Efficient Identity Based Parameter Selection for Elliptic Curve Cryptosystems

Arjen K. Lenstra

Citibank, N.A., 4 Sylvan Way, Parsippany, NJ 07054, U.S.A.
arjen.lenstra@citicorp.com

Abstract. A method is proposed that allows each individual party to an elliptic curve cryptosystem to quickly determine its own unique pair of finite field and Weierstraß equation, in such a way that the resulting pair provides adequate security. Although the choice of Weierstraß equations allowed by this proposal is limited, the number of possible finite fields is unlimited. The proposed method allows each participant to select its elliptic curve cryptosystem parameters in such a way that the security is not affected by attacks on any other participant, unless unanticipated progress is made affecting the security for a particular Weierstraß equation irrespective of the underlying finite field. Thus the proposal provides more security than elliptic curve cryptosystems where all participants share the same Weierstraß equation and finite field. It also offers much faster and less complicated parameter initialization than elliptic curve cryptosystems where each participant randomly selects its own unique Weierstraß equation and thus has to solve the cumbersome point counting problem.

1 Introduction

Elliptic curve cryptosystems come in many different flavors. However, they all have the following in common: a point of high prime order in the group of points of an elliptic curve over a finite field. In this paper, let F_p denote the finite field (of cardinality p for a prime power p), let E be an equation defining the elliptic curve over F_p, let $E(F_p)$ be the group of points of that elliptic curve over F_p, and let Q be the point of high prime order q in $E(F_p)$. In some elliptic curve cryptosystems all participants share the elliptic curve data (F_p,E,Q,q), in others each participant selects its own curve data. In this paper these systems are referred to as shared and non-shared systems, respectively.

In either case, given shared or non-shared curve data, each participant selects its own private key and computes the corresponding public key in the following manner: party A selects a random positive integer $m_A < q$ and computes the point $G_A = m_A \bullet Q$ in $E(F_p)$ (with $\bullet$ denoting scalar multiplication in $E(F_p)$). In a shared curve system the public key of party A consists of G_A, with (F_p,E,Q,q) implicitly defined as system wide constants and presumably determined beforehand by a central authority. In a non-shared system A's public key consists of (F_p,E,Q,q,G_A), with (F_p,E,Q,q) unique to party A, and therefore more appropriately referred to as $(F_{p(A)},E_A,Q_A,q_A)$. In both systems A's private key consists of the integer m_A.

Shared and non-shared systems have their own advantages and disadvantages. An advantage of shared systems is that the part of A's public key data that is unique to A consists of just G_A instead of the five-tuple $(F_{p(A)},E_A,Q_A,q_A,G_A)$, which facilitates key certification and communication set-up. Another advantage of the shared approach is that, since the computation involved in determining the curve data (F_p,E,Q,q) has been carried out beforehand by a central authority, none of the parties has to perform complicated or lengthy computations to generate their private and public keys: selecting a random integer and performing a scalar multiplication in $E(F_p)$ suffices. Key generation in a non-shared system may be quite involved: if party A picks finite fields and curve equations at random, highly non-trivial counting techniques have to be employed to check if an appropriate point of high prime order exists. These are probably the reasons why most elliptic curve cryptosystem proposals these days are shared systems.

Non-shared systems may, however, be preferable if security and not efficiency is the top-priority. In a shared system, it is conceivable that an attack on one participant's public key affects some other participant's security as well, in particular if an index calculus type attack against elliptic curve cryptosystems would be found. In such attacks individual problems often become relatively easy once an initial database has been built. With the current state of the art of methods to solve the discrete logarithm problem in groups of elliptic curves, however, data generated to attack a particular non-shared public key have no effect at all on the security of any other non-shared public key. This is the case even if only the underlying finite fields are different but the curve equation remains the same.

In this paper a non-shared system is proposed for which the public key data is relatively short and can easily be constructed by each participant, and such that an attack on one participant does not affect the security of any other participant. Thus the proposal combines the advantages of the more traditional shared and non-shared systems, without having any of their disadvantages. The Weierstraß models used in this proposal are not new. At least one of them even goes back to Gauss. Also, their application to elliptic curve cryptosystems has been proposed before, e.g. [3: page 158]. What is new, however, is the way each party uses its identity to select an appropriate elliptic curve (i.e., a Weierstraß model and a finite field F_p) and a suitable point of high order, and how other users reconstruct that party's public key data based on the same identity and a small number of additional bits.

The paper is organized as follows. In Section 2 the theoretical background is reviewed, in Section 3 the new non-shared elliptic curve key generation method is presented, and in Section 4 a more detailed comparison with shared and other non-shared systems is provided.

2 Background

The theoretical background on elliptic curves reviewed in this section can be verified by anyone with adequate background in elliptic curves (cf. [5]), or using any standard textbook, e.g. [6].

Table 1 lists eight Weierstraß models $Y^2 = X^3 + uX + v$ for elliptic curves. If $p \equiv 3 \bmod 4$ is a prime that satisfies the conditions in one of the rows of Table 1, then the Weierstraß model $Y^2 = X^3 + uX + v$ in that row defines a non-supersingular elliptic curve $E_{u,v} = E$ over F_p. In this paper only prime fields F_p are considered. The discriminant Δ of the endomorphism ring and the cardinality $|E(F_p)|$ of the group of points $E(F_p)$ of E over F_p are also listed in Table 1, where $j(m,n) = 1$ if m is a square modulo n and $j(m,n) = -1$ if m is not a square modulo n. The last column of Table 1 lists a fixed divisor of $|E(F_p)|$ (i.e., in some cases the group order is not prime). Note that the rows for d = 7, 11, 19, 43, 67, and 163 share the same fourth, fifth and sixth columns, with exceptions for d = 11 and d = 7 for the fifth and sixth column, respectively.

For example, if d = 7 and p is a prime that is 3 mod 4 for which there are non-negative integers a, b such that $a^2 + 7b^2 = 4p$ and $a \neq 1$, then $Y^2 = X^3 - 35X - 98$ defines a non-supersingular elliptic curve $E_{-35,-98} = E$ over F_p, and the cardinality of the group of points $E(F_p)$ is $p+1-j(2a,7)a$. Furthermore, $p+1-j(2a,7)a$ is divisible by 8.

Table 1. Weierstraß models.

$\Delta = -d$, with d	$Y^2 = X^3 + uX + v$, with u	v	conditions on p, d, and $a, b \in \mathbf{Z}_{\geq 0}$	group cardinality	fixed divisor
3	0	16	$a^2+3b^2 = 4p$, $p \equiv 1 \bmod 3$, $a \equiv 1 \bmod 3$, $b \equiv 0 \bmod 3$	$p+1+a$	9
8	–270	–1512	$a^2+2b^2 = p$, $a \equiv 1 \bmod 4$ if $p \equiv 3 \bmod 16$, $a \equiv 3 \bmod 4$ if $p \equiv 11 \bmod 16$	$p+1-2a$	2
7	–35	–98	$a^2+db^2 = 4p$, $a \neq 1$	$d \neq 11$: $p+1-j(2a,d)a$, $d = 11$: $p+1+j(2a,11)a$	$d \neq 7$: 1, $d = 7$: 8
11	–9504	–365904			
19	–608	5776			
43	–13760	621264			
67	–117920	15585808			
163	–34790720	78984748304			

The same Weierstraß model used over two different finite fields gives rise to two different and, from a security point of view, independent groups. With the current algorithms, ability to solve discrete logarithms in one of those groups does not make it easier to compute discrete logarithms in the other group.

3 Non-shared elliptic curve key generation

Table 1 can be used for elliptic curve key generation in the following manner (for d = 3 this is implied by [3: page 158]): generate a random prime $p \equiv 3 \bmod 4$ of a specified size, try and solve the equation for a, b, d, and p for all d's in Table 1, if successful for some d then check if the group cardinality for the corresponding curve satisfies the obvious security requirements, and repeat with another prime p until a good curve has been found. This straightforward approach has the advantage that

primes can be chosen having advantageous computational properties. But since it requires a relatively complicated computation (namely the attempt to solve for a given p the equation $a^2+tb^2 = 4p$ for various t's) the following even simpler approach is used here: pick a random integer a of appropriate size, and search for a non-negative integer b until the conditions in at least one row of Table 1 are satisfied for a prime p that is 3 modulo 4, and such that the group cardinality satisfies the security requirements.

The resulting elliptic curve key generation method makes use of the following function to check if a candidate a,b pair is satisfactory for a certain d.

(3.1) Check conditions on d, a, b, and x.

This function not only checks that d, a, and b lead to a 'good' curve according to Table 1, but also checks that a point of high prime order can easily be computed on the curve as a function of the additional input parameter x. Curves that are otherwise 'good' are rejected if the prescribed x does not lead, in some standard way that is described below, to a point of high prime order (cf. Remark (3.6)).

Input: Positive integers d, a, b, x, where d is one of the values in the first column of Table 1.

Output: Either 'Failure', or d, p, q, and Q, where p and q are two primes, and Q is a point of order q in the group of the curve corresponding to d over F_p (i.e., all public key data required, except for the point G).

1. Check if the conditions in the row referred to by d and the fourth column of Table 1 are satisfied for d, a, b, and if the resulting number p is a prime that is 3 mod 4. If not, then output 'Failure' and terminate.
2. Compute the group cardinality according to the fifth column (and appropriate row), and check if it can be written as $q*f$ for a prime q and a positive integer $f \leq 32$ (the fixed divisor from the sixth column will be a factor of f). If not, then output 'Failure' and terminate. (The bound of 32 on f is arbitrarily chosen and may be replaced by any value that is convenient and that still offers acceptable security.)
3. Check that there is no integer m with $m(\ln(m*\ln(p)))^2 \leq 0.02(\ln(p))^2$ for which q divides p^m-1. If such an integer m exists, then output 'Failure' and terminate. (Existence of a small m for which q divides p^m-1 implies that the group of points of the corresponding curve over F_p is susceptible to a subexponential-time attack based on the Weil or Tate pairing. It is well known that such curves should be avoided for cryptographic applications. The bound on m follows trivially from the heuristic runtime estimate of the Number Field Sieve based subexponential-time discrete logarithm algorithm.)
4. Compute $y = (x^3 + ux + v)^{(p+1)/4}$ modulo p, with u and v corresponding to d according to the second and third columns of Table 1, check if $y^2 = x^3 + ux + v$ modulo p (i.e., if the y thus computed is indeed the squareroot modulo p of $x^3 + ux + v$) and if so put P = (x mod p, y). If $y^2 \neq x^3 + ux + v$ modulo p, i.e., if no point with x-coordinate equal to x modulo p is on the curve indicated by d, then output 'Failure' and terminate. (The condition that $p \equiv 3 \bmod 4$ is required for the efficient modular squareroot computation in this step.)

5. Compute $Q = f \bullet P$ in $E_{u,v}(F_p)$ with u and v as in the previous step, and check that Q is not equal to the identity element in $E_{u,v}(F_p)$. If Q is equal to the identity element, then output 'Failure' and terminate. (Due to the way Q is constructed it has order either 1 or q; a point Q of order 1 is worthless. If d = 3 then f may be replaced by f/3 because the order 9 subgroup is not cyclic.)
6. Check that $q \bullet Q$ is the identity element in $E_{u,v}(F_p)$. If not, then output 'Failure' and terminate. (This failure indicates an inconsistency, either due to an implementation error or, when called from (3.3), to an intentional attempt to reconstruct a 'wrong' key.)
7. Output d, p, q, and Q.

For the key generation it is assumed that each participant to the system has a unique ID: a bitstring that uniquely identifies a participant and that is recognized by all other participants. Commonly such IDs are exchanged whenever two participants set up a communication channel, either as part of the respective certificates or in plain-text. Furthermore, it is assumed that all participants to the system share three hash functions $R_1(B,s)$, $R_2(B,s)$, $R_3(B,s)$, each mapping a positive integer B and a bitstring s of arbitrary length to a positive integer of B bits. These hash functions do not have to satisfy any fancy cryptographic requirements and neither do they have to be particularly efficient – anything that is convenient and that takes all bits of s into account will do.

(3.2) Non-shared identity based elliptic curve key generation.

Input: The identifying information ID of the participant generating the key and a security parameter $B \in \mathbf{Z}_{>0}$.

Output: A public key (F_p,E,Q,q,G_{ID}) suitable for use in elliptic curve cryptosystems, the corresponding private key m_{ID}, and additional data from which F_p, Q, and q can easily be reconstructed given E or d: a bitstring s of length 32 and an 8-bit integer b_1.

This protocol should be performed by the party uniquely identified by ID (though steps 1 through 5 may in principle be carried out by any party).

1. Pick a random bitstring s of length 32 and compute $a = R_1(B,ID||s)$, $b_0 = R_2(B,ID||s)$, and $x = R_3(2*B,ID||s)$, where ID||s denotes the concatenation of ID and s.
2. Let i = 0.
3. For the eight different d's given in Table 1 do the following:
   - Check the conditions on d, a, $b = b_0+i$, and x as described in (3.1). If (3.1) outputs d, p, q, and Q, then let $b_1 = i$ and jump ahead to step 6.
4. Otherwise, if (3.1) outputs Failure for all 8 different d's, then replace i by i+1 and jump back to step 3 if $i < 256$.
5. Return to step 1 (because all $8*256$ calls to (3.1) resulted in 'Failure' a new s is needed).
6. Let d, p, q, and Q be as output by (3.1) and let E be the curve indicated by d according to Table 1. Pick a random positive integer $m_{ID} < q$ and compute $G_{ID} = m_{ID} \bullet Q$ in $E(F_p)$.

7. Output (F_p,E,Q,q,G_{ID}) as public key, m_{ID} as private key, and the key reconstruction data s and b_1.

Because a and b are B-bit integers the resulting p and q have approximately 2B and at least 2B−5 bits, respectively, where the '−5' corresponds to the bound 32 on f in step 2 of (3.1). Thus, the security of the resulting key is believed to be at least B−2.5 bits.

An important aspect of (3.2) is that the (F_p,Q,q) part of a party's public key can be reconstructed very easily given the party's ID, E (or d), and the s and b_1 resulting from the construction of (F_p,E,Q,q). Since E and d can be encoded using 3 bits, any other party only needs 3 + 32 + 8 bits plus the ID and value for B to reconstruct F_p, Q, and q. The details are as follows.

(3.3) Public key reconstruction.
Input: Identifying information ID, security parameter B, 32-bit string s, 8-bit integer b_1, and 3 bits indicating E and d.
Output: Either 'Failure' or (F_p,Q,q).

1. Compute $a = R_1(B,ID||s)$, $b = R_2(B,ID||s) + b_1$, and $x = R_3(2*B,ID||s)$.
2. Check conditions on d, a, b, and x as described in (3.1). If (3.1) returns 'Failure', then output 'Failure' and terminate.
3. Otherwise, output (F_p,Q,q) with p, q, and Q as output by (3.1).

(3.4) Cheap public key reconstruction. If the public key reconstruction (3.3) is performed on data retrieved from a certificate, then the check of the conditions on d, a, b, and x in step 2 of (3.3) can be sped-up considerably, under the assumption that the certification authority performed the full check as described in (3.1). For instance, the two primality checks on p and q can be omitted (where q can be found after trial division up to at most 32), and also the check that Q has indeed order q can be omitted. This implies that reconstruction of the (F_p,Q,q)-part of a public key from certified data can be performed at the cost of essentially a single (p+1)/4-th powering in F_p. The time required for this exponentiation is very small compared to the time required for the 'standard' operation in elliptic curve cryptosystems, namely full scalar multiplication in the group of the elliptic curve.

Lengths of s and b_1. The lengths of s and b_1 should in principle depend on B. In (3.2) the lengths are arbitrarily chosen in such a way that the choices work satisfactorily for any reasonable value of B. Smaller s and b_1 imply that fewer bits have to be exchanged and/or certified for public key exchange. Larger s and b_1 mean that for a given ID and B more different curves may be selected. The values 32 and 8 lead to more different curves than any particular ID will ever want to generate for any reasonable fixed value of B and do not cause noticeable communication overhead.

(3.5) Performance. If for an integer d as in Table 1 and a randomly picked pair a, b the conditions on a are satisfied, then the probability that a, b, and d lead to a prime p as specified in the fourth column of Table 1 is approximately the same that a randomly picked number of the same size as p is prime. Thus, it may be expected that step 1 of (3.1) (as called by step 3 of (3.2)) is successful for some d after O(B)

attempts. The probability of success of step 2 of (3.1) is of the same order of magnitude, but with a much better constant because cofactors up to 32 are allowed. So, $O(B^2)$ attempts may be expected before step 2 of (3.1) is successful. Steps 3, 5, and 6 of (3.1) have a negligible probability of failure, and Step 4 of (3.1) fails with probability approximately 0.5. It follows that the runtime of (3.2) is dominated by the $O(B^2)$ probabilistic compositeness tests on approximately 2B-bit numbers, in steps 1 and 2 of (3.1). In practice (3.2) runs quite fast. For instance, for B = 90 public keys are on average produced in less than ten seconds on a 133MHz Pentium.

Key reconstruction (3.3) is dominated by the two probabilistic compositeness tests on approximately 2B-bit numbers in steps 1 and 2 of (3.1), the (p+1)/4-th powering in step 4 of (3.1), and the check that $q \bullet Q$ is the identity element in step 6 of (3.1). In practice it takes a fraction of a second. The runtime required by the 'cheap' key reconstruction (3.4) is dominated the (p+1)/4-th powering in step 4 of (3.1), and thus almost negligible in practice.

(3.6) Remark. In (3.1) a curve that would otherwise be good (i.e., a curve for which step 4 of (3.1) is reached) is rejected if a point with x-coordinate equal to x mod p is not on the curve (step 4 of (3.1)), or if such a point is on the curve but does not lead to a point of order q (as in step 5 of (3.1)). Here x is chosen in (3.2) as $R_3(2*B,ID||s)$ so it can easily be reconstructed. If speed of the public key generation process is important, then (3.1) can trivially be modified so that it looks for the smallest non-negative j such that (x+j) mod p instead of x mod p satisfies the requirements in steps 4 and 5 of (3.1). Obviously, the resulting j would have to be included in the key reconstruction data s and b_1, thereby increasing the number of bits required for key reconstruction. A two or three bit j may speed up the key generation process by a factor two (cf. (3.5)), without affecting the length of the key reconstruction data in a substantial way.

Implementation. Elliptic curve arithmetic can be implemented in many different ways (cf. [1, 4]). The most efficient choice depends on hardware characteristics of the device to be used and is outside the scope of this paper. Assuming that elliptic curve arithmetic is available, implementation of the key generation and reconstruction method proposed in this paper is entirely straightforward based on the descriptions in (3.1), (3.2), (3.3), and (3.4).

4 Comparison

As explained in Section 1, in a shared elliptic curve cryptosystem the part of A's public key data that is unique to A consists of a single point G_A on the curve. If an L-bit finite field is used, then G_A can trivially be encoded in 2L bits: L bits for the x-coordinate and L bits for the y-coordinate. It is common practice, however, not to encode the full y-coordinate, but to let the recipient of the public key perform a modular squareroot computation to derive the y-coordinate from the x-coordinate. As argued above this squareroot computation is negligible compared to the ensuing cryptographic operations (in particular if $p \equiv 3 \bmod 4$). Since one additional bit is needed to indicate which of the two squareroots should be used, G_A can be encoded in

L + 1 bits. It follows that the amount of data to be exchanged or certified is L + 1 + |ID| bits, where |ID| denotes the length of the identifying information ID.

In a non-shared elliptic curve cryptosystem party A's public key data consists of the five-tuple $(F_{p(A)},E_A,Q_A,q_A,G_A)$. Encoding of these data requires L bits for p(A), in general 2L bits for a randomly selected Weierstraß model to specify E_A, L + 1 bits each for Q_A and G_A (at the cost of two modular squareroot computations), and approximately L/2 bits to encode the difference between p(A) and the group order, plus the cofactor of q_A in the group order. The total number of bits to encode $(F_{p(A)},E_A,Q_A,q_A,G_A)$ is equal to 5.5L+2. The amount of data to be exchanged or certified is 5.5L + 2 + |ID| bits.

In the non-shared system from Section 3 the (F_p,Q,q)-part of the public key data can be reconstructed, at the cost of one modular squareroot, from ID, B, and an additional 3 + 32 + 8 = 43 bits, as shown in Section 3. Since B may be assumed to be a system wide parameter (of value approximately equal to L/2), the total amount of data to be exchanged or certified is L + 44 + |ID| bits. Compared to the shared system, 43 more bits have to be carried along, and one additional modular squareroot has to be performed during key reconstruction. An additional 43 bits and single modular squareroot are a small price to pay for the additional security obtained.

Compared to the traditional non-shared system, during key exchange or certification 4.5L − 42 bits are saved by the method from Section 3, and the same number of modular squareroots is required. Furthermore, public key generation according to the new method is straightforward, whereas traditional non-shared systems require curve point counting software. The latter can be done using either complex multiplication (CM) techniques (thereby restricting the range of curves that can be used) or using the Schoof-Elkies-Atkin (SEA) algorithm (for truly random curves). Both the CM-based and the SEA-based methods are considerably more complicated than the approach from Section 3. Intuitively, the security of the SEA approach ranks highest, followed by CM, followed by the new method (cf. [2]), but as explicitly stated by the same authoritative source, there is no evidence whatsoever that this intuition is correct. Thus, given the current state of the art, the security offered by the newly proposed non-shared system and either type of traditional non-shared system (CM or SEA) seems to be the same.

Acknowledgments. Acknowledgments are due to Rene Schoof for providing the rows for d > 3 of Table 1 and for helpful discussions.

References

1. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.): Advances in Cryptology – Asiacrypt'98. Lecture Notes in Computer Science, Vol. 1514. Springer-Verlag, Berlin Heidelberg New York, (1998) 51-65
2. Frey, G.: Remarks made during lecture at ECC'98, Waterloo, 1998

3. Koblitz, N.: Constructing elliptic curve cryptosystems in characteristic 2. In: Menezes, A.J., Vanstone, S.A. (eds.): Advances in Cryptology – Crypto‘90. Lecture Notes in Computer Science, Vol. 537. Springer-Verlag, Berlin Heidelberg New York, (1991) 156-167
4. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. 48 (1987) 243-264
5. Schoof, R.: Private communication, 1997
6. Silverman, J.H.: The Arithmetic of Elliptic Curves, Graduate Texts in Mathematics, Vol. 106. Springer-Verlag, Berlin Heidelberg New York, (1986)