A New Distribution Version of Boneh-Goh-Nissim Cryptosystem: Security and performance analysis

Oualid Benamara\textsuperscript{a}, Fatiha Merazka\textsuperscript{b}

\textsuperscript{a}University of Science and Technology Houari Boumediene, Institute of mathematics
\textsuperscript{b}University of Science and Technology Houari Boumediene, Institute of electronics

Abstract

The aim of this paper is to provide two distributed versions of Boneh elliptic curve cryptography (BECC) algorithm. We give a proof of semantic security for the first one. This guarantees that our algorithm is semantically secure in the contest of active non-adaptive adversary. Furthermore, we prove that the second version of our distributed scheme is computationally more efficient in time computation than ElGamal elliptic curve threshold cryptosystem and secure under the Subgroup Decision Problem assumption.

Keywords: Elliptic curve cryptography, Threshold scheme, ElGamal cryptosystem.

1. Introduction

Public key cryptosystems are widely used nowadays in electronic banking, online browsing, voting systems and so on. While symmetric systems are more efficient than the asymmetric ones, those later are suitable for key handling in an intrusted setup. A well known drawback in public key cryptography, is that the knowledge of the secret keys (like a server) provide full control, and may be viewed as a weakness. Hence a new direction in research named distributed systems. Instead of relying on a single trusted party, the key is distributed among \( n \) parties and at least \( m \) parties must collaborate in

\textit{Email addresses:} benamara.oualid@gmail.com (Oualid Benamara), fmerazka@usthb.dz (Fatiha Merazka)
order to recover the secret key. $m$ is chosen as a parameter of the system, at the setup level.

The definition of semantic security of threshold cryptosystem used in this paper follows the one stated in [7].

The decryption operation of the BGNC (?) used in this paper is done by computing the discrete logarithm on elliptic curve. We use Pollard Lambda algorithm (reference) which has complexity $O(\sqrt{T})$ in time of computation, where $T$ is the length of the interval to which the message $m$ belongs.

Our second algorithm Boneh elliptic curve cryptography (BECC) is more efficient than EECC in terms of computation efficiency. We evaluate the computational time of our algorithm and compare it to EECC[6].

In our paper, we focus on encryption and decryption time.

1.1. Organisation

This paper is organized as follows. In Section 3.1 we recall the basic definition and the scenario when dealing with threshold cryptography. We present our novel threshold scheme together with a proof of security of this later in Section 4. The other distribution version together with its complexity analysis are in Section 6.

The security of the Paillier, ElGamal ECC (EECC) and BGNC cryptosystems rely on the Decisional Composite Residuosity assumption (DCRA) [12], the Elliptic Curve Discrete Logarithm Problem (ECDLP) [1] and the Sub-group Decision Problem assumptions [2] respectively. So, building on the ECDLP and the SDP assumptions, we prove that our distributed scheme is semantically secure, in the meaning introduced in [7]. For the best of our knowledge, this has never been done.

2. Related work

Several elliptic curve distributed systems have been proposed in the literature [7], [8], [11], [4], [14]. Our paper aims to introduce two novel distributed schemes for the BGNC introduced in [2] following the paradigm of [7]. Fouque et al. [7] proposed a first distribution of the Paillier system together with a proof of semantic security. In [8], Fournaris proposed a distributed version of ECC. However, no proof of security was given.
3. Preliminaries

3.1. Zero Knowledge (ZK) Proof of Equality

In this section we will recall the ZK proof of equality algorithm given in [14] to produce proofs of validity in step 3 of our threshold ECC, see Section 4.

Suppose that party B needs to prove that it has a valid secret share key $s$ to party A without divulging any information about $s$.

1. A selects two random integers $a$ and $b$ smaller than the order of the cyclic group generated by a point of an elliptic curve.
2. It computes $(a_1, a_2) = asg \mod p$ and $(b_1, b_2) = bg \mod p$, where $p$ is a prime number related to the finite field $GF(p)$ on which the elliptic curve $E$ is constructed and $g$ is a generator of the group $G$ of the points on $E$.
3. It also computes $t_1 = a_1b_1 \mod p$ and $t_2 = a_2b_2 \mod p$ and sends $(ag, t_1, t_2)$ to party B.
4. B computes $sag = (r_1, r_2) \mod p$.
5. B computes $z_1 = t_1r_1^{-1} \mod p$ and $z_2 = t_2r_2^{-1} \mod p$ and sends $(z_1, z_2)$ to A.

At the end, A verifies that $(b_1, b_2) = (z_1, z_2)$. If this is true, B has the secret $s$. Otherwise, the proof $proof = (z_1, z_2)$ is not valid.

The correctness of this scheme can be verified from the Menezes–Vanstone elliptic curve cryptosystem [14], section 4.

3.2. Formal Definition

A threshold cryptosystem consists of the following four components:

- A key generation algorithm takes as input a security parameter $k$, the number $l$ of decryption servers, the threshold parameter $t$ and a random string $\omega$. It outputs a public key $PK$, a list $PK_1, \ldots, PK_l$ of private keys and a list $VK, VK_1, \ldots, VK_l$ of verification keys.

- An encryption algorithm takes as input the public key $PK$, a random string $\omega$ and a clear text $M$. It outputs a ciphertext $c$.

- A share decryption algorithm takes as input the public key $PK$, an index $1 \leq i \leq l$, the private key $SK_i$ and a ciphertext $c$. It outputs a decryption share $c_i$ and a proof of its validity $proof_i$. 

3
A combining algorithm takes as input the public key $PK$, a ciphertext $c$, a list $c_1,\ldots,c_l$ of decryption shares, the list $VK,VK_1,\ldots,VK_l$ of verification keys and a list $proof_1,\ldots,proof_l$ of validity proofs. It outputs a cleartext $M$ or fails.

3.3. The Players and the Scenario

The game includes the following players: a dealer, a combiner, a set of $l$ servers $P_i$, an adversary and users. All are considered as probabilistic polynomial time Turing machines, playing in the following scenario:

- In an initialization phase, the dealer uses the key generation algorithm to create the public, private and the verification keys. The public key $PK$ and all the verification keys $VK,VK_i$; where $1 \leq i \leq l$ are publicized and each server receives its shares $SK_i$ of the secret key $SK$.

- To encrypt a message, any user can run the encryption algorithm using the public key $PK$.

- To decrypt a ciphertext $c$, the combiner first forwards $c$ to the servers. Using their secret key $SK_i$ and their verification keys $VK,VK_i$, each server runs the decryption algorithm and outputs a partial decryption $c_i$ with a proof of validity of the partial decryption $proof_i$. Finally, the combiner uses the combining algorithm to recover the cleartext, if enough partial decryptions are valid.

3.4. Security requirements

We recall that an active non-adaptive adversary completely controls the behavior of the corrupted servers and he chooses which servers he wants to corrupt before key generation. A threshold cryptosystem is said to be $t$-robust if the combiner is able to correctly decrypt any ciphertext, even in the presence of an adversary who actively corrupts up to $t$ servers. Let us consider an attacker who first issues two messages $M_0$ and $M_1$. We randomly choose one of these messages. We encrypt it and send this ciphertext to the attacker. Finally, he answers which message has been encrypted. We say that the encryption scheme is semantically secure if there exists no such polynomial time attacker able to guess which of the two messages has been encrypted with a non-negligible advantage. The semantic security for threshold cryptosystem is defined as follow:
Let be an attacker who actively and non-adaptively corrupts $t$ servers learns the public parameters, the secret keys of the corrupted servers, the public verification keys, all the decryption shares and the proof of validity of those shares. We consider the following game A:

A1: The attacker chooses to corrupt $t$ servers. He learns all their secret information and he actively controls their behavior.

A2: The key generation algorithm is run; the public keys are publicized, each server receives its secret keys and the attacker learns the secrets of the corrupted players.

A3: The attacker chooses a message $M$ and a partial decryption oracle gives her $l$ valid decryption shares of the encryption of $M$, along with proofs of validity. This step is repeated as many times as the attacker wishes.

A4: The attacker issues two messages $M_0$ and $M_1$ and send them to an encryption oracle who randomly chooses a bit $b$ and sends back an encryption $c$ of $M_b$ to the attacker.

A5: The attacker repeats step A3, asking for decryption shares of encryptions of chosen messages.

A6: The attacker outputs a bit $b'$.

A threshold encryption scheme is said to be semantically secure against active non-adaptive adversaries if for any polynomial time attacker, $b = b'$ with probability only negligibly greater than $1/2$.

4. The Boneh-Goh-Nissim Cryptosystem

We recall that the BGNC presented in this section is semantically secure under the subgroup decision problem (SDP) assumption[^2].

4.1. Description

**Key Generation:** The public key is $(n, G, g, h)$, where $G$ is a group of order $n$, $n = q_1q_2$, $g$ and $u$ are generators of $G$ and $h = u.q_2$. The private key is $q_1$. 
**Encryption**: Pick a random \( r \in [0, n - 1] \). Let \( m \in [0, \ldots, T] \) be a message to encrypt. So the ciphertext is computed as

\[
C = m \times g + r \times h.
\]

(1)

**Decryption**: Compute

\[
m = \log_{q_1 g} q_1 C.
\]

(2)

4.2. Correctness

The equation 2 can be written in the following form:

\[
mq_1 g = q_1 C
\]

(3)

We have to prove that the above equation is equivalent to equation 1. Multiply the two sides of equation 1 by \( q_1 \) we get:

\[
q_1 C = q_1 (m \times g + r \times h)
\]

\[
= q_1 \times m \times g + q_1 \times r \times h
\]

\[
= q_1 mg + q_1 r(uq_2) \quad \text{(recall that } h = uq_2)\)
\]

\[
= mq_1 g + ru(q_1 q_2)
\]

\[
= mq_1 g + run \quad \text{(since } n = q_1 q_2)\)
\]

\[
= mq_1 g. \quad \text{(since } n \text{ is the order of the group } G \text{ and } u \text{ is a generator of the group.)}
\]

4.3. Distribution Version of BGNC

**Key Generation**: The dealer chooses an elliptic curve \( E \) such that the order of the group of the point of this curve is \( p + 1 = n \times s \) and such that \( n = q_1 q_2 \), \( q_1 \) and \( q_2 \) are two sufficiently large prime numbers. So there exist a subgroup \( G \) of order \( n \). Let \( g \) and \( u \) be two generators of \( G \), set \( h = q_2 u \) and \( g_0 = q_1 g \). The public key \( PK \) will be \( PK = (n, G, g, h, g_0) \) and the secret key \( SK = q_1 \). It is shared using Shamir’s secret sharing scheme: let \( f_0 = SK \) and, for \( i = 1, \ldots, t \), \( f_i \) is randomly chosen in \( \mathbb{Z}_n \). Let \( f(X) = \sum_{i=0}^{t} f_i X^i \); the secret key \( SK_i \) is \( d_i = f(i) \mod n \).

The \( d_i \)'s must be different from \( q_1 \) and \( q_2 \) so that the decryption shares \( c_i \) may exist. We have to ensure that \( c_i \in [0, T] \) for a suitable chosen \( T \).

**Encryption**: Compute \( C = m \times g + r \times h \) as in the BGNC.
Share Decryption Algorithm: Let
\[ d_i = q_1 \alpha_i + \beta_i, \quad \beta_i < q_1 \]
\( \beta_i \) here is the remaining of the division of \( d_i \) by \( q_1 \). \( \alpha_i \) is the quotient. Define
\[ \gamma_i = q_1 \alpha_i \]
Compute
\[ c_i = \log_{q_1} \gamma_i C \]
for \( i = 1, \ldots, l \). The share of each server will be the couple \((c_i, g_i)\). To convince anyone that the server \( i \) has a valid share \( d_i \), it uses zero knowledge proof presented in section 3.1 resulting in a proof of validity.

Combining Algorithm: Let \( S \) be a set of valid decryption shares \( c_i, i = 1, \ldots, t + 1 \). Compute
\[ m = \log_B A, \]
where \( B = \sum_{j \in S} \mu_{0,j}^S g_j \) and \( A = \sum_{j \in S} \mu_{0,j}^S (c_i g_0 + \beta_i C) \). The coefficients \( \mu_{i,j}^S \) are the Lagrange coefficients defined by
\[ \mu_{i,j}^S = D \times \prod_{j' \in S \setminus j} (i - j') \prod_{j' \in S \setminus j} (j - j') \in \mathbb{Z} \]
for any \( i \in \{0, \ldots, l\} \) and any \( j \in S \).

We will use this Lagrange interpolation formula:
\[ Df(i) = \sum_{j \in S} \mu_{i,j}^S f(j) \mod n \]
for any \( i \in \{0, \ldots, l\} \) and any \( j \in S \).
So for \( i = 0 \):
\[ Df(0) = \sum_{j \in S} \mu_{0,j}^S f(j) \mod n. \]

Thus
\[ Dd_0 = \sum_{j \in S} \mu_{0,j}^S d_j. \quad (4) \]

We have that
c_i = \log_{q_1} \gamma_i C \iff c_i q_1 g = \gamma_i C
\iff c_i g_0 = \gamma_i C \text{ since } g_0 = q_1 g
\iff c_i g_0 = (d_i - \beta_i) C \text{ since } \gamma_i = d_i - \beta_i
\iff c_i g_0 = d_i C - \beta_i C.

Finally
\[ c_i = \log_{q_1} \gamma_i C \iff c_i g_0 + \beta_i C = d_i C \quad (5) \]
and
\[ m = \log_{d_0} d_0 C \iff md_0 g = d_0 C. \]

Multiplying the above formula by $D$ we obtain:
\[ m D d_0 g = d_0 DC, \]
so by formula $4$
\[ m \Sigma_{j \in S} \mu_{0,j}^S d_j g = \Sigma_{j \in S} \mu_{0,j}^S d_j C \]
and by formula $5$
\[ \iff m \Sigma_{j \in S} \mu_{0,j}^S g_j = \Sigma_{j \in S} \mu_{0,j}^S (c_i g_0 + \beta_i C), \]
or
\[ m B = A. \]

So the proposed scheme is correct.

5. Numerical Example

We have used Sage software to make a numerical example.

Let $p = 100000000000000003$ and $q = 100000000000000013$. With those parameters we obtain an elliptic curve of order: 960000000000000015360000000000003744. We choose for illustration purpose $l = 20$ and $t = 8$. We have computed two points of order $n = pq$:
\[ H = (901696848398424513792042624183449898 : 919534227604146025519668445930592752 : 1) \]
and
\[ D = (51558236239313508339332053520498553 : 934365455532976888281795609449803454 : 1) \]
A generator of the curve is: $G = (7441145344472669657966133843400531746 : 7743547579143442084416408483805765 : 1)$. We chose to encrypt a message $m = 10$. We obtain the shares: $[320, 15530, 213640, 1457210, 6593360, 22845370, 65658680, 164354090]$. We apply the formula stated in the algorithm, we recover the clear message $m = 10$. 

8
5.1. Details on Sage Commands Used for the Above Example

To obtain the two prime numbers and the suitable order for the field we use the following code:

```python
p = next_prime(100000000000000000); # this will generate a prime greater then 100000000000000000.
q = next_prime(p); # a prime number greater than p.
n = p * q;
l = 1;
pp = l*n-1;
while (not(is_prime(pp)) or mod(pp,3)==1):
    l = l+1;
    pp = l*n-1;
```
After running this code, we obtain a prime number \( pp \) with \( pp + 1 = l \cdot n \) and this will ensure that the following command:

\[
E = \text{EllipticCurve}(\text{GF}(pp),[0,1]);
\]

will generate an elliptic curve of order \( pp \) as required in our key generation step. When looking for a generator of the curve, we use

\[
E.gens();
\]

To obtain a point of order \( n \) we generate a random point and test if its order is as desired:

\[
o = 1;
while (o <> n):
    H = E.random_point();
    o = H.order();
\]

The discrete logarithm in Sage when the range of the logarithm is known is achieved in Sage with the command:

\[
\text{discrete_log_lambda}(P,Q,[0,T],operation='+');
\]

Here \( Q \) is the base, \( T \) is the range and the operation is relative to the group on which we are working on.

**Theorem 1.** Under the SDP and the ECDLP assumptions, the distributed version of BGNC (DBGNC) is semantically secure against active non-adaptive adversaries.

*Proof.* We prove using the reduction process. We show that if an adversary can break the semantic security of the DBGNC, then we can construct an attacker who can break the semantic security of BGNC. Following this procedure, we have to simulate data received by the adversary in steps A2, A3 and A5 of the game A. So the proof consists of proving that the data simulated during the steps of the game A are indistinguishable from real one.

Let us assume the existence of an adversary \( A \) able to break the semantic security of the threshold scheme. We now describe an attacker which uses \( A \) in order to break the semantic security of BGNC. In a first phase the attacker obtains the public key \((n, G, g, h)\) and he chooses two messages \( M_0 \) and \( M_1 \) which are sent to an encryption oracle who randomly chooses a bit
and return an encryption $c$ of $M_b$. In a second phase the attacker tries to guess which message has been encrypted.

We now describe how to feed an adversary $A$ of the threshold scheme in order to make a semantic attacker. In step $A1$ of game $A$, the adversary chooses to corrupt $t$ servers $P_1, \ldots, P_t$. In the find phase, the attacker first obtains the public key $PK = (n, G, g, h)$ of the BGNC. He randomly chooses $t$ values $d_1, \ldots, d_t$ in the range $\{0, \ldots, n\}$. $T$ is suitably chosen to efficiently compute the discrete logarithm on the elliptic curve. As $q_2$ is unknown and $T < q_2$. We have to simulate the value of $T$. However, as the computation efficiency of the function $\log$ is well known, a suitable choice of $T$ may be obtained.

In step 2 of game $A$ the attacker sends $(n, G, g, h, d_1, \ldots, d_t)$ to $A$. During step $A3$, $A$ chooses a message $M$ and send it to the attacker. He compute a valid encryption of $M$ given by $c = r \times h + M \times g$, where $r$ is a random number. The decryption shares of the corrupted players are correctly computed using the $d_i$'s: $c_i = \log_{d_i}^g \gamma_i c$, and $g_i = d_i g$ for $i = 1, \ldots, t$. The other shares are computed from the following formula

$$c_i = \log_{g_i} b,$$

where $a = D g_0$ and $b = \left( \sum_{j \in S} \mu_{ij}(\beta_j c + g_0 c_j) \right) - D \beta_i c$. We have that $c_i = \log_{g_0} \gamma_i c$. So $c_i g_0 = \gamma_i c$, and by multiplying by $D$: $D c_i g_0 = D \gamma_i c = D(d_i - \beta_i)c = D d_i c - D \beta_i c$. We have also $D d_i = \sum_{j \in S} \mu_{ij} d_j$, this is why we chosen the values of $c_i$ as in the above formula.

Finally, the adversary returns

$$(c, c_1, \ldots, c_l, proof_1, \ldots, proof_l).$$

In step $A4$, $A$ chooses and outputs two messages $M_0$ and $M_1$. The attacker outputs those two messages as the results of the find phase.

The encryption oracle for the non-threshold ECC scheme chooses a random bit and sends an encryption $c$ of $M_b$ to the attacker. He forwards $c$ to the adversary $A$.

Step $A5$ is similar to step $A3$. Finally, in step $A6$, $A$ answers a bit $b'$ which is returned by the attacker in the guess phase.

We have that $d_1, \ldots, d_t$ and the verification keys are randomly chosen on the interval $[0, T]$. So the distribution received by $A$ during the key generation step is indistinguishable from a real one.
Also, the value received in step A3 and A5 are computed from the secret keys and the values of the non-corrupted servers are randomly chosen on the interval $[0,n]$.

Finally, all the data simulated by the attacker cannot be distinguished from real ones by $A$. Consequently, if there exists a polynomial time adversary $A$ able to break the semantic security of the threshold scheme, we have made an attacker able to break the semantic security of the original ECC scheme.

6. A Second Distribution of BGNC

6.1. The ElGamal Distributed Version

First we recall the distributed version of the ElGamal elliptic curve cryptosystem (DEGECC) taken from [11]. Let the parameters of the cryptosystem be a finite field $GF(p)$, where $p$ is a prime number, an elliptic curve $E_p(a,b)$ and a point $G$ of order $q$ on the group of point of $E$. Then we run the cryptosystem as follow:

Bob’s private key is $n_B$ with $0 < n_B < q$ and the public key is $K_B = n_BG$.

1. First we choose a prime number $p > \max(M,n)$, and define $a_0 = M$ to be the message. Then we randomly select $k−1$ independent coefficients $a_1, \ldots, a_{k-1}$, with $0 \leq a_j \leq p-1$; which define the random polynomial $f(x)$ over a Galois prime field $GF(p)$.
2. We compute $n$ shares, $M_i = f(x_i) \mod p$, $1 \leq i \leq n$, where $x_i$ can be just the public index $i$ for simplicity. Convert each index $i$ to a point $P_i$ on the elliptic curve $E$.
3. Alice picks a random number $r$ and send $rG$ and $P_i + rK_B$ to Bob with index $t$.
4. Bob recovers each elliptic curve point by calculating $P_i + rK_B - n_BrG = P_i$.
5. Bob converts $P_i$ to $M_i$ and deduces $M$ by using Lagrange interpolation formula.

Our novel distributed cryptosystem runs as follows: The public and private data are as in section 4.

1. First Define $a_0 = M$, the message. Then we select $k-1$ random, independent coefficients $a_1, \ldots, a_{k-1}$, $0 \leq a_j \leq p - 1$, defining the random polynomial $f(x)$ over $GF(p)$. 
2. We compute $n$ shares, $M_i = f(x_i) \mod p$, $1 \leq i \leq n$, where $x_i$ can be just the public index $i$ for simplicity.

3. Alice picks a random number $r \in [0, n - 1]$. So the ciphertext is computed as

$$C_i = M_i \times g + r \times h.$$ 

4. Bob recovers each elliptic curve point by calculating

$$M_i = \log_{q_1} q_1 C_i.$$ 

5. Bob deduces $M$ by using Lagrange interpolation formula.

If an adversary wishes to cryptanalyse our scheme, he will have to break the SDP assumption, which has not been broken to the best of our knowledge.

6.2. Implementation Details and Computation Efficiency

In this section we compare our second version of the distributed BGNC to DEGECC. We focus on the ciphering plus the deciphering time of the two cryptosystems. We have used Sage software [13] to compare the computational efficiency of the two algorithms. Rather then implementing the hole algorithm, we restricted to the encryption plus the decryption functions, as the other parts of the two algorithms are the same. As we deal with the computation time, we take the average value of all the random values $r$. Also, we have chosen $q/2$ as the private key $n_B$.

For the BGNC, we have used $T = 100$. So, for a given share $M_i < p$, write $M_i$ in base 100. Thus, we obtain $(M'_1,\ldots,M'_s)$, $s$ depending on the value of $p$. Hence, for all $0 < i < s + 1$, $0 < M'_i < 100$.

We compute the number of point addition during the ciphering plus the deciphering process of the two cryptosystems. This is because this operation is the more expensive in time computation. So, for the DEGECC, we will need $r + r + 1 + n_B + 1 = 2r + 2 + n_B$ point addition for encryption and decryption of one single point $P_i$. For the BGNC, we will need $r + M_i$ point addition plus $s/2$ computation of the logarithm with the Pollard Lambda algorithm. As the value of $s$ will have logarithm behavior, DEGECC will be more costly then BGNC for large keys.

We have taken 3 curves of different key lengths from [9] and computed the average time consumption of 100 runs of the two algorithms. We plotted that and obtained figure[1] which shows the good performance of our algorithm over the ElGamal one.
7. Conclusion

We have designed two distributions of the BGNC. The first one is proved semantically secure and the second one is more efficient than the DEGECC. While efficiency is considered a poor comparison parameter, it can be an essential one in the setting of a sensor network, wherein energy consumption is of concern, rather than the security level of the scheme.

8. References

References

[1] O. Benamara, *Elliptic Curve Discrete Logarithm Problem*, General Mathematics Notes, vol. 15, pp. 84–91, March 2013.

[2] D. Boneh, E.J. Goh and N. Kobbi, *Evaluating 2-DNF formulas on ciphertexts*, Proceedings of the Second international conference on Theory of Cryptography, no. 17, pp. 325–341, 2005.

[3] R. Broker, P. Stevenhagen, *Constructing of elliptic curves of prime order*, Contemporary Mathematics, vol. 20, 2007.

[4] P. Changgen, and L. Xiang, *Threshold Signcryption Scheme Based on Elliptic Curve Cryptosystem and Verifiable Secret Sharing*, Proc. Intern. Conference on Wireless Communications, pp. 1182–1185, 2005.
[5] S. Farah, M.Y. Javed, A. Shamim and T. Nawaz, An experimental study on Performance Evaluation of Asymmetric Encryption Algorithms, Recent Adv. in Information Science, vol. 62, no. 1, pp. 31–42, 2012.

[6] O. Farras et al., Linear threshold multisecret sharing schemes, Inform. Processing Letters, vol. 112, no. 17-18, pp. 667–673, 2012.

[7] P.A. Fouque, G. Poupard and J. Stern, Sharing decryption in the context of voting or lotteries, Financial Cryptography, LNCS, pp. 90–104, 2000.

[8] A.P. Fournaris, A Distributed Approach of a Threshold Certificate-Based Encryption Scheme with No Trusted Entities, Information Security Journal: A Global Perspective, vol. 22, no. 3, pp. 126–139, 2013.

[9] H. Ivey-Law and R. Rolland, Constructing a database of cryptographically strong elliptic curves, Crypto’Puces, 9th-12th May 2011, http://galg.acrypta.com

[10] K. Lauter, The advantages of elliptic curve cryptography for wireless security, Wireless Communications, IEEE, vol. 11, no. 1, pp. 62–67, 2004.

[11] E. Levent and L. Weimin, ECC based threshold cryptography for secure data forwarding and secure key exchange in MANET (i), Proceedings of the 4th IFIP-TC6 international conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems, NETWORKING 05, no.12, pp. 102–113, 2005.

[12] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, Proceedings of the 17th international conference on Theory and application of cryptographic techniques, Springer-Verlag, Berlin, Heidelberg, no. 16, pp. 223–238, 1999.

[13] A. Stein and others, Sage Mathematics Software (Version 5.11), The Sage Development Team, 2013.

[14] X. Zhang, F. Zhang, Z. Qin, J. Liu, ECC Based threshold decryption scheme and its application in web security, Journal of Electronic Science and Technology of China, vol. 2, no. 4, 2004.