An Efficient Distributed Key Generation Protocol for Secure Communications with Causal Ordering

Caimu Tang (Student Member, IEEE)
Department of Computer Science
University of Southern California, Los Angeles
caimut@cs.usc.edu

Anthony T. Chronopoulos (Senior Member, IEEE) [1]
Department of Computer Science
University of Texas at San Antonio, San Antonio
atc@cs.utsa.edu

Abstract— **In this paper, we propose an elliptic curve based distributed key generation protocol in communication systems with causal ordering semantics on broadcast messages.**

I. Introduction

The seminal work of Shamir [8] and later Feldman [2] on the (t, n) (where n is the number of shares of a secret and t is the threshold) threshold scheme is based on a simple polynomial interpolation over a Galois field, GF(q), where q is a prime or some power of a prime p by which it is an extension field of $Z/_pZ$. Pedersen [7] gives an extension to Feldman's non-interactive approach which allows that each party can verify the information about the secret without communicating to other parties, and any t of these parties can later find the secret ($1 \leq t \leq n$), and fewer than t parties get no information about the secret.

Existing distributed key generation (DKG) protocols are based on either discrete logarithm problem (DLP) over a finite field or integer factorization problem (IFP). In order to maintain a certain level of secrecy, key lengths in both cases have to be long enough to be secure due to recent developed subexponential algorithms on IFP and DLP. Elliptic curve cryptosystems (ECC), on the other hand, are safe against some common algorithmic techniques, e.g. index-calculus [4]. There is no specific subexponential algorithm for elliptic curve discrete logarithm problem (ECDLP) if some precaution is exercised upon selecting a proper curve and associated parameters.

In [2], a non-distributed version of verifiable secret sharing (VSS) scheme is presented in which the *dealer* (a dealer is defined as the threshold coordinator which distributes the shares to each player) selects and encrypts a "secret message", s, and gives a "share" of s, to each of n players. All communications use broadcast messages and the players can verify the authenticity of the dealer. Shamir's $(n, t, t{+}1)$ threshold cryptography can be used as a building block to VSS. The first distributed version of VSS denoted by DF-VSS in this paper is presented in [6], and it is based on Feldman VSS (where each player acts as a dealer). In [3], an improved (in terms of its secrecy) version of DF-VSS is presented and it is called DKG. This protocol can tolerate the attack where the adversary can force the public key to have a biased distribution. DKG tolerates up to t halting players for $n \geq 2t + 1$ and t eavesdropping players for $n \geq t + 1$ and t static malicious adversary for $n \geq 3t + 1$.

ECC has been also applied to smart card [9] applications and sub-second key generation performance on signature verification and key generation has been reported. In summary, the advantages of using ECC compared to competing approaches are given as follows: 1) Much more flexibility with many curves to choose from. 2) More efficient key generation, validation algorithms with a low processing overhead. 3) Smaller key size for a similar level of secrecy.

In this paper, we propose a new protocol called elliptic curve based distributed key generation (ECDKG), which is based on DF-VSS and uses ECC as the building block. It is built upon common adversary models and is immune to these adversaries. Compared to DKG, this new protocol enjoys all advantages of DKG besides short key length and moreover flexibility of protocol setup. Our initial implementation results show that it takes time in the sub-second range to generate keys even in a large network, and this allows ECDKG to be used in many applications. This protocol can be used for efficient threshold signature algorithms.

In Section II, our models are presented and our key generation protocol is shown in Section III. Implementation results are given in Section IV. We conclude this paper in Section V.

II. Our Models

We assume that there are two kinds of channels available, broadcast channel and private channel, and any two players can communicate via their respective private channel. Private channels are assumed to be at least as secure as the building block cryptosystems. Message broadcasting uses a flooding mechanism, i.e., scoped flooding with a time-to-live scope. Messages from ECDKG follow different semantics. For a broadcast message, it either reaches all recipients or none. Furthermore, if it reaches all recipients,

[1]This research of Dr. Chronopoulos was supported, in part, by a grant from the Center for Infrastructure Assurance and Security at The University of Texas at San Antonio and by NSF CCR-0312323.

the reception order by these players is random. We denote this as message broadcasting semantics (MBS). We also require a causal ordering (i.e. happened-before) on message-delivery semantics between a given pair of sender and recipient, i.e., message m_1 from sender p_1 reaches recipient p_2 before message m_2 from p_1 if p_1 sends m_1 before m_2. We denote this as message ordering semantics (MOS). In this communication model, we also assume that no message loss can occur during transmission. Once a message is sent by a player (faulty or not), the message will reach its intended recipient(s) in a uniformly bounded time interval. For a halting adversary, messages which would have been delivered if the protocol were followed, are not considered to be sent. Furthermore, due to the happened-before semantics, all successive messages from this halting adversary are blocked.

The adversaries can be categorized into two types: 1) static, 2) dynamic and adaptive. For a static adversary, decision on which player to break into is made before the run of the protocol. That is all players are taken as the same to the adversary. It can immediately read any message sent on a non-private channel, and when a player is corrupted, all its states and partial results are exposed to the adversary. The message processing time by the adversary is ignored in a static adversary model under the assumption that the adversary has more computing power than any of the honest players. There are four types of adversaries dealt with in this paper: 1) *Halting adversary*. In the protocol run, a player may not respond to a message either deliberately or due to stop-fail type of failure. 2) *Eavesdropper*. An adversary passively monitors the channel, and accesses all public messages. 3) *Static malicious adversary*. Before the protocol executes, this type of adversaries have already decided which player to corrupt during the execution of the protocol. This decision can not be changed by exploiting the runtime information obtained during protocol execution. 4) *Replay adversary*. A replay adversary buffers message and sends these out whenever necessary to impersonate a honest player. This type of attacks is applicable to a multi-stage protocol. One static attack example is presented in [3] in which two faulty players collude to make a bias on the public key. DF-VSS fails to prevent this attack, because it may cause that the generated public key does not follow a uniform distribution over the given field. We call this the *GJKR attack*. The design of our proposed protocol has taken into account these adversaries.

III. Our Proposed Key Generation Protocol

Our distributed key generation protocol is based on the improved version of DF-VSS [3] with enhancement on efficiency and protection against adversaries enumerated in Section II. A description of this protocol is given below. We use field $\mathrm{GF}(q)$ as the ground field where q is prime or some proper power of a prime. We assume that each player has a unique random identification number $p_i \in \mathrm{GF}(q)$ and players know these numbers of each other.

Notation: In this paper, $\mathrm{GF}^*(q)$ denotes the induced multiplicative group of $\mathrm{GF}(q)$. $\mathcal{G}$ denotes the main subgroup of order p which is derived from a point T, and used as the base group of ECDKG; $\oplus$ denotes the point add operator over $\mathcal{G}$ and $\sum^{\oplus}$ denotes the point summation under $\oplus$, and $\mathcal{S}_i = \{p_j | j \neq i, 1 \leq j \leq n\}$ is the set of peer players of p_i.

Let n be the total number of players who want to form a secure group, and they are identified by the distinct IDs $(p_1, p_2, \cdots, p_n)$, where $p_i \in \mathrm{GF}^*(q)$. We use "*player i*" or p_i interchangeably in this paper. Let $E/\mathrm{GF}(q)$ be an additive group based on a properly preselected elliptic curve E, and T be a point in $E/\mathrm{GF}(q)$. The cardinality of $E/\mathrm{GF}(q)$ is a prime number or has a large prime factor for the cryptographical purpose. We use p to denote this prime hereafter.

In this paper, we assume that point multiplication (a point multiplied by a scalar in $\mathrm{GF}^*(q)$) and point addition are performed in $\mathcal{G}$, all other arithmetic operations are performed in finite field $\mathrm{GF}(q)$ unless otherwise specified. To evaluate $Q(x)T$, we first evaluate $Q(x)$ using field arithmetic operations in $\mathrm{GF}(q)$, then we take modular p to the result of $Q(x)$, and $(Q(x) \bmod p)$ is the point multiplication scalar on point T to get the result point in $\mathcal{G}$. Note that the bit length of $Q(x)$ is normally longer than that of p. We also assume that there is another point T' in $\mathcal{G}$ whose discrete logarithm with respect to T is not known to any of these n players. ECDKG consists of four algorithms: the key distribution algorithm (KD), the key verification algorithm (KV), the key check algorithm (KC) and the key generation algorithm (KG). The protocol at p_i is given next.

Protocol 1: $\mathrm{ECDKG}(p_i, \tau, Q = \{p_1, p_2, \cdots, p_n\})$

Given: τ, timeout value; Q, a set of non-disqualified members. Execute:

```
set $Q_i = Q$
$\mathrm{KD}(p_i, t, T)$
$\mathrm{KV}(p_i, t, T)$
while(1) {
    if(timeout with $\tau$) {
        $\mathrm{KG}(p_i, Q_i)$; exit
        }
    $\mathrm{KC}(p_i, t, Q_i)$
} ■
```

Algorithm 1: **KD**(i, t, T)

1) Initialization: pick $(2t + 2)$ random numbers uniformly, $a_{ik} \in \mathrm{GF}(q)$ and $b_{ik} \in \mathrm{GF}(q)$ $(0 \leq k \leq t)$, as polynomial coefficients to generate two polynomials of degree t as follows:

$$f_i(z) = \sum_{k=0}^{t} a_{ik} z^k \qquad f'_i(z) = \sum_{k=0}^{t} b_{ik} z^k$$

- compute $s_{ij} = f_i(p_j)$ mod p, and $s'_{ij} = f'_i(p_j)$ mod p ($j \in \mathcal{S}_i$).
- compute $(t+1)$ public values: $P_{ik} = (a_{ik}T) \oplus (b_{ik}T')$ $(0 \le k \le t)$.

2) Dissemination of private information: sends a message containing s_{ij} and s'_{ij} to p_j using the private channel between p_i and p_j ($j \in \mathcal{S}_i$).
3) Dissemination of public information: broadcasts a message containing $\{P_{ik} | 0 \le k \le t\}$. ■

Algorithm 2: $\mathbf{KV}(i, t, T)$

Receive s_{ji} and s'_{ji} sent by p_j ($j \in \mathcal{S}_i$), then for $j \in \mathcal{S}_i$, do the following:

1) verify

$$(s_{ji}T) \oplus (s'_{ji}T') = \sum_{k=0}^{t}{}^{\oplus} \left(p_i{}^k P_{jk}\right) \tag{1}$$

2) broadcast a complaint against p_j, if (1) fails for p_j.
3) broadcast s_{ij} and s'_{ij} that satisfy (1), if p_i receives a complaint to him from p_j. ■

Algorithm 3: $\mathbf{KC}(i, t, Q_i)$

Update share s_i: p_j is removed from Q_i and update $s_i = \sum_{j \in Q_i} s_{ji}$,

if one of the following two conditions holds:

1) received $t+1$ or more distinct complaints against p_j.
2) received a re-broadcasted s_{ji} and s'_{ji}, but the received s_{ji} and s'_{ji} still falsifies (1). ■

Algorithm 4: $\mathbf{KG}(i, Q_i)$

Generate public key:

1) computes $A_{i0} = a_{i0}T$ and broadcasts A_{i0}.
2) receives A_{j0} ($j \in Q_i$) and compute public key as $y_i = \sum_{j \in Q_i}^{\oplus} A_{j0}$. ■

Remarks: 1) Since at p_j,

$$s_{ji} = \left(\sum_{k=0}^{t} p_i^k a_{jk}\right) \qquad s'_{ji} = \left(\sum_{k=0}^{t} p_i^k b_{jk}\right)$$

Equation (1) should hold at p_i for $j \in \mathcal{S}_i$. This explains the necessity of Step 2 of Algorithm 2 if (1) is violated. 2) When $t+1$ or more complaints received against one player, the contribution of secret from that player is a public knowledge by Lagrangian interpolation. Therefore, it is necessary to exclude p_j in Step 1 in Algorithm 3. 3) A timeout mechanism is used in ECDKG to countermeasure the GJKR attack and the halting adversary. ECDKG does not require a strong synchronization since the timeout value can absorb the clock drifts among players, i.e. adjusting $\tau' = \tau + d_{\max}$, where $d_{\max}$ is the maximum clock drift among these n players and τ' is the adjusted timeout value. In fact, in practice, DKG implicitly requires a similar timeout mechanism for public key extraction and against the halting adversary. 4) Although ECDKG is essentially a two-round protocol, the actual duration of the second round is much smaller than that of the first round. By noticing that the key generation is usually run once at the initialization stage of an application, it should not be considered as an issue. This timeout value should be set equal to half of the longest roundtrip time between any two players. This value is proven to be sufficient as shown in the proof of Proposition 5. 5) T' can be pre-computed in a distributed fashion based on the standard DKG with all the n players involved. A simple scheme is given as follows: i) all players run DKG to generate a uniform random number $r \in \mathrm{GF}(q)$ and no single player knows r and each has a shared piece of it. ii) each player broadcasts a point with its shared piece multiplied to point T, and set $T' = (r_1T) \oplus (r_2T) \oplus \cdots \oplus (r_nT)$. No single player in this simple scheme knows the discrete logarithm of T' with respect to T. This point is pre-computed. 6) The information dissemination order is private information first followed by the public information. 7) The secrecy of ECDKG partially depends on the intractability of ECDLP. If ECDLP can be solved efficiently, the problem to find the shared secret of ECDKG can be solved efficiently; however, the inverse does not hold in general. The following propositions hold for ECDKG.

Proposition 1: Uniqueness of Q: When the protocol ECDKG terminates, the set of non-disqualified players is the same across all uncorrupted players if the number of corrupted players is less than $t+1$ for $n \ge 3t+1$.

Proof: We prove that any two players p_i and p_j have the same set. The proof proceeds as follows: we first show that $Q_i \subset Q_j$ and then by symmetry, $Q_j \subset Q_i$, we conclude that $Q_i = Q_j$ for any two uncorrupted players p_i and p_j.

Let $p_u \subset Q_i$, since p_k passed the KV algorithm, i.e. there are at most t complaints against it and all his complaints (if any) are correctly resolved, we have,

$$(s_{ui}T) \oplus (s'_{ui}T') = \sum_{k=0}^{t}{}^{\oplus} \left(p_i{}^k P_{uk}\right)$$

Since p_u's public information has been sent to p_i, p_u's private information is already available to p_i when p_i receives the public information. By the MBS, the same public information should also be available to p_j by this time. By the MOS, the private information s_{uj} and s'_{uj} should be already available to p_j at this time. After the KD algorithm, all messages are broadcast messages. If p_i receives them, so does p_j. If p_u is uncorrupted, s_{uj} and s'_{uj} are correct at p_j, then,

$$(s_{uj}T) \oplus (s'_{uj}T\prime) = \sum_{k=0}^{t}{}^{\oplus} \left(p_j{}^k P_{uk}\right)$$

This means that p_u will pass the KV algorithm.

If there are at most t corrupted players, the number of complaints against p_j is at most t. Therefore, p_u will pass

Step 1 of the KC algorithm. Since p_u is always able to rebroadcast s_{uk} and s'_{uk} to any complaining player (corrupted or not) p_k, all uncorrupted players will include p_u in their respective non-disqualified set. Therefore, p_u will pass Step 2 of the KC algorithm. Since $n \geq 3t+1$, the protocol exits with nonempty non-disqualified set of size at least $t+1$. So, $Q_i \subset Q_j$. This completes the proof by noticing the symmetry ■

Now that all non-disqualified sets are the same, we denote this unique set by Q hereafter.

Proposition 2: The public keys generated by all players in Q are the same.

This follows directly from Proposition 1. Hereafter, we denote by y this common public key.

Proposition 3: Threshold secret sharing: if $t+1$ players of Q collaborate, the secret corresponding to y can be revealed. However, no secret can be revealed if less than $t+1$ players collaborate.

Proof: Without loss of generality, assume that there are n players in Q $(n \geq t+1)$, and we consider the first $t+1$ players $(p_1, p_2, \cdots, p_{t+1})$. We consider the following polynomial:

$$F(z) = \left(\sum_{k\in Q} a_{k0}\right) + \left(\sum_{k\in Q} a_{k1}\right) z + \cdots + \left(\sum_{k\in Q} a_{kt}\right) z^t$$

where the coefficients are unknown. There are available shares s_i $(1 \leq i \leq t+1)$.

Based on the construction of s_i in ECDKG, we have $s_i = F(p_i)$ $(1 \leq i \leq t{+}1)$. By Lagrangian interpolation, we can uniquely compute all the coefficients of $F(z)$, therefore $F(0)$. By noticing that $y = F(0)T$, the shared secret is revealed. When less than t players collaborate, in order to use the Lagrangian interpolation, one has to solve the ECDLP in order to recover at least one secret share from public information. ■

Proposition 4: ECDKG is immune to the GJKR attack.

Proof: Denote the time to execute the KG algorithm in ECDKG by t_0. When $t < t_0$, A_{i0} is hidden inside P_{i0} $(i \in Q)$. This is due to the fact that the discrete logarithm of T' is unknown, and a_{i0} and b_{i0} are unknown. Therefore GJKR attack can not bias the public key distribution before timeout.

When $t \geq t_0$, A_{i0} and Q are public knowledge. However, secret shares correspond to y are already decided. First, generation of public key simply collects those A_{i0} in Q. Second, any bias on y will render it useless. ■

Proposition 5: Protocol soundness: when there are less than $t+1$ corrupted players, ECDKG will terminate in a uniformly bounded time. In particular, this duration is less than $4\tau + \delta$, where τ equals half of the longest roundtrip time between any two players and $\delta > 0$.

Proof: At the beginning of Step 2 of the KV algorithm, one τ is required for all public and private information to be delivered. There is a 2τ duration needed for complaints to be sent and satisfied. Therefore, the timeout value can be set at $3\tau + \delta$ for some margin $\delta > 0$. In the end, one more τ is needed for the generation of the public key.

If there are more than t corrupted players, they can decide at will to allow or disallow any player to be included in Q. This can be done via controlling the complaints in Step 2 of the KV algorithm. ■

Proposition 6: In ECDKG, when the number of corrupted players is less than $t+1$ and $n \geq 3t+1$, corrupted players can only complain against each other.

Proof: Without loss of generality, assume that $p_1, p_2, \cdots, p_t$ are the corrupted players under control of some adversary and the rest are the honest players.

If p_i $(1 \leq i \leq t)$ complains against p_j for some $j > t$, there are two cases to consider: 1) p_j can correctly convince all uncorrupted players by revealing its private information as done in Step 3 in the KV algorithm; 2) there are at most t complaints against p_j, so p_j can pass Step 2 in the KV algorithm. Therefore, any complaint from the first t players can not affect any uncorrupted player. When $n \geq 3t+1$, ECDKG will succeed since the cardinality of Q is always greater than t during the course of the protocol execution. ■

Proposition 7: Secrecy of ECDKG: ECDKG enables $(n, t, t+1)$ threshold secret sharing.

Remarks on Proposition 7: In order to show the secrecy, an oracle uses a simulator and proves that a transcript can be produced with knowledge of only public available information, and this transcript is indistinguishable from that produced by the protocol. The actual proof is omitted here and it is similar to that of DKG in [3] with an exception on that the discrete logarithm is based on the additive group $\mathcal{G}$ instead of GF(q).

IV. Implementation Results

In general, ECC has advantages over other cryptosystems, but the choice of parameters including the curve used could significantly affect the overall performance. ABC curves (i.e. Koblitz curves) are used in our implementation. For the use of ABC curves, the order of the group always has a large prime factor. The use of ABC curves makes our protocol suitable for applications in networks with resource constrained devices. This type of curves has this form: $y^2 + xy = x^3 + ax^2 + 1$, where $a \in$ GF(2). The curve is represented by a quintuple $[1, a, 0, 0, 1]$ in PARI/GP. The domain parameters related to ECDKG are given as follows: 1) threshold value t and one field element for the coefficient a corresponding to a unique quintuple. 2) field representation type, for polynomial basis, the irreducible polynomial and the coefficients of it are required; for normal basis, which is most efficient for raising the unique identification number p_i to a power less than $(t+1)$, the base element θ of the basis is needed. 3) point representation, point compression type, coordinate system type and the selected point whose order must have a large prime factor p ($p > 2^{160}$) which is almost guaranteed in the case of ABC curves. 4) cofactor h which can be either 2 or 4 since m has to be a prime, and h multiplied by p gives the number of points in the group $E/\mathrm{GF}(2^m)$.

We first itemize overhead of ECDKG including computation and communication. Table I shows the communication cost where $U(t) = t\log_2(t)$. Each cell in Tab. I consists of two expressions in (x, y) format where x is the arithmetic cost on $\mathcal{G}$ – the elliptic curve main subgroup, and y is the arithmetic cost on GF(q). Table II shows the number of messages involved in each algorithm, and the transmission column consists of two expressions in (x, y) format where x is the transmission cost in private channel and y is that in broadcast channel. Note that in Tab. I, the evaluation of

TABLE I
WORST-CASE COMPUTATION COST OF ECDKG

	Multiplication	Addition
KD	$(2t+2, 2nU(t+1)$	$(t+1, 2nt)$
KV	$(n(t+3), nU(t+1)$	$(n-1)(t+1), 0)$
KC	$(2t, 0)$	$(t, n-1)$
KG	$(1, 0)$	$(n-1, 0)$

an exponent in GF(q) uses repeated squaring and that of point multiplication in $\mathcal{G}$ uses τ-adic Non-Adjacent Form which makes point doubling almost 'free' and converts point multiplication operation to a few point addition operations.

TABLE II
WORST-CASE COMMUNICATION COST OF ECDKG

	Reception	Transmission	Adversary
KD	0	$(n-1, 1)$	all $(n > t)$
KV	$n-1$	$(0, t+1)$	all $(n > t)$
KC	$t(n-2t-1)+t$	$(0, 0)$	static $(n > 3t)$
KG	$n-1$	$(0, 1)$	all $(n > t)$

Note that in Tab. II for the reception overhead in the KC algorithm, although the KC algorithm is included in a loop, the number of received messages must be upper bounded by the total number of messages actually transmitted. The worst-case memory requirements are given as follows: For the KD algorithm, this is $2(t+1)\log_2(p)+C$, where C is the worst-case memory requirement for performing individual point multiplication. For the KV algorithm, the worst-case memory requirement is $(2n+t+1)\log_2(p)+C$. For the KC algorithm, the static malicious adversary yields the worst-case memory requirement which is $(n-1)\log_2(p)+C$. The memory requirement for the KG algorithm is a constant.

We used Intel PXA 255 200 MHz processor as the platform running PARI/GP to simulate the protocol. Table III shows the key generation times for 5 players under varying key sizes and message roundtrip times where "RT-x" is for the roundtrip time at x ms and "K-x" is for the NIST ABC curves [1] in the field GF(2^x). Table IV shows the key generation times for 10 players under varying key sizes and message roundtrip times. In both cases, the threshold is set at 3.

TABLE III
KEY GENERATION TIMING (N = 5, T = 3)

	$\log_2(p)$	Curve	RT-10	RT-100	RT-500
K-163	163	[1, 1, 0, 0, 1]	90 ms	279 ms	1.079 s
K-233	232	[1, 0, 0, 0, 1]	204 ms	384 ms	1.184 s
K-283	281	[1, 0, 0, 0, 1]	292 ms	472 ms	1.272 s

TABLE IV
KEY GENERATION TIMING (N = 10, T = 3)

	$\log_2(p)$	Curve	RT-10	RT-100	RT-500
K-163	163	[1, 1, 0, 0, 1]	163 ms	343 ms	1.143 s
K-233	232	[1, 0, 0, 0, 1]	356 ms	535 ms	1.336 s
K-283	281	[1, 0, 0, 0, 1]	516 ms	696 ms	1.496 s

In above tables, the costs of field additions in GF(q) and the point additions in $\mathcal{G}$ are negligible, and the computation cost is dominated by the field multiplication in GF(q). The point multiplication is efficiently performed using the τ-adic non-adjacent form. This makes the computation of ECDKG much more efficient compared to distributed key generation based on DLP or IFP which takes time in the range of seconds to generate a key on Intel PXA 255 200 MHz.

V. CONCLUSION

In this paper, we proposed the elliptic curve distributed key generation protocol. It is well suited for secure applications with resource constraints and provides high level of secrecy with efficiency and small key size.

REFERENCES

[1] ANSI X9.62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Key Agreement and Key Transport Protocols.

[2] P. Feldman, "A Practical Scheme for Non-Interactive Verifiable Secret Sharing", *Proc. 28th IEEE FOCS*, 1987.

[3] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems", *Proceeding Eurocrpt, 1999.*

[4] M.-D. Huang, K. L. Kueh, and K. Tan, "Lifting Elliptic Curves and Solving the Elliptic Curve Discrete Logarithm Problem", *Lecture Notes in Computer Science 1838*, pp. 377-384.

[5] N. Koblitz, "The State of Elliptic Curve Cryptography," *Designs, Codes and Cryptography*, 19, 173-193(2000), Kluwer Academic Publishers, Boston.

[6] T. Pedersen, "A Threshold Cryptosystem Without a trusted Party", *Advances in Cryptology – Eurocrypt '91*. LNCS 547, Springer-Verlag.

[7] T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing", *Advances in Cryptology – Crypto'91*, LNCS 576, Springer-Verlag.

[8] A. Shamir, "How to Share a Secret", *Communications of the ACM*, Vol. 22, No. 11, Nov. 1979.

[9] C. P. Schnorr, "Efficient Signature Generation by Smart Cards", *Journal of Cryptology*, Vol. 4, pp. 161-174, 1991.