of Mathematics, Queensland University of Technology, GPO Box 2434, Brisbane Q 4001, Australia
e-mail: g.carter@qut.edu.au

Published online: 5 November 2004 –

Abstract. This paper examines the cryptographic security of fixed versus random elliptic curves over $GF(p)$. It assumes a precomputation for use in breaking the elliptic curve discrete logarithm problem (ECDLP) can be made for fixed curves. A lower bound for the efficiency of a variation of Pollard's rho method for solving multiple ECDLPs is presented, as well as an approximation of the expected time remaining to solve an ECDLP when a given size of precomputation is available. We conclude that adding 4 bits to the order of a fixed curve to avoid general software attacks plus 6 bits to avoid attacks on curves with special properties provides equivalent security.

Keywords: Elliptic curve discrete logarithm problem (ECDLP) – Baby-step giant-step method (BSGS) – Pollard's rho method – Fixed elliptic curve – Precomputation

1 Introduction

Elliptic curves were first proposed as a basis for public key cryptography in the mid 1980s independently by Koblitz and Miller. Elliptic curves provide a discrete-log-based public key cryptosystem and can use a much shorter key length than other public key cryptosystems to provide an equivalent level of security. Elliptic curve cryptosystems (ECCs) can also provide a faster implementation than RSA or discrete log (DL) systems and use less bandwidth and power [5]. These issues can be crucial in lightweight applications such as smart cards. In the last few years, ECCs have been included or proposed for inclusion in internationally recognized standards (specifically IEEE 1363, WAP (Wireless Application Protocol), ANSI X9.62, ANSI X9.63 and ISO CD 14888-3). Thus ECCs are set to become an integral part of lightweight applications in the immediate future.

One drawback of an ECC is the complexity of generating a secure elliptic curve. The complexity is high enough to render it infeasible to generate a randomly chosen but secure elliptic curve on a mobile device (e.g. telephones, PDAs and smart cards) due to the time, memory and code size required to count the points on the curve and ensure that other security requirements [1, Section V.7] are met. For example, the Schoof-Elkies-Atkin (SEA) point counting algorithm is the best known point counting algorithm for a randomly chosen EC over $GF(p)$ and has complexity $O\left(\log^8(p)\right)$ [1]. It has been implemented in the MIRACL library in conjunction with Pollard's lambda method and takes 2–3 min on a 180-MHz Pentium Pro to count the points on a 160-bit curve and 3.5–5.5 min for a 192-bit curve [14]. On a smart card platform, it would take much longer – a 10-MHz smart card could be expected to take at least 36 min to count the points on a 160-bit curve based on processor speed. However, it is likely that code size and memory limitations would preclude the algorithm from being programmed onto such a smart card in the first place.

Even if a mobile device could generate a secure elliptic curve, there would still be other costs, such as the bandwidth required to transmit the curve to other parties. The cost of transmitting a curve over $GF(p)$ is that of transmitting four numbers modulo p. These numbers are the two curve constants a and b as well as the base point and the mobile device's public key in compressed format. Added to the bandwidth and time costs, there is also the cost of a substantially increased code size associated with generating a curve on the mobile device.

* An abridged version of this paper appeared as "The Security of Fixed versus Random Elliptic Curves in Cryptography" in ACISP 2003 [6].

** *Corresponding author*: Address as above, tel.: +61-7-3864-9570, fax: +61-7-3221-2384

On the other hand, if a fixed curve is used, we know that it is feasible to implement an associated ECC on a mobile device since various implementations have been reported (for example the implementation in [5]). When using a fixed curve, the mobile device is only required to generate a secret key using a random number generator and to transmit the corresponding public key to the other parties. The random number generator is needed in any case by some signature algorithms, and the scalar multiplication routine required to generate the public key will already be available for use in the protocols utilized by the mobile device. Therefore, any extra code associated with key selection is minimal when using a fixed curve. Other advantages of using fixed curves include being able to choose special parameters to increase the efficiency of the implementation (Sect. 3.3) and a reduced bandwidth requirement since only one number modulo p (the mobile device's compressed public key) must be transmitted to other parties. The fact that the fixed curve parameters are required to be publicly available is not a disadvantage when compared with random curves because the curve parameters of random curves must also be made public before the curve can be used. While these issues may not be major for all mobile devices (e.g. in some applications the random curve could be generated by a server and bandwidth usage might not be a problem), the difficulties associated with using random curves have caused various standards organizations to include fixed curves in their standards, such as the WAP curves [19] and the NIST curves [11].

Whilst a fixed curve may be an attractive option for efficiency reasons, it also offers a single target for people all over the world to attack. On the other hand, if random curves are utilized, there are many more curves in use throughout the world, so that a group of attackers no longer has one target, but many targets to attack. The random curves used may also be constantly changed, making the number of possible targets to attack even greater. Furthermore, attacking one curve will not give the attackers any advantage if they wish to attack a different curve at a later date. Thus the computational power deployed to break a fixed curve is likely to be much greater than that deployed to break a random curve. In addition to this, if a fixed curve is broken, all users of that curve are affected. On the other hand, if a random curve used by a small number of people is broken, the overall impact is much smaller than if a fixed curve used by many people all over the world is broken.

Given the above observations, it would appear intuitively obvious that using a random curve provides a higher level of security than a fixed curve. However, exactly how much extra security a random curve provides and whether the amount of extra security is significant is much less clear. To date, there have been no publications examining whether the decision to use a fixed curve compromises the security of a cryptosystem or the significance of any such compromise. This issue is therefore investigated here in detail.

The discussion is restricted to curves over the field $GF(p)$ where p is a large prime. These curves consist of the set of points (x, y) satisfying the equation $y^2 \equiv x^3 + ax + b \pmod{p}$ where a and b are constants such that $4a^3 + 27b^2 \not\equiv 0 \pmod{}p$. The paper firstly examines existing methods of software attack and their impact on fixed-curve security, including a variant of Pollard's rho method, which can be used to break more than one ECDLP on the one curve. We then present a lower bound on the expected number of iterations required to solve a subsequent ECDLP using this method, as well as an approximation for the number of remaining iterations to solve an ECDLP when a given number of iterations have already been performed. We also investigate threats from hardware attacks and optimizations for curves with special properties. Finally, recommendations are made for the size increase required for a fixed curve to have an equivalent security level to a random curve.

2 Existing methods of attack

In this section the efficiencies of different methods available to attack the ECDLP are examined. Only those attacks applicable to arbitrary elliptic curves are considered. These attacks are then used in the following section to analyse the security of fixed curves compared to random curves. Attacks such as differential side channel analysis that pose an equal threat to both fixed and random curves are not discussed.

2.1 Pohlig–Hellman algorithm

The Pohlig–Hellman [12] algorithm breaks the ECDLP down into several different ECDLPs, one in each prime-order subgroup of the elliptic curve group. Obviously, the hardest one of these to solve is in the subgroup of largest prime order, and thus the attack is resisted by requiring the order of this subgroup to be at least 160 bits [1, p. 98]. We assume for the remainder of this analysis that (if applicable) the Pohlig–Hellman algorithm has been used to reduce the ECDLP to an ECDLP in the subgroup of largest prime order.

2.2 Index calculus and related methods

There are currently no index calculus or related methods applicable to elliptic curves. Indeed, it is believed to be unlikely that such attacks will ever be possible [7]. Therefore, these methods are not considered further here.

2.3 Shanks's baby-step giant-step method

The baby-step giant-step (BSGS) method of Shanks [16] has a precomputation for each curve. A balanced version is often given in the literature (e.g. [1]). We give an un-

balanced version below which takes advantage of the fact that the negative of an elliptic curve point can be calculated "for free", in a similar manner to Shanks's original proposal. Let n, Q, z, m, R and d be defined as follows:

$$\begin{aligned} n &= \text{The prime order of the base point } P \\ Q &= \text{The point whose ECDL is to be found} \\ z &= \text{The value of the ECDLP. That is, } Q = [z]P \\ m &= \text{Number of points in the precomputation} \\ d &= \left\lceil \frac{n}{2m} \right\rceil \\ R &= [d]P \end{aligned}$$

Then the precomputation of giant steps can be calculated as:

$$S_\alpha = [\alpha]\, R \quad \text{for } 0 \le \alpha < m$$

and the ECDLP can be solved by finding the baby steps:

$$R_\beta = Q - [\beta]\, P \quad \text{for } 0 \le \beta < d$$

until an R_β value is found which is the same as S_α or $-S_\alpha$ for some α. The solution to the ECDLP is then:

$$\begin{aligned} & z = \alpha\, d + \beta \quad \text{if } R_\beta = S_\alpha \\ \text{or} \quad & z = n - \alpha\, d + \beta \quad \text{if } R_\beta = -S_\alpha\,. \end{aligned}$$

There are approximately m elliptic curve additions required in the precomputation and on average $\frac{d}{2}$ further elliptic curve additions required to solve the ECDLP. Thus, on average, approximately $\frac{4m^2+n}{4m}$ operations are required to solve one ECDLP. This value is at its minimum of $\sqrt{n}$ operations when $m \approx \frac{\sqrt{n}}{2}$.

2.4 Pollard's rho method

Pollard's rho method [13] is currently the best method known for solving the general ECDLP [20]. The method searches for a collision in a pseudo-random walk through the points on the curve. If the iterating function defining the pseudo-random walk is independent of the point whose discrete logarithm is to be found, then the same calculations can be used to find more than one discrete logarithm on the one curve. Kuhn and Struik [8] provide an analysis of the expected running time of such a method, which is described as follows. Let the definitions in Table 1 be given. The pseudo-random walk function to solve the kth ECDLP, $g(R_{k,i})$, is defined to be as follows:

$$g(R_{k,i}) = [h_{f(R_{k,i})}]R_{k,i} + [c_{f(R_{k,i})}]P\,,$$

where h_j and c_j are constants. Note that the next value in the pseudo-random walk to solve the kth ECDLP $R_{k,i+1}$, is determined only by P and $R_{k,i}$, not P, Q_k and $R_{k,i}$. In order to maximize efficiency, h_j should be set to 1 for all but one of the possible values of j, in which case h_j should be set to 2 and c_j should be set to zero. If this is done, each iteration of the method will require only one elliptic curve addition. This random walk is similar to a special case of the "combined walk" proposed by Teske [17]. We note that currently there is no proof that the above random walk is sufficiently random for the theoretical results (which assume the randomness of the walk) to hold. However, it differs from the random walk with such a proof proposed by Teske [17] in approximately $1/s$ cases where s is the number of equations defining the pseudo-random walk and $s \approx 20$ gives optimal performance [17]. Since the random walk proposed here differs from Teske's random walk in only about $1/20$ cases, it is expected to perform randomly enough.

Table 1. Definitions for Pollard's rho method

$n =$	The prime order of the base point P.
$Q_k =$	The points whose ECDLs are to be found. That is, $Q_k = [z_k]P$, where we wish to find z_k for $k \ge 0$.
$R_{k,0} =$	$[u_{k,0}]P + [w_{k,0}]Q_k$, where $u_{k,0}$ and $w_{k,0}$ are randomly chosen constants and $w_{k,0} \neq 0$.
$R_{k,i} =$	The ith point in the pseudo-random walk to solve the ECDLP for Q_k. Note that $R_{k,i} = [u_{k,i}]P + [w_{k,i}]Q_k$.
$s =$	The number of equations defining the pseudo-random walk.
$f(R) =$	A function mapping a point R to a number between 1 and s.
$g(R_{k,i}) =$	A function returning the next value in the pseudo-random walk, $R_{k,i+1}$. It is defined as: $g(R_{k,i}) = [h_{f(R_{k,i})}]R_{k,i} + [c_{f(R_{k,i})}]P$, where c_j and h_j are constants for $1 \le j \le s$.
$u_{k,i+1} \equiv$	$h_{f(R_{k,i})}u_{k,i} + c_{f(R_{k,i})} \pmod n \quad \text{for } 0 \le i\,.$
$w_{k,i+1} \equiv$	$h_{f(R_{k,i})}w_{k,i} \pmod n \quad \text{for } 0 \le i\,.$

There are two different types of collisions which can occur, a collision with a point on the current pseudo-random walk and a collision with a point on a previous pseudo-random walk. They can be solved as follows:

$$\begin{aligned} \text{If } R_{k,i} &= R_{k,j} \\ \text{then } [u_{k,i}]P + [w_{k,i}]Q_k &= [u_{k,j}]P + [w_{k,j}]Q_k \\ \text{with a solution of } Q_k &= \left[\frac{u_{k,j} - u_{k,i}}{w_{k,i} - w_{k,j}}\right] P\,. \end{aligned}$$

$$\begin{aligned} \text{Otherwise, } R_{k,i} &= R_{l,j} \\ \text{where } R_{l,j} &= [u_{l,j}]P + [w_{l,j}]Q_l \\ \text{and } Q_l &= [z_l]P\,. \\ \text{Therefore, } [u_{k,i}]P + [w_{k,i}]Q_k &= [u_{l,j}]P + [w_{l,j}z_l]P \\ \text{with a solution of } Q_k &= \left[\frac{u_{l,j} + w_{l,j}z_l - u_{k,i}}{w_{k,i}}\right] P\,. \end{aligned}$$

In order to detect collisions, we need to store the points on the pseudo-random walk $R_{k,i}$ and compare the current point on the random walk with previous points.

However, in order to save storage space, $R_{k,i}$, $u_{k,i}$ and $w_{k,i}$ are stored only if $R_{k,i}$ is a *distinguished point*. Distinguished points are defined as those points with some easily checkable property (such as having ten leading zeros in the x coordinate).

Because only distinguished points are stored, we will not always detect a collision as soon as it occurs, but rather at the next distinguished point on the pseudo-random walk. Because $R_{k,i+1}$ only depends on $R_{k,i}$ and P, once a collision occurs between $R_{k,i}$ and $R_{l,j}$ we know that $R_{k,i+m} = R_{l,j+m}$ for all $m \geq 0$. For this reason, a collision is guaranteed to be detected at the next distinguished point.

We emphasize that if distinguished points are used, the random walk definition *must* be independent of the values $Q_1, Q_2, \ldots$ in order for the results in this section to hold. We note that the particular random walk recommended by Kuhn and Struik in [8] (originally recommended by Teske [17]) to solve a single ECDLP should not be used to solve multiple ECDLPs when using distinguished points because it must depend on Q_i in order to be useful. Although Kuhn and Struik provide an analysis of the time required to solve multiple ECDLPs, they do not specify a suitable random walk to use in this situation. The problem with the random walk depending on any Q_i is that a different random walk must be used for each ECDLP to be solved. This in turn means that any collisions of non-distinguished points from different random walks are not detected because the random walks take different paths after the collision. Only collisions of distinguished points are detected in this case. Of course, if all points are distinguished then the random walk may depend on Q_i since all collisions are detected. However, in most practical situations, not all points will be distinguished. Figure 1 shows the results of using a different random walk for each ECDLP compared to using the one random walk described above. It is easily seen that while some advantage is gained from the distinguished points from previous (different) random walks, that advantage is quite small compared to the advantage gained if only one random walk is used.

We now wish to know how much of an improvement previous calculations can offer to the speed with which the solution to a subsequent ECDLP is found.

Let $Z_i =$ The number of iterations needed to solve the ith ECDLP after the $(i-1)$th ECDLP's solution.

$T_i =$ The total number of iterations to solve the first i ECDLPs.

Note: $T_i \geq i+1$.

Obviously, the expected value of Z_1, $\mathrm{E}(Z_1)$, is the same as that of the traditional Pollard's rho method, namely [17]:

$$\mathrm{E}(T_1) = \mathrm{E}(Z_1) \approx \sqrt{\frac{\pi n}{2}}.$$

We note that Wiener and Zuccherato [20] have been able to improve this figure by a factor of $\sqrt{2}$ by restricting the random walk to points with distinct x coordinates. For simplicity, we have not included this optimization in the description of Pollard's rho method in this section. However, by changing n to $n/2$ in the following discussion, its effect can be taken into consideration. In Sect. 3 this optimization is taken into account in the calculations performed.

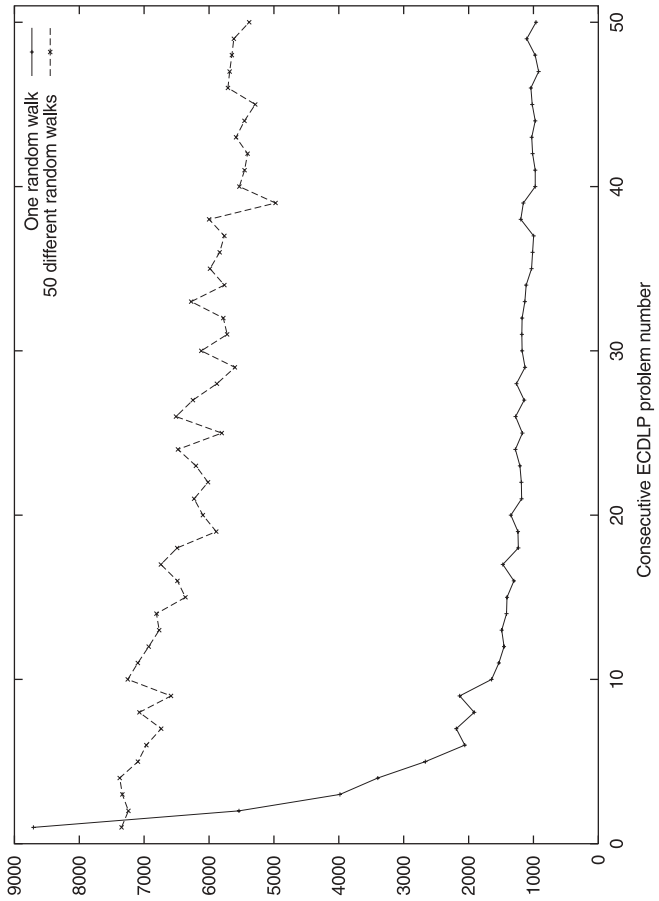

Fig. 1. Actual iterations to solve 50 ECDLPs on a 25-bit curve averaged over 200 trials with 1 in 400 points distinguished and $s = 5$

It is also possible to parallelize Pollard's rho algorithm [18] to obtain a linear speedup. However, it is not necessary to include such parallelization directly in the model since it can be taken into account by increasing the speed at which calculations can be made. For example, a parallelized version running on five computers each at speed x will complete in the same time as a non-parallelized version on a single computer running at speed $5x$. We therefore account for any increase in speed due to parallelization by setting the speed at which computations are performed to an appropriate value.

We now wish to find $\mathrm{E}(T_i)$ and $\mathrm{E}(Z_i)$ for $i > 1$. Kuhn and Struik [8] provide an approximation for the expected value of Z_{i+1} as:

$$\mathrm{E}(Z_{i+1}) \approx \sqrt{\frac{\pi n}{2}} \binom{2i}{i} \frac{1}{4^i} \quad \text{for } i \ll n^{\frac{1}{4}} \tag{1}$$

and the expected value of T_{i+1} as:

$$\mathrm{E}(T_{i+1}) \approx \sqrt{\frac{\pi n}{2}} \sum_{t=0}^{i-1} \frac{\binom{2t}{t}}{4^t} \quad \text{for } i \ll n^{\frac{1}{4}} \tag{2}$$

$$\approx 2\sqrt{\frac{i}{\pi}}\mathrm{E}(Z_1)\,. \tag{3}$$

This paper proves a new result which gives a lower bound on the expected values above. The lower bound is quite similar to the approximations above but replaces $\sqrt{\frac{\pi n}{2}}$ with $E(Z_1)$. This is stated formally in the following theorem:

Theorem 1. *Let Z_i and T_i be defined as above. Then the following inequalities hold:*

$$\mathrm{E}(Z_{i+1}) \geq \binom{2i}{i}\frac{1}{4^i}\mathrm{E}(Z_1) \quad \textit{for } i \geq 1\,, \tag{4}$$

$$\mathrm{E}(Z_{i+1}) \geq \frac{1}{2i}\mathrm{E}(T_i) \quad \textit{for } i \geq 1\,. \tag{5}$$

Substituting the first few values of i into (4) gives:

$$\mathrm{E}(Z_2) \geq \frac{1}{2}\mathrm{E}(Z_1)\,,$$
$$\mathrm{E}(Z_3) \geq \frac{3}{4}\cdot\frac{1}{2}\mathrm{E}(Z_1)\,,$$
$$\mathrm{E}(Z_4) \geq \frac{5}{6}\cdot\frac{3}{4}\cdot\frac{1}{2}\mathrm{E}(Z_1)\,.$$

The proof of Theorem 1 is dependent on the following lemma, whose proof is provided in Appendix A:

Lemma 1.

$$\sum_{k_1=1}^{t-\alpha}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_\alpha=1}^{k_{\alpha-1}}\frac{k_\alpha}{n} \geq \frac{t(t-\alpha)}{2\alpha n}\sum_{k_1=1}^{t-\alpha}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\frac{k_{\alpha-1}}{n} \tag{6}$$

It is now possible to prove Theorem 1 as follows:

Proof. In order to find $\mathrm{E}(T_i)$ and $\mathrm{E}(Z_i)$ for $i > 1$ we start with the formulae given in [18], where W_i may be T_i or Z_i:

$$\Pr(T_1 > t) = \prod_{j=1}^{t-1}\left(1-\frac{j}{n}\right)$$

$$\mathrm{E}(W_i) = \sum_{j=1}^{\infty} j\cdot\Pr(W_i = j), \tag{7}$$

$$= \sum_{j=1}^{\infty} j\cdot(\Pr(W_i > j-1) - \Pr(W_i > j))$$

$$= \sum_{j=0}^{\infty}\Pr(W_i > j)\,. \tag{8}$$

We now give $\Pr(T_i = t)$ and $\Pr(T_{i+1} > t)$:

$$\Pr(T_i = t) = \frac{t-i}{n}\left(\prod_{k=1}^{t-i-1}\left(1-\frac{k}{n}\right)\right)\sum_{k_1=1}^{t-i}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_{i-1}=1}^{k_{i-2}}\frac{k_{i-1}}{n}, \tag{9}$$

$$\Pr(T_{i+1} > t) = \Pr(T_i > t) + \Pr(T_{i+1} > t \text{ and } T_i \leq t)$$
$$= \Pr(T_i > t) + \left(\sum_{k_1=1}^{t-i}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_i=1}^{k_{i-1}}\frac{k_i}{n}\right)\prod_{k=1}^{t-i-1}\left(1-\frac{k}{n}\right)\,. \tag{10}$$

Using (7) and (9), we derive an expression for $\mathrm{E}(T_i)$:

$$\mathrm{E}(T_i) = \sum_{t=i+1}^{n+i} t\frac{t-i}{n}\left(\sum_{k_1=1}^{t-i}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_{i-1}=1}^{k_{i-2}}\frac{k_{i-1}}{n}\right)\cdot \prod_{k=1}^{t-i-1}\left(1-\frac{k}{n}\right)$$
$$= (i+1)\left(\frac{1}{n}\right)^i + \sum_{t=i+2}^{n+i} t\frac{t-i}{n}\left(\sum_{k_1=1}^{t-i}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_{i-1}=1}^{k_{i-2}}\frac{k_{i-1}}{n}\right)\cdot \prod_{k=1}^{t-i-1}\left(1-\frac{k}{n}\right) \tag{11}$$

We now derive an expression for $\mathrm{E}(T_{i+1})$ from (8) and (10):

$$\mathrm{E}(T_{i+1}) = \sum_{t=0}^{\infty}\Pr(T_{i+1} > t)$$
$$= 2 + i + \sum_{t=i+2}^{n+i}\Pr(T_{i+1} > t)$$
$$= 2 + i + \sum_{t=i+2}^{n+i}\Pr(T_i > t) + \sum_{t=i+2}^{n+i}\left(\sum_{k_1=1}^{t-i}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_i=1}^{k_{i-1}}\frac{k_i}{n}\right)\prod_{k=1}^{t-i-1}\left(1-\frac{k}{n}\right)$$

$$
\begin{aligned}
&\mathrm{E}(T_{i+1}) \\
&= \mathrm{E}(T_i) + 1 - \Pr(T_i > i+1) + \\
&\quad \sum_{t=i+2}^{n+i} \left(\sum_{k_1=1}^{t-i} \frac{k_1}{n} \sum_{k_2=1}^{k_1} \frac{k_2}{n} \cdots \sum_{k_i=1}^{k_{i-1}} \frac{k_i}{n} \right) \prod_{k=1}^{t-i-1} \left(1 - \frac{k}{n}\right) \\
&= \mathrm{E}(T_i) + \left(\frac{1}{n}\right)^i + \\
&\quad \sum_{t=i+2}^{n+i} \left(\sum_{k_1=1}^{t-i} \frac{k_1}{n} \sum_{k_2=1}^{k_1} \frac{k_2}{n} \cdots \sum_{k_i=1}^{k_{i-1}} \frac{k_i}{n} \right) \prod_{k=1}^{t-i-1} \left(1 - \frac{k}{n}\right) . \qquad (12)
\end{aligned}
$$

We then substitute into (12) the inequality of Lemma 1, which is given again in (13):

$$
\begin{aligned}
&\sum_{k_1=1}^{t-\alpha} \frac{k_1}{n} \sum_{k_2=1}^{k_1} \frac{k_2}{n} \cdots \sum_{k_\alpha=1}^{k_{\alpha-1}} \frac{k_\alpha}{n} \\
&\geq \frac{t(t-\alpha)}{2\alpha n} \sum_{k_1=1}^{t-\alpha} \frac{k_1}{n} \sum_{k_2=1}^{k_1} \frac{k_2}{n} \cdots \sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} \frac{k_{\alpha-1}}{n} \qquad (13)
\end{aligned}
$$

$$
\begin{aligned}
&\therefore \mathrm{E}(T_{i+1}) \\
&\geq \mathrm{E}(T_i) + \left(\frac{1}{n}\right)^i + \\
&\quad \sum_{t=i+2}^{n+i} \frac{t(t-i)}{2ni} \left(\sum_{k_1=1}^{t-i} \frac{k_1}{n} \sum_{k_2=1}^{k_1} \frac{k_2}{n} \cdots \sum_{k_{i-1}=1}^{k_{i-2}} \frac{k_{i-1}}{n} \right) \cdot \\
&\qquad \prod_{k=1}^{t-i-1} \left(1 - \frac{k}{n}\right) \qquad (14)
\end{aligned}
$$

and substituting (11) into (14) gives:

$$
\begin{aligned}
\mathrm{E}(T_{i+1}) &\geq \mathrm{E}(T_i) + \frac{1}{2i}\mathrm{E}(T_i) \qquad (15) \\
&= \frac{2i+1}{2i}\mathrm{E}(T_i) \\
\therefore \mathrm{E}(T_{i+1}) &\geq \prod_{k=1}^{i} \left(\frac{2k+1}{2k}\right) \mathrm{E}(T_1) .
\end{aligned}
$$

Similarly,

$$
\mathrm{E}(T_i) \geq \prod_{k=1}^{i-1} \left(\frac{2k+1}{2k}\right) \mathrm{E}(T_1) \qquad (16)
$$

and note that

$$
\mathrm{E}(T_{i+1}) = \mathrm{E}(Z_{i+1}) + \mathrm{E}(T_i) . \qquad (17)
$$

By substituting (17) into (15) and then (16) into (18) we obtain:

$$
\begin{aligned}
\mathrm{E}(Z_{i+1}) + \mathrm{E}(T_i) &\geq \mathrm{E}(T_i) + \frac{1}{2i}\mathrm{E}(T_i), \qquad (18) \\
\mathrm{E}(Z_{i+1}) &\geq \frac{1}{2i}\mathrm{E}(T_i), \qquad (19) \\
\mathrm{E}(Z_{i+1}) &\geq \frac{1}{2i} \prod_{k=1}^{i-1} \left(\frac{2k+1}{2k}\right) \mathrm{E}(T_1) \\
&= \prod_{k=1}^{i} \left(\frac{2k-1}{2k}\right) \mathrm{E}(Z_1) .
\end{aligned}
$$

That is, we have:

$$
\begin{aligned}
&\mathrm{E}(Z_2) \geq \frac{1}{2}\mathrm{E}(Z_1), \\
&\mathrm{E}(Z_3) \geq \frac{3}{4} \cdot \frac{1}{2}\mathrm{E}(Z_1), \\
&\mathrm{E}(Z_4) \geq \frac{5}{6} \cdot \frac{3}{4} \cdot \frac{1}{2}\mathrm{E}(Z_1), \\
&\mathrm{E}(Z_i) \geq \frac{(2\,(i-1))!}{2^{2(i-1)}\left((i-1)!\right)^2}\mathrm{E}(Z_1) \quad \text{for} i \geq 2 , \qquad (20) \\
&\mathrm{E}(Z_{i+1}) \geq \binom{2i}{i} \frac{1}{4^i}\mathrm{E}(Z_1) \quad \text{for } i \geq 1 . \qquad (21)
\end{aligned}
$$

Equations (21) and (19) complete the proof of (4) and (5) respectively. □

It is easily seen that the left- and right-hand sides of Lemma 1 and (13) are equal when $\alpha = 1$, since they give the formula for the sum of numbers from 1 to $t-1$ in that case. Therefore, in the case of $i = 1$ in the above proof, the greater than or equal to signs can be replaced with an equals sign in (14), (15) and (21), giving the results:

$$
\begin{aligned}
\mathrm{E}(Z_2) &= \frac{1}{2}\mathrm{E}(Z_1) \\
\text{and } \mathrm{E}(T_2) &= \frac{3}{2}\mathrm{E}(Z_1) .
\end{aligned}
$$

In order to study the behaviour of $\mathrm{E}(Z_i)$ more easily, (20) can be approximated using Stirling's formula, which states [4, p. 373]:

$$
n! \approx (2\pi)^{\frac{1}{2}}\, n^{n+\frac{1}{2}} e^{-n} \quad \text{for large } n . \qquad (22)
$$

Substituting (22) where possible into (20), we obtain:

$$
\mathrm{E}(Z_i) \geq \frac{1}{\sqrt{\pi i}}\mathrm{E}(Z_1) \quad \text{for large } i . \qquad (23)
$$

The above results lead us to expect that the second ECDLP can be solved in half the time of the first, the third ECDLP can be solved in no less than three-eighths the time of the first, and so on. As stated in [8], (1) and (2) are good approximations. We provide experimental evidence of this in Fig. 2, which shows the actual number of iterations to solve 50 ECDLPs on a 32-bit curve averaged over 200 trials, as well as the bound in (4). Note that since

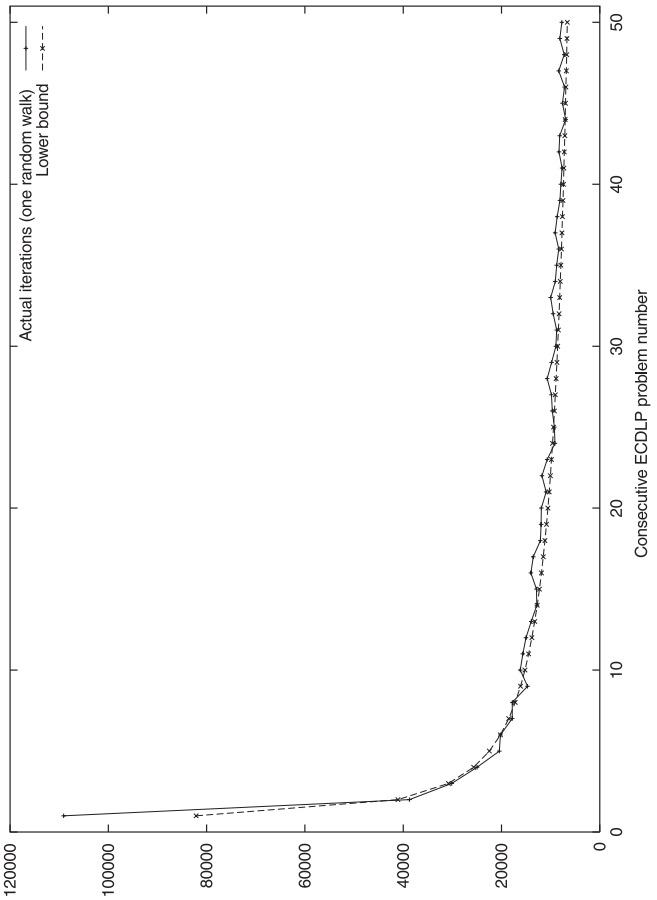

Fig. 2. Actual iterations to solve 50 ECDLPs on a 32-bit curve, averaged over 200 trials, with $s = 21$ and 1 in 800 points distinguished. The theoretical bound from Eq. (4) is also shown

$\mathrm{E}(Z_1)$ has been taken to be $\sqrt{\frac{\pi n}{2}}$, the bound is the same as the expected value provided in (1).

We also use $\mathrm{E}(Z_i \mid r$ previous iterations) in our analysis in the next section. An approximation is given by Theorem 2:

Theorem 2. *Let Z_i and n be defined as above and let r be the number of iterations previously performed. Then:*

$$\mathrm{E}(Z_i \mid r \text{ prev. iters.}) \approx \sqrt{\frac{\pi n}{2}} e^{\frac{r^2}{2n}} \left(1 - \Phi\left(\frac{r}{\sqrt{2n}}\right)\right) \quad (24)$$
$$\text{where } \Phi(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt\,.$$

Proof. Suppose that during a random walk to solve previous ECDLPs, as defined in Table 1, r iterations have been performed. Then for Z_i, the number of iterations to solve a new ECDLP,

$$\Pr(Z_i > z) = \left(1 - \frac{r}{n}\right)\left(1 - \frac{r+1}{n}\right) \cdots \left(1 - \frac{r+z-1}{n}\right),$$

and taking logarithms we have:

$$\begin{aligned}&\ln\left(\Pr\left(Z_i > z\right)\right)\\ &= \ln\left(1 - \frac{r}{n}\right) + \ln\left(1 - \frac{r+1}{n}\right) + \cdots\\ &\quad + \ln\left(1 - \frac{r+z-1}{n}\right). \end{aligned} \quad (25)$$

Now we can expand $\ln(1+x)$ using a Maclaurin series as:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots.$$

Using this expansion and assuming $r,\, z \leq \sqrt{n}$, we can approximate (25) as:

$$\begin{aligned}\ln\left(\Pr\left(Z_i > z\right)\right) &\approx -\sum_{j=r}^{r+z-1} \frac{j}{n} - \frac{1}{2} \sum_{j=r}^{r+z-1} \frac{j^2}{n^2} - \cdots\\ &\approx -\frac{z\,(2r+z-1)}{2n} - O\left(\frac{1}{\sqrt{n}}\right)\\ \therefore \Pr\left(Z_i > z\right) &\approx e^{-\frac{z}{2n}(z+2r-1)}. \end{aligned} \quad (26)$$

Substituting (26) into (8) produces:

$$\begin{aligned}&\mathrm{E}(Z_i)\\ &\approx \sum_{z=0}^{\infty} e^{-\frac{z}{2n}(z+2r-1)}\\ &\approx \int_0^{\infty} e^{-\frac{z}{2n}(z+2r-1)} dz\\ &= \int_{r-\frac{1}{2}}^{\infty} e^{-\frac{1}{2n}\left(y^2 - \left(r-\frac{1}{2}\right)^2\right)} dy \quad \text{where } y = z + r - \frac{1}{2}\\ &= e^{\frac{1}{2n}\left(r-\frac{1}{2}\right)^2} \int_{r-\frac{1}{2}}^{\infty} e^{-\frac{y^2}{2n}} dy\\ &= e^{\frac{1}{2n}\left(r-\frac{1}{2}\right)^2} \sqrt{2n} \int_{\frac{1}{\sqrt{2n}}\left(r-\frac{1}{2}\right)}^{\infty} e^{-t^2} dt \quad \text{where } t = \frac{y}{\sqrt{2n}}. \end{aligned} \quad (27)$$

Now the error function $\Phi(x)$ is defined as [3, p. 938]:

$$\Phi(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

and from [3, p. 354],

$$\begin{aligned}&\int_0^{\infty} e^{-t^2} dt = \frac{\sqrt{\pi}}{2}\\ &\therefore 1 - \Phi(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt\,. \end{aligned} \quad (28)$$

Substituting (28) into (27) gives:

$$\begin{aligned}\mathrm{E}(Z_i) &\approx e^{\frac{1}{2n}\left(r-\frac{1}{2}\right)^2} \frac{\sqrt{2\pi n}}{2} \left(1 - \Phi\left(\frac{r-\frac{1}{2}}{\sqrt{2n}}\right)\right)\\ &\approx e^{\frac{r^2}{2n}} \sqrt{\frac{\pi n}{2}} \left(1 - \Phi\left(\frac{r}{\sqrt{2n}}\right)\right). \end{aligned} \quad (29)$$

This completes the proof. □

If we take into account the optimization of Wiener and Zuccherato [20], (24) becomes:

$$\mathrm{E}(Z_i \mid r \text{ prev. iters.}) \approx \frac{\sqrt{\pi n}}{2} e^{\frac{r^2}{n}} \left(1 - \Phi\left(\frac{r}{\sqrt{n}}\right)\right) \qquad (30)$$

with $\Phi(x)$ defined earlier.

3 Security comparison

This section contains a detailed comparison of the security of fixed versus random curves. Firstly, it examines *"equivalent security"* by assuming that a precomputation can be made to aid in solving the ECDLP on a fixed curve. All other attributes of the curves are assumed to be equal for this comparison. Secondly, special properties of curves which may give a greater amount of speed to an ECC implementation, and are therefore likely to be used in fixed curves, are examined in relation to the ECDLP. Unfortunately, such properties also make it faster to find an ECDL, and this issue is addressed. Thirdly, special-purpose hardware is considered, and finally all of the results are combined into an overall recommendation.

Throughout this section, we use n_{F} and n_{R} to denote the orders of the prime (sub)groups of the fixed and random curves respectively, where both curves have an "equivalent" level of security.

3.1 Equivalent Security I

In order to compare the security of fixed and random curves, we need a definition of equivalence of security. We give one possible definition below:

Definition 1 (Equivalent Security I). *Assume that there are 2^ν users of a fixed curve (for example, 2^{32} users), and that if these users moved to random curves, there would be 2^ω users per curve, and a total of $2^{\nu-\omega}$ random curves in use. Then we say that the fixed curve has* equivalent security I (ES-I) *to the random curves if it takes the same expected number of computations to break the* ECDLP *for all users of the fixed curve as it does to break the* ECDLP *for all users of the random curves.*

3.1.1 BSGS and ES-I

In order to apply the ES-I definition to attacks using the BSGS method, we assume that there are 2^ν users of a fixed curve and that if random curves were used, each user would have a different random curve (i.e. $\omega = 0$). In the case of the fixed curve, the precomputation need be performed only once. Therefore, the total amount of computation to find all ECDLs is $V \approx m_{\mathrm{F}} + 2^\nu \frac{n_{\mathrm{F}}}{4m_{\mathrm{F}}}$, where m_{F} is the size of the precomputation. We find that V is at its minimum value of $V_{\min} \approx 2^{\nu/2}\sqrt{n_{\mathrm{F}}}$ for any value of n_{F} when $m_{\mathrm{F}} \approx 2^{(\nu/2)-1}\sqrt{n_{\mathrm{F}}}$.

We now set $V_{\min}$ equal to the minimum time taken to solve all ECDLPs on the random curves (using Pollard's rho method with the optimization of Wiener and Zuccherato since it is fastest) and solve for n_{F}:

$$V_{\min} = 2^{\nu/2}\sqrt{n_{\mathrm{F}}} = 2^\nu \sqrt{\pi n_{\mathrm{R}}}/2$$
$$n_{\mathrm{F}} \approx 2^\nu n_{\mathrm{R}} .$$

Therefore, we need to increase the order of a fixed curve by ν bits to satisfy ES-I under the assumption of a BSGS attack.

3.1.2 Pollard's rho and ES-I

Using the definition for ES-I and the result in (3), if the fixed and random curves have the same order, then when $(\nu - \omega)$ is sufficiently large, it takes approximately $2^{\frac{\nu-\omega}{2}}$ times as long to solve all of the ECDLPs on the random curves as it does on the fixed curve. Therefore, to decide how much bigger the fixed curve should be than the random curves for an equivalent level of security, we set:

$$\frac{\sqrt{\pi n_{\mathrm{F}}}}{2} = 2^{\frac{\nu-\omega}{2}} \frac{\sqrt{\pi n_{\mathrm{R}}}}{2} .$$

That is, $n_{\mathrm{F}} = 2^{\nu-\omega} n_{\mathrm{R}}$.

3.1.3 ES-I result

By setting $\nu = 32$ (for the approximate number of people in the world) and $\omega = 0$ (every user has his/her own random curve) we can give a very conservative worst-case estimate that about 32 bits should be added to the curve order of a fixed curve compared to a random curve. This then ensures that the number of group operations required to solve all ECDLPs on the fixed curve is equivalent to the number required to solve all ECDLPs on random curves.

3.2 Equivalent Security II

The analysis using ES-I is based on the assumption that we must break all users' ECDLPs on a fixed curve to break the cryptosystem. In practice, we actually find it unacceptable for even one ECDLP on the fixed curve to be broken and therefore present a new definition of *"equivalent security"*. The definition is based on the observation that a curve becomes insecure at the same time as the first ECDLP is able to be solved on that curve. Therefore, for a fixed curve, assume that a precomputation with the number of iterations equal to $\mathrm{E}(Z_1) \approx \frac{\sqrt{\pi n}}{2}$ is possible, since this is the maximum number of iterations that can be performed before the curve becomes insecure. A fixed curve with an equivalent size can then be found using the following definition.

Definition 2 (Equivalent Security II). *The approximation of $\mathrm{E}(Z_1) \approx \frac{\sqrt{\pi n}}{2}$ can be taken as the number of iterations used to create a precomputation after the fixed*

curve has been released. This value can be used to find how many iterations are expected to remain to solve the first ECDLP*. A calculation can then be made to determine how many extra bits need to be added to the order of the fixed curve to ensure that the first* ECDLP *on the fixed curve (using a precomputation) is as hard as an* ECDLP *on a random curve. We define a fixed curve and a random curve with curve orders satisfying these conditions to have* equivalent security II (ES-II)*.*

This definition will give a better estimate of the required increase in curve order than that given by using the definition of ES-I. However, we stress that there is really no significant improvement in the time to solve the first ECDLP on a fixed curve compared to a random curve if the precomputation time is included. This has been shown by Kuhn and Struik [8], who prove the following bounds on the time to solve one out of k ECDLPs (denoted as $Z_{\mathrm{DLP}(1:k)}$):

$$\begin{aligned} \mathrm{E}(Z_1) - k &\leq \mathrm{E}(Z_{\mathrm{DLP}(1:k)}) \\ \mathrm{E}(Z_{\mathrm{DLP}(1:k)}) &\leq \mathrm{E}(Z_1)\,. \end{aligned}$$

Since k is much less than $\mathrm{E}(Z_1)$, we can approximate the value $\mathrm{E}(Z_{\mathrm{DLP}(1:k)})$ with $\mathrm{E}(Z_1)$.

We ignore the precomputation time when calculating ES-II because there is a greater incentive to break ECDLPs on a fixed curve than a random curve (this is because more ECDLPs may be solved for the same effort). Again, we note that if it is feasible to solve one ECDLP on a fixed curve of a particular size, then it is also possible to solve one ECDLP on a random curve of the same size in the same manner.

If the BSGS method is used, the number of iterations, m, in the precomputation is:

$$m = \frac{\sqrt{\pi n}}{2}\,.$$

For the BSGS method, any ECDL can then be found using an average of

$$\frac{n}{4m} = \frac{n}{2\sqrt{\pi n}} = \frac{1}{2}\sqrt{\frac{n}{\pi}}$$

iterations. We then set this equivalent to the number of iterations required to solve an ECDLP on a random curve with a curve order of unknown size using Pollard's rho method (since it is faster than the BSGS method):

$$\begin{aligned} \frac{1}{2}\sqrt{\frac{n}{\pi}} &= \sqrt{\pi n_{\mathrm{R}}}/2 \\ \frac{n}{\pi} &= \pi n_{\mathrm{R}} \\ n_{\mathrm{R}} &= \frac{n}{\pi^2} \\ &\approx \frac{n}{2^{3.3}}\,. \end{aligned}$$

Therefore about four bits[1] of security have been lost if the BSGS method is used to solve the ECDLP on the fixed curve. Note that while the time for each iteration on the fixed curve will be slightly different from that required on the random curve, because the curves are close in size, the effect is minimal.

For Pollard's rho method, we substitute the size of the precomputation ($\frac{\sqrt{\pi n}}{2}$) as r in (30) to find how many iterations are expected to remain to solve the first ECDLP using a precomputation:

$$\begin{aligned} \mathrm{E}(Z_1|\text{precomputation}) &\approx \frac{\sqrt{\pi n}}{2}\left(e^{\frac{\pi n}{4n}}\right)\left(1-\Phi\left(\sqrt{\frac{\pi n}{4n}}\right)\right) \\ &= \frac{\sqrt{\pi n}}{4}\left(e^{\frac{\pi}{2}}\right)\left(1-\Phi\left(\frac{\sqrt{\pi}}{2}\right)\right) \\ &\approx 0.461\frac{\sqrt{\pi n}}{2}\,. \end{aligned}$$

We then ascertain the number of bits in the order of a random curve which has equivalent security to the fixed curve (as defined by ES-II) as follows:

$$\begin{aligned} 0.461\frac{\sqrt{\pi n}}{2} &\approx \frac{\sqrt{\pi n_{\mathrm{R}}}}{2} \\ n_{\mathrm{R}} &\approx 0.461^2 n \\ &\approx \frac{n}{2^{2.236}}\,. \end{aligned}$$

Therefore, in this case, using Pollard's rho method, less than 3 bits of security are lost, as opposed to 4 bits using the BSGS method. However, because (30) is an approximation, the Pollard's rho figure for the loss in security is also an approximation. However, its correctness is supported by the fact that when no precomputation is used, $\mathrm{E}(Z_2) \approx \frac{1}{2}\mathrm{E}(Z_1) \approx \frac{\sqrt{\pi n}}{4}$ from (1). Therefore, if the size of the precomputation was exactly Z (not $\mathrm{E}(Z)$), we would set

$$\begin{aligned} \frac{\sqrt{\pi n}}{4} &\approx \frac{\sqrt{\pi n_{\mathrm{R}}}}{2} \\ n_{\mathrm{R}} &\approx \frac{n}{4} \end{aligned}$$

and find that only 2 bits of security had been lost, which is close to the 2.236-bit figure above.

Given that Pollard's rho method is generally accepted as being more efficient than the BSGS method on average, it may at first seem surprising that a greater loss of fixed-curve security occurs due to the BSGS method than to Pollard's rho method. This apparent contradiction can be explained by observing that the calculations assume that the first ECDLP to be solved is not known until after the calculation of the precomputation. If the first ECDLP had been known before the precomputation for Pollard's rho method began, the precomputation could be directed at

[1] A figure of five bits was previously published in [6] by using a different method of calculation. The discrepancy is probably due to rounding error introduced in [6] by using figures from the table in [9].

solving that particular ECDLP, and it would be expected to be solved at about the same time as the precomputation finished. This is not surprising since the curve becomes insecure at the same time as this occurs. Although the final result on fixed-curve security hinges on the efficiency of the BSGS method, the new results for Pollard's rho method are still necessary to the analysis because they enable us to show that this is in fact the case.

We conclude that, given the above assumptions, adding about 4 bits to the order of a fixed curve compared to a random curve will give approximately the same level of security as defined by ES-II when attacks are performed using the BSGS or Pollard's rho method.

3.3 Curves with special properties

Standardized fixed curves often have special properties in order to increase the speed at which ECCs can operate. For example, the curves over $GF(p)$ specified by NIST [11] use primes for which fast modular reduction algorithms are available as well as setting the a parameter of the elliptic curve to -3 for faster point addition algorithms. On the other hand, the WAP specification [19] sometimes sets the a parameter to 0 for the same reason. Unfortunately, these settings mean that attacks on these curves can be performed at a faster rate also.

Using one of the generalized Mersenne primes specified by NIST as the modulus for the curve can make an implementation up to 3.7 times as fast, based on figures from [2]. Even if the modular reduction took no time at all, this would only lead to an implementation up to 4.5 times as fast. Therefore, based on the relationship between the curve order and the complexity of the BSGS and Pollard's rho methods, adding $2 \cdot \log_2(4.5) = 4.34$ bits to the curve order would overcome this problem.

Using an a parameter of -3 can reduce the number of squares and multiplies required for a point addition from 10 to 8, so that addition is about 1.25 times as fast. Using an a parameter of 0 can reduce the number of squares and multiplies to 7 instead of 10, so that addition is about 1.43 times as fast. Increasing the curve order by $2 \cdot \log_2(1.43) = 1.03$ bits would overcome problems due to special values for the a parameter.

To avoid having any attacker obtain an advantage when attacking a fixed curve with a special modulus or a parameter, we suggest that an increase of 5.4 bits $(1.03 + 4.34)$ would provide a more than adequate level of security.

3.4 Special-purpose hardware

It is possible to build special-purpose hardware to attack elliptic curve cryptosystems which would be considerably faster than a software attack using equipment of the same value. Lenstra and Verheul [9, Section 3.2.5] provide an analysis of the difference in cost between hardware and software implementations and conclude that in the elliptic curve case, for curves over the field $GF(p)$, software is more than $2000 \approx 2^{11}$ times more expensive than hardware.

As another example, an MPC190 security processor from Motorola [10] running at 66 MHz can perform 1000 Internet key exchanges (IKE) on a 155-bit elliptic curve per second. Therefore, one scalar multiplication on such a device takes less than 1 ms. A Pentium IV 1.8-GHz machine can compute one scalar multiplication on a 160-bit curve in 2.66 ms, or about $73 \approx 2^{6.2}$ times slower taking into account the processor speed.

While some would suggest that extra bits should be added to fixed curves over $GF(p)$ to resist attacks due to special-purpose hardware, we argue that hardware availability constitutes an equal threat to both fixed and random curves. Hardware such as the MPC190 security processor is able to perform calculations for any elliptic curve, not just a single curve. If attackers are able to invest in hardware to attack fixed curves, then that hardware can just as easily be used to attack random curves. While those who see hardware as a greater threat to fixed curves than random curves may suggest adding some extra bits to the fixed curve (22 bits based on the estimation of Lenstra and Verheul or 13–15 bits based on the MPC190 speed), we believe that such an action is unnecessary, given the equal susceptibility of fixed and random curves to hardware attacks.

3.5 Results of analysis and performance effects

By combining the results of the previous subsections we can determine how many extra bits should be added to a fixed curve for security equivalent to a random curve. Very conservative users may wish to add a total of 38 bits for curves over $GF(p)$ (6 bits for special curve attacks and 32 bits to achieve ES-I). However, we believe a more realistic approach is to add approximately 10 bits for curves over $GF(p)$ (being 4 bits to achieve ES-II and 6 bits for special curve attacks). Of course, if the fixed curve has neither a special modulus nor a special a parameter, the 6 bits for special curve attacks need not be added, and in that case only an extra 4 bits are required for the fixed curve.

While adding extra bits to a fixed curve does increase the time required to perform elliptic curve operations on such curves, the increase is still small enough for fixed curves to be attractive. For comparison, Table 2 shows timings using the MIRACL library [15] for a single scalar multiplication on both a fixed and a random curve. In all cases, the Comba optimization from the MIRACL library has been used which unravels and reorganizes the programme loops implicit in the field multiplication and reduction processes. The curves recommended by NIST [11] were used as the fixed curves. These curves have an a parameter of -3 and a generalized Mersenne number as the modulus, allowing a fast modular reduction algorithm. The table shows that a fixed curve with these properties is still faster than a random curve 32 bits smaller than it.

Table 2. Time in milliseconds of elliptic curve scalar multiplication using the MIRACL library [15] on a Pentium IV 1.8 GHz

Bit size	Random curve	Fixed curve
160	2.66	
192	4.46	2.19
224	6.76	3.34

Therefore, if fixed curves take advantage of the availability of special moduli and parameters, the 10 extra bits we recommend adding to the order of a fixed curve will not have any serious impact on performance compared to random curves. In fact, the fixed curve may be faster than a random curve implementation not using these special features but with an equivalent level of security. We note that in practice 32 (rather than 10) extra bits are likely to be added to make the modulus size a multiple of the word size of the processor being used, but that this does not change our conclusion.

4 Conclusion

We have analysed the ECDLP on a fixed versus a random curve over $GF(p)$ and found that if the order of the fixed curve with special properties is 10 bits larger than that of the random curve, an equivalent level of security is achieved if previously published attacks are used to solve the ECDLP.

We have given a lower bound on the expected value to solve more than one ECDLP on the one curve using Pollard's rho method, given an approximation of the expected time to solve an ECDLP using Pollard's rho method on the assumption that a precomputation of a certain size already exists, and proposed two definitions of *"equivalent security"*. We have given examples which support the conclusion that approximately 32 bits should be added to the order of a fixed curve to have equivalent security to that of a random curve using the first definition, ES-I, but only 4 bits need to be added using the preferred definition, ES-II.

Attacks taking advantage of special-purpose hardware have been considered, but it was concluded that special-purpose hardware forms an equal threat to both fixed and random curves, implying that this attack does not require the order of fixed curves to be increased compared to random curves. Also, attacks taking advantage of fixed curves using a special modulus or a parameter have been investigated and a recommendation made to add 6 bits to the order of the fixed curve to resist these attacks.

Taking all attacks into consideration, we recommend adding 10 bits to the order of a fixed curve compared to a random curve, being 6 bits to resist attacks due to a special modulus or a parameter and 4 bits to achieve ES-II. However, if the fixed curve does not have a special modulus or a parameter, the addition of only 4 bits to the curve order is necessary. These results show that there is no security problem associated with the use of fixed curves, provided the order of the fixed curve is increased by a small amount. Such an increase in the size of the fixed curve has a minimal performance impact, whilst allowing realization of the many benefits associated with the use of fixed curves.

Acknowledgements. This research is part of an ARC SPIRT project (C10024103) undertaken jointly by Queensland University of Technology and Motorola.

References

1. Blake I, Seroussi G, Smart N (1999) Elliptic curves in cryptography. London Mathematical Society Lecture Note Series, vol 265. Cambridge University Press, Cambridge
2. Brown M, Hankerson D, López J, Menezes A (2001) Software implementation of the NIST elliptic curves over prime fields. In: Topics in Cryptology – CT-RSA 2001. Lecture notes in computer science, vol 2020. Springer, Berlin Heidelberg New York, pp 250–265
3. Gradshteyn IS, Ryzhik IM (1994) Table of integrals, series, and products, 5th edn. Academic, San Diego
4. Greenspan HP, Benny DJ (1973) Calculus: an introduction to applied mathematics. McGraw-Hill Kogakusha, Tokyo, International student edition
5. Hasegawa T, Nakajima J, Matsui M (1998) A practical implementation of elliptic curve cryptosystems over $GF(p)$ on a 16-bit microcomputer. In: Practice and Theory in Public Key Cryptography – PKC '98. Lecture notes in computer science, vol 1431. Springer, Berlin Heidelberg New York, pp 182–194
6. Hitchcock Y, Montague P, Carter G, Dawson E (2003) The security of fixed versus random elliptic curves in cryptography. In: Australasian Conference on Information Security and Privacy – ACISP 2003. Lecture notes in computer science, vol 2727. Springer, Berlin Heidelberg New York, pp 55–66
7. Huang M-DA, Kueh KL, Tan K-S (2000) Lifting elliptic curves and solving the elliptic curve discrete logarithm problem. In: Proceedings of Algorithmic Number Theory: 4th international symposium – ANTS-IV 2000. Lecture notes in computer science, vol 1838. Springer, Berlin Heidelberg New York, pp 377–384
8. Kuhn F, Struik R (2001) Random walks revisited: extensions of Pollard's rho algorithm for computing multiple discrete logarithms. In: Selected Areas in Cryptography – SAC 2001. Lecture notes in computer science, vol 2259. Springer, Berlin Heidelberg New York, pp 212–229
9. Lenstra AK, Verheul ER (2001) Selecting cryptographic key sizes. J Cryptol 14(4):255–293
10. Motorola Inc (2003) MPC190: Security processor, 1994–2003. [Online] `http://e-www.motorola.com/webapp/sps/site/prod_summary.jsp?code=MPC190&nodeId=01DFTQ42497721` [accessed 13/02/2003]
11. NIST (National Institute of Standards and Technology), US Department of Commerce (2001) FIPS 186-2, digital signature standard (DSS). Federal Information Processing Standard (FIPS), January 2000. [Online] `http://www.csrc.nist.gov/publications/fips/` [accessed 07/06/2001]
12. Pohlig SC, Hellman ME (1978) An imporved algortihm for computing logarithms in $GF(p)$ and its cryptographic significance. IEEE Trans Inf Theory 24(1):106–111
13. Pollard JM (1978) *Monte Carlo* methods for index computation (mod p). Math Comput 32(143):918–924
14. Scott M (1999) Comments in the file sea.cpp which implements the Schoof-Elkies-Atkin algorithm for the Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL). Shamus Software Ltd. [Online] `ftp://ftp.computing.dcu.ie/pub/crypto/sea.cpp` [accessed 04/06/2003]

15. Shamus Software Ltd (2000) Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL). [Online] http://indigo.ie/~mscott/ [accessed 23/6/2000]
16. Shanks D (1971) Class number: a theory of factorization, and genera. In: Proceedings of Symposia in Pure Mathematics 1969 Number Theory Institute, vol XX. AMS, Providence, RI, pp 415–440
17. Teske E (1998) Speeding up pollard's rho method for computing discrete logarithms. In: Proceedings of Algorithmic Number Theory: 3rd international symposium – ANTS-III 1998. Lecture notes in computer science, vol 1423. Springer, Berlin Heidelberg New York, pp 541–554
18. van Oorschot PC, Wiener MJ (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
19. WAP (Wireless Application Protocol Forum Ltd) (2001) Wireless application protocol: wireless transport layer security. [Online] http://www1.wapforum.org/tech/terms.asp?doc=WAP-261-WTLS-20010406-a.pdf [accessed 31/07/2002]
20. Wiener MJ, Zuccherato RJ (1999) Faster attacks on elliptic curve cryptosystems. In: Selected Areas in Cryptography – SAC '98. Lecture notes in computer science, vol 1556. Springer, Berlin Heidelberg New York, pp 190–200

A Proof of Lemma 1

This appendix provides the proof of Lemma 1:

Lemma 1.

$$\sum_{k_1=1}^{t-\alpha}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_\alpha=1}^{k_{\alpha-1}}\frac{k_\alpha}{n} \geq \frac{t(t-\alpha)}{2\alpha n}\sum_{k_1=1}^{t-\alpha}\frac{k_1}{n}\sum_{k_2=1}^{k_1}\frac{k_2}{n}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\frac{k_{\alpha-1}}{n} \tag{31}$$

Proof. With a change of variables, (6) can be rewritten as:

$$\sum_{k_1=1}^{c}\sum_{k_2=1}^{k_1}\cdots\sum_{k_\beta=1}^{k_{\beta-1}} k_1k_2\ldots k_\beta \geq \frac{c(c+\beta)}{2\beta}\sum_{k_1=1}^{c}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\beta-1}=1}^{k_{\beta-2}} k_1k_2\ldots k_{\beta-1}\,.$$

By subtracting $\left(c\sum_{k_1=1}^{c}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\beta-1}=1}^{k_{\beta-2}} k_1k_2\ldots k_{\beta-1}\right)$ from both sides we obtain:

$$\sum_{k_1=1}^{c-1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_\beta=1}^{k_{\beta-1}} k_1k_2\ldots k_\beta \geq \frac{c(c-\beta)}{2\beta}\sum_{k_1=1}^{c}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\beta-1}=1}^{k_{\beta-2}} k_1k_2\ldots k_{\beta-1}\,. \tag{32}$$

We now proceed to prove (32) by using two nested proofs by induction. Firstly, we prove by induction on β, and to complete the proof, we need to use induction on c for a specific value of β.

Firstly, we prove (32) true for $\beta=2,\ c\geq 1$. When $\beta=2$, we have:

$$\begin{aligned}\sum_{k_1=1}^{c-1}\sum_{k_2=1}^{k_1} k_1k_2 &= \sum_{k_1=1}^{c-1}\frac{{k_1}^2(k_1+1)}{2}\\ &= \frac{1}{2}\left(\sum_{k_1=1}^{c-1}{k_1}^3+\sum_{k_1=1}^{c-1}k_1^2\right)\\ &= \frac{1}{2}\left(\frac{c^2(c-1)^2}{4}+\frac{c(c-1)(2c-1)}{6}\right)\\ &= \frac{c(3c-2)(c-1)(c+1)}{24}\\ &= \frac{c(c-\frac{2}{3})(c-1)(c+1)}{8}\\ &\geq \frac{c^2(c-2)(c+1)}{8}\\ &= \frac{c(c-2)}{4}\sum_{k_1=1}^{c}k_1\,,\end{aligned}$$

which completes the proof.

We now assume (32) true for $\beta=\alpha-1$ and $c\geq 1$ and prove that this implies (32) is true for $\beta=\alpha$. In particular, assume (32) for $\beta=\alpha-1$ and $c=n+1$, as in (33):

$$\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1} \geq \frac{(n+1)(n+2-\alpha)}{2(\alpha-1)}\sum_{k_1=1}^{n+1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-2}=1}^{k_{\alpha-3}} k_1k_2\ldots k_{\alpha-2}\,. \tag{33}$$

That is, when $n+2>\alpha$ we assume:

$$\sum_{k_1=1}^{n+1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-2}=1}^{k_{\alpha-3}} k_1k_2\ldots k_{\alpha-2} \leq \frac{2(\alpha-1)}{(n+1)(n-\alpha+2)}\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}\,. \tag{34}$$

We are now able to proceed to prove:

$$\sum_{k_1=1}^{c-1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_\alpha=1}^{k_{\alpha-1}} k_1k_2\ldots k_\alpha \geq \frac{c(c-\alpha)}{2\alpha}\sum_{k_1=1}^{c}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}\,, \tag{35}$$

which is obviously true when $\alpha\geq c\geq 1$, since the LHS ≥ 0 and the RHS ≤ 0. We prove (35) when $c>\alpha$ by induction. We let the "first" case be $c=\alpha-1$. This is obviously true since $\alpha\geq c$. We now assume that (35) is true for some value n of c such that $n\geq\alpha-1$. That is, $n+2>\alpha$.

Substituting n for c in (35) gives (36), which we assume is true:

$$\sum_{k_1=1}^{n-1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_\alpha=1}^{k_{\alpha-1}} k_1k_2\ldots k_\alpha$$
$$\geq \frac{n(n-\alpha)}{2\alpha}\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}, \qquad (36)$$
$$\therefore \sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_\alpha=1}^{k_{\alpha-1}} k_1k_2\ldots k_\alpha$$
$$\geq n\left(\frac{n-\alpha}{2\alpha}+1\right)\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}$$
$$= \frac{n(n+\alpha)}{2\alpha}\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}. \qquad (37)$$

We now proceed to prove (35) for $c=n+1$ given (36). That is, we proceed to prove:

$$\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_\alpha=1}^{k_{\alpha-1}} k_1k_2\ldots k_\alpha$$
$$\geq \frac{(n+1)(n+1-\alpha)}{2\alpha}\sum_{k_1=1}^{n+1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}. \qquad (38)$$

From (37), (38) is true if:

$$n(n+\alpha)\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}$$
$$\geq (n+1)(n+1-\alpha)\sum_{k_1=1}^{n+1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}. \qquad (39)$$

Substituting (34) into the (rearranged) RHS of (39) gives:

$$(n+1)(n+1-\alpha)\sum_{k_1=1}^{n+1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}$$
$$= (n+1)(n+1-\alpha)\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\prod_{i=1}^{\alpha-1} k_i$$
$$+ (n+1)^2(n+1-\alpha)\sum_{k_1=1}^{n+1}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-2}=1}^{k_{\alpha-3}}\prod_{i=1}^{\alpha-2} k_i$$
$$\leq (n+1)(n+1-\alpha)\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\prod_{i=1}^{\alpha-1} k_i$$
$$+ \frac{2(\alpha-1)(n+1)^2(n+1-\alpha)}{(n+1)(n-\alpha+2)}\cdot \left(\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\prod_{i=1}^{\alpha-1} k_i\right)$$
$$= \frac{(n+1)(n+1-\alpha)(n+\alpha)}{n+2-\alpha}\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\prod_{i=1}^{\alpha-1} k_i$$
$$\stackrel{\text{def}}{=} Q.$$

Now LHS of (39) $\geq Q \Longrightarrow$ LHS of (39) $\geq$ RHS of (39). Therefore, in order to prove LHS of (39) $\geq$ RHS of (39), we prove LHS of (39) $\geq Q$, or LHS of (39) $- Q \geq 0$, from which the desired result follows immediately:

LHS of (39) $- Q$
$$= n(n+\alpha)\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\prod_{i=1}^{\alpha-1} k_i$$
$$- \frac{(n+1)(n+1-\alpha)(n+\alpha)}{n+2-\alpha}\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}}\prod_{i=1}^{\alpha-1} k_i$$
$$= \frac{(n+\alpha)(\alpha-1)}{n+2-\alpha}\sum_{k_1=1}^{n}\sum_{k_2=1}^{k_1}\cdots\sum_{k_{\alpha-1}=1}^{k_{\alpha-2}} k_1k_2\ldots k_{\alpha-1}$$
$$\stackrel{\text{def}}{=} R.$$

Now $R \geq 0$ when $n+2>\alpha$. Thus we have proved that (39) is true when $n+2>\alpha$, and hence (38) is true given $n+2>\alpha$, (36) and (34).

Since given (34) is true and (35) is true for $c=n$ and $n+2\geq\alpha$ implies that (35) is true for $c=n+1$ and since (35) is obviously true for $1\leq c\leq\alpha$, (35) is true for all $c\geq 1$ given (34).

Since (32) is true for $\beta=2$ and (32) is true for $\beta-1$ implies (32) is true for β, (32) is true for all $\beta\geq 2$. This completes the proof. □