U.P.B. Sci. Bull., Series C, Vol. 72, Iss. 3, 2010 ISSN 1454-234x

ZERO-KNOWLEDGE PROTOCOLS IMPLEMENTED WITH ELLIPTIC CURVES

Bogdan TUDOR[1]

În acest material se abordează protocoalele de tip 'cunoastere zero' în perspectiva curbelor eliptice. Elementele originale sunt optimizarea algoritmilor, alegerea parametrilor curbelor eliptice şi calcularea împerecherii Tate. Algoritmii se încadrează într- un cadru de dezvoltare şi împreună cu funcţiile de distorsiune se pot aplica pentru orice împerechere.

In this paper 'zero-knowledge' protocols are treated in the elliptic curves perspective. The original elementes are the optimization of the algorithms, elliptic curves parameters and the Tate pairing calculus. The algorithms can be treated as a framework and through the distortion maps they can be applied on every elliptic curve pairing.

Keywords: elliptic curve, Tate pairing, zero-knowledge, distortion maps

1. Introduction

Zero-knowledge protocols play an important role in the data transfer systems nowdays. In the case of the wireless networks these protocols are being more important because of the propagation medium. In this moment, conforming to the latest standards the authentication infomation is protected through the best methods available but despite to these protections there are always posibilities to an virtual oponent to decrypt the the data through some kinds of cryptographic attacks [10]. The zero-knowledge protocols solve in part this kind of treat because the sensitive information is not even transmitted to the verifier. The new cryptographic framework introduced in paragraph 3 and the new parameters computed in paragraph 5 are the original contributions of this paper.

2. Protocols with pairing

In the beginning of the communication setup, is the asymmetric cryptographic part. The majority of the protocols use the Diffie-Hellman or RSA methods. Using the elliptic curves pairings is a relatively new perspective [4]. Three of the pairings are mostly used: Weil, Tate and Ate (a short version of Tate). These pairings are through definition bilinear mappings for elliptic curves.

[1] Eng., Advanced Technology Institute, Bucharest, Romania, bogdantudor74@yahoo.com

In the present case two kinds of pairings are being used: symmetric and asymmetric. The main difference between them is about the definition domains.

Every bilinear pairing used in cryptography for an elliptic curve G satisfies the following properties:

- bilinear: $e(ag,bh)=e(g,h)^{ab}$ for every g,h∈G and b∈Z, in the multiplicative form (can be written as exponentials).
- non-degenerate: for every generator element g and h from G with *e(g,h)≠1* , in this point is important to note that most of this paper results are based on Tate pairing, where $e(g,g)\neq 1$
- computability: there is an efficient algorithm for calculating $e(g,h)$

An admissible map is defined as a function $e:G\times G\rightarrow G_1$ where G is an subgroup from E(F_p) and G_1 from E(F_{p^2}). These groups have the same order q and we choose g as generator element from G. The G group is chosen from the condition that the Gap Diffie-Hellman problem is intractable (if we know the values g, ag, bg calculate the element abg – in multiplicative form).

The Hufschmitt scheme [6] is a zero-knowledge one through the value of abg obtained in l iterations comprising three steps as below.

In this algorithm public parameters are (g, ag, bg, e(g,g), v=e(g,g)^{ab}) and private key is $S=abg$. The public key is (ag, bg, v), who is calculated only once. The proposed scheme is backed by the following equation:

$$e(g,Y)=e(g,rg\ \ c(abg\ \ )\ )=e(g,g)^{r+abc}=e(g,g)^{r}\times(e(g,g)^{ab})^{c}=W\times v^{c}$$

Table 1

The Hufschmitt algorithm

Prover	Verifier
choose $r\in[0,q]$	
calculate $W=e(g,g)^{r}$	
send W to the verifier	
	choose $c\in[0,2^{k}]$
	send c to the Prover
verify that $c\in[0,2^{k}]$	
computes $Y=g^{r}\times S^{c}$	
Send Y to the verifier	
	Verify that $e(g,Y)=W\times v^{c}$

The probability of guessing the c value is after l rounds $\frac{1}{2^{lk}}$.

The scheme proposed by Shao, Lu and Cao [13] is based on the security of the Diffie-Hellman algorithm. The public parameters are g, e(g,g) and the private key is $s \in (0, q]$. The public key is $v = sg$.

Table 2

The Shao algorithm

Prover	Verifier
choose $r \in [0, q]$	
Calculate $W = rg$	
Send W to the verifier	
	Choose $\beta \notin (0, q]$
	Send β to the prover
Calculate $Y = g^{\frac{1}{r+s\beta}}$	
Send Y to the verifier	
	Verifies that $e(Y, v^{\beta} W) = e(g, g)$

This scheme is more efficient than those of Kim and Kim or Yao and Wang [13]. Below is a comparison between the two algorithms presented above.

Table 3

Comparison between Hufschmitt and Shao algorithms

	Hufschmitt	Shao
Nr of exp for prover	$2 + \varepsilon$	2
Nr of pairings for prover	0	0
Nr of exp for verifier	ε	1
Nr of pairings for verifier	1	1

These two algorithms are almost equals from the calculus point of view. Balancing the calculus power between the prover and the verifier is done based on the usage conditions. In the case of client-server implementations (the most used case) it is an advantage in Hufschmitt algorithm. In other applications (like smart devices) the Shao implementation is superior. The ε number from the evaluations is from the fact that the exponent of the final verification v^c is smaller than q, having only k bits.

The initial calculus of e(g,g) is done only once and is possible to be done outside the system, subsequent values will be calculated based on this result. This is the reason that number of pairings for the prover is 0.

As it can be seen from the above comparison the number of the required computations of pairings is 1, a minimum value for the algorithms. The only operation that can be moved from one part to another is the exponentiation one. The sum is 3 for Shao and 2+2ε for Hufschmitt.

Another perspective for using pairings in these kinds of protocols is presented by Zhang and Kim in [17] and has its roots in DVB (Digital Video Broadcasting). In that case is mandatory the presence of the third trusted party. That entity generates a user's signature and put it in the user's hardware (usually an smart-card). In the original scheme, proposed by the Boneh and Franklin exist a difference in the calculus power needed between the user and the provider. This is the main reason of the following scheme. In the beginning the system generates S_{ID}=s*h(ID), where s is a secret random parameter. The public key is P_{pub}=sP.

Table 4

Zhang and Kim algorithm

Prover	Verifier
Has ID, S_{ID}	
Transmit ID to the verifier	
	Calculate $Q_{ID} = h(ID)$
	Choose r – random and R=rP
	Transmit R to the prover
Choose a at random	
Calculate A=aP, C=H_1(A)S_{ID}+a^2R	
Transmit A, C to the verifier	
	$e(C,P) = e(Q_{ID}, P_{pub})^{H_1(A)} e(A,A)^r$

The final step (verification) is done based on the equality:

$$e(C,P) = e(H_1(A)S_{ID} + a^2R, P) = e(H_1(A)S_{ID}, P)e(aR, aP) =$$
$$e(H_1(A)Q_{ID}, sP)e(aP, aP)^r = e(Q_{ID}, P_{pub})^{H_1(A)} e(A,A)^r$$

Here H is a hash function defined on R and H_1 is a function defined on the image of the elliptic curve with results in R.

In this context we can observe that filtering the values of points R (from C one cannot compute S_{ID}) we can drop the H_1(A) term – which helps in diffusion of information. In that case:

$$C = S_{ID} + a^2R \text{ and } e(C,P) = e(Q_{ID}, P_{pub})e(A,A)^r$$

In this way it's possible to introduce supplementary operations in the network, but the number of mathematical operations needed is decreased. The same is also in the case of $e(A,A)^r$, where we can use $e(A,R)$ through $C = S_{ID} + a\ R$, which reduces an exponentiation from the client and server part. These improvements do not affect the overall scheme security, but reduces the implementation effort. In a presentation made by Xavier Boyen (A Roadmap of IBE Systems and their applications) on NIST workshop in 2008 he treats the Shao-like algorithm as bilinear DH inversion and the frameworks derived from it as "Exponent Inversion".

3. Cryptographic framework

All the previous schemes are following the classical three steps protocol of zero-knowledge protocols. The Shao ones is minimal from the computations point of view. So optimizing further the scheme following the classical setup is not possible (every another scheme involves a minimum 2 pairing computations). The key in this case is to minimize the number of steps and the message flow the protocol requires. The scheme in this case it would be composed from only two steps, thus requiring much lower network traffic. All the algorithms below will skip the first step from the classical zero-knowledge protocols (the prover first value - witness). In the classical algorithms the verifier is required to send only one random (challenge) value. In the new algorithm it must transmit 2 random values.

In the setup, some entity generates g and e(g,g). The prover chooses some secret value s as private key and publishes $g^s = Y$ as public key.

Table 5

First algorithm

Prover	Verifier
	Choose random values b and r and transmit them to the prover
Choose random t value Computes $X = g^t, Z = g^{\frac{1}{tb+rs}}$	
Send X, Z to the verifier	
	Verify that $e(Z, Y^r X^b) = e(g,g)$

This is backed by the following equality:

$$e(Z, Y^r X^b) = e(g^{\frac{1}{tb+rs}}, g^{sr} g^{tb}) = e(g,g)$$

We must to analyze the computational costs involved.

Table 6

Comparison between first algorithm and Shao

	This scheme	Shao
Nr of exp for prover	2	2
Nr of pairings for prover	0	0
Nr of exp for verifier	2	1
Nr of pairings for verifier	1	1

From the table above it’s easy to see that these two algorithms are comparable from this point of view. In fact, the logical operations in the prover part are the same (the inversion of the sensitive parts). The implication of r and b in the final verification excludes the possibility of forgery. If one can compute some values X, Z with the desired properties, he could solve the Y^b equation and thus find the secret value r. The proposed algorithm is very useful in the wireless area, where network transmissions are expensive. From X and Z the verifier cannot learn anything about t and r, except for public information. The extra exponentiation in the verifier part is a little extra cost for dropping the witness value transmission. This scheme balances the network load with verifier's computations.

An analog scheme exists for identification with some credential (ID).

Table 7

ID-based algorithm

Prover	Verifier
Calculate $ID \rightarrow h(ID) \in F_q$ Choose s – secret key Calculate $Z = g^{h(ID)}$, $Y = g^s$	
Transmit ID, Z	Calculate $ID \rightarrow h(ID) \in F_q$ Calculate $Z = g^{h(ID)}$
	Send three random values a, b, c
Choose t random value and compute $X = g^t$ $T = g^{\frac{1}{a*h(ID)+b*s+c*t}}$	
Send X, T to verifier	
	Verifies that $e(X^c Y^b Z^a, T) = e(g, g)$

The equality is evident and the system could be extended to more informations (in this case there are two informations: secret key s and ID value). For every piece of sensitive information the verifier must generate a random number. If k is the number of valuable informations (like private key), the number of random variables must be k+1. Of course if the number increases, extra care must be put in random values generation in the verifier part. This condition appears from the fact that information must be randomized through multiplying. This completes this kind of authentication protocols with something like a framework. For current case it is important to compare the number of operations required:

Table 8

Comparison between previous ID algorithm and Zhang's one

	ID scheme	Zhang scheme
Prover expon	4	3
Prover pairings	0	0
Verifier expon	4	1
Verifier pairings	1	3

From the table above we see the difference in number of pairings (an operation who is far more time consuming than exponentiation). For every supplementary parameter we can add an extra exponentiation for verifier. Unlike the previous scheme the computations are more balanced between the prover and verifier. The schema with only two random numbers is the same with c=0. From these examples we can write the general case:

Table 9

Generalized version

Prover	Verifier
In the setup, it chooses m secret values $n_i, i = 0...m$. Publish $X_i = g^{n_i}, i = 0...m$	
	Generates m random values a_i Generates an binary vector of length m b_i with $b_i = 0,1$ Sends them to the prover
Computes $Y = g^{\frac{1}{\sum a_i b_i n_i}}$ and send	
	Verifies $e(Y, \prod_i X_i^{b_i^{a_i}}) = e(g,g)$

Through the judicious use of binary vector b_i who selects what secret values are used in verification (the exponentiation in the end of the final formula is trivial when b_i=0) the verifier can use a single step instead of using multiple steps for minimizing the forgery probability. This probability is $\prod (\frac{1}{p_i})^{b_i}$ where p_i is the probability of guessing a secret number n_i and we omit the influence of a_i parameters. When the Hamming weight of the b_i vector is sufficiently high, this probability is not an issue. This can be seen as an analogue of generalized Feige-Fiat-Shamir protocol [10] in the elliptic curve domain.

4. Pairing computation

The main part of the algorithms consists in pairing's computation. The Tate pairing seems to be much faster and appropriate than other pairings for cryptographic applications. It is based on the degree 0 divisors from $E(F_q)$

represented by $P - P_\infty$ with $P \in E(F_q)$. The calculus is done based on the observation that if we choose two points P and P' from $E(F_q)$ we can find another point from the curve B and a function G with $P + P' - B - P_\infty = div(G)$. G is the equation of line through the two points and B is the negative of the sum. If we note L_1 the above line and L_2 the vertical line through B we can write the following equations:

$$div(L_1) = P + P' + C - 3P_\infty$$

$$div(L_2) = C + B - 2P_\infty$$

$$div(L_1 / L_2) = P + P' - B - P_\infty = div(G)$$

The G line has $G = \dfrac{Y - \lambda(X - x_1) - y_1}{X + (x_1 + x_2) - \lambda^2}$ where λ is the angle of the line through P and P' (or the tangent if that two points are equal).

For some kinds of pairings (like Weil) the result is trivial if the points are dependent of each other. This situation is solved in [15] through distortion functions. In this case the pairings are called modified pairings. The distortion function is an endomorphism who transform an point P with order l in another point of order l but from $E(F_{q^k})$, where k is the embedding degree: $\phi(P) \in E(F_{q^k})$. The advantage in using supersingular elliptic curves is that the distortion function always exists in this case and for regular ones does not exist (with some minor exceptions). The modified pairing can be defined in this way:

$$e'_l(P, Q) = e_l(P, \phi(Q))$$

Examples of distortion functions are:

Table 10

Distortion functions

Characteristic	Curve	Embedding degree	ϕ
$p > 3$ and $p \equiv 2 \bmod 3$	$y^2 = x^3 + a$	2	ϕ_1
$p > 3$ and $p \equiv 3 \bmod 4$	$y^2 = x^3 + ax$	2	ϕ_2

Corresponding distortion functions:

$$\phi_1(x, y) = (\xi x, y) \text{ where } \xi^2 + \xi + 1 = 0$$

$$\phi_2(x, y) = (-x, iy) \text{ where } i^2 + 1 = 0$$

The definition field does the security level of the operations.

5. Choosing the parameters

We choose the second example where $p \equiv 3 \bmod 4$. It is possible to show that the curve is supersingular and $u = \#E(F_p) = p + 1 = lc$, where l is the point order and c represents the cofactor. Because of the nature of the chosen curve, the MOV attack is possible so the discrete logarithm problem must be defined over a minimum field of 2048 bits (a level required by today's computing possibilities). That field is defined by F_{p^2}, so p must be a prime number of 1024 bits. From the previous works, the point order must be at a minimum of 320 bits and the cofactor must fill the rest. From the fact that l must be a prime number, follows $c \equiv 0 \bmod 4$. The main criterion in choosing these parameters is the Hamming weight who has to be as low as possible. The final Hamming weight is a combination between all the parameter's weights.

To find the parameters is sufficient to build a predefined model for searching numbers:

$$c = 2^x + 2^y \text{, with } x \in [1025 - l, 1024] \text{ and } x > y > 1$$

this model results from the condition that c is a multiple of 4.
There are multiple possibilities for l but we try the lowest Hamming:

$$l = 2^x + 1 \text{ and } l = 2^x + 2^y + 1 \text{ where } x \in [1025 - l, 1024] \text{ and } x > y > 1$$

Trying all the possibilities between chosen ranges performs searching. The first model cannot reveal any number who satisfies all the requirements. Because the term 2^y from c representation has the main influence in final weight, it has to be as low as possible. We fix $x = 704$ in c representation and the results for $c = 2^{704} + 2^x$ and $l = 2^y + 2^z + 1$ are:

Table 11

Parameters for low Hamming weight

X	y	z	Hamming p
2	395	290	7
3	365	87	8
4	333	286	9
2	499	94	7
2	612	513	7

It's obvious that lower than 7 we cannot obtain valid Hamming weights (for x=2). So the values with 7 are optimum values.

Similar result indicates Hamming weights of around 10 for 1024 bits security [3]. The primality tests used are not pseudo-primality tests, but combinations of Pocklington-Lehmer and APRCL tests, which slows down entire search process. These parameters are being computed only once, so the computing time is not important.

It is possible to compute $f_Q(\phi^{-1}(R))$ instead of $f_{\phi(Q)}(R)$ where R is the random point used in pairings. These bring the required operations from the field extension in base field operations [3].

An important point is finding and representing the torsion group. The multiplication can be done through division polynomials, but this is slow. Instead we can use the fact that the torsion group is generated by at most two independent points. This group is rather cyclic. In our case is sufficient to take some random points from the curve and multiply them with c. If the results are non-zero, the points are from torsion group of order l (because lc=p+1). We can use the fact that all the points kP with $k < l$ are from the same torsion group.

For finding torsion points we can use three methods [14], but in this case the useful one is that who utilizes group structure. The group $c* < E(F_q) >$ contains all the l torsion points and the c torsion points.

An algorithm for generating a torsion point of order l for a given curve:
1. we choose a random point $x \in F_q$
2. verifies if the curve's equation has solutions in F_q
3. compute $Q = cP \in E(F_q)$ where $P = (x, y) \in E(F_q)$
4. if $Q = (0,1,0)$ we roll back to step 1, if not we step out to step 5
5. Q is the l torsion point

In this case, the probability of choosing in step 1 an element that is not on the definition field of the curve is $\frac{1}{2}$. If we consider the probability of torsion points, the final probability in succeeding is $\frac{l}{2(l+c)}$. The second model brings us multiple choices for parameters, and we choose the lowest values:

$$c = 2^{704} + 2^{2}$$
$$l = 2^{395} + 2^{290} + 1$$
$$p = l * c - 1$$

Value of p is a prime number, which can be verified through more specialized programs. The number of nonzero bits in this representation is minimum. That's why the number of computations required by the Tate pairing is reduced. In the process to find out points on this curve, we can try all the natural numbers: 1, 2, and 3. We find that 3 is a point on the curve with coordinates:
(3 :
40416501322504352981471165734467770466041742250766374256961772915453322512442791335859806926093956724398275954478782129550707259996515911662948619647856050907441888089034351515134441190720072751271868753530912425114394508596312197533593156458564831966653570935835014182805930786219072959689960871118530233313894629625319043002883 9 : 1)

Multiplying this point with c factor gives us a point of order l from the curve:
(5541204497330535607189738758875110555367283006161559465026645999149516987755975116885649291111089310917344704719385715267177884705118996737139278867324546082819330979446617397574931384597308306729793421414950410360615836675722545614520044155707742901046430260832881141414304536704360279081448936361113124476900307179369450449198376 :
3252778554606901935798705641760330660284414712966536699317451892894736574934303991992252765081503998148844640991945431689490996191220055103558634533552278423810308914873429164141985437994312881097614375740648169180165127714603990127837588598678368608340033325227449177777440228686263519742052368127749567645336494234811900751813074 : 1)

This is the point required by the algorithms. If the preferred representations of the parameters are not these, choosing an addition/subtraction chain instead of binary chain presented by the classical Miller algorithm can do improving the computing speed. There are many possibilities [9], among these we refer to:

- Morain and Olivo's methods (based on Booth representations) [11]
- window methods by Downey and Schonhage
- Koyama methods
- Lee and Kim sliding method

In a particular case a method or another can be the best choice, but from statistical point of view for numbers with 512 bits the average chain lengths are:

Table 12

Chain's lengths for different methods

Algorithm	Length
binary	766.5
modified binary	681.7
window methods	~606
Koyama	602.6
sliding window	595.6

For current case these methods (that works in general setup) cannot be applied because the number of the bits equal to 1 is minimal by the selection of parameters. The effort is moved on finding parameters. That's why it is preferred to make the effort to find optimal parameters only once in the setup stage of the algorithm instead of applying the methods above.

5. Conclusions

The proposed algorithms can be seen as a framework and introduces the original idea of skipping the first step of protocol through an additional random challenge. The elliptic curve parameters are selected for minimum Hamming weight, thus the pairing implementation does not require sophisticated addition/subtraction chains and it's calculus is optimal. The Hamming weight value of 7 obtained in table 11 is the minimum one for the requirements. That value is better from computations point of view than those found in current literature.

R E F E R E N C E S

1. *P.S.L.M. Barreto, B. Lynn, M. Scott*, Constructing elliptic curves with prescribed embedding degrees, SCN, LNCS 2576, pp.257–267, Springer-Verlag, 2002
2. *Daniel J. Bernstein, Tanja Lange*, Faster addition and doubling on elliptic curves, Pages 29--50 in Advances in cryptology---ASIACRYPT 2007, 13th international conference, December 2--6, 2007 Lecture Notes in Computer Science 4833, Springer, 2007
3. *G. Bertoni, L. Chen, P. Fragneto, K. Harrison, G. Pelosi,* Computing Tate Pairing on Smartcards, White Paper STMicroelectronics, 2005
4. *D. Boneh, M. Franklin*, Identity-based encryption from the Weil pairing, SIAM J. of Computing, **vol. 32**, No. 3, pp. 586-615, 2003 Extended abstract in Crypto 2001, LNCS 2139, pp. 213-229, 2001
5. *T. Hadano*, Elliptic curves with a torsion point, Nagoya Math. J 66. (1977) 99-108
6. *E.Hufschmitt*, A zero-knowledge identification scheme in Gap Diffie-Hellman groups, Western European Workshop on Research in Cryptography Workshop Record (2005)
7. *J.-S. Hwu, R.-J. Chen, Y.-B. Lin*, An Efficient Identity-based Cryptosystem for End-to-end Mobile Security, Accepted and to appear in IEEE Transactions on Wireless Communications
8. *M.Kim , K.Kim*, A new identification scheme based on the bilinear Diffie-Hellman problem, Springer-Verlag 2002
9. *Y. Lee, H. Kim*, Expansion of sliding window method for finding shorter addition-subtraction chains, International Journal of Network Security, **vol. 2**, No.1, pp.54-60, 2006.1
10. *A.Menezes*, Handbook of applied cryptography, CRC Press 1996
11. *F. Morain, J. Olivos*, Speeding up computations on an elliptic curve using addition-subtraction chains, Inform.TheoryAppl.24(190), p.531-543
12. *M. Scott*, Computing the Tate pairing, CT-RSA, **vol. 3376** of Lecture Notes in Computer Science, pages 293—304, Springer-Verlag, 2005
13. *J. Shao, R. Lu, Z. Cao*, A new efficient identification scheme based on the strong Diffie-Hellman assumption, International Symposium on Future Software Technology 2004x
14. *J. Shikata, Y. Zheng*, Optimizing MOV algorithm for non-supersingular elliptic curves, Lecture Note on Computer Science, Asiacrypt '99, 1999
15. *E. Verheul*, Evidence that XTR is more secure than supersingular elliptic curve cryptosystems, Journal of Cryptology, **vol. 17**, Number 4, September 2004, pp. 277-296(20),Springer-Verlag
16. *G.Yao, G. Wang , Y.Wang*, An improved identification scheme, Berkhauser- Verlag 2003
17. F. Zhang, K. Kim, Signature-Masked Authentication Using the Bilinear Pairings, Cryptology & Information Security Laboratory (CAIS), Information and Communications University, technical report, 2002.