Henk van Tilborg
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
H.C.A.v.Tilborg@tue.nl

Overzichtsartikel

Elliptic curve cryptosystems;

Er bestaan vele publieke-sleutel cryptosystemen. Het vakgebied is sterk in beweging en wordt gekenmerkt door hevige controverses. Dat is logisch, want met het beveiligen van gegevens zijn grote commerciële belangen gemoeid. Op het Mathematisch Congres 2001 in Amsterdam presenteerde Eric Verheul (Pricewaterhouse Coopers) in een van de hoofdvoordrachten het cryptosysteem gebaseerd op de discrete logarithme. Hij beweerde ondermeer dat cryptosystemen gebaseerd op elliptische krommen trager zijn en eigenlijk achterhaald. Henk van Tilborg geeft hier een andere visie.

An exponentiation, say the computation of 3^i, can be performed pretty efficiently by using the binary expansion of the exponent i. For instance, the binary expansion of $i = 2185$ is given by 100010001001 and results in:

$$3^{2185} = \left(\left(\left(\left(\left(3^{2^4}\right)3\right)^{2^4}\right)3\right)^{2^3}\right)3.$$

Also *modular* exponentiation, say the computation of $3^{2185} \pmod{3583}$, can be done efficiently: after each squaring or after each multiplication by 3 reduce the result modulo 3583 and then continue. (The answer is 220.)

To solve $3^i \equiv 2217 \pmod{3583}$, i.e., to determine $\log_3 2217$ in $\mathbf{Z}_{3583}$, no method is known with a similar low complexity. This observation forms the basis of a modern public key cryptosystem: the discrete logarithm system (see [2]). The reason why it is easy to take a logarithm but not a modular/discrete logarithm may become clear by comparing the two pictures in Figure 1.

For the sake of completeness we mention that it is easy to solve $g^{2185} \equiv 2217 \pmod{3583}$. Indeed, 3583 is a prime number and so we know by a theorem of Fermat that $g^{3582} \equiv 1 \pmod{3583}$. It follows that if we raise both hands to the power 1741 (the multiplicative inverse of 2185 modulo 3582) we find the solution $g = 2217^{1741} = 68$. However, if the modulus is the product of two primes, finding the solution becomes, in general, again infeasible for large moduli, simply because the factorization of large numbers is infeasible. The RSA cryptosystem makes use of this observation (see [11]).

Here, we shall discuss the Discrete Logarithm system. In the next section, we shall explain how this system can be used by two parties who can only communicate over a public channel (radio, telephone, internet; all with potential eavesdroppers), to agree on some secret number. This method is called the Diffie-Hellman key exchange [2]. The common, secret number can be used by them as a key in a conventional cryptosystem. In the same section, we shall give an idea what kind of techniques are available to take discrete logarithms. More importantly, the computational complexities of these methods will be given. This is relevant for the choice of the field $\mathbf{Z}_p$. If we take p sufficient large, even the fastest method to take discrete logarithms is no threat to the security of the Diffie-Hellman key exchange.

In 1985, Victor Miller (IBM, [9]) and Neil Koblitz (University of Washington) realized that the additive group associated with an elliptic curve can be used for a similar 'public' key-exchange method. In the same year, Scott Vanstone together with other re-

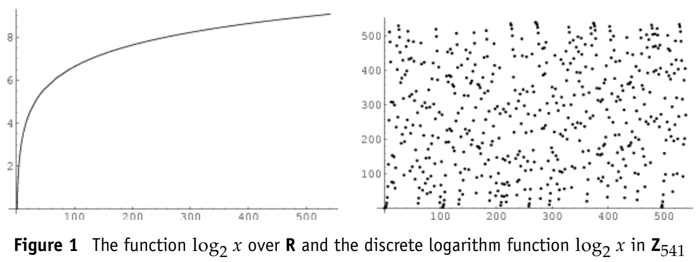

Figure 1 The function $\log_2 x$ over **R** and the discrete logarithm function $\log_2 x$ in $\mathbf{Z}_{541}$

too good to be true?

searchers of the University of Waterloo in Ontario, Canada, started a company called Mobius, later renamed into Certicom, to commercialize this idea. Now, about three hundred people are employed by Certicom. Apparently, the people around Vanstone realized that the invention by Miller and Koblitz was not just an interesting mathematical generalization of the original Diffie-Hellman idea, but that some aspects of it could lead to a very promising practical (read: commercial) alternative. We shall explain their idea in greater detail in the section *Elliptic curve systems* below. The crux of their observation is that the only methods to take discrete logarithms with subexponential complexity (time and memory) can not be generalized to the elliptic curve setting. So, only algorithms remain with exponential complexity. This allows the choice of much smaller prime numbers and thus more manageable parameters with the same level of security.

The discrete logarithm cryptosystem

Two people, say Alice and Bob, want to communicate privately over a connection that is not secure. They can use a *symmetric* or *secret key* cryptosystem to tackle this problem. In such a system, both parties need to know a common secret, called the *key*. In the Enigma, which was used by the Germans in WW-II, the key consisted of the choice and the starting positions of three rotors [5]. In modern secret key cryptosystems the key is a string of 128, 192 or even more bits. The problem is that Alice and Bob have never met before and do not have a common secret key. They use the Diffie-Hellman key exchange protocol to agree on this key. Note that their communication takes place over a public channel, where eavesdroppers may listen in.

Alice and Bob use a very large prime number p and an element g in $\mathbf{Z}_p$ of sufficiently large multiplicative order q (think of 128 bits long). In the sequel, we shall assume that q is prime (note that q divides $p-1$). Alice can choose the parameters p and g and tell them to Bob or they can use standard parameters given by the communication network. In all cases, an eavesdropper will also know these parameters.

Next, Alice and Bob each choose a random exponent less than q, say s_A and s_B. They compute $k_A = g^{s_A}$, resp. $k_B = g^{s_B}$ in $\mathbf{Z}_p$ and exchange these values over the public channel. They keep their exponent s_A resp. s_B secret. The common, secret value of Alice and Bob is given by

$$g^{s_A s_B} \text{ in } \mathbf{Z}_p,$$

which Alice can compute from $(k_B)^{s_A}$ and Bob from $(k_A)^{s_B}$.

There is a practical complication with this system. How does Alice know for sure that s_B is indeed Bob's public value? You can think of a public directory (like a telephone book) in which all these values are listed, but that is not a very practical solution. More likely, you want some trustworthy authority to sign each public value as authentic.

Clearly, the key exchange system, described above, is broken as soon as an eavesdropper can determine s_A from the known k_A (or s_B from k_B). This forces us to look at various techniques to solve

$$g^s \equiv k \pmod{p}, \tag{1}$$

where g, k, and p are known and s needs to be determined. The problem of how to solve (1) is called the *discrete logarithm* problem. For further reading on the techniques described below, we refer the reader to [8], an excellent handbook, or [14], a textbook with interactive cd-rom.

Ways to determine a discrete logarithm

A *brute-force* way to find s would be to try $s = 0, 1, 2, \ldots$ until a solution is found or, alternately, to put $g^0, g^1, g^2, \ldots$ in a table and

9	(4,6)	(5,5)	(1,7)	(7,5)	(0,0)	(5,3)	(1,8)	(3,5)	(8,4)	(7,9)
8	(1,8)	(2,2)	(6,5)	(9,9)	(9,6)	(2,0)	(4,4)	(6,1)	(2,3)	(9,1)
7	(7,9)	(1,0)	(2,4)	(2,3)	(4,8)	(0,3)	(3,7)	(8,2)	(4,2)	(6,5)
6	(9,9)	(0,6)	(1,5)	(8,5)	(7,3)	(8,4)	(9,5)	(4,3)	(3,8)	(1,3)
5	(4,2)	(9,9)	(7,8)	(4,8)	(3,1)	(5,1)	(9,7)	(1,9)	(3,7)	(6,9)
4	(1,2)	(0,4)	(7,6)	(6,4)	(0,5)	(1,6)	(8,7)	(5,7)	(1,7)	(5,8)
3	(0,2)	(5,0)	(0,9)	(4,9)	(2,6)	(3,3)	(7,3)	(5,0)	(8,7)	(6,9)
2	(3,9)	(1,8)	(5,5)	(4,7)	(0,1)	(4,7)	(0,1)	(4,8)	(3,7)	(0,6)
1	(1,1)	(2,2)	(7,9)	(8,5)	(5,1)	(4,6)	(8,7)	(5,2)	(1,6)	(5,3)
0	(3,1)	(3,2)	(0,6)	(4,3)	(0,2)	(1,7)	(5,4)	(7,9)	(0,1)	(9,5)
	0	1	2	3	4	5	6	7	8	9

Table 1 A random mapping f from $\{0,1,\ldots,9\}^2$ to itself.

look for k. Either way, the complexity is p. If p consists of t bits, we can say that the complexity is given by 2^t, so the complexity grows exponentially in t.

A nice way to balance the time complexity with the available memory is the *baby-step giant-step* method. Suppose that one has enough memory available to store m elements of $\mathbf{Z}_p$. Compute $g^0, g^1, g^2, \ldots, g^{m-1}$, sort these elements to facilitate an easy look-up and put them in a table. Note that the exponents increase by 1, the baby-steps. Next one checks if k is in the table, if not one checks if k/g^m is in the table, if not check k/g^{2m}, etcetera (the giant-steps). When k/g^{jm} is in the table, say it equals g^i, $0 \le i \le m-1$, one has found the unknown exponent: $s = jm + i$. The time complexity of the baby-step giant-step method is p/m, so the product of memory requirement and time complexity is still $p \approx 2^t$.

Two related techniques to determine the solution s of (1) are called the *Pollard-ρ* and the *Pollard-λ* method [10]. We shall only explain the first method. We still assume that g generates a subgroup of prime order q. We start with the following observation about random functions.

Let f be a random mapping from a finite set A to itself. Select a random element a_0 in S and define the sequence $\{a_i\}_{i\ge 0}$ recursively by $a_{i+1} = f(a_i)$. The sequence $\{a_i\}_{i\ge 0}$ will eventually cycle, because A is finite. The expected length of this cycle and the expected length of the beginning segment until the cycle starts are both given by $\sqrt{\pi|A|/8}$ (see [3]). This situation is depicted graphically by the ρ in Figure 2, where we have taken $A = \{0,1,\ldots,9\} \times \{0,1,\ldots,9\}$ and where $f(i,j)$ is given by the entry at place (i,j) in the rectangle given in Table 1.

Returning to the problem of finding the solution of (1) the idea now is to define a recursion on $\mathbf{Z}_p$. It turns out that we need to keep track of two additional parameters. We define the mapping $F : \mathbf{Z}_p \times \mathbf{Z}_q \times \mathbf{Z}_q \to \mathbf{Z}_p \times \mathbf{Z}_q \times \mathbf{Z}_q$ by

$$F(x,u,v) = \begin{cases} (x^2, 2u, 2v), & \text{if } x \equiv 0 \pmod 3, \\ (kx, u, v+1), & \text{if } x \equiv 1 \pmod 3, \\ (gx, u+1, v), & \text{if } x \equiv 2 \pmod 3. \end{cases}$$

The sequence $\{(x_i, u_i, v_i)\}_{i\ge 0}$ is defined recursively by

$$\begin{cases} (x_0, u_0, v_0) = (1,0,0), \\ (x_{i+1}, u_{i+1}, v_{i+1}) = F(x_i, u_i, v_i). \end{cases}$$

It is easy to verify with an induction argument that $x_i = g^{u_i} k^{v_i}$ for all $i \ge 0$. For instance, if $x \equiv 0 \pmod 3$, one has $g^{u_{i+1}} k^{v_{i+1}} = g^{2u_i} k^{2v_i} = (g^{u_i} k^{v_i})^2 = (x_i)^2 = x_{i+1}$.

We need to find indices $0 \le i < j$ with $x_i = x_j$ but we want to avoid having to store all intermediate values $x_0, x_1, \ldots$. To this end, we only compare the first coordinates of (x_i, u_i, v_i) and (x_{2i}, u_{2i}, v_{2i}) for $i > 0$. If $x_i \ne x_{2i}$, we calculate $(x_{i+1}, u_{i+1}, v_{i+1}) = F(x_i, u_i, v_i)$ and $(x_{2i+2}, u_{2i+2}, v_{2i+2}) = F(F(x_{2i}, u_{2i}, v_{2i}))$ and compare their first coordinates again.

When $x_i = x_{2i}$, $i > 0$, we have $g^{u_i} k^{v_i} = g^{u_{2i}} k^{v_{2i}}$, i.e., $g^{u_i} g^{s.v_i} = g^{u_{2i}} g^{s.v_{2i}}$. From this almost always the unknown exponent s can be calculated: $s = (u_{2i} - u_i)/(v_i - v_{2i}) \pmod{q-1}$. (In case that $\gcd(v_i - v_{2i}, p-1) \ne 1$, one can solve $kg^l \equiv g^{s+l} \pmod p$ with Pollard's method for $l = 1, 2, \ldots$ until a solution is found.)

It follows from the ρ shape of the $\{x_i\}_{i\ge 0}$-walk that the expected running time of this algorithm is $\sqrt{\pi p/2} \approx 2^{t/2}$, while the memory requirements are a very small constant.

As an example, we try to solve $121^s \equiv 3435 \pmod{4679}$. Note that 121 has multiplicative order $q = 2339$. Starting with $(x_0, u_0, v_0) = (1,0,0)$, we find that $x_{76} = x_{152}$ with $a_{76} = 84, b_{76} = 2191, a_{152} = 286, b_{152} = 915$. From this one can obtain $s \equiv (286-84)/(2191-915) \equiv 1111 \pmod{2339}$.

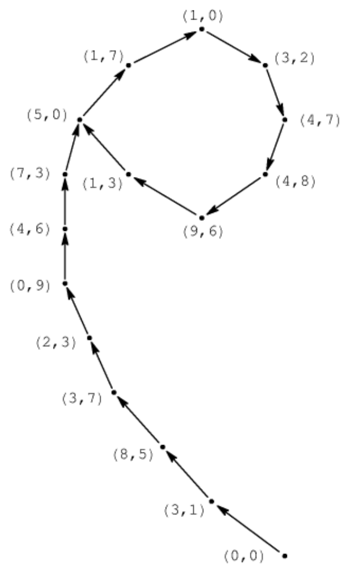

Figure 2 The ρ-shaped walk in $\{0,1,\ldots,9\}^2$ starting in $(0,0)$.

Index-calculus

We shall only give a typical example of this method. Suppose that one needs to solve:

$$11^k \equiv 3333 \pmod{4679}.$$

We start with a precalculation that is independent of the right-hand side 3333. We consider the set $\{2,3,5,7,11\}$, consisting of the first five prime numbers (this set is called a *factor base*) and try to solve the logarithm problem for all elements in the factor bases. In order words, we want to solve

$$\begin{aligned} 11^{k_1} &\equiv 2 \pmod{4679}, \\ 11^{k_2} &\equiv 3 \pmod{4679}, \\ 11^{k_3} &\equiv 5 \pmod{4679}, \\ 11^{k_4} &\equiv 7 \pmod{4679}, \\ 11^{k_5} &\equiv 11 \pmod{4679}. \end{aligned}$$

At first glance, it may look like we have made the situation worse instead of better, but that is not true. Select a random exponent r, compute $11^r \pmod{4679}$, and check if the remainder can be factored completely by means of the factor base. For instance, $11^{2208} \equiv 4182 \pmod{4679}$ but $4182 = 2^1 \times 3^1 \times 697$ and 697 can not be factored further over $\{2,3,5,7,11\}$.

Note that we do not need to factor the righthand side, we only have to divide it by the elements in the factor base. Note also that the larger the factor base, the easier we find righthand sides that do completely factor with respect to the factor base, but the price we pay is having more unknowns k_i to determine. As soon as a right hand side does completely factor with respect to the factor base, we get a linear relation between the unknown k_i's. For instance,

$$11^{1006} \equiv 315 = 3^2.5.7 \pmod{4679}$$

gives the relation

$$1006 \equiv 2 * m_2 + m_3 + m_4 \pmod{4678}.$$

Collect enough linear relations to solve the unknown k_i's. For instance, from

$$\begin{aligned}
11^{104} &\equiv 1280 = 2^8.5 \pmod{4679},\\
11^{208} &\equiv 750 = 2.3.5^3 \pmod{4679},\\
11^{1006} &\equiv 315 = 3^2.5.7 \pmod{4679},\\
11^{2303} &\equiv 198 = 2.3^2.11 \pmod{4679},\\
11^{3506} &\equiv 4050 = 2.3^4.5^2 \pmod{4679},
\end{aligned}$$

we get the linear relations

$$\begin{aligned}
104 &\equiv 8k_1 + k_3, \quad \pmod{4678},\\
208 &\equiv k_1 + k_2 + 3k_3 \pmod{4678},\\
1006 &\equiv 2k_2 + k_3 + k_4 \pmod{4678},\\
2303 &\equiv k_1 + 2k_2 + k_5 \pmod{4678},\\
3506 &\equiv k_1 + 4k_2 + 2k_3 \pmod{4678}.
\end{aligned}$$

The solution is given by $k_1 \equiv 352$, $k_2 \equiv 3314$, $k_3 \equiv 1966$, $k_4 \equiv 1768$, and $k_5 \equiv 1$, all modulo 4678.

We are now ready to solve the original problem: $11^k \equiv 3333 \pmod{4679}$. Pick a random exponent r and check if $3333 \times 11^r \pmod{4679}$ completely factors over the factor base. After a few tries we find that

$$3333 \times 11^{573} \equiv 540 = 2^2.3^3.5 \pmod{4679}.$$

From this we get

$$k + 573 \equiv 2k_1 + 3k_2 + k_3 \pmod{4678}.$$

Since the k_i's are now known, it is easy to determine k. One obtains the solution $k \equiv 2 \times 352 + 3 \times 3314 + 1 \times 1966 - 573 \equiv 2683 \pmod{4678}$. Indeed, $11^{2683} \equiv 3333 \pmod{4679}$.

The time complexity of the above method is given by $e^{1.923\, t^{1/3}(\ln t)^{2/3}}$ (see [4]). The memory requirement typically equals the square root of this number. This makes the index calculus algorithm and its variations the only method for taking discrete logarithms with a subexponential complexity (see also Table 2).

	time	memory
Exhaustive key search	2^t	1
Baby-step Giant-step	$2^t/m$	m
Pollard	$2^{t/2}$	1
Index calculus	$e^{1.923\, t^{1/3}(\ln t)^{2/3}}$	$e^{1.923\, t^{1/3}(\ln t)^{2/3}/2}$

Table 2 The complexity of different methods to take discrete logarithms for $p \approx 2^t$.

Elliptic curve cryptosystems

Although it is very natural to use the multiplicative group of a finite field to set up the discrete logarithm system, one may equally well use a (sufficient large) subgroup of it or even any other cyclic group. Victor Miller and Neil Koblitz proposed in 1985 a variation that makes use of the additive group associated with an elliptic curve. In our explanation, we restrict ourselves again to $\mathbf{Z}_p$ with p prime.

Definition 1. *Let $a, b, c \in \mathbf{Z}_p$. The* elliptic curve *$\mathcal{E}$ over $\mathbf{Z}_p$ is the set of points (x, y) satisfying*

$$y^2 = x^3 + ax^2 + bx + c, \tag{2}$$

together with a point $\mathcal{O}$, called the point at infinity.

Two examples of elliptic curves over $\mathbf{R}$ are depicted in Figure 3. The point $\mathcal{O}$ can be best viewed as the intersection point at infinity of all vertical lines. Clearly, if the right hand side in (2) is positive for some value of x there will be two points on $\mathcal{E}$ with that x coordinate, one with a positive y-coordinate and its mirror image. If the right hand side is zero there is only a single value (being $y = 0$) and there is no solution if the right hand side is negative.

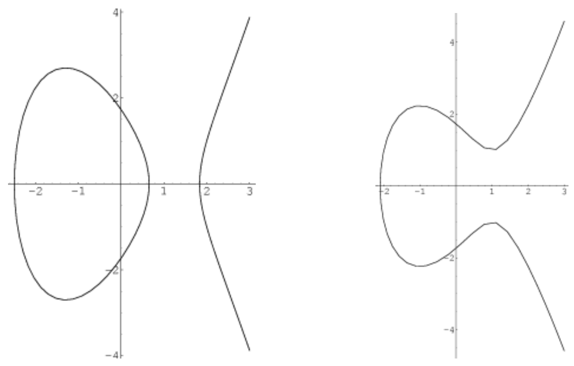

Figure 3 The EC-curves $y^2 = x^3 - 5x + 3$ and $y^2 = x^3 - 3x + 3$ over **R**.

Over $\mathbf{Z}_p$ the situation is exactly the same: two solutions if $x^3 + ax + bx + c$ is a quadratic residue modulo p, one solution if it is zero and no solution otherwise. There is no exact formula for the number of points on a EC-curve over $\mathbf{Z}_p$. In [13] one can find the following bound.

Theorem (Hasse) *The number of points $\mathcal{N}$ on an elliptic curve over $\mathbf{Z}_p$ satisfies:*

$$|N - (p+1)| \le 2\sqrt{p}.$$

Algorithms exist to count the number of points on a EC-curve precisely, e.g. [12]. However, these methods are not as efficient as one would like them to be and much research remains to be done in this area.

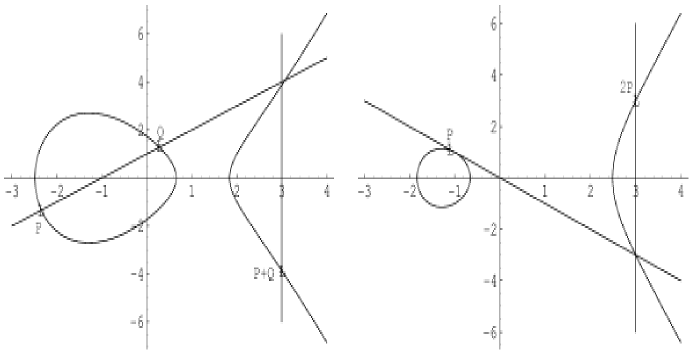

Figure 4 Addition over EC-curves

Elliptic curves have the nice property that any line through two points on $\mathcal{E}$ will intersect $\mathcal{E}$ in a third point (for a vertical line the point $\mathcal{O}$ plays that role). The same statement is true for a (double) tangent on $\mathcal{E}$. It will intersect $\mathcal{E}$ in a third point. The reason for this is simple. Let $y = mx + n$ be the equation of the line l through (x_1, y_1) and (x_2, y_2) both on $\mathcal{E}$. Substitution of $y = mx + n$ in (2) gives the third degree equation $(mx + n)^2 = x^3 + ax^2 + bx + c$. Since x_1 and x_2 are two of its roots, there must be a third one: x_3. The point (x_3, y_3), with $y_3 = mx_3 + n$, is the third point of intersection of l with $\mathcal{E}$. Other cases can be verified similarly.

We are now ready to define the group operation.

Definition 2. *Addition on $\mathcal{E}$ is defined by:*
1. $\mathcal{O} + \mathcal{P} = \mathcal{P} + \mathcal{O} = \mathcal{P}$,
2. *if* $\mathcal{P} = (x, y)$ *then* $-\mathcal{P} = (x, -y)$,
3. *otherwise* $\mathcal{P} + \mathcal{Q} = -\mathcal{R}$, *where* $\mathcal{R}$ *is the third of intersection of the line through* $\mathcal{P}$ *and* $\mathcal{Q}$.

In Figure 4, typical cases for this addition are depicted over $\mathbf{R}$. The reader should realize that the addition formulas do not involve complicated functions. Indeed, in the general case, $x_1 + x_2 + x_3$ is equal to minus the coefficient of x^2 in the third degree equation above.

Since addition on $\mathcal{E}$ is easy, it is also simple to compute a scalar multiple $s\mathcal{P}$. Again, the binary expansion of the scalar s helps. For instance, for $s = 2185$ with binary expansion 100010001001, one gets:

$$2185\mathcal{P} = 2(2(2(2(2(2(2(2(2(2(2\mathcal{P}))) + \mathcal{P})))) + \mathcal{P}))) + \mathcal{P}.$$

When the order of $\mathcal{P}$ is sufficiently high, the opposite problem, i.e. to determine s such that $s\mathcal{P} = \mathcal{Q}$ for given $\mathcal{P}$ and $\mathcal{Q}$ on $\mathcal{E}$ is again, as with the discrete logarithm, intractable in general.

As an example, we consider the elliptic curve $\mathcal{E}$ given by $y^2 = x^3 + 100x^2 + 10x + 1$ over $\mathbf{Z}_{863}$. The point $\mathcal{P} = (121, 517)$ on $\mathcal{E}$ has order 432. Computing $300\mathcal{P}$ from $\mathcal{P}$ is straightforward, but to solve $s\mathcal{P} = \mathcal{Q}$ for $\mathcal{Q} = (101, 496)$ on $\mathcal{E}$ one can not do a lot better than to try $s = 1, 2, 3, \ldots$

It is now easy to set up a Diffie-Hellman-like key exchange system on an elliptic curve (or to copy all later generalizations). Start with an elliptic curve $\mathcal{E}$ (not all are suitable, just like not all prime numbers are suitable in the original scheme) and a point $\mathcal{P}$ on $\mathcal{E}$ of sufficiently high (additive) order. Let this order be denoted by v.

Alice and Bob (and every other participant) choose a random coefficient, say s_A and s_B, respectively, both less than v. Alice and Bob compute the scalar multiples $\mathcal{K}_A = s_A\mathcal{P}$ and $\mathcal{K}_B = s_B\mathcal{P}$, respectively, and make them public or exchange them in some public way. They can now both compute

$$\mathcal{S}_{A,B} = s_A s_B \mathcal{P} = s_A \mathcal{K}_B = s_B \mathcal{K}_A.$$

Indeed, Alice knows the public $\mathcal{K}_B$ of Bob and her own secret s_A and can compute $s_A\mathcal{K}_B = \mathcal{S}_{A,B}$. For Bob a similar situation holds. Since the x-coordinate of $\mathcal{S}_{A,B}$ determines the y-coordinate apart from a single bit (the sign), Alice and Bob should only use the x-coordinate as common secret key.

An overview of the Diffie-Hellman key exchange system over elliptic curves is given in Table 3. Note that an exponentiation in the original system translates into a scalar multiplication, just like a multiplication in the original system translates into an addition.

system parameters	elliptic curve $\mathcal{E}$ over $\mathbf{Z}_p$ point $\mathcal{P}$ on $\mathcal{E}$ of high order v
secret key of user U	s_U, $0 < s_U < v$
public key of user U	the point $\mathcal{K}_U = s_U\mathcal{P}$
common key of Alice and Bob	the point $\mathcal{S}_{A,B} = s_A s_B \mathcal{P}$
Alice computes	$\mathcal{S}_{A,B} = s_A\mathcal{K}_B$
Bob computes	$\mathcal{S}_{A,B} = s_B\mathcal{K}_A$

Table 3 The Diffie-Hellman key exchange system over elliptic curves.

The importance of EC-cryptosystems

So, why is it interesting to generalize the Diffie-Hellman key exchange system to the elliptic curve setting? Maybe at the beginning, some sceptics may have thought that this was a generalization just for the sake of the generalization or that it was motivated by patent aspects. Both assumptions were incorrect, as it turned out that there was much more to it. To understand this, one has to look at the various methods to take discrete logarithms and see if they can be adapted to the elliptic curve setting.

Obviously, exhaustive search and baby-step giant step can be generalized. For instance, one can make a table of $\mathcal{O}, \mathcal{P}, 2\mathcal{P}, \ldots, 9\mathcal{P}$ and then check if $\mathcal{Q}$ is in the table. If it is not, check if $\mathcal{Q} - 10\mathcal{P}$ is in the table, otherwise check if $\mathcal{Q} - 20\mathcal{P}$ is in the table, etcetera.

Also Pollard's method can be generalized very naturally (see [1]). The index calculus method has defeated any attempt to generalize it to the elliptic curve setting. The reason seems to lie in the fact that while one can define a 'natural' factor base in the set of integers (e.g. the first so many prime numbers) and then check quite easily if an integer factors completely over this factor base, a similar thing can not be done for the points on an elliptic curve. (In the ring of polynomials a natural factor base consists of a finite set of irreducible polynomials. This is relevant for the elliptic curve cryptosystems defined over finite fields.)

The above observation is extremely important and explains all the interest in EC-cryptosystems (ECC's). To keep the Diffie-Hellman key exchange over $\mathbf{Z}_p$ secure even the index-calculus method with its subexponential complexity should be infeasible. This forces the designers of the system to use very large prime numbers p (or, if the finite field $GF(q)$ is used instead of $\mathbf{Z}_p$, q has to be taken very large). With the Diffie-Hellman key exchange over an elliptic curve one does not have to worry about the index calculus method (or related techniques). This leaves the designer with Pollard's method which has an exponential running time. To see how these complexities compare, see Table 4. Quite clearly, much smaller prime numbers suffice when using elliptic curve cryptosystems than when working over finite fields. This comparison is not completely fair. For instance, more constants need to be stored to describe an elliptic curve. To get an impression of the security of a cryptosystem we refer the reader to [6]. Depending on assumptions that one wants to make, like the expected increase of computer power per year and the number of years that the system has to remain secure, the program in [6] will calculate the necessary key lengths.

In the regular Diffie-Hellman key exchange some parameter choices are not safe. For instance, prime numbers p for which $p-1$ only has small prime factors have to be avoided. A similar thing holds for elliptic curve cryptosystems. The strongest attack seems to be the MOV attack [7]. It reduces the logarithm problem over an elliptic curve to the regular logarithm problem over a finite field. The MOV attack can be applied to so-called singular and super-singular elliptic curves. This means that the subexponential algorithms can be applied again and that nothing can be gained by using cryptosystems over such curves! Luckily, it is quite easy to test and avoid these curves.

Only one worry remains among applied cryptographers: since elliptic curves have not been studied as intensively as the classical discrete logarithm problem over $\mathbf{Z}_p$ and much less than the factorization problem, new insights may make the whole idea of elliptic curve cryptosystems worthless (from an application point of view). Only the future can show if this worry is justified. On the other hand, with every year that elliptic curve cryptosystems are around such dramatic events seem less and less likely. For further reading, we refer the reader to [1].

As a final remark, we note that it is only natural to consider other group structures to set up a Diffie-Hellman-like key exchange system. However, all current proposals of this kind seem to lack an efficient implementation.

# digits of p	Pollard	index calculus
100	1.13 10^{15}	6.79 10^{15}
150	3.78 10^{22}	1.03 10^{19}
200	1.27 10^{30}	4.05 10^{21}
250	4.25 10^{37}	6.87 10^{23}
300	1.43 10^{45}	6.49 10^{25}

Table 4 Pollard's method compared with the index calculus method.

Applications

As was noted before, the security of elliptic curve based cryptosystems grows exponentially in its parameter, while alternative systems as RSA have a subexponential security. To give another comparison: a 256 bits EC cryptosystem should be compared to a 3072 bits RSA modulus. The computational overhead of both systems grows like n^3, where n is the key length in bits.
When implementing cryptosystems one often has to deal with practical constraints like memory requirement, processing time, or power consumption (for instance on smartcards). A fast growing application area is wireless communication. It is here that ECC compares very favorably with other systems. ECC's can be implemented on smartcards without mathematical coprocessor. Contactless smartcards only work with ECC because other systems require too much induction energy. Also for the wireless communication with or between handhelds, like the Palm Pilot, EEC offers advantages that other systems can not offer: a shorter key generation, a shorter handshaking protocol, etc. For this reason, one will very likely see ECC applied in future applications of e-commerce, for instance when people use their handheld to sign their business transactions. Already at this moment, there are many international standards involving ECC: ISO, ANSI, IEEE, and SECG.

Acknowledgement

The author likes to thank Rene Struik for his valuable comments and input.

References

1 I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series, Vol. 265, Cambridge: Cambridge University Press, 1999.

2 W. Diffie and M.E. Hellman, *New directions in cryptography,* IEEE Trans. Inf. Theory, IT-22, pp. 644–654, Nov. 1976.

3 P. Flajolet and A.M. Odlyzko, *Random mapping statistics,* in Advances in Cryptography: Proc. of Eurocrypt '89, J.-J. Quisquater and J. Vandewalle, Eds., Lecture Notes in Computer Science 434, Springer Verlag, Berlin etc., pp. 329–354, 1990.

4 D.M. Gordon, *Discrete logarithms in $GF(p)$ using the number field sieve,* SIAM Journal on Discrete Mathematics, **6**, pp. 124-138, 1993.

5 A.G. Konheim, *Cryptography, a primer,* John Wiley & Sons, New York, etc., 1981.

6 A.K. Lenstra and E.R. Verheul, *Selecting Cryptographic Key Sizes,* http://www.cryptosavvy.com/

7 A. Menezes, T. Okamoto, and S. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field,* IEEE Transactions on Information Theory, IT-39, pp. 1639–1646, 1993.

8 Menezes, A.J., P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography,* CRC Press, Boca Raton, etc. 1997.

9 S. Miller, *Use of elliptic curves in cryptography,* abstract in Advances in Cryptography: Proc. of Crypto '85, H.C. Williams, Ed., Lecture Notes in Computer Science 218, Springer Verlag, Berlin etc., p. 417, 1986.

10 J.M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. Comp. **32**, pp. 918–924, 1978.

11 R.L. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems,* Comm. ACM, Vol. **21**, pp. 120–126, Febr. 1978.

12 R. Schoof, *Counting points on elliptic curves over finite fields*, Journal de Théorie des Nombres de Bordeaux, 7, pp. 219–254, 1995.

13 J.H. Silverman, *The Arithmetic of Elliptic Curves,* Springer Verlag, Berlin, etc., 1986.

14 Henk C.A. van Tilborg, *Fundamentals of cryptology; A professional reference and interactive tutorial,* Kluwer Academic Publishers, Boston etc., 2000.