

COMPRESSED ECC PARAMETERS

N.P. SMART

We do not believe that a single elliptic curve across all installations is neither a sensible or secure way to achieve interoperability for elliptic curve systems. To achieve interoperability in a multi-curve environment either the different ECC parameters need to be held in some database for all systems or the ECC parameters need to be transported between the different devices/systems.

Current methods to define the bit representation of ECC parameters are very heavy. They rely on ASN.1 syntax to define all the parameters. Given that a lot of parameters need to be passed this creates a huge amount of wastage given the actual amount of information that needs to be passed.

In this note we explain how the ECC parameters can be represented in a very small number of bits, namely $34 + 3n/2$, where n is the bit size of the curve. We assume that $150 \leq n \leq 255$, but other values of n can clearly be accommodated using only minor changes to the following method.

Hence for $n = 191$ we represent an ECC key in 321 bits. A similar strength set of parameters for *DSA* would require 1024 bits for p , 160 bits for q and 1024 bits for the generator, making 2208 bits in all. The standard representation of an ECC set of parameters would require at least 600 bits.

We note that various different schemes for compressing the ECC parameters can be thought up along the following lines and what follows is just a simple example.

1. COMPRESSING ECC PARAMETERS

All the details we require about elliptic curve systems can be found in the book [1].

1.1. The Finite Field. We first need to specify which finite field to use and whether it is of even or odd characteristic. We wish to choose an elliptic curve, E , over a field \mathbb{F}_q of order around 2^n . Note that in all current practical elliptic curve systems, one would choose $n \leq 255$.

To specify an even characteristic field can then be done by restricting the fields to those with a trinomial basis. We can represent the field using

$$X^n + X^c + 1$$

so we require 8 bits to represent c . In practice it is common to take n to be odd in this situation, which we shall indeed do (for reasons to become apparent later). Hence to specify the even characteristic field requires at most 15 bits, since we are assuming n is always odd.

In many systems for odd characteristic fields we use fields of prime order equal to

$$q = 2^n + c.$$

If c is small this gives a very efficient way of performing the field operations with no known loss of security. We shall restrict to fields of the above form with $n \leq 255$

and $1 \leq c \leq 255$. There are 174 such primes with $150 \leq n \leq 255$ and $1 \leq c \leq 255$, which we list in Table 1. Hence we have more choice of odd characteristic fields than even characteristic fields. In addition we will require 15 bits to represent q , 8 bits for n and 7 bits for c , since clearly c must be odd.

Since both odd and even fields can be represented using 15 bits we can represent the choice of field in 16 bits by using a single additional bit to specify whether we are in the odd or even case. We shall call the resulting bit string f .

1.2. The Elliptic Curve. Having chosen a field, we now find an elliptic curve over \mathbb{F}_q using Schoof's algorithm [2]. Notice that we are not specifying a special elliptic curve and so we are not compromising security with our choice of elliptic curves. The elliptic curve is given by an equation of the form

$$\begin{aligned} Y^2 &= X^3 + aX + b & q \text{ odd,} \\ Y^2 + XY &= X^3 + a_2X^2 + a_6 & q \text{ even.} \end{aligned}$$

Since in the even characteristic case we have insisted that n is odd, we can choose $a_2 \in \{0, 1\}$.

In the odd characteristic case we can assume that a will sit in a signed eight bit word, i.e. between -127 and 127 . To see why this is so, notice that $Y^2 = X^3 + aX + b$ is isomorphic to $Y^2 = X^3 + au^4X + bu^6$ for some $u \in \mathbb{F}_q$. Now if $q \equiv 1 \pmod{4}$ then we have a 25 percent change of replacing a by any number a' , by simply trying to extract the fourth root of a'/a . If $q \equiv 3 \pmod{4}$ then we have an even better chance, namely 50 percent, of this working for any given a' . If we insist that a is specified by only 8 bits then we have an, at most,

$$.75^{256} \approx 10^{-32}$$

chance that our curve found by Schoof's algorithm cannot be put in a form with $a \notin \{-127, \dots, 127\}$. If we are so unlucky we could always try and find another curve.

Since b and a_6 both require n bits to specify them, we can represent the elliptic curve E using only $n + 8$ bits. We call the resulting string e .

1.3. The Group Order. We assume that the curve order is divisible by a large prime l and

$$N_q = \#E(\mathbb{F}_q) = \begin{cases} l & q \text{ odd,} \\ 2l & q \text{ even and } a_2 = 1, \\ 4l & q \text{ even and } a_2 = 0. \end{cases}$$

This is common practice and such elliptic curves are quite easy to find using Schoof's algorithm for the values of q in use today. By Hasse's theorem we have that $t = q + 1 - \#E(\mathbb{F}_q)$ is bounded by $|t| \leq 2\sqrt{q}$. Hence to specify the group order, and hence l , requires $2 + n/2$ bits. The resulting bit string we denote by o . Note by choosing the group orders in this way a cofactor need not be specified as this is implicit in the definition of the field and the curve.

1.4. The Generating Point. We now need to specify a point which generates a subgroup of order l . We pick a random bit string x of length 7 and consider the field element $\{x\}$ represented by x padded with $n - 7$ zero bits. The choice of 7 bits is to make the final representation of the generating point fit into 8 bits in total

(we are reserving one bit for the y -coordinate). With probability $1/2$ we can find a field element $y \in \mathbb{F}_q$ such that $(\{x\}, y) \in E(\mathbb{F}_q)$. If we then compute

$$P = [N_q/l](\{x\}, y)$$

then with probability around 2^{2-n} we have $P = \mathcal{O}$ and we reject this value of x and start again. Otherwise we have found a point P which is a non-trivial element of order l in $E(\mathbb{F}_q)$. So we can represent the generating point, P , by compressing the elliptic curve point $(\{x\}, y)$ into a bit string, p , of length 8 bits. With overwhelming probability such a method is guaranteed to find a suitable generating point.

Hence to represent all the parameters of our elliptic curve we require the bit string

$$X = f \parallel e \parallel o \parallel p$$

of length $t = 16 + (n + 8) + (2 + n/2) + 8 = 34 + 3n/2$ bits.

REFERENCES

- [1] I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic Curves in Cryptography*. To appear, CUP, 1999.
- [2] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p . *Math. Comp.*, **44**, 483–494, 1985.

HEWLETT-PACKARD LABORATORIES, FILTON ROAD, STOKE GIFFORD, BRISTOL, BS12 6QZ, U.K.
E-mail address: nsma@hplb.hpl.hp.com

TABLE 1. The 174 primes of the form $2^n + c$ with $150 \leq n \leq 255$ and $1 \leq c \leq 255$

n	c	n	c
150	147, 163	151	253
152	27	153	45, 115, 133
154	97, 189, 253	155	7
156	19, 49, 225	157	117, 135, 151, 231
158	85	159	87, 117
160	49, 55, 79, 85, 97, 133, 135, 225	161	105, 177
162	7, 37, 73, 129	164	87, 99, 189, 193
165	15, 115, 193	166	49, 207
167	27, 63	168	205, 249
169	57, 123, 153	170	129, 223
171	133	172	27, 67, 103
173	21	174	15, 49, 169
175	235	176	67, 189
177	7, 157, 213, 247	178	33, 75, 169, 229, 247
180	33, 177	181	57, 223
182	7, 217	184	163
185	217	186	49, 57, 109, 127, 163
189	3	190	67, 163, 183
192	43, 225, 255	193	25, 97, 193, 237
194	67, 189, 199	195	115
196	69	197	133, 235
199	81	200	25
202	79, 153, 163, 207, 247	204	73
205	223	206	79
207	175	208	133, 135, 193
209	43, 111, 121	210	127, 133
211	57, 81, 163, 223	212	63, 67, 177
213	157	214	117
215	45	216	159, 189
218	43	219	127
221	13, 91	222	67, 169, 205, 249
223	61, 163, 247	225	157
227	37	228	45, 73, 117
229	27, 81	230	15, 189
231	91	232	93
233	87	235	27
236	219	238	249
239	15	240	93
243	241	244	97, 99
246	15, 85	247	231
249	241	251	55, 81, 223
255	105		