Electronic Journal of Differential Equations, Vol. 2017 (2017), No. 237, pp. 1–21.
ISSN: 1072-6691. URL: http://ejde.math.txstate.edu or http://ejde.math.unt.edu

ELLIPTIC CURVES DIFFERENTIATION WITH APPLICATION TO GROUP SIGNATURE SCHEME

ALIN IONUŢ GOLUMBEANU, OANA ADRIANA ŢICLEANU

Communicated by Vicentiu D. Radulescu

Abstract. Starting with the presented concept by Chaum and van Heijst and its refers to digitally signing for a document by a group member, such signatures allows the signers to remains anonymous but any verifier can confirm that the signer is a group member. The signatory anonymity can be revealed only by a designated group authority that has some auxiliary information. We present a complexity efficient group signature scheme based on zero knowledge and Schnorr signature algorithm. The scheme has two phases: the first one demonstrates that the signer is a member of the group while the second generates the message signature. In the end, we modify the classic scheme using differential elliptic curve cryptography to increase the system's performance against differential attacks.

1. Introduction

The group signature concept firstly was presented by Chaum and van Heijst [8] in 1991. Using such a scheme, any group member can sign a message on the groups behalf such that everybody can verify the signature but no one can find out which group member provided it. However, there is a designed group authority (named group manager) who can reveal the signer's identity in the case of a dispute.

An important characteristic of such a scheme is the fact that is hard to decide if two signatures were provided by the same group member. Most of the group signature schemes become increasingly inefficient for large groups since the group public key and the length of the signature depend on the size of the group. The first efficient group signature was introduced by Camenisch [6]. Its efficiency relies on the fact that the group public key and the signature length have a constant size.

Definition 1.1 ([4])**.** A digital signature scheme that has the following procedures is called a group signature scheme:

- *Setup:* it is a procedure made between the group manager and all the group members and its result is the group public key made from all the group members' public keys, the group members' secret keys and the group manager's secret key.

2010 *Mathematics Subject Classification.* 35H20, 35S15, 12H20, 11G07.
Key words and phrases. Group signature; zero knowledge; discrete logarithm; Schnorr signature.

Submitted July 6, 2017. Published September 29, 2017.

- *Join:* it is a protocol between the group manager and a user for making the respective user a member of the group.
- *Sign:* it is a probabilistic algorithm that has on input a group member's private key, and a message and returns a signature on that message.
- *Verify:* it is an algorithm which takes as input the group public key, the signature, and the signed message. This algorithm outputs yes if and only if the signature is correct.
- *Open:* it is a deterministic algorithm which takes as input the message m, the signature, and the group manager's secret key and returns the identity of the group member who issued the signature together with a proof of this fact.

A secure group signature must respect the following properties [39]:

- Correctness – the signatures provided by a group member using the *Sign* procedure must be accepted by the *Verify* procedure.
- Unforgeability – only the group members can sign messages on behalf of the group.
- Anonymity – the identity of the member who provided a valid signature must be computationally hard to find for anyone except the group manager.
- Unlinkability – given two valid signatures provided by the same member it is computationally hard to verify this fact for anyone except the group manager.
- Traceability – the group manager is always able to use the *Open* procedure to identify the signer of a valid signature.
- Exculpability – the group manager and the group members are not able to provide valid signatures for any other groups. The group manager is not able to give responsibility to a group member for a valid signature that he did not provide.
- Coalition-resistance – no subset of group members can provide a group signature that cannot be open by the group manager.

This type of signature is highly used in big companies where an employee has the permission to sign documents on behalf of the company. In this case the verifier does not need to check that particular employee, he has to know only the company's public key to check the signature validity.

Group signature are also used in electronic commerce [29]. In this case a costumer can use coins issued by several banks. The seller cannot find out which bank issued the coin used by the costumer to buy a certain product. So, the central bank plays the group manager role while the other banks are the group members [24].

Various group signature schemes have been proposed so far. Some of them have the efficiency level depending on the group public key length [4, 8, 9, 33] while others are independent of the number of the group members [5, 6].

This paper presents the security issues that must be taken in consideration when designing a group signature scheme and also the algorithms and concepts underlying the most popular schemes. We also propose an elliptic curve group signature scheme which uses much more shorter keys and has the same security level as the classic method.

2. Number-theoretic problems

The security of many systems relies on the intractability of solving number-theoretical problems. A problem is considered intractable if there is no algorithm that solves it using a reasonable amount of resources. A reasonable amount of resources means that the resources needed by the best algorithm are at least exponential in the number of bits needed to describe the problem. In this section we present the discrete logarithm problem and factoring large integers problems. For further details see [10, 30].

2.1. **The discrete logarithm problem.** The discrete logarithm can be considered the analogous of the common logarithm for groups. The common logarithm $\log_a b$ is the solution of the equation

$$a^x = b$$

So, given g and h elements from a finite cyclic group G the solution of the equation

$$g^x = h$$

is called the discrete logarithm of the base g to h in the group G.

Definition 2.1 ([41])**.** Let G be a finite cyclic group with n elements and b a generator of it. Then each element $g \in G$ can be written as

$$\log_b : G \to Z_n$$

where Z_n is a ring of the integers modulo n. This function is an isomorphism of groups and is called discrete logarithm of base b.

Changing the base for a discrete logarithm is done in the same way as for a common one:

$$\log_c g = \log_c b \cdot \log_b g$$

The discrete logarithm problem is similarly with solving common logarithms using real numbers. The major difference is given by the fact that unlike the common logarithms where the solution is approximated, the discrete logarithm problem is defined over a discrete domain where the solution must necessarily be exact.

A brute solution for computing a discrete logarithm $\log_b g$ is raising b to the power k. k will grow increasingly more until g is found. This algorithm needs a linear time proportional with G's size. Thus, the time consuming is exponential.

There are other algorithms more difficult but, also, more efficient based on integers factorization. However, none of these algorithms has a polynomial running time. These algorithms are grouped in three:

(1) Generic algorithms that work in arbitrary groups.
(2) Algorithms that work in arbitrary groups but especially efficient if the group's order is smooth.
(3) Special algorithms that are designed for special groups.

The generic algorithms are the ones based on exhaustive search. Two of the most efficient algorithms from this category are Baby-Step Giant-Step [25] and Pollard's rho algorithm [34]. The Baby-Step Giant-Step algorithm computes discrete logarithms in $O(\sqrt{n}\log n)$ group operations and stores $\sqrt{n}$ group elements. Pollard's rho algorithm has the same complexity as Baby-Step Giant-Step algorithm but the storage is negligible. These algorithms are exponential.

When the group has the form Z_p^* or $Z_{2^m}^*$, where p is a prime integer, there are sub-exponential algorithms that have a running time upper-bounded by

$$O(\exp((c+o(1))\sqrt{\ln q \ln\ln q}))$$

group operations, where q is p or 2^m and $c>0$ is a constant.

Together with the rho algorithm, Pollard also proposed a lambda algorithm. This algorithm computes discrete logarithms with a parameter-dependent success probability in $O(p\sqrt{w})$ group operations when the solution lies within a restricted interval of width w.

The hardness of the discrete logarithm problem depends on the representation of the group elements. So, there are groups in which the DLP can be easily solved. Such a group is $(Z_m,+)$ where for finding the discrete logarithm of an element a to a base b we have to solve the equation

$$bx \equiv a \pmod{m}$$

To solve this we have to compute $gcd(b,m)$ which needs $O(\log^2 m)$ group operations using the extended Euclidean algorithm.

2.2. **Factoring large integers.** The factoring large integers problem underlies on the security of the most popular public key cryptosystem: RSA.

Definition 2.2 ([15])**.** Solving the integer factorization problem means finding the prime factorization for a given positive integer n.

$$n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$$

The algorithms for factoring an integer are classified in:

(1) general purpose algorithms – the running time depends only on the size of n
(2) special purpose algorithms – the running time depends on a special property of n; this property can be the size of the largest prime factor.

One of the most popular special purpose algorithms is the trial division. The worst case of this algorithm consists in trying all primes smaller than $\sqrt{n}$ for factoring n. Another exponential algorithm is the Pollard's rho [35] which has a running time $O(\sqrt{n})$.

The elliptic curve method finds a small prime factor p in

$$O\big(\exp((1+o(1))\sqrt{2\ln p \ln\ln p})\big) \text{ time.}$$

The quadratic sieve algorithms and the number field sieve algorithm are general purpose algorithms. The first one was introduced in [36]. The latter was presented in [27] and used for breaking the RSA-130 [17].

In contrast with the discrete logarithm problem, when trying to solve a factorizing problem it is not clear which kind of algorithm is best when only given the integer n. However, the special purpose algorithms are recommended whenever possible. In [30] we can find a possibility for ordering the applying algorithms:

(1) trial division by small primes up to a bound b_1
(2) Pollard's rho algorithm in order to find any small factors than a given bound $b_2 > b_1$
(3) an elliptic curve factoring algorithm in order to find any small factors smaller than a given bound $b_3 > b_2$
(4) a general purpose algorithm.

3. Public key cryptosystems

In this section we present two public key cryptosystems that are often used in group signature schemes.

3.1. **RSA Cryptosystem.** RSA is a public key cryptosystem which was introduced in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman [38].

The algorithm for generating the keys has the following steps:

(1) generate two large primes p and q that are kept secret
(2) compute $n = p \cdot q$ which is made public
(3) compute $\Phi = (p-1)(q-1)$
(4) choose an integer $e < n$ and $gdc(e, \Phi) = 1$
(5) choose an integer d such that $d \cdot e = 1 \mod \Phi$
(6) the public key is (e, n)
(7) the private key is (d, n)

For encrypting the message m we use

$$c = m^e \mod n$$

and for decrypting it

$$m = c^d \mod n.$$

The RSA algorithm is described in algorithm 3.1

Algorithm 1 RSA algorithm

1: generate large primes p and q
2: $n = p \cdot q$
3: $\Phi = (p-1)(q-1)$
4: choose an integer $e < n$ and $gdc(e, \Phi) = 1$
5: choose an integer d such that $d \cdot e = 1 \mod \Phi$
6: encrypt the message m

$$c = m^e \mod n$$

7: decrypt the encrypted message c

$$m = c^d \mod n$$

To compute d (the private key) we use:

$$d \cdot e = 1 + t \cdot \Phi$$

where t is an integer.

We can use the Extended Euclidean algorithm to compute the gcd:

$$ax + bz = gcd(a, b)$$

The security of this system relies on the factorization of n. If p and q are found then the cryptosystem can be broken. To reach a high security level p and q must be large primes such that the security will rely on the factorizing large numbers problem (see section 2).

To verify if a large number is prime there can be used different methods. We will shortly present the Rabin test [37] for prime numbers, which is a probabilistic test, and Cohen and Lenstra test [11] for prime numbers, which is a deterministic

one. The probabilistic tests are much faster than the deterministic ones, but the latter are much more efficient. Both test use Fermat Theorem.

Theorem 3.1 (Fermat Theorem [15]). *If m is a prime and a an integer then*

$$a^m \equiv a \pmod{n}$$

Rabin test is based on the fact that the following equation

$$x^2 \equiv 1 \pmod{p}$$

has only two solutions $x \equiv \pm 1 \pmod{p}$. Suppose m is the integer that we want to test. We assume that m is prime and from Fermat Theorem (Theorem 3.1) we have that $gcd(a, m) = 1$ satisfies

$$a^{m-1} \equiv 1 \pmod{m} \quad \text{for all integer } a$$

Because $m - 1$ is even we have:

$$a^{(m-1/2)} \equiv \pm 1 (mod\ m)$$

If $a^{(m-1)/2} = +1$ and $(m-1)/2$ is even then

$$a^{(m-1)/4} \equiv \pm 1 \pmod{n}$$

This is the reasoning used to demonstrate the following lemma.

Lemma 3.2. [31] *Let p be a prime number and $p - 1 = a \cdot 2^f$ where a is uneven. Let u be an integer such that $1 < u < p - 1$ then we have:*

$$u^a \equiv 1 \pmod{p}$$

or

$$u^{a \cdot 2^f} \equiv -1 \pmod{p} \quad \forall 0 \leq i < f$$

To test if an uneven integer is prime we write $m - 1 = a \cdot 2^f$ where a is uneven. Then we randomly choose an integer u such that $2 \leq u < m$ and compute from left to right

$$u^a, u^{a \cdot 2}, u^{a \cdot 2^2}, \ldots, u^{a \cdot 2^f}$$

When we have a number that is not equal with -1 or +1 and the one right next to it is 1, or $u^{a \cdot 2^f} \neq 1 \pmod{n}$ we can say m is compound. The test must pe repeated for k times where k is a security parameter.

The Cohen and Lenstra test is based on Fermat Theorem stated above. Suppose m is the number that we want to test. If there exists a single integer a that does not satisfy the relation $a^m \equiv a \pmod{n}$ then m is not prime. Because the reverse is not true we can also use another Theorem 3.3.

Theorem 3.3 ([30]). *An integer m is prime if and only if every divisor of m is a power of m.*

When p and q are chosen correctly the RSA cryptosystem cannot be broken. For a detailed study on RSA security see [26].

3.2. **ElGamal cryptosystem.** Unlike the RSA cryptosystem, the security of the ElGamal system is based on the discrete logarithm problem. The three algorithms for the system are described below:

(1) ElGamal Key Generation
- choose a large prime number p such that $p-1$ has a large prime factor and a primitive root $g \in Z_p^*$
- randomly choose a number x such that $0 \le x \le p-2$
- compute $y = g^x \pmod p$
- the public key is (p, g, y)
- the private key is (p, g, x)

(2) Elgamal Encryption
- randomly choose $k \in Z_p^*$
- compute $K = y^k \pmod p$
- compute
$$c_1 = g^k \pmod p$$
$$c_2 = Km \pmod p$$
- the encrypted message is (c_1, c_2)

(3) Elgamal Decryption
- compute $K = c_1^x \pmod p$
- compute $m = c_2/K \pmod p$

Another way to decrypt the message is:

$$x_1 = p - 1 - x$$

$$\begin{aligned} c_1^{x_1} c_2 &= g^{kx_1} Km \pmod p \\ &= g^{k(p-1-x)Km \pmod p} \\ &= g^{k(p-1-x)} y^k m \pmod p \\ &= (g^{p-1})(g^x)^{-k} y^k m \pmod p \\ &= y^{-k} y^k m \pmod p \\ &= m \pmod p \end{aligned}$$

To avoid a plaintext attack for an ElGamal system the values of k must be changed as often as possible. To exemplify such an attack we assume to use the same value for k for two times. The messages are m_1 and m_2. The encrypted messages will be:

$$(c_1^{(1)}, c_2^{(2)}) = (g^k \pmod p, Km_1 \pmod p)$$

and

$$(c_1^{(2)}, c_2^{(2)}) = (g^k \pmod p, Km_2 \pmod p)$$

So we have

$$m_1/m_2 = c_1^{(1)}/c_2^{(2)} \pmod p$$

Thus, if one of the two messages is known, the attacker can easily find out the other message, too.

As we have already mentioned, the security of the ElGamal system is based on the discrete logarithm problem. So, we have to carefully choose the primes such that their discrete logarithm to be difficult to solve.

4. Digital signature

The concept of digital signature was first presented by Whitfield Diffie and Martin Hellman in "New Directions in Cryptography" [18]. A digital signature scheme consists in three algorithms:

(1) an algorithm for generating keys; this algorithm takes as input a private key and returns that private key and its public key.
(2) an algorithm for signing the message
(3) an algorithm for verifying the signature

A digital signature scheme must respect at least three properties [28]:

(1) authentication – the receiver must always be able to identify the sender
(2) non-repudiation – the sender cannot later deny the sending and the singing of the message
(3) integrity – the receiver must be able to verify if the message was modified during its transmission.

4.1. **RSA digital signature scheme.** To generate the keys for the RSA digital signature scheme will be used the same algorithm from the RSA cryptosystem described in the previous section. Thus, we have the public key (e, n) and the private key (d, n). The private key will be used for signing the message while the public one will be used for verifying the signature. The signature is given by

$$s = RSA_{d,n}(m) = m^d \mod n$$

where m is the message. To verify the signature s we use

$$m' = RSA_{e,n}(s) = s^e \mod n$$

If $m' = m$ then the signature is valid.

The security conditions of the RSA digital signature scheme are almost the same as for the RSA system. So, if n can be easily factorized then an attacker can discover the signing key. A big disadvantage of the RSA digital signature scheme is represented by its multiplicative properties. To exemplify this, we take two messages m_1 and m_2 with the signatures s_1 and s_2, respective. Thus, we have:

$$s = s_1 s_2 = (m_1 m_2)^d \pmod n$$

which is a valid signature for the message $m = m_1 m_2 \pmod n$. Another problem appears when a message is signed and encrypted with the RSA system. The difficulty is caused by the fact that nobody knows which of these two operations must be done first. Generally, the answer depends on the cryptographic purpose. However, usually, is recommended to first sign the document and then encrypt it. In such a situation the length of the chosen modulus must be carefully treated because the resulted signature length can be greater than the maximum input accepted by the RSA encryption algorithm. For further study on this issue and the measures that must be taken in such a case see [32].

4.2. **Schnorr's digital signature scheme.** Schnorr's digital signature scheme is based on ElGamal digital signature scheme, the main difference being caused by the fact that the former uses a much smaller group of primes. This property does not affect the system's security which is also based on the discrete logarithm problem. The Schnorr's digital signature scheme is considered to be the simplest digital signature scheme whose security can be proven.

Usually, for generating the keys is used a Schnorr group.

Definition 4.1. *Schnorr Group[*[40]*]* A Schnorr group is a large subgroup of Z_p^* of prime order. To generate such a group there must be chosen p, q, r such that

$$p = qr + 1$$

where p and q are primes. Then h is randomly chosen such that $1 < h < p$ and

$$H^r \neq 1 \pmod{p}$$

A generator of a subgroup of Z_p^* has the value

$$g - h^r \pmod{p}$$

and x is an element of a Schnorr group if $0 < x < p$ and

$$x^q = 1 \pmod{p}$$

From the above definition it is obvious that every element, except 1, of a Schnorr group is a generator of the group. All the systems based on a Schnorr group must use a large p such that the discrete logarithm problem to be difficult to solve. q must also be large to avoid a birthday attack on the discrete logarithm problem. A big advantage of the Schnorr group is that such a group has a prime order and because of that it does not have non-trivial subgroups. Thus, an attack of subgroups is impossible.

To use the Schnorr digital signature scheme we choose a Schnorr group of order q and generator g, and a hash function H. Having a private key x the generating key algorithm will return a public key

$$y = g^x \pmod{p}$$

To sign a message m we choose a number k such that $0 < k < q$ and compute

$$r = g^k \pmod{p}$$
$$e = H(m||r)$$
$$s = (k - xe) \pmod{q}$$

where $||$ denotes the concatenation operation. So the signature will be (e, s) where $0 \leq s < q$. If The order q of the Schnorr group is smaller than 2^{160} the signature will have a maximum length of 40 bytes. To verify the signature we compute:

$$r_c = g^s y^e$$
$$e_v = H(m||r_v)$$

If $e_v = e$ then the signature is valid.

Algorithm 2 Schnorr generating key algorithm

1: generate large primes p
2: g is the group's generator
3: choose the private key x
4: $y = g^x \pmod{p}$
5: $\Phi = (p-1)(q-1)$
6: the public key is (p, g, y)
7: the private key is (p, g, x)

Algorithm 3 Schnorr signing algorithm

1: (p, g, x) is the private key
2: randomly choose k such that $0 < k < q$
3: $r = g^k \pmod p$
4: $e = H(m||r)$
5: $s = (k - xe) \pmod q$
6: the signature is (e, s)

Algorithm 4 Schnorr verifying algorithm

1: (p, g, y) is the public key
2: (e, s) is the signature
3: $r_v = g^s y^e$
4: $e_v = H(m||r_v)$
5: **if** $(e_v = e)$ **then**
6: (e, s) is valid
7: **end if**

5. Zero knowledge proof

A zero-knowledge proof is a proof of some statement which reveals nothing else but the veracity of the statement. To define a zero-knowledge proof we first need to define an interactive proof system.

Definition 5.1 (Interactive proof system)**.** An interactive proof system for a set A is a process between a verifier which executes a probabilistic polynomial-time strategy and a prover which executes a computationally unbounded strategy satisfying:

- **Completeness** – the verifier always accepts the common input $a \in A$ (after interacting with the prover).
- **Soundness** – having some polynomial p, any $x \notin A$ and any potential strategy S, the probability that verifier rejects the common input a is at least $\frac{1}{p(|a|)}$ (after interacting with S).

So, a proof is considered complete only if an honest verifier is always convinced of the veracity of a statement from an honest prover. A proof is considered sound if the probability that a cheating prover can convince an honest verifier that a false statement is true is very small.

Interactive proof systems were introduced by Babai et al. [2, 3] and by Goldwasser et al. [21] who also invented the concept of zero-knowledge. In the model used in these papers, the prover is computationally unbounded, while the verifier is a probabilistic polynomial-time machine (Definition 5.1).

Definition 5.2 (Zero-knowledge)**.** A strategy S is zero-knowledge on the set A if for any feasible strategy B exists a feasible computation C so that the following are computationally indistinguishable:

- the output of B after interacting with S on common input $a \in A$
- the output of C on input $a \in A$

Thus, any information obtained by interacting with S on some input a, can also be obtained from a without interacting with S [20].

After executing a zero-knowledge protocol between two entities, the verifier must be convinced of the validity of the statement. A big advantage of the zero-knowledge protocol is that the verifier is not able to prove the statement to other people.

5.1. **Fiat-Shamir identification protocol.** Fiat-Shamir identification protocol represents the basis for the most popular zero-knowledge protocols. The most important protocols derived from it are Feige-Fiat-Shamir [19] and Guillou-Quisquater.

Algorithm 5 Fiat-Shamir Identification Protocol

1: p and q are generated
2: $n = pq$ is made public
3: the prover selects Se coprime to n such that $1 \leq Se \leq n-1$
4: the prover computes $v = Se^2 \mod n$ which is his public key
5: the prover chooses r such that $1 \leq r \leq n-1$
6: the prover computes $x = r^2 \mod n$ and sends it to the verifier
7: the verifier chooses a bit $e \in \{0, 1\}$ and sends it to the prover
8: **if** e=0 **then**
9: the prover computes $y = r$
10: **else**
11: the prover computes $y = rs \mod n$
12: **end if**
13: the prover sends y to the verifier
14: the verifier rejects if $y = 0$ or $y^2 \neq x * v^e \pmod n$

This protocol is used in cryptography mostly for authenticating a person. Suppose Alice wants to identify herself to Bob. She has a secret Se known only by her. To successfully identify herself she has to prove her identity to Bob by proving that she possesses Se without revealing it. Since the secret is not revealed to Bob, no adversary can find it from the prover response. For this protocol, is needed a trusted part which generates two secret prime numbers p and q, and computes the public value $n = pq$. The steps that follow this operation are repeated t times, each time using independent random numbers. If the verifier has repeated the steps t times then he accepts.

The algorithm for such an identification is described in (5) and the repeating steps begin with the fifth one. The first two steps are executed by the third trusted part, while the steps three and four are executed by the prover only one time each. The number t is chosen by the verifier, if the verifier is easy to convince, t can be smaller. For further study on this algorithm see [31].

To better understand the algorithm we present a numeric example. Suppose $p = 5$ and $q = 11$ then $n = 55$ is made public. Suppose Alice (prover) chooses her secret $Se = 12$ and computes $v = 12^2 \mod 55 = 34$. Bob is an easy to convince verifier and choses $t = 2$.

(1) Alice chooses $r = 9$
(2) Alice sends $x = 9^2 \mod 55 = 26$ to Bob
(3) Bob sends $e = 0$ to Alice
(4) Alice sends $y = r = 9$ to Bob
(5) Bob verifies $y \neq 0$ and $9^2 \mod 55 = (26 * 34^0) \mod 55 \Leftrightarrow 19 = 19$
(6) Alice chooses $r = 15$

(7) Alice sends $x = 15^2 \mod 55 = 5$ to Bob
(8) Bob sends $e = 1$ to Alice
(9) Alice sends $y = rs \mod 55 = 45$ to Bob
(10) Bob verifies $y \neq 0$ and $45^2 \mod 55 = (5 * 34^1) \mod 55 \Leftrightarrow 45 = 45$

The completeness property of this protocol is caused by the fact that the prover which possesses the secret Se can also compute $y = r$ or $y = rs$ and sends it to the verifier. Because of that an honest verifier will always complete all t iterations and accept with the probability 1. The soundness is demonstrated by supposing the fact that the prover does not possess the secret Se. So, on a given round he cannot compute $y = r$ or $y = rs$. Thus, the rejection probability will be $\frac{1}{2}$ in each round. The zero-knowledge is provided by the fact that the only values made public in one round are x and y. A (x, y) pair can be simulated by choosing a random y and then computing $x = y^2$ or $x = \frac{y^2}{v}$. We can observe that such pairs are computationally indistinguishable from the ones computed in the protocol.

5.2. **Schnorr's zero-knowledge protocol.** The discrete logarithm problem represents the base for many cryptosystems' security. For an authentication protocol there are often used techniques for proving knowledge and properties of secret keys without revealing the keys. These are for instance proofs of knowledge of discrete logarithms.

Schnorr's zero-knowledge protocol is one of the methods used for proving the knowledge of discrete logarithms. For this protocol is used a cyclic group G_q of order q and generator g. The prover knows the secret key $x = \log_g y$ where y is public. So, the prover wants to convince the verifier that he knows the value of x without revealing it. The algorithm for this protocol is described below (6).

Algorithm 6 Schnorr's zero-knowledge Protocol

1: the prover randomly chooses r
2: $t = g^y$ is made public
3: the verifier sends a random number c to the prover
4: the prover computes $s = r + cx$ and sends it to the verifier
5: **if** $g^s = ty^c$ **then**
6: protocol successfully completed
7: **end if**

For a numeric example we take G_{11} with $g = 7$, so $x = log_7 y$. We take $x = 3$, so $y = 7^3 (mod\ 11) = 343 \mod 11 = 2$. The prover chooses $r = 3$ and computes $t = g^y = 7^2 \mod 11 = 5$. The random number of the verifier, named the challenger, is $c = 5$. The prover then computes

$$s = r + cx = 3 + 5 \cdot 3 = 18 \mod 11 = 7$$

The verifier computes

$$g^s = 7^7 \pmod 1 1 = 6$$

and

$$ty^c = 5 \cdot 2^5 \pmod 1 1 = 6$$

So the prover has convinced the verifier that he knows the discrete logarithm without revealing the value of x.

6. A group signature scheme based on the discrete logarithm problem

In [4, 23, 1, 12, 14] there are presented some efficient group signature schemes which are based on the discrete logarithm problem or Elliptic Curves DLP. Some schemes have been based on a variation of Elgamal encryption system (3.2) and of Schnorr's digital signature scheme (4.2). In this section we describe the scheme with all its phases since it represents the bases for the group signature scheme we propose.

6.1. **Premises.** Suppose we have a group with n members $M = \{M_1, M_2, \dots, M_n\}$ and a group manager M^G. Let G be a finite cyclic group of prime order q and $g, g1, \dots, g_n \in G$ be generators of G such that computing discrete logarithms to any of the bases is infeasible. Each member has a private key $x_i \in Z_q$ where $i = 1, 2, \dots, n$ and a public key $y_i = g^{x_i}$. The group manager's private key is denoted with ω and the public one is $z = g^{\omega}$.

6.1.1. *Elgamal variation scheme.* Suppose M_1 wants to encrypt a message message and sends it to M_2. He uses M_2's public key $y_2 = g^{x_2}$ and a random value $\alpha \in Z_q$. He then computes $A = {y_2}^{\alpha}$ and $B = g^{\alpha}$message. After receiving the pair (A, B), M_2 can decrypt the message by computing:

$$\frac{B}{A^{(x_2)^{-1}}} = \frac{g^{\alpha}\text{message}}{{y_2}^{\alpha x_2{}^{-1}}} = \frac{g^{\alpha}\text{message}}{g^{x_2 \alpha x_2{}^{-1}}} = \text{message}$$

Algorithm 7 Elgamal variation scheme

1: the sender randomly chooses $\alpha \in Z_q$
2: the verifier makes public $y = g^x$ is made public
3: the sender computes $A = y^{\alpha}$
4: the sender computes $B = g^{\alpha}$message
5: the sender sends the pair (A, B) to the verifier
6: the receiver computes $\frac{B}{A^{x^{-1}}} = \text{message}$

6.1.2. *Proving knowledge of discrete logarithms.* For proving knowledge of discrete logarithms the author used Schnorr's signature but he modified the argument for the hash function and he named the result *signature of knowledge*. To compute such a signature we need a private key x and its public key $y = g^x$ and we randomly choose a value $\alpha \in Z_q$. T hen we compute (c, s) as it follows:

$$c = H(g||y||g^r||\text{message})$$

and

$$s = r - cx \pmod{q}$$

So, we can define the pair (c, s) as a signature which satisfies

$$c = H(g||y||g^s y^c||\text{message})$$

because

$$g^s y^c = g^{r-cx}(g^x)^c = g^{r-cx+xc} = g^r$$

The author denoted this signature $SKDL(g, y, \text{message})$ which proves knowledge of the discrete logarithm of the public key y to the base g.

By combining two such signatures we obtain a signature that the logarithms of two group elements with respect to two different bases are the same. So, suppose we have $SKDL(g, y, \text{message})$ and $SKDL(f, z, \text{message})$ and we form a signature $SEQDL(g, f, y, z, \text{message})$. The new signature is a signature of equality of the discrete logarithm of the group element y with respect to the base g and the discrete logarithm of the group element z with respect to the base f for the message message. The pair (c, s) for this new signature is given by

$$c = H(g||f||y||z||g^s y^c||f^s z^c||\text{message}).$$

To compute a signature of knowledge of the discrete logarithm of one group element out of the list $\{y_1, y_2, \dots, y_n\}$ to the base g for the message message there must be known at least one of the secret keys. Assume that this secret key is x_1. Then the prover chooses random values $r, s_2 \dots, s_n, c_2 \dots c_n \in Z_q$ and computes $h_1 = g^r,\ h_i = g^{s_i} y_i{}^{c_i}$ where $i \in \{2, 3, \dots, n\}$. Then c_1 is given by

$$c_1 = H(g||y_1|| \dots ||y_n||h_1|| \dots ||h_n||\text{message}) - \sum_{i=2}^{n} c_i \pmod q$$

and s_1 is given by

$$s_1 = r - x_1 c_1 \pmod q.$$

We denote this signature with

$$SKDL_1^n(g, y_1, \dots, y_n, \text{message}) = (c_1, \dots c_n, s_1, \dots, s_n)$$

We can obtain the signature system $SEQDL_1^n(g, f, y_1, z_1, \dots, y_n, z_n, \text{message})$ by using several $SKDL$ in parallel.

6.2. **Group signature algorithm.** The algebraic settings were described above. So the group's public key is formed from all the members' public key $Y = (y_1, \dots, y_n)$ along with the manager's public key z. The idea behind this algorithm is that if a member wants to sign a message on behalf of the group he has to encrypt a public key from Y and to prove that

(1) the encrypted key is from Y
(2) he knows the discrete logarithm of the encrypted key.

From the second condition it follows that the member has encrypted his own public key since the secret key is in fact the discrete logarithm. To describe the algorithm we choose M_i as the signing member from the group. He first chooses a random value $\alpha \in Z_q$. Then he encrypts his own key y_i by computing $A = z^\alpha$ and $B = y_i g^\alpha$. He computes $(c_1, \dots, c_n, s_c, \dots, s_n)$ from

$$SEQDL_1^n(z, g, A, \frac{B}{y_1}, \dots, A, \frac{B}{y_n}, \text{message})$$

Finally, he computes the pair $(\tilde{c}, \tilde{s} = SKDL(g, B, \text{message}))$.

So the group signature is $(A, B, c_1, \dots, c_n, s_c, \dots, s_n, \tilde{c}, \tilde{s})$. To prove that the two conditions mentioned above are accomplished we discuss the last two signatures. So, the first signature proves that the member has encrypted a key from Y. To better understand this we take M_3 from a group of 5 members to be the one who will sign the message. So the first signature will look like this: $(c_1, \dots c_5, s_1 \dots s_5)$ is

$$SEQDL_1^5(z, g, A, \frac{B}{y_1}, \dots, A, \frac{B}{y_5}, \text{message}).$$

So we have

$$SEQDL_1^5(z, g, A, \frac{B}{y_1}, A, \frac{B}{y_2}, A, \frac{B}{y_3}, A, \frac{B}{y_4}, A, \frac{B}{y_5}, \text{message}).$$
$$SEQDL_1^5(z, g, A, \frac{y_3 g^\alpha}{y_1}, A, \frac{y_3 g^\alpha}{y_2}, A, \frac{y_3 g^\alpha}{y_3}, A, \frac{y_3 g^\alpha}{y_4}, A, \frac{y_3 g^\alpha}{y_5}, \text{message}).$$
$$SEQDL_1^5(z, g, z^\alpha, \frac{B}{y_1}, z^\alpha, \frac{B}{y_2}, \underline{z^\alpha, g^\alpha}, z^\alpha, \frac{B}{y_4}, z^\alpha, \frac{B}{y_5}, \text{message}).$$

The second signature proves the knowledge of the discrete logarithm, and also the identity of the member since he encrypted his own key. So

$$(\tilde{c}, \tilde{s} = SKDL(g, B, \text{message}))$$
$$(\tilde{c}, \tilde{s} = SKDL(g, y_3 g^\alpha, \text{message}))$$

To open a group signature the manager has first to decrypt (A, B).

$$\frac{B}{A^{\omega^{-1}}} = \frac{y_3 g^\alpha}{z^{\alpha\omega^{-1}}} = \frac{y_3 g^\alpha}{g^{\omega\alpha\omega^{-1}}} = y_3$$

So the manager has easily found out the public key of the signer. Then he computes the signature of equality

$$SEQDL(g, z, B/(y_3), A, M_3)$$
$$SEQDL(g, z, y_3 g^\alpha/(y_3), z^\alpha, M_3)$$
$$SEQDL(g, z, g^\alpha, z^\alpha, M_3)$$

The efficiency of this signature is linear in the number of group members. Only the algorithm *Open* is independent of the group's size but finding the identity of a signer given his key requires a look up in a database. The group key and the signatures' length are also linear in the number of group members. This can be a big problem if we are dealing with a large group. The author offers a solution for reducing the size of the group's public key but he himself mentions the problems caused by applying it. This technique was proposed by Blom and uses Φ which is a public generator matrix of an $(n, k)MDS$ code over Z_q. So the group's public key will be $\{y_1, \ldots, y_k\}$ and the public key of a member M_i will be:

$$\tilde{y_i} = \prod_{i=1} k y_i^{\phi_{ij}}$$

where ϕ_{ij} is the element of Φ in row i and column j. There are two big disadvantages when using this matrix. First is that there is needed a third party to compute the public and the private keys, while the second is that if a group of members collude they can find out all secret keys.

7. Group signature scheme based on elliptic curves

To solve the problem raised above regarding the size of the group's public key and, implicitly, the efficiency of the algorithm we chose to modify the Camenisch's algorithm using elliptic curves.

7.1. **Premises.**

7.1.1. *Elliptic Curves Basics.* The elliptic curves are an area of mathematics and have been independently proposed by Neal Koblitz and Victor Miller to be used in cryptography in 1985. Since then, they are widely studied and the cryptographic systems based on elliptic curves become more and more popular.

Definition 7.1 (Weierstrass equation [4])**.** An elliptic curve over a field K is given by

$$E:\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_i \in K$.

Definition 7.2 ([28])**.** The discriminant of an elliptic curve given in the Weierstrass form is:

$$\Delta = d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$$

where:

$$d_2 = a_1 + 4a_2$$
$$d_4 = 2a_4 + a_1a_3$$
$$d_6 = a_3^2 + 4a_6$$
$$d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$$

and $\Delta \neq 0$.

For two extension fields of K the points of the curve are:

$$E(L) = \{(x, y) \in L \times L | E = 0\} \cup O$$

where O is the infinity point.

If $K = F_p$ where $p > 3$ is a prime the Weierstrass equation can be simplified to:

$$E : y^2 = x^3 + ax + b$$

The discriminant of this curve is $\Delta = -16(4a^3 + 27b^2)$. If we have the point $P(x, y)$ then the inverse will be $-P(x, -y)$. If we have $P(x_1, y_1)$ and $Q(x_2, y_2)$ then $P + Q = R(x_3, y_3)$ is given by:

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1,$$

where $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$. For doubling a point $2P(x_3, y_3)$ we use the formulas:

$$x_3 = \lambda^2 - 2x_1$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

where $\lambda = \frac{3x_1^2 + a}{2y_1}$.

The elliptic curve cryptosystems are very efficient and they, also, have a high security level. The efficiency depends the most on the scalar multiplication. This operation has a high time-consuming. The computational speed of such an operation is influenced by:

- finite field operations;
- curve point operations;
- representation of the scalar k [44, 22].

The scalar multiplication kP, is in fact the adding of the point P to itself k times. That means

$$kP = \underbrace{P + P + P + \ldots + P}_{k \text{ times}}$$

and $-kP = k(-P)$.

ECDH. Like some classic cryptosystems' security is based on the DLP (discrete logarithm problem), the elliptic curve cryptosystems' security is based on the ECDLP (elliptic curve discrete logarithm problem). This states that given $P \in E$ and $Q = kP \in E$, k is very hard to find (almost impossible)[13]. For some curves the ECDLP has been solved efficiently [41]. To avoid this problem the elliptic curve must be chosen carefully. NIST recommends fifteen elliptic curves. Specifically, FIPS 186-3 has ten recommended finite fields. There are five prime fields F_p for $p = 192, 224, 256, 384, 521$. For each of the prime fields one elliptic curve is recommended. There are five binary fields F_{2^m} for $2^{163}, 2^{223}, 2^{283}, 2^{409}, 2^{571}$. For each of the binary fields one elliptic curve and one Koblitz curve was selected. The curves were chosen for optimal security and implementation efficiency [43, 7]. The main reason for using elliptic curve cryptography instead of classic systems is because resolving ECDLP implies the same complexity even if the keys are much smaller than the ones used in DLP. This is a big advantage because operating with smaller numbers increases the performance of the algorithm and decreases the amount of storage resources.

7.2. **Algebraic settings.** The algebraic settings are almost the same as the previous algorithm the only difference being the domain parameters of the elliptic curve. Suppose we have a group with n members $M = \{M_1, M_2, \ldots, M_n\}$ and a group manager M^G. For every elliptic curve cryptosystem we have to declare the domain parameters, similarly with [16]. We choose a nonsupersingular elliptic curve E defined over a prime field. The domain parameters are $(F, p, a_E, b_E, G, n, h)$ where F_p is the prime field, a_E, b_E define the curve $E : y^2 = x^3 + a_E x + b_E$, $P \in E$ is a point of order n (this means that n is the smallest positive number for which $nG = O$), $h = |E(F_p)|/n$ is the cofactor. To meet the above conditions it is recommended for $|E(F_p)|$ to be prime or $|E(F_p)| = h \cdot n$ where n is a large prime and $h \in \{1, 2, 3, 4\}$ [15]. Each member has a private key $x_i \in Z_p$ where $i = 1, 2, \ldots, n$ and a public key $y_i = x_i P$. The group manager's private key is denoted with ω and the public one is $z = \omega P$.

7.2.1. *Proving knowledge of elliptic curve discrete logarithms.* The two signature schemes used remain the same with the exception that y is computed by multiplying the public point P with the secret key x. So we have:

$$SKECDL(P, y, \text{message}) = SKECDL(P, xP, \text{message})$$
$$SEQECDL_1^n(P, Q, y_1, z_1, \ldots, y_n, z_n, \text{message})$$

where Q is also a point from the curve E. Thus, the pair (c, s) is given by

$$c = H(P||y||rP||\text{message})$$

where r is a random scalar from Z_p and $s = r - cx$. So the definition of (c, s) becomes

$$c = H(P||y||sP + yc||m)$$

because

$$sP + yc = (r - cx)P + xPc = RP - cxP + xPc = rP.$$

Analogously, the equality signature becomes

$$SEQECDL_1^n(P, Q, y_1, z_1, \dots, y_n, z_n, \text{message}).$$

7.3. **Elliptic curve group signature algorithm.** The group public key is $Y = (y_1, \dots, y_n)$ along with z, the manager's public key. Suppose M_i is the member which will sign the message on behalf of the group. The two conditions stated above remain valid even if we use elliptic curves. First he chooses a random value $\alpha \in Z_p$ and he encrypts his public key y_i by computing

$$A = \alpha z$$
$$B = \alpha P y_i.$$

He finds $(c_1, c_2 \dots c_n, s_1, s_2, \dots s_n)$ from

$$SEQECDL_1^n(z, P, A, \frac{B}{y_1}, \dots A, \frac{B}{y_n}, \text{message})$$

and $(\tilde{c}, \tilde{s}) = SKECDL(P, B, \text{message})$.

To open the elliptic curve signature $(A, B, c_1, \dots, c_n, s_1 \dots, s_n)$ the group manager first decrypts (A, B) by computing

$$\frac{B}{\frac{1}{\omega}A} = \frac{y_i \alpha P}{\frac{1}{\omega}\alpha\omega P} = y_i$$

Then he opens the signature

$$SKECDL(P, z, B/y_i, A, M_i)$$
$$SKECDL(P, z, \alpha P y_i/y_i, \alpha z, M_i)$$
$$SKECDL(P, z, \alpha P, \alpha z, M_i)$$

Thus, the manager can be sure on the validity of the signature and on the identity of the signer.

Algorithm 8 Elliptic Curve Group Signature

1: the signer randomly chooses $\alpha \in Z_p$
2: the manager makes public $z = \omega P$
3: the signer computes $A = \alpha z$
4: the signer computes $B = y\alpha P$
5: the signer computes $(c_1, c_2 \dots c_n, s_1, s_2, \dots s_n)$
6: the signer computes $(\tilde{c}, \tilde{s})$
7: the signature is $(A, B, c_1, c_2 \dots c_n, s_1, s_2, \dots s_n, \tilde{c}, \tilde{s})$

8. Conclusions and comparison

The group signature schemes become more and more popular mainly because companies tend to allow a trusted employee to sign documents on behalf of the institute in order to save time and resources. Various schemes have been proposed most of them using classic cryptography. We have described such a scheme and modified it in order to obtain a more efficient one using elliptic curves. This fact provides a methodology for obtaining high-speed implementations of authentication

protocols and encrypted message techniques while using fewer bits for the keys. In the last years, the cryptosystems based on elliptic curves are increasingly used. This is because their security level and the efficiency one are very high. While the complexity of the classic scheme described depends on the number of the group members, the complexity of our elliptic curve scheme depends the most on the scalar multiplications. In future we intend to study various methods for computing point multiplications on elliptic curves and to choose the most suitable one for the proposed group signature scheme.

Acknowledgments. The authors acknowledges the support through Grant of The Executive Council for Funding Higher Education, Research and Innovation, Romania-UEFISCDI, Project Type: Advanced Collaborative Research Projects - PCCA, Number 23/2014.

References

[1] Ramzi Alsaedi, Nicolae Constantinescu, Vicentiu Radulescu; Nonlinearities in Elliptic Curve Authentication, *Entropy*, Vol. **16**(9), pp. 5144–5158, 2014.

[2] L. Babai; Trading group theory for randomness, *ACM Symposium on Theory of Computing*, pp. 421–429, Providence, Rhode Island, 6-8 May 1985.

[3] L. Babai, S. Moran; Arthur - Merlin games: A randomized proof system, and a hierarchy of complexity classes, *Journal of Computer and System Sciences*, Vol. **36**, 1988.

[4] J. Camenisch; Effcient and generalized group signatures, *In Advances in Cryptology EUROCRYPT '97*, Vol. **1233** of Lecture Notes in Computer Science, Springer-Verlag, pp. 465–479, 1997.

[5] J. L. Camenisch; *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*, PhD thesis, ETH Zurich, Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz, 1998.

[6] J. Camenisch, M. Stadler; Efficient group signatures schemes for large groups, *Advances in Cryptology-Crypto 1997*, Vol. **1294** of Lecture Notes in Computer Science, Springer-Verlag, pp. 410–424, 1997.

[7] Certicom Research; SEC 2: Recommended Elliptic Curve Domain Parameters, *Standards for efficient Cryptography*, Version 1.0, Sep. 2000

[8] D. Chaum, E. van Heyst; Group signatures, *Advances in Cryptology EUROCRYPT '91*, Vol. **547** of Lecture Notes in Computer Science, Springer-Verlag, pp. 257–265, 1991.

[9] L. Chen, T. P. Pedersen; New group signature schemes, *Advances in Cryptology - EUROCRYPT '94*, Vol. **950** of Lecture Notes in Computer Science, Springer-Verlag, pp. 171–181, 1995.

[10] H. Cohen; *A Course in Computational Algebraic Number Theory*, No. 138 in Graduate Texts in Mathematics, Springer-Verlag, Berlin, 1993.

[11] H. Cohen, H. W. Lenstra jr.; Primality Testing and Jacobi Sums, *Mathematics of Computation*, Vol. **42**(165), pp. 297–330, 1984.

[12] Nicolae Constantinescu; Authentication ranks with identities based on elliptic curves, *Annals of the University of Craiova, Mathematics and Computer Science Series*, Vol. **XXXIV**(1), pp. 94–99, 2007.

[13] Nicolae Constantinescu; Security System Vulnerabilities, *Proceedings of the Romanian Academy Series A-Mathematics Physics Technical Sciences Information Science*, Vol. **13**(2), pp. 175–179, 2012.

[14] Nicolae Constantinescu; Authentication hierarchy based on blind signature, *Journal of Knowledge Communication and Computing Technologies*, Vol. **1**(1), pp. 77–84, 2010.

[15] N. Constantinescu; *Criptografie*, Ed. Academiei Romane, Bucuresti, 2009.

[16] Nicolae Constantinescu, George Stephanides, Mirel Cosulschi, Mihai Gabroveanu; RSA-Padding Signatures with Attack Studies, *International Conference on Web Information Systems and Technologies: Internet Technology/Web Interface and Applications*, Portugal, ISBN 978-972-8865-46-7, pp. 97–100, 2006.

[17] J. Cowie, B. Dodson, R. M. E.-Huizing, A. K. Lenstra, P. L. Montgomery, J. Zayer; A world wide number field sieve factoring record: On to 512 bits, *Advances in Cryptology-ASIACRYPT '96*, Vol. **1163** of Lecture Notes in Computer Science, Springer-Verlag, pp. 382–394, 1996.

[18] W. Diffie, M. E. Hellman; New Directions in Cryptography, *IEEE Transactions on Information Theory*, Vol. **22**(6), pp. 644–654, Nov. 1976.

[19] U. Feige, A. Fiat, A. Shamir; Zero knowledge proofs of identity, *Journal of Cryptology*, Vol. **1**, pp. 77–94, 1987.

[20] O. Goldreich, S. Micali, Avi Wigderson; Proofs that yield nothing but their validity, *Journal of the ACM*, Vol. **38**(3), pp. 690–728, July 1991.

[21] S. Goldwasser, S. Micali, C. Rackoff; The knowledge complexity of interactive proof systems, *Proc. 27th Annual Symposium on Foundations of Computer Science*, pp. 291–304, 1985.

[22] J. Guajardo, C. Paar; Itoh-tsuji inversion in standard basis and its application in cryptography and codes. Design, *Codes and Cryptography*, Vol. **25**(2), pp. 207–216.

[23] Ion Iancu, Nicolae Constantinescu, Mihaela Colhon; Fingerprints Identification using a Fuzzy Logic System, *International Journal of Computers, Communications & Control*, Vol. **5**(4), pp. 525–531, 2010.

[24] J. Kilian, E. Petrank; Identity escrow, *Advances in Cryptology - CRYPTO '98*, Vol. **1642** of Lecture Notes in Computer Science, Springer-Verlag, pp. 169–185, Berlin, 1998.

[25] D. E. Knuth; *The Art of Computer Programming*, Addison-Wesley, second edition, 1981.

[26] A. G. Konheim; Computer Security and Cryptography, *Wiley*, 2007.

[27] A. K. Lenstra, H. W. Lenstra Jr.; *The Development of the Number Field Sieve*, Vol. **1554** of Lecture Notes in Mathematics, Springer-Verlag, 1993.

[28] A. Lysyanskaya; Signature Schemes and Applications to Cryptographic Protocol Design, PhD thesis, *MIT*, 2002.

[29] A. Lysyanskaya, Z. Ramzan; Group blind digital signatures: A scalable solution to electronic cash, *Proc. of Second International Conference on Financial Cryptography*, 1998.

[30] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone; *Handbook of Applied Cryptography*, CRC Press, 1997.

[31] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone; Handbook of Applied Cryptography, 5th Ed. *CRC Press*, 2001.

[32] Rolf Oppliger; *Contemporary Cryptography*, Artech House, 2005.

[33] H. Petersen; *How to convert any digital signature scheme into a group signature scheme*, Security Protocols Workshop, Paris, 1997.

[34] J. M. Pollard; Monte Carlo methods for index computation (mod p), *Mathematics of Computation*, Vol. **32**(143), pp. 918–924, July 1978.

[35] J. M. Pollard; A monte carlo method for factorization, *BIT*, pp. 331–334, 1975.

[36] C. Pomerance; The quadratic sieve factoring algorithm, *Advances in Cryptology*, Vol. **209** of Lecture Notes in Computer Science, Springer-Verlag, pp. 169–182, 1985.

[37] M. O. Rabin; Probabilistic algorithm for testing primality, *Journal of Number Theory*, Vol. **12**(1), pp. 128–138, 1980.

[38] R. Rivest, A. Shamir, L. Adleman; A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM*, Vol. **21**(2), pp. 120–126, 1978.

[39] Emil Simion, Nicolae Constantinescu; Complexity Computations in Code Cracking Problems, *Concurrent Engineering in Electronic Packaging, IEEE Communication*, may 05-09, pp. 225–232, ISSE 2001.

[40] C. P. Schnorr; Efficient identification and signatures for smart cards, *Advances in Cryptology - Crypto '89*, Vol. **435** of Lecture Notes in Computer Science, Springer-Verlag, pp. 239–252, Springer-Verlag, 1990.

[41] N. Smart; How secure are elliptic curves over composite extension fields?, *EUROCRYPT 2001*, Vol. **2045** of Lecture Notes in Computer Science, Springer-Verlag, pp. 30–39, 2001.

[42] Oana Ticleanu; Nicolae Constantinescu, Daniel Ebanca, Intelligent data retrieval with hierarchically structured information, *KES-IIMS*, Vol. **254**, pp. 345–351, Jun 26-28, Portugal, 2013.

[43] U.S. Dept of Commerce/NIST; Digital Signature Standard (DSS), *FIPS PUB* 186, Jan. 2000.

[44] D. Yong, G. Feng; High speed modular divider based on GCD algorithm over GF(2m), *Journal on Communications*, Vol. **29**(10), pp. 199–204, oct. 2008

Alin Ionuţ Golumbeanu
University of Craiova, Department of Mathematics, Street: A. I. Cuza 13, 200585 Craiova, Romania
E-mail address: alin.golumbeanu@inf.ucv.ro

Oana Adriana Ţicleanu
University of Craiova, Department of Informatics, Street: A.I. Cuza 13, 200585 Craiova, Romania
E-mail address: oana.ticleanu@inf.ucv.ro