A One Round Protocol for Tripartite Diffie–Hellman

Antoine Joux

SCSSI, 18, rue du Dr. Zamenhoff
F-92131 Issy-les-Mx Cedex, France
Antoine.Joux@ens.fr

Abstract. In this paper, we propose a three participants variation of the Diffie–Hellman protocol. This variation is based on the Weil and Tate pairings on elliptic curves, which were first used in cryptography as cryptanalytic tools for reducing the discrete logarithm problem on some elliptic curves to the discrete logarithm problem in a finite field.

1 Introduction

Since its discovery in 1976, the Diffie–Hellman protocol has become one of the most famous and largely used cryptographic primitive. In its basic version, it is an efficient solution to the problem of creating a common secret between two participants. Since this protocol is also used as a building block in many complex cryptographic protocols, finding a generalization of Diffie–Hellman would give a new tool and might lead to new and more efficient protocols.

In this paper, we show that the Weil and Tate pairings can be used to build a tripartite generalization of the Diffie–Hellman protocol. These pairings were first used in cryptography as cryptanalytic tools to reduce the complexity of the discrete logarithm problem on some "weak" elliptic curves. Of course, the problem of setting a common key between more than two participants has already been addressed (see the protocol for conference keying in [1]). However, all the known techniques require at least two round of communication. In some protocols having these two rounds can be somewhat cumbersome, and a single round would be much preferable. To give an example, exchanging an email message key with a two round Diffie–Hellman protocol would require both participants to be connected at the same time, which is a very undesirable property for a key exchange protocol. For this reason, we believe that the one round tripartite Diffie–Hellman presented here is a real improvement over conference keying even though the computational cost will be somewhat higher.

2 The Discrete Logarithm Problem on Weak Elliptic Curve

The discrete logarithm problem on elliptic curves is now playing an increasingly important role in cryptography. When elliptic curve cryptosystems where first proposed in [9], computing the number of points of a given curve was a challenging task, since the Schoof, Elkies and Atkin algorithm was not yet mature (for a survey of this algorithm see [6]). For this reason and also to simplify the addition formulas, the idea of using special curves quickly arose. However, it was shown later on that some of these special cases are not good enough. Today, three weak special cases have been identified. In one of them, the discrete logarithm problem becomes easy (i.e. polynomial time) as was shown in [11,10]. This easiest case happens when the number of points of the elliptic curve over $\mathbb{F}_p$ is exactly p. In the two other cases, the discrete logarithm problem on the elliptic curve is transformed into a discrete logarithm problem in a small extension of the field of definition of the elliptic curve. These two reductions are called the Menezes, Okamoto, Vanstone (MOV) reduction [8] and the Frey, Rück (FR) reduction [3]. A survey of these reductions was published at Eurocrypt'99 [4], and gave a comparison of these two reductions. The conclusion was the FR reduction can be applied to more curves than the MOV reduction and moreover that it can be computed faster than the MOV reduction. Thus for all practical usage, the authors recommend the FR reduction. However, they claim that the computation of the FR and MOV reduction may be a heavy load. We will show that in fact this is not the case and that these reductions can be turned from cryptanalytic to cryptographic tools.

Pairings on Elliptic Curve

The MOV and FR reductions are both based on a bilinear pairing, in the MOV case it is the Weil pairing and in the FR case it is (a variation of) the Tate pairing. In the sequel, we describe these pairings for an elliptic curve E defined over $\mathbb{F}_p$. In order to define these pairings, we first need to introduce the function field and the divisors of the elliptic curve. Very informally, the function field $K(E)$ of E is the set of rational map in x and y modulo the equation of E (e.g. $y^2 - x^3 - ax - b$). A divisor D is an element of the free group generated by the points on E, i.e. it can be written as a finite formal sum: $D = \sum_i a_i(P_i)$, where the P_i are points on E and the a_i are integers. In the sequel, we will only consider divisors of degree 0, i.e. such that $\sum_i a_i = 0$.

Given any function f in $K(E)$, we can build a degree 0 divisor $div(f)$ from the zeros and poles of f simply by forming the formal sum of the zeroes (with multiplicity) minus the formal sum of the poles (with multiplicity). Any divisor $D = div(f)$ will be called a principal divisor. In the reverse direction, testing whether a degree 0 divisor $D = \sum_i a_i(P_i)$ is principal or not, can be done by evaluating $\sum a_i P_i$ on E. The result will be the point at infinity if and only if D is principal.

Given a function f in $K(E)$ and a point P of E, f can be evaluated at P by substituting the coordinates of P for x and y in any rational map representing f. The function f can also be evaluates at a divisor $D = \sum_i a_i(P_i)$, using the following definition:

$$f(D) = \prod_i f(P_i)^{a_i}.$$

Using these notions, we can now define the Weil pairing: it is a bilinear function from the torsion group $E[n]$ to the multiplicative group μ_n of n-th roots of unity in some extension of $\mathbb{F}_p$, say $\mathbb{F}_{p^k}$. Given two n-torsion points P and Q, we compute their pairing $e_n(P,Q)$ by finding two functions f_P and f_Q such that $div(f_P) = n(P) - n(O)$ and $div(f_Q) = n(Q) - n(O)$, and by evaluating:

$$e_n(P,Q) = f_P(Q)/f_Q(P).$$

This pairing $e_n : E[n] \times E[n] \to \mu_n$ is bilinear and non-degenerate. This means that $e_n(aP, bQ) = e_n(P,Q)^{ab}$ and that for some values of P and Q, we have $e_n(P,Q) \neq 1$. We can easily see that given a point X "independent" from P and Q, we can reduce the discrete logarithm problem $Q = \lambda P$ on the elliptic curve to the discrete logarithm problem $e_n(Q,X) = e_n(P,X)^\lambda$ in $\mathbb{F}_{p^k}$.

The variant of the Tate pairing described in [3] is more complicated, since it operates on divisors instead of points. The Tate pairing operates on n-fold divisors, i.e. divisors D such that nD is principal, it takes values in μ_n and it is bilinear and non-degenerate. Given two n-fold divisors D_1 and D_2 defined over an extension $\mathbb{F}_{p^k}$ that contains the n-th roots of unity, we find f_{D_1} and f_{D_2} such that $div(f_{D_1}) = nD_1$ and $div(f_{D_2}) = nD_2$. The Tate pairing of D_1 and D_2 is then defined as:

$$t_n(D_1, D_2) = f_{D_1}(D_2)^{(p^k-1)/n}.$$

This pairing is also bilinear and non-degenerate. Moreover, for the purpose of discrete logarithm reduction, the Tate pairing $t_n(D_1, D_2)$ can easily be transformed into a pairing that involves points. One can simply fix two points R and S, and remark that $t_n((\lambda P) - (O), (R) - (S)) = t_n((P) - (O), (R) - (S))^\lambda$.

For more details about the properties and definitions of the Weil and Tate pairing, we refer the reader to [8,3,4].

3 A Tripartite Diffie–Hellman Protocol

In this section, we want to build an analog of the Diffie–Hellman protocol, that involves three participants A, B and C, requires a single pass of communications and allows the construction of a common secret $K_{A,B,C}$. By a single pass of communication, we mean that each participant is allowed to talk once and broadcast some data to the other two. The main idea is as in ordinary Diffie–Hellman, we start from some elliptic curve E and some point P. Then A, B and C each chose a random number (a, b or c) and they respectively compute $P_A = aP$, $P_B = bP$ and $P_C = cP$ and broadcast these values. Then they respectively compute $F(a, P_B, P_C)$, $F(b, P_A, P_C)$ and $F(c, P_A, P_B)$, where the function F is chosen in a way that ensures that these numbers will be equal and that this common value $K_{A,B,C}$ will be hard to compute given P_A, P_B and P_C. The problem now is to find such an F.

Using the Weil pairing, it is seems very easy to define such an F using the following formula:

$$F_W(x, P, Q) = e_n(P,Q)^x.$$

With this definition, one can easily check that:

$$F_W(a, P_B, P_C) = F_W(b, P_A, P_C) = F_W(c, P_A, P_B) = F_W(1, P, P)^{abc}.$$

However, this function is not satisfying because $e_n(P, P) = 1$ and thus $K_{A,B,C}$ is a constant. Nevertheless, the basic idea is quite sound and can in fact be implemented if we use two independent points P, and Q and if we have the three participants compute and broadcast (P_A, Q_A), (P_B, Q_B) and (P_C, Q_C). Then A, B and C can respectively compute $F_W(a, P_B, Q_C) = F_W(a, Q_B, P_C)$, $F_W(b, P_A, Q_C) = F_W(b, Q_A, P_C)$ and $F_W(c, P_A, Q_B) = F_W(c, Q_A, P_B)$. Moreover, all these values are equal and thanks to the independence of P and Q, they are not constant.

Moreover, using two points P and Q, it is easy to use the Tate pairing instead of the Weil pairing, and to define another function F as:

$$F_T(x, D_1, D_2) = t_n(D_1, D_2)^x.$$

Then A, B and C can respectively compute:

$$\begin{aligned}
F_T(a, (P_B) - (Q_B), (P_C + Q_C) - (O)) &= \\
&F_T(a, (P_C) - (Q_C), (P_B + Q_B) - (O)), \\
F_T(b, (P_A) - (Q_A), (P_C + Q_C) - (O)) &= \\
&F_T(b, (P_C) - (Q_C), (P_A + Q_A) - (O)), \\
F_T(c, (P_B) - (Q_B), (P_A + Q_A) - (O)) &= \\
&F_T(c, (P_A) - (Q_A), (P_B + Q_B) - (O)).
\end{aligned}$$

Because of the bilinearity of the pairing, all these numbers are equal and because of the non-degeneracy, their common value

$$F_T(1, (P) - (Q), (P + Q) - (O))^{abc}$$

is not independent from the choice of a, b and c.

Since F_T is based on the Tate pairing, it will be faster to evaluate then F_W (see the general remark about the efficiency of the Tate pairing versus that of the Weil pairing in [4]). Finally, our tripartite Diffie–Hellman protocol can be summarized as follows:

Alice	Bob	Charlie
Choose a Compute (P_A, Q_A)	Choose b Compute (P_B, Q_B)	Choose c Compute (P_C, Q_C)
Broadcast P_A, P_B, P_C and Q_A, Q_B, Q_C.		
Compute the common key as: $F_T(a, (P_B) - (Q_B), (P_C + Q_C) - (O))$ $F_T(b, (P_A) - (Q_A), (P_C + Q_C) - (O))$ $F_T(c, (P_B) - (Q_B), (P_A + Q_A) - (O))$		

Choice of Parameters and Construction of the Elliptic Curve

For the tripartite Diffie–Hellman protocol to be efficient, we need to choose elliptic curves such that the pairing can be efficiently computed. This means that the group μ_n should be in a small extension $\mathbb{F}_p$, i.e. k should be small. Moreover, we need to choose two points P and Q such that the pairing will be non-degenerate, this point can easily be checked by testing whether $e_n(P,Q)$ or $t_n((P)-(Q),(P+Q)-(O))$ is 1 or not. Note that when $k \neq 1$ at least one of the points P and Q must be defined over the extension $\mathbb{F}_{p^k}$ rather than over $\mathbb{F}_p$ otherwise the pairing will always be degenerate.

Two kind of curves are very promising for this tripartite Diffie–Hellman: supersingular curves (which leads to $k = 2$ according to the MOV reduction), and curves of trace 2 (which leads to $k = 1$ according to the FR reduction). It might seem strange to use elliptic curves which are known to be weaker than random curves, however, since we are also mixing in exponentiation in $\mathbb{F}_{p^k}$, we need to choose a large enough p for the discrete logarithm in $\mathbb{F}_{p^k}$ to be hard and then nobody knows how to compute discrete logarithms on the elliptic curve. The first kind of curve, i.e. supersingular curves, is well known and very easy to build. However, curves of trace 2 are not so easy to construct, in fact, we only known how to construct such curve when $p-1$ is a square or a small multiple of a square (see [5] or for some examples [4]). This is a pity because curves of trace 2 with a squarefree $p-1$ would allow us to work with a single point over $\mathbb{F}_p$ instead of two which would be very nice and efficient.

4 Efficient Implementation of the Pairing

The main step when computing the Weil or the Tate pairing is given a n-fold divisor $D = (X) - (Y)$, to write the principal divisor nD as the divisor of a (bivariate) function f denoted by $div(f)$. Then we need to evaluate f at some other point Z. There exists a standard method to do that, which is based on the fact that every divisor can be written as $(P) - (O) + div(f)$ for some point P and some function f, and that adding two divisors of that form is easy. Indeed, if

$$D = (P) - (O) + div(f),$$
$$D' = (P') - (O) + div(f')$$

then

$$D + D' = (P + P') - (O) + div(ff'g),$$

where $g = l/v$ with l the line through P and P' and v the vertical line through $P + P'$.

As explained in [7], when writing nD as $div(f)$, f cannot be expressed as an expanded polynomial (which would be exponentially large) but should be kept in factored form. However, even in factored form, writing down f is quite costly. As an example, the data in [4] shows that such a computation took about 40000 seconds for a supersingular curve when using a 50-digit prime p. This is not acceptable since with supersingular curves we want to work with a prime number of at least 100 or 150 digits.

In fact, a much better approach is to avoid the computation of f and to directly compute $f(Z)$. This is easily done by keeping for each intermediate divisor D the values of P and $f(Z)$ and by forgetting f. Computing $ff'g(Z)$ is easily performed by multiplying $f(Z)$, $f'(Z)$ and $g(Z)$. Thus at each step, we only need to evaluate two linear polynomials, to compute one inverse and to multiply a couple of numbers. Using this approach and the ZEN library [2], we see in the following example that the Tate pairing can be computed in a single second on a Pentium II-400 processor for a supersingular curve defined over a prime field of more than 150 digits.

A Small Example

In this section, we give an example of the tripartite Diffie–Hellman using a supersingular curve. We chose a prime p of more than 512 bits:

$$\begin{aligned} p = {} & 48267777815770043535044410856360047038953960729113574 \\ & 29530850774144832990078179684573230519991072031530329 \\ & 37333023591271636050696817523671646492380723773419011. \end{aligned}$$

We are working on the supersingular curve defined by $y^2 = x^3 + x$. Since we need to work in an extension field with p^2 elements, we define this field from the irreducible polynomial $x^2 + 1$, and we denote the square root of 1 by the letter i.

Remark that p was chosen in such a way that the large (160 bits) prime q divides $p + 1$, where:

$$q = 593917583375891588584754753148372137203682206097.$$

We then choose our two points P and Q as points of order q:

$$\begin{aligned} P = {} & (44190300200219570605979955052143576952357255551511568 \\ & 68511701918183168420954869076254808843953176168634019 \\ & 27551006066189692708095924815897927498508535823262371, \\ & 26090947680860922395540330613428690525406329616428470 \\ & 73807303133884126088547738030713042022034220476530186 \\ & 51634802037575702236646062353815408010755638011118751) \end{aligned}$$

$$
\begin{aligned}
Q = (&41741839015179817915732768381465901446084951835050843\\
&64114477814173114302373312329585774568654291610400898\\
&0621722645598334824826033527206878334398341068564562 0,\\
&8598407943832806682953550380640284842511375568804261 4\\
&5346094353988820150684505043538654728150635315316572 1\\
&0019063972911218641810155964304683033635085838106425i)
\end{aligned}
$$

Using the Tate pairing we can compute

$$
\begin{aligned}
&F_T(1, (P) - (Q), (P + Q) - (O)) =\\
&3212260441330924846356567690530493333930589751352981900 55\\
&1491951878703681174480221600106557183904342214112647184 01\\
&2057960459613431923269557790286442357677246 55i +\\
&1882486718083976251736310342313163726675921997728969820 55\\
&0034390807159246606942885382186286577575700984687232892 23\\
&2549741868148348246686465425921848080385170 84
\end{aligned}
$$

Then for $a = 4$, $b = 7$ and $c = 28$ we compute

$$
\begin{aligned}
&F_T(a, (bP) - (bQ), (cP + cQ) - (O)) =\\
&2170465527325859502018505803671466158543295222385734483 5\\
&6777395721055102020058687041606605791667561999196950219 2\\
&641850458307828001561451703866966014963187271 19i +\\
&1854796754535600500024199532873596699011379170363502841 6\\
&2348376178652213528456277384398902756897609415503827104 8\\
&944364817877003701614538998745627383212540261 46
\end{aligned}
$$

and we check that indeed

$$F_T(a, (bP) - (bQ), (cP + cQ) - (O)) = F_T(1, (P) - (Q), (P + Q) - (O))^{abc}$$

Each evaluation of F_T took 1 second on a Pentium II-400 PC running under linux, which is very efficient compared to the 40000 seconds (on a Pentium-75) in [4].

5 Security Issues

Clearly in order to be secure the tripartite Diffie–Hellman described here requires the discrete logarithm on the chosen elliptic curve to be hard, and the discrete logarithm in the finite field $\mathbb{F}_{p^k}$ to be hard. Since we placed ourselves in the cases where either the MOV or the FR reduction applies, the hardness of the elliptic curve discrete log implies the hardness of the finite field discrete log and we can remove the second condition. This is a simple restatement of the fact that when the finite field discrete log, then to solve the elliptic curve discrete log we simply transport the problem in the finite field using the pairing and then solve the problem in the finite field. However, it is not known whether the elliptic curve discrete logarithm on a weak curve is as hard as the discrete logarithm in the corresponding finite field (in the sense of the MOV or FR reduction). In fact, this is a very interesting open problem. Moreover, as in the Diffie–Hellman case this is not the whole story, some Diffie–Hellman like problem and Diffie–Hellman like decision problem should be hard in order to get security.

Quite amusingly, we should note that on curves where either the MOV or FR reduction applies, the usual Diffie–Hellman decision problem is mostly easy. Remember that the usual Diffie–Hellman problem is given a quadruple (g, g^a, g^b, g^c) to decide whether $c = ab$. This problem can also be expressed with the following formulation which is slightly different. Given a quadruple (g, g^a, h, h^b), decide whether $a = b$. Now on an elliptic curve where the MOV reduction applies, we can easily test for a quadruple (P, aP, Q, bQ) whether $a = b$, it suffices to compute $e_n(aP, Q)$ and $e_n(P, bQ)$ and to compare them. This test works as soon as P and Q are independent (i.e. when $e_n(P, Q) \neq 1$). Of course, in the FR case, such a test also exists. More precisely, one can test for the equality of $t_n((aP) - (O), (\lambda Q) - (Q))$ and $t_n((P) - (O), (\lambda bQ) - (bQ))$, where λ is essentially any constant number (some values of λ are excluded, for example $\lambda = 1$ is not allowed). Note than when P and Q are not independent, the test usually doesn't work, thus some cases of the usual Diffie–Hellman decision problem are still hard on these elliptic curves.

With the current knowledge of elliptic curves, we believe that this system is secure in practice as soon as the discrete logarithm in $\mathbb{F}_{p^k}$ is hard. For the supersingular case ($k = 2$), we think that p should be a 512 bits prime. In the trace 2 case ($k = 1$), we recommend to choose a 1024 bits prime. Moreover, the usual precautions should be taken, i.e. some large prime q should divide the order of the elliptic curve, all the points involved in the computation should be of order q, and we should use the pairing e_q or t_q.

6 Conclusion

In this article, we described a generalization of the Diffie–Hellman protocol to three parties using the Weil or Tate pairing on elliptic curves. We also showed that this pairing can be implemented much more efficiently than previously shown in [4]. Therefore, this new protocol seems quite promising as a new building block to construct new and efficient complex cryptographic protocols. On the other hand, we sincerely hope that people will try to attack it, since finding a weakness in this protocol would certainly give some new insight in the difficulty of the discrete logarithm on elliptic curves.

References

1. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Comput. Sci.*, pages 275–286. Springer, 1995.
2. F. Chabaud and R. Lercier. The ZEN library. `http://www.dmi.ens.fr/~zen`.
3. G. Frey and H. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.
4. R. Harasawa, J. Shikata, J. Suzuki, and H. Imai. Comparing the MOV and FR reductions in elliptic curve cryptography. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT'99*, volume 1592 of *Lecture Notes in Comput. Sci.*, pages 190–205. Springer, 1999.
5. G.-J. Lay and H. Zimmer. Constructing elliptic curves with given group order over large finite fields. In L. Adleman, editor, *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 250–263. Springer, 1994.
6. R. Lercier. *Algorithmique des courbes elliptiques dans les corps finis.* thèse, École polytechnique, June 1997.
7. A. Menezes. *Elliptic curve public key cryptosystems.* Kluwer Academic Publishers, 1994.
8. A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transaction on Information Theory*, 39:1639–1646, 1993.
9. V. Miller. Use of elliptic curves in cryptography. In H. Williams, editor, *Advances in Cryptology — CRYPTO'85*, volume 218 of *Lecture Notes in Comput. Sci.*, pages 417–428. Springer, 1986.
10. I. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Mathematics of Computation*, 67:353–356, 1998.
11. N. Smart. The discrete logarithm problem on elliptic curves of trace one. preprint, 1997.