POINT COMPRESSION FOR KOBLITZ ELLIPTIC CURVES

P. N. J. EAGLE, STEVEN D. GALBRAITH, AND J. ONG

Abstract. Elliptic curves over finite fields have applications in public key cryptography. A Koblitz curve is an elliptic curve E over $\mathbb{F}_2$; the group $E(\mathbb{F}_{2^n})$ has convenient features for efficient implementation of elliptic curve cryptography.

Wiener and Zuccherato and Gallant, Lambert and Vanstone showed that one can accelerate the Pollard rho algorithm for the discrete logarithm problem on Koblitz curves. This implies that when using Koblitz curves, one has a lower security per bit than when using general elliptic curves defined over the same field. Hence for a fixed security level, systems using Koblitz curves require slightly more bandwidth.

We present a method to reduce this bandwidth when a normal basis representation for $\mathbb{F}_{2^n}$ is used. Our method is appropriate for applications such as Diffie-Hellman key exchange or Elgamal encryption. We show that, with a low probability of failure, our method gives the expected bandwidth for a given security level.

Keywords: Elliptic curve cryptography, Koblitz curves, point compression.

AMS Classifications: 94A60 11T71

1. Introduction

Let E be an elliptic curve over $\mathbb{F}_q$. The elliptic curve discrete logarithm problem (ECDLP) is: Given a point $P \in E(\mathbb{F}_q)$ of large prime order ℓ and $Q \in \langle P \rangle$, find an integer $0 \leq a < \ell$ such that $Q = [a]P$. Pollard [8] gave algorithms to solve the DLP in a generic group of prime order ℓ using pseudorandom walks. Van Oorschot and Wiener [12] showed how to use distinguished points so that the DLP can be solved in close to the expected $\sqrt{\pi\ell/2}$ group operations.

A Koblitz curve is an ordinary elliptic curve $E : y^2 + xy = x^3 + a_2x^2 + 1$ over $\mathbb{F}_2$. We consider the group $E(\mathbb{F}_{2^n})$ where n is prime. For certain values of n one obtains group orders of the form $c \cdot \ell$ where c is a small even cofactor and ℓ is a large prime. Koblitz [5] demonstrated the performance benefits of using the group $E(\mathbb{F}_{2^n})$. The advantage is that point multiplication can be accelerated using the 2-power Frobenius map $\psi(x, y) = (x^2, y^2)$ (see Solinas [11] for more details). Throughout the paper we assume that $\ell^2 \nmid \#E(\mathbb{F}_{2^n})$ so that if $P \in E(\mathbb{F}_{2^n})[\ell]$ then $\psi(P) \in \langle P \rangle$.

Wiener and Zuccherato [13] and Gallant, Lambert and Vanstone [2] showed that one can accelerate the Pollard rho method using equivalence classes. For general elliptic curves one can always use the equivalence relation $P \equiv -P$ (i.e., the equivalence classes $\{P, -P\}$) and therefore solve the ECDLP in a group of order ℓ in expected $\sqrt{\pi\ell/4}$ group operations. For Koblitz curves one can define the equivalence relation

$$P \equiv \pm\psi^i(P)$$

for $0 \leq i < n$. The equivalence classes are of size $2n$. It follows that one can solve the ECDLP in a subgroup of $E(\mathbb{F}_{2^n})$ of order ℓ in expected $\sqrt{\pi\ell/4n}$ group operations.

Definition 1. *An elliptic curve E over $\mathbb{F}_q$ has a k–*bit security level *if the expected running time of the Pollard rho algorithm to solve the ECDLP is 2^k group operations*[1]*. For convenience we often round k to the nearest integer.*

When working modulo a large prime q one can find elliptic curves E over $\mathbb{F}_q$ such that $\#E(\mathbb{F}_q) = \ell$ is prime. It follows that one has k-bit security level when $\sqrt{\pi l/4} \approx 0.886\sqrt{q} > 2^k$. This certainly holds if $q \geq 2^{2k+1}$. When working with ordinary elliptic curves over $\mathbb{F}_{2^n}$ (this is the setting of our paper) the group order is always even so one hopes for $\#E(\mathbb{F}_{2^n}) = 2 \cdot \ell$ where ℓ is prime. It is not proven that there exist elliptic curves over every field $\mathbb{F}_{2^n}$ whose number of points is twice a prime, but this conjecture is widely believed. For k-bit security, by the argument above, one therefore takes $n \geq 2k + 2$.

[1]Some authors might define ‘security level’ to be 2^k bit operations, or 2^k operations of the AES function.

For Koblitz curves over $\mathbb{F}_{2^n}$, due to the equivalence classes, one needs $n \geq 2k + 2 + \log_2(n) \geq 2k + 2 + \log_2(2k)$. It follows that when using Koblitz curves one needs more bits of storage and communication compared with general elliptic curves. An alternative way to view this is that we have lower security per bit than when using general curves over the same field. A natural problem is to achieve bandwidth for Koblitz curves which matches the case of more general curves. This problem does not seem to have been considered previously in the literature.

For a given security level k there may not be a prime n close to $2k+2+\log_2(n)$ such that $\#E(\mathbb{F}_{2^n}) = 2 \cdot \ell$ for a large prime ℓ. If the next useful value for n is much larger than $2k+2+\log_2(n)$ then one may prefer not use Koblitz curves and our methods give no improvement (though one might be able to use subfield curves over a different field, e.g., $\mathbb{F}_{2^2}$ and obtain a very small amount of compression using our approach).

Given a point $P = (x_P, y_P) \in E(\mathbb{F}_q)$ one can transmit P by sending x_P and a single bit to determine y_P. As pointed out by Miller [7], in many applications it is possible to ignore y_P altogether and perform cryptography using the equivalence classes $\{P, -P\}$. This often goes under the name of elliptic curve cryptography using x-coordinates only. In other words, when using elliptic curves over prime fields one usually expects $2k + 1$ bits or $\lceil (2k + 1)/8 \rceil$ bytes bandwidth for k bits of security.

When using elliptic curves over $\mathbb{F}_{2^n}$ there is further potential for compression. From now on we assume that a normal basis representation for $\mathbb{F}_{2^n}$ is used. Alternatively, one could convert to normal basis representation just for the compression process. Seroussi [10] gave a method to save one bit of the x-coordinate (we recall the details in §4.1). It follows that one can transmit $P \in E(\mathbb{F}_{2^n})$ using n bits, or $n - 1$ bits if one ignores y_P and works with the equivalence classes $\{P, -P\}$. The minimal bandwidth for elliptic curve cryptography over $\mathbb{F}_{2^n}$ for security level k is therefore $2k + 1$ bits or $\lceil (2k + 1)/8 \rceil$ bytes. When using Koblitz curves this becomes $n = 2k + 1 + \log_2(n)$ bits or $\lceil (2k + 1 + \log_2(n))/8 \rceil$ bytes.

King [4] gives an alternative method to compress points on elliptic curves which applies when $\mathrm{Tr}(a_2) = 0$. His method requires $n - 1$ bits to send (x, y) (i.e., allowing unique recovery of the y-coordinate). While this is better than Seroussi (who needs n bits to send (x, y)) it is not better for our application since we discard y.

The main result of this paper is to give a method to reduce the bandwidth when using Koblitz curves and normal basis representation for finite fields. Our method is a generalisation of working with the equivalence class $\{P, -P\}$. Recall that the reason for the overhead is that one can attack the system using the Pollard rho algorithm on a set of equivalence classes

$$[P] = \{\pm\psi^i(P) : 0 \leq i < n\}.$$

Hence it is natural to do cryptography using these equivalence classes. We show that, with a low probability of failure, one can obtain the desired bandwidth for certain applications. We deal with the $\pm$ by discarding y_P. The equivalence class is then determined by the n-bit string representing x_P (with respect to a normal basis for $\mathbb{F}_{2^n}$) up to rotation. The idea is to rotate this binary string so that a certain pattern of bits appears at one end (our proposal looks for a pattern of the form `011`$\cdots$`110`). This pattern can then be deleted and just the remaining string sent. An extra bit is saved by using the Seroussi trick.

The plan of the paper is as follows. Section 2 makes the notion of bandwidth overhead more precise and gives some targets to achieve. Section 3 explains how to perform Diffie-Hellman key exchange on equivalence classes. Section 4 gives a technical description and justification of the method. Section 5 analyses how well the method is expected to work in practice. Section 6 mentions some other ideas for compression which do not seem to have any advantages over the method we propose. Section 7 discusses security implications and possible generalisations.

2. Overhead in Koblitz Curve Cryptosystems

To make the problem precise we need to consider how elliptic curve points are transmitted. We consider three communication models for transmitting binary data.

- (Fixed length bitstring) The receiver expects to get an m–bit string;
- (Fixed length bytestring) The receiver expects to get an m–byte string (this is a special case of the previous one);
- (Variable length bitstring) The receiver expects to get a bit–string of variable length $\leq m$ terminated by an end–of–transmission (EOT) symbol.

Parameters	n	a_2	c	Security level k	r_{bit}	r_{byte}
sect163k1	163	1	2	77	7	1
sect233k1	233	0	4	111	9	1
sect239k1	239	0	4	114	9	1
sect283k1	283	0	4	136	9	1
sect409k1	409	0	4	199	9	1
sect571k1	571	0	4	280	9	1

Table 1. Bandwidth overhead when using Koblitz curves.

The additional bandwidth required to send a point using a Koblitz curve system, compared with using general curves over $\mathbb{F}_{2^n}$, for an equivalent security level k, is called the *overhead.* We denote by r_{bit} the number of additional bits required to be sent (this is the *overhead in a bitstring communication model*), and r_{byte} the number of additional bytes (the *overhead in a bytestring communication model*). When it is clear from the context we will simply refer to these as *bit and byte overheads.*

Table 1 lists the values of n for which there is a Koblitz curve over $\mathbb{F}_2$ such that $\#E(\mathbb{F}_{2^n}) = c\ell$ where ℓ is a large prime and $c \in \{2, 4\}$. The security level is $k = \log_2(\sqrt{\pi 2^n/4cn})$ (which we round to the nearest integer). One expects to achieve this security level using a general elliptic curve over $\mathbb{F}_{2^m}$ with $m = 2k + 2$ (ignoring the possibility of Weil descent attacks for this value of m [3]). We therefore have

$$r_{\text{bit}} = n - (2k + 2)$$

(for example, with $n = 163$ we have $k = 77$ and so $m = 2 \cdot 77 + 2 = 156$ and $r_{\text{bit}} = 7$). Applying the Seroussi trick for both Koblitz curves and general curves over $\mathbb{F}_{2^n}$ gives $r_{\text{bit}} = (n - 1) - (2k + 1)$ which is the same overhead. Similarly, the number of extra bytes to be transmitted (when using the Seroussi trick) is

$$r_{\text{byte}} = \lceil (n - 1)/8 \rceil - \lceil (2k + 1)/8 \rceil \,.$$

For example, with $n = 163$, we have $\lceil (n - 1)/8 \rceil = 21$ while $\lceil (2 \cdot 77 + 1)/8 \rceil = 20$ so $r_{\text{byte}} = 1$. Similarly, for $n = 233$ we have $\lceil (n - 1)/8 \rceil = 29$ while $\lceil (2 \cdot 111 + 1)/8 \rceil = 28$, which again means we are sending one byte more than could be achieved using other curves. Even worse, if we didn't use the Seroussi trick we would be sending $\lceil n/8 \rceil = 30$ bytes.

The reason why 9 bits of overhead can mean only one byte of overhead is that $2k + 1$ is not necessarily a multiple of 8 and so there are already some spare bits in the byte representation.

3. Cryptography using Equivalence Classes

As mentioned in the introduction, one can perform elliptic curve cryptography using x-coordinates only (i.e., using equivalence classes of the form $\{P, -P\}$). Our proposal is to extend this idea to equivalence classes of the form $\{\pm\psi^i(P)\}$ for $0 \le i < n$. The goal of this section is to explain, using Diffie-Hellman key exchange as an example, that one can do cryptography with these equivalence classes. The crucial fact is that one can define point multiplication on equivalence classes: For $a \in \mathbb{N}$, one defines $[a][P] = [[a]P]$.

Lemma 1. *The operation $[a][P]$ is well-defined.*

Proof. Let $P_1 \in [P]$ so that $P_1 = \pm\psi^i(P)$ for some i. Since $\pm\psi^i$ is a group homomorphism we have $[a]P_1 = \pm\psi^i([a]P)$ and so $[[a]P_1] = [[a]P]$. □

Definition 2. *Let $\mathcal{S}$ be the set of equivalence classes of points in $E(\mathbb{F}_{2^n})[\ell]$. Let $r > 0$ be an integer. Let $\mathcal{C} : \mathcal{S} \to \{0,1\}^{n-r}$ and $\mathcal{D} : \{0,1\}^{n-r} \to \mathcal{S}$ be functions.*

Let $P \in E(\mathbb{F}_{2^n})[\ell]$ and write $[P]$ for the equivalence class of P. If

$$\mathcal{D}\big(\mathcal{C}([P])\big) = [P]$$

then we call $\mathcal{C}$ and $\mathcal{D}$ compression and decompression *functions.*

We now show that one can use compression and decompression functions to obtain a compressed Diffie-Hellman key exchange protocol. We assume the system parameters include a Koblitz curve $E(\mathbb{F}_{2^n})$ with point $P \in E(\mathbb{F}_{2^n})$ of prime order ℓ.

Alice picks a random $1 \leq a < \ell$, computes $Q_A = [a]P$ and sends $\mathtt{x}'_A = \mathcal{C}\big([Q_A]\big)$ to Bob. Similarly, Bob picks $1 \leq b < \ell$, computes $Q_B = [b]P$ and sends $\mathtt{x}'_B = \mathcal{C}\big([Q_B]\big)$ to Alice. Alice computes

$$k_A = \mathcal{C}\big([a]\mathcal{D}(\mathtt{x}'_B)\big)$$

and Bob computes

$$k_B = \mathcal{C}\big([b]\mathcal{D}(\mathtt{x}'_A)\big).$$

Lemma 2. *Alice and Bob compute the same key.*

Proof. Alice computes

$$\begin{aligned} k_A &= \mathcal{C}\big([a]\mathcal{D}(\mathtt{x}'_B)\big) \\ &= \mathcal{C}\big([a]\mathcal{D}\big(\mathcal{C}\big([[b]P]\big)\big)\big) \\ &= \mathcal{C}\big([a][[b]P]\big) \\ &= \mathcal{C}\big([[ab]P]\big) \end{aligned}$$

using the property $\mathcal{D}(\mathcal{C}([P])) = [P]$. It is easy to check that Bob computes the same value. □

In practice (to avoid small subgroup or invalid point attacks) one should verify that the resulting point does lie on the curve and that it has odd order. We discuss this later.

Obviously the same ideas can be used for any other cryptosystem which is fundamentally based on Diffie-Hellman key exchange (for example, modern versions of Elgamal encryption, which use a bitstring derived from $[ab]P$ for a symmetric encryption key).

4. Short Representatives of Equivalence Classes

In the following subsections we explain the compression and decompression algorithms. Recall that a *normal basis* for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ is a vector space basis of the form $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{n-1}}\}$. Theorem 2.35 of [6] states that a normal basis exists for every finite extension of finite fields. One represents elements of $\mathbb{F}_{2^n}$ with respect to the normal basis as an n-bit string. The action of the 2-power Frobenius map ψ is simply a rotation of the binary string, which is fast to compute.

4.1. Seroussi's Point Compression for Curves $E/\mathbb{F}_{2^n}$.

We describe Seroussi's method [10] for saving one bit in the representation of points in $E(\mathbb{F}_{2^n})$; also see [1]. Recall that for $\alpha \in \mathbb{F}_{2^n}$ the trace map is defined to be $\mathrm{Tr}(\alpha) = \sum_{i=0}^{n-1} \alpha^{2^i}$. It is well known that $\mathrm{Tr}(\alpha) \in \mathbb{F}_2$, $\mathrm{Tr}(\alpha + \gamma) = \mathrm{Tr}(\alpha) + \mathrm{Tr}(\gamma)$ and $\mathrm{Tr}(\alpha^2) = \mathrm{Tr}(\alpha)$ for all $\alpha, \gamma \in \mathbb{F}_{2^n}$. Seroussi's main result is the following.

Lemma 3. *Let $E/\mathbb{F}_{2^n}$ be defined by the Weierstrass equation*

$$E : y^2 + xy = x^3 + a_2x^2 + a_6$$

and let $P = (x_P, y_P) \in E(\mathbb{F}_{2^n})$ be of odd prime order ℓ. Then one has

$$\mathrm{Tr}(x_P) = \mathrm{Tr}(a_2).$$

The following result is standard, but we include a proof for completeness.

Lemma 4. *Let $\mathbb{F}_{2^n}$ be represented using a normal basis $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{n-1}}\}$ over $\mathbb{F}_2$. Then $\mathrm{Tr}(\beta) = 1$. Let $x \in \mathbb{F}_{2^n}$ be represented by the vector $(x_{n-1}, x_{n-2}, \ldots, x_0)$ in $\mathbb{F}_2^n$. Then*

$$\mathrm{Tr}(x) = \sum_{i=0}^{n-1} x_i.$$

Proof. First note that $\alpha = \sum_{i=0}^{n-1} \beta^{2^i} = \mathrm{Tr}(\beta)$ is an element of $\mathbb{F}_2$ and it is not zero since $\{\beta, \ldots, \beta^{2^{n-1}}\}$ is a linearly independent set over $\mathbb{F}_2$. Hence $\alpha = 1$. For $x_i \in \{0, 1\}$ it follows that $\mathrm{Tr}(x_i\beta^{2^i}) = x_i$. The result follows from the additivity of the trace. □

It follows that given a point $P = (x_P, y_P) \in E(\mathbb{F}_{2^n})$ of odd order one can represent x_P, with respect to a fixed normal basis for $\mathbb{F}_{2^n}$, as a binary string of length n. One can then remove any bit agreed between sender and receiver (it is natural to use the most or least significant bit) before transmission. The receiver obtains a bitstring of length $n-1$ and can append the correct bit so that the sum of the bits is equal to $\mathrm{Tr}(a_2)$.

4.2. **Compressing Abscissæ of Points on Koblitz Curves.** As mentioned, to compress a point $P = (x_P, y_P)$ we first throw away y_P and then represent x_P, with respect to a normal basis, as an n-bit string. We consider all the rotations of this bitstring.

Definition 3. *Let $n \geq 3$ and let $\mathtt{x} = \mathtt{x}_{n-1}\mathtt{x}_{n-2}\cdots\mathtt{x}_1\mathtt{x}_0$ be a binary string with $\mathtt{x}_i \in \{0,1\}$. We say $\mathtt{x}$ contains a* right padded run *of length t if and only if*

$$\mathtt{x} = \mathtt{x}_{n-1}\cdots\mathtt{x}_{t+2}\mathtt{011}\ \cdots\mathtt{110}.$$

In other words, bit 0 and the $(t+1)$–th bit are $\mathtt{0}$ and the intermediate t bits are $\mathtt{1}$.

Lemma 5. *Up to rotation only three strings of length ≥ 3 do not have a right padded run. Namely*

$$\mathtt{11}\cdots\mathtt{11} = (\mathtt{1})^n, \qquad \mathtt{00}\cdots\mathtt{00} = (\mathtt{0})^n \qquad \textit{and} \qquad (\mathtt{1})^{n-1} \parallel \mathtt{0}.$$

The first two of these do not correspond to points over $\mathbb{F}_{2^n}$ of odd order on a Koblitz curve. The third also does not correspond to a point of odd order when $a_2 = 1$ and n is odd.

Proof. The first claim is obvious. For the later claim first note the all zero and all one strings correspond to $x = 0, 1 \in \mathbb{F}_2$. Such values satisfy the equation $y^2 + xy = x^3 + a_2x^2 + 1$ for $y \in \mathbb{F}_2$ or $y \in \mathbb{F}_{2^2}$, and either way correspond to points of even order. The last statement follows from Lemma 3 since, when n is odd, $\mathrm{Tr}(x) = 0$ and $\mathrm{Tr}(a_2) = 1$. □

The compression algorithm can now be described. Given $P = (x_P, y_P)$ of odd order on a Koblitz curve consider all n rotations of the binary string representing x_P and determine which has the longest right padded run. In the case there are two or more runs of the same length we choose the binary string with lowest lexicographical ordering (i.e., smallest value when the bitstring is interpreted as an integer). Call the binary string $\mathtt{x}$ and let t be the length of the run of ones. One has $\mathtt{x} = \mathtt{x}_{n-1}\cdots\mathtt{x}_{t+2}\mathtt{011}\cdots\mathtt{10}$. In Section 5 we discuss the expected size of t. Applying the Seroussi trick one can delete $\mathtt{x}_{n-1}$.

We first consider the variable length bit string communication model. It is only necessary to send the $n-t-3$ bits $\mathtt{x}_{n-2}\cdots\mathtt{x}_{t+2}$. When the receiver is given these $n-t-3$ bits, knowing n, she can compute t and thus append the length $t+2$ pattern $\mathtt{01}\cdots\mathtt{10}$. Finally, the receiver obtains the most significant bit using the Seroussi method.

More realistically, we are in a fixed bit or byte communication model. The receiver expects to get a fixed number m of bits (or a fixed number of bytes) and must determine the $n-m$ missing bits. One subtlety here is that the receiver cannot determine the length t of the run of ones by the data sent to them (since the run of ones could be very long, or only just the minimum length, and the difference is whether or not to include a zero). Hence the receiver must assume that the missing bits include only a substring of $\mathtt{011}\cdots\mathtt{110}$. If $n-t-3 \geq m$ then the sender fails to obtain the desired level of compression and the algorithm must terminate (the sender can perhaps repeat the cryptographic protocol with different random choices). Hence the receiver assumes $m \geq n-t-2$. The sender sends $\mathtt{x}_{n-2}\cdots\mathtt{x}_{t+2}$ and an initial segment of the run $\mathtt{011}\cdots$ to make up the m bits. The receiver then adds $n-m-2$ ones, followed by a zero, and then adds the bit coming from the Seroussi trick.

One easily checks that the compression and decompression functions satisfy Definition 2. In particular, $\mathcal{D}(\mathcal{C}([P])) = [P]$ for any equivalence class $[P]$.

As noted earlier, one should verify that the resulting point does lie on the curve and that it has odd order. Since we are using the Seroussi trick it is automatic that $\mathrm{Tr}(x_P) = \mathrm{Tr}(a_2)$. To check that the point lies on the curve one should check that $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(x_P + a_2 + 1/x_P^2) = 0$ (which simplifies to $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(1/x_P^2) = 0$ in this case). If the number of points on the curve is 2ℓ where ℓ is prime then no further checks are necessary. If the number of points is 4ℓ then one should compute y_P, perform point halving of $P = (x_P, y_P)$, and check whether the resulting point $Q = (x_Q, y_Q)$ satisfies $\mathrm{Tr}(x_Q) = \mathrm{Tr}(a_2)$; we refer to Section 5 of King [4] for the details.

		bit–size n						
t	$t+3$	163	233	239	283	409	571	2047
3	6	1.00	1.00	1.00	1.00	1.00	1.00	1.00
4	7	0.99	1.00	1.00	1.00	1.00	1.00	1.00
5	8	0.94	0.98	0.98	0.99	1.00	1.00	1.00
6	9	0.74	0.86	0.86	0.90	0.97	0.99	1.00
7	10	0.48	0.62	0.62	0.68	0.81	0.90	1.00
8	11	0.28	0.38	0.38	0.43	0.56	0.68	0.98
9	12	0.15	0.22	0.21	0.24	0.33	0.43	0.86

TABLE 2. Estimated probability that the binary representation of the x-coordinate of a randomly chosen point on a Koblitz curve has, up to rotation, a run of at least t ones.

5. EXPECTED BANDWIDTH SAVING

We now analyse what values t can be expected in practice, and thus how effective our method is. Our compression function acts by finding the longest run of ones in the binary representation of the x-coordinate. Hence, the question is to determine the probability of certain lengths of runs of ones. For random binary strings of length n there is a large literature on this problem (for a survey see [9]). The basic result is that the expected value for the longest run of ones in a random binary sequence of length n is approximately $\log_2(n) - 1$.

Working with binary sequences of length n up to rotation can only increase the expected length of the longest run of ones, but our experiments show that this effect is not noticeable for moderate values of n. Also note that we are not studying random binary strings, but strings corresponding to normal basis representations of x-coordinates of elliptic curve points of odd order (and hence with an imposed parity condition $\mathrm{Tr}(x) = \mathrm{Tr}(a_2)$). Our experiments suggest that the expected length of the longest run of ones is the same (up to 2 decimal places) in this case as for random binary strings. In any case, it is clear that $r \approx \log_2(n)$ is the best we can hope for, and is unlikely to be achievable for a large proportion of points.

One crucial feature of our method (in the variable length bitstring case at least) is that we do not just look for a run of ones, but use the fact that a run of ones has zeroes on each end. This allows us to get closer to the desired saving of $\log_2(n)$ bits.

We now present the results of some simulations. We computed an approximation to the probability that the x-coordinate of a random point of odd order on the Koblitz curves of Table 1 has a run of ones of length at least t for $3 \leq t \leq 9$. Our method allows us to omit $t + 3$ bits when sending such a point in the variable bitstring model.

The results are given in Table 2. The black coloured cells indicate when $t+3 = \lceil \log_2 n \rceil$ and the grey cells are when $t + 3 = \lfloor \log_2 n \rfloor$ when these values are different. The case $n = 2^{11} - 1 = 2047$ has been included purely for theoretical interest.

5.1. **Variable Length Bitstring Model.** We first consider the case of variable length bitstrings. One sees from the tables that, with high probability, 7 bits can be saved when $n = 163$, 8 bits saved when $233 \leq n \leq 283$ and 9 bits saved when $n \geq 409$. These savings are achieved in key exchange by an algorithm which fails (in other words, has to be repeated) between 1 and 3 times in every 100 executions, on average. Depending on the application, the saved bandwidth could be worth the added inconvenience of occasionally having to repeat an exponentiation. Hence, we have a good solution to the original problem.

5.2. **Fixed Length Bitstring Model.** We now consider the more standard setting of fixed length bit or byte communications. Note that the naive method sends $\lceil n/8 \rceil$ bytes. The results of Table 2 show how successfully the number of bytes to be sent can be reduced. It follows that one can save one byte in all cases with very high probability, and this achieves the expected number of bytes of communication for $n \in \{163, 239, 283, 571\}$. The cases $n \in \{233, 409\}$ are harder since $233 \equiv 409 \equiv 1 \pmod 8$ and we wish to save 2 bytes for these cases. We give our results in Table 3.

n	Number bits to remove	Probability to save 1 byte	Probability of full compression
163	3	1.00	1.00
233	9	1.00	0.62
239	7	1.00	1.00
283	3	1.00	1.00
409	9	1.00	0.81
571	3	1.00	1.00

Table 3. Experimental estimates for the probability of successful compression in the fixed byte length model.

6. Alternative Methods

There are several other ways to approach this problem. We believe the one presented in the paper is the most simple and efficient, but we briefly mention some of the other ideas that were considered.

- Search for the longest run of zeroes *or* ones, remove the run together with the bit to the right of it, but keep the bit to the left of the run (this bit indicates whether the run was of zeroes or ones). This seems to give no overall improvement. The expected length of the longest run is now one bit more, but this saving is lost by having to keep a bit to specify whether the run is of ones or zeroes. Note that this method deals with the third case of Lemma 5 (though of course one would still compress this string by removing part of the run of ones).
- Consider a different property than runs of zeroes or ones, such as repeated patterns of bits, palindromic patterns of bits etc. The problem with this approach is that such patterns are no more likely than runs. Such patterns also do not have the crucial additional feature of runs, namely that the adjacent bits have opposite value.
- Use general data compression techniques (such as run-length coding or the Lempel-Ziv-Welch algorithm). These methods do not exploit the fact that we consider the binary string up to rotation, which is the main novelty of our proposal. The problem is that random binary strings are not compressible. Hence, when applied to random x-coordinates these methods cannot be competitive with our approach.

7. Conclusions

We have given a compression method and have given experimental results that show it works well in both the fixed and variable length bit string communication models.

There can be no significant loss of security when using a compression method which succeeds with probability more than 0.95, since the potential key space is only reduced by a small amount. Even if it were possible to modify the Pollard rho algorithm to take this information into account, the impact on the difficulty of the discrete logarithm problem would be very small. For the same reason we believe the full compression for $n = 233$ and $n = 409$ causes no loss in security.

We remark that similar ideas can be applied to obtain compression with other subfield curves but with much less success. For example, with elliptic curves E over $\mathbb{F}_{2^2}$ with group $E(\mathbb{F}_{2^{2n}})$ the expected length of the longest run is only $\approx \log_4(n) = \log_2(n)/2$. Since there are four possible types of run we either focus only on runs of ones (which could be shorter), or use two bits to specify which "digit" appears in the longest run. A further impediment in this case is that n is now half as big for a given security level.

Acknowledgements

The second author was supported by EPSRC grant EP/D069904/1. We are grateful to Tanja Lange and the anonymous referees for comments.

References

1. I.F. Blake and G. Seroussi and N.P. Smart, Elliptic Curves in Cryptography, Cambridge, 1999.
2. R. P. Gallant, R. Lambert, and S. A. Vanstone, Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves, *Mathematics of Computation*, **69** (2000) 1699-1705.

3. P. Gaudry, F. Hess and N. Smart, Constructive and Destructive Facets of Weil Descent on Elliptic Curves, *Journal of Cryptology*, 15 (2002) 19-46.
4. B. King, A Point Compression Method for Elliptic Curves Defined over $GF(2^n)$, in F. Bao, R. H. Deng and J. Zhou (eds.), PKC 2004, Springer LNCS 2947 (2004) 333-345.
5. N. Koblitz, CM-Curves with Good Cryptographic Properties, in J. Feigenbaum (ed.), CRYPTO '91, Springer LNCS 576, (1992) 279-287.
6. R. Lidl and H. Niederreiter, Introduction to Finite Fields and their Applications, Cambridge, 1994.
7. V. S. Miller, Use of Elliptic Curves in Cryptography, in H. C. Williams (ed), CRYPTO '85, Springer LNCS 218 (1986) 417-426.
8. J. Pollard, Monte Carlo Methods for Index Computation mod p, Mathematics of Computation, **32** (1978) 918-924.
9. M. F. Schilling, The Longest Run of Heads, *The College Mathematics Journal*, Vol. 21, No. 3 (1990) 196-207.
10. G. Seroussi, Compact Representation of Elliptic Curve Points over $\mathbb{F}_{2^n}$, Tech. report, HP Labs Tech. Report HPL-98-94R1, September 1998.
11. J. A. Solinas, Efficient Arithmetic on Koblitz Curves, *Designs, Codes and Cryptography*, **19**, nos. 2-3 (2000) 195-249.
12. P. C. van Oorschot and M. J. Wiener, Parallel Collision Search with Cryptanalytic Applications, *Journal of Cryptology*, **12** (1999) 1-28.
13. M. J. Wiener and R. J. Zuccherato, Faster Attacks on Elliptic Curve Cryptosystems, in S. E. Tavares and H. Meijer (eds.), SAC 1998, Springer LNCS 1556 (1999) 190-200.

E-mail address: `Philip.Eagle@hsbc.com.mx`

Information Security Group, Mathematics Department, Royal Holloway University of London, Egham, Surrey TW20 0EX, UK.

E-mail address: `S.Galbraith@math.auckland.ac.nz`

E-mail address: `jong017@aucklanduni.ac.nz`

Mathematics Department, The University of Auckland, Private Bag 92019 Auckland 1142 New Zealand.