

ECDKG: A Distributed Key Generation Protocol Based on Elliptic Curve Discrete Logarithm *

Caimu Tang

Abstract

In public key based cryptosystems, key distribution has been a crucial issue of protocol design. Many attack methods are orchestrated by making use of some weakness at the stage of key distribution. In this report, we also propose a distributed key generation protocol based on elliptic curve discrete logarithm, called ECDKG. Existing distributed key generation protocols use either cryptosystems based on discrete logarithm over a finite field or Integer factorization as a building block. The secrecy has been proven and the secrecy is as strong as the building block systems. However, there are subexponential algorithms for solving this discrete logarithm problem over a field field and the integer factorization problem. Elliptic curve cryptosystem provides a promising alternative for building a distributed key generation scheme and no known subexponential algorithms solve a general version of the discrete logarithm over an additive group derived from a point on an elliptic curve. Main advantages using this building block include the space and time efficiency and flexibility. We also propose the adaptive version and proactive version of ECDKG. An example on how to use ECDKG on ElGamal based message encryption is also given. We will also address some practical implementation issues.

Key Words and Phases: *Elliptic Curve Cryptosystems (ECC), Public-key Based Cryptography, Threshold Cryptography. Distributed Key Generation(DKG), ECDKG, Elliptic Curve Discrete Logarithm Problem (ECDLP), Discrete Logarithm Problem (DLP),*

1. Introduction

As collaboration via Internet is common nowadays, security has always been a major challenge. One type of applications could require many parties to jointly sign a document or share certain secure information and no individuals are allowed to have a total control on this information. It is not always possible to have a trusted party presented as this is required by many existing protocols of the wired counterpart [29] [28] [1]. Distributed collaborative security mechanism has been recognized as an alternative. Distributing trust

* The author is greatly gratitude to Prof. Ming-Deh Huang for his encouragement on this research and many constructive discussions during and after the author took his class on Elliptic curve cryptosystems. This is a revision to two ealier reports dated back in Dec. 2001 and May 2002.

and share secrets among a group of parties have attracted research interest for a decade also [37] [7]. In particular, in the Internet, in order to provide a trusted service model over a group of non-trustable entities, a model on how to distribute trust has been proposed [2].

1.1. Motivation, Past Work and Our Contribution

Motivation: Since the seminal work [37] and later [7] on the proposed (k, n) (where n is the number of shares of a secret and k is the threshold) threshold scheme based on a simple polynomial interpolation over a finite field, $\text{GF}(q)$, where q is a prime or power of some prime p by which it is an extension field of field Z/pZ (it is also called a Galois Field (GF)). And [32] extends Feldman's non-interactive approach and presents a scheme in which each party can verify the information about the secret without communicating to other parties, and any k of these parties can later find the secret ($1 \leq k \leq n$), where fewer than k parties get no information about the secret. It randomly selects two polynomials and only $E(F_i, G_i)$ s ($E(s, t) = g^s h^t$ for two preselected generators g and h in $\text{GF}(q)$) are put in public while in [7] g^{F_i} is put in public. It has been seeing some progresses on applying this threshold scheme for various security purposes, e.g. distributed digital signature [18], distributed key generation [11].

Encryption using keys generated by distributed key generation (DKG) can be used in many applications, e.g. multi-party digital signature. DKG can also be used solving some consensus problems, e.g. Byzantine Agreement [23] [21]. Existing DKG protocols are based on either discrete logarithm problem (DLP) over a finite field or integer factorization problem (IFP). In order to maintain certain level of security, key lengths in both cases have to be long enough to detain cryptanalysis due to recent developed subexponential algorithms on integer factorization and DLP. Elliptic curve cryptosystem (ECC), on the other hand, is safe against some common algorithmic techniques, e.g. index-calculus. There is no specific subexponential algorithm for ECDLP if some discretion is exercised on selecting the curve and associated parameters. It is reasonable to believe that shorter key length can provide similar level of security compared to that based on DLP or IFP.

Past Work: For a given secret s and a threshold k , s can be partitioned into n shares as (s_1, s_2, \dots, s_n) . Only u out of n shares is sufficient to recover s and less than t shares cannot. This scheme is called (n, t, u) scheme. In [37], a $(n, t, t + 1)$ scheme is presented. The computation involved are the polynomial evaluation and interpolation over a $\text{GF}(q)$ where q is defined as above ($\text{GF}(q)$ denotes the field, $\text{GF}^+(q)$ denotes the induced additive group from $\text{GF}(q)$ and $\text{GF}^*(q)$ denotes the induced multiplicative group from $\text{GF}(q)$). Note if q itself is a prime, all operations are reduced to modular arithmetic with prime q . Given any $t + 1$ interpolation points, there corresponds a unique polynomial of degree t . The basic formula of the Lagrangian interpolation is as follows:

$$Q(i) = \sum_{l=1}^{t+1} \left(\prod_{h \neq l} (x - a_h)(a_l - a_h)^{-1} \right) N_{a_l}$$

for $(t + 1)$ points $(a_1, N_1), (a_2, N_2), \dots, (a_{t+1}, N_{t+1})$ over $\text{GF}(q)$. Also as pointed out in [37], there exists $O(t \log^2 t)$ polynomial interpolation algorithm over finite field.

In [7], 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 communication uses broadcast message and also the players can verify the authenticity of the dealer, therefore the so-called non-interactive VSS with communication and computation complexity as $O(nk)$ and $O((n \log n + k)(nk \log k))$, respectively. It uses homomorphic encryption as the building block for the scheme. Shamir’s $(n, t, t + 1)$ threshold cryptography is one of this kind homomorphic encryption schemes and it can be used as one of the building block. To combat a dynamic adversary, the scheme employs the permutation instead of direct use of the indices. The adversary simulation is based on zero-knowledge [12] [45]. The first distributed VSS version is presented in [31] which is based on Feldman VSS (where each player acts as a dealer). It specifies n parallel runs of all the players, each player selects a random secret $z_i \in \text{GF}(q)$ and shares it among other players. The player collaboratively constructs a non-disqualified set Q in which the secrets are shared. The random secret x is set to the sum of the properly received shares from others in Q . This protocol called Distributed-Feldman verifiable secret sharing (DF-VSS) is presented in the appendix for convenience. In [11], an improved version of DF-VSS is presented and it is called DKG which can countermeasure the GJKR attack. 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 $n \geq 3t + 1$.

Since the first proposals of using elliptic curve for cryptographic purpose, [27], [17], attempts of using existing techniques or new methods to solve ECDLP in subexponential time [14] [15] [24] are still ongoing. Regardless this endeavor, no promising results have been found. Only generic exponential algorithms, i.e. square-root [42] type algorithms, are available for a broad class of ECDLP. Only in some restricted cases (see Section 1.2), such subexponential algorithms exist [25], [39]. Cryptosystems based on ECDLP can use small key size to provide comparable levels of security as long as certain discretion has been taken into account to select the candidate curve and domain parameters. This prominent feature makes them attractive for many applications, e.g. electronic commerce where computing and communication resources are concerned. Small key size means saving on storage, processing time and bandwidth. Protocols based on ECDLG, e.g. ECDSA [16], have been adopted by some standards, and incorporated into many drafts [19]. Efficiency issue on software implementations of ECC have being studied from the aspect on fast arithmetic operation algorithms over $\text{GF}(2^n)$ [43] and over $\text{GF}(p^n)$ [22]. ECC has been also applied to smart card applications[44] [38] and sub-second performance on signature verification and key generation has been reported. In summary, the advantages of using ECC compared to other schemes are given as follows:

- 1) Much more flexibility using curve selection than whatother public-key cryptosystems can provide as compared to cryptosystems based on DLP or IFP.
- 2) Efficient key generation, validation algorithms which allow a low processing overhead.
- 3) Smaller key size for similar security than that used by other public-key cryptosystems.

Our Contribution: 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. The group public key is fixed once it is generated. New member can join this group via the joining primitive and member can leave the group via simply stopping response to any request by the protocol. Compared to DKG, this new protocol enjoys all advantages of DKG besides short key length and flexibility of protocol setup. The message decryption involves a threshold numbers of players and an encrypted message can be decrypted without explicitly construction of the group private key at any player.

This report first gives a brief introduction on the mathematical underpinning of elliptic curve cryptosystems (ECC). In Section 2, the system model is presented and our key generation protocol is shown in Section 3. Implementation related issues are discussed in Section 4. Section 5 concludes this report with remarks on some implementation issues of our proposed approach.

1.2. Elliptic Curve Discrete Logarithm Problem and Elliptic Curve Cryptosystems

Let F be a finite field and E/F be the additive abelian group derived by an elliptic curve E defined in F . For practical purposes, two types of ground fields are often considered: $\text{GF}(p)$ for p a prime and $\text{GF}(2^p)$ for p a prime. For $\text{GF}(p)$, a smooth elliptic curve can be represented as: $y^2 = x^3 + ax + b$ in affine form or $Y^2Z = X^3 + aXZ^2 + bZ^3$ in projective form, where $a, b \in \text{GF}(p)$ and $4a^3 + 27b^2 \neq 0$. For $\text{GF}(2^p)$, a nonsupersingular elliptic curve can be represented as: $y^2 + xy = x^3 + ax^2 + b$ in affine form or $Y^2Z + XYZ = X^3 + aX^2Z + bZ^3$ in projective form, where $a, b \in \text{GF}(2^p)$, and $b \neq 0$. Group E/F is attractive for cryptography partially due to the intractability of the so-called elliptic curve discrete logarithm problem (ECDLP). Furthermore, there are only few cases of ECDLP which have polynomial or subexponential algorithms, there are huge number of candidate curves for cryptographic use.

Definition 1: ECDLP: given an elliptic curve E over a finite field F , a point $T \in E/F$ whose order contains a large prime factor p , and a point $Q = xT$ for some $x \in [1, p-1]$, to determine the unknown x .

There are the following known attacks specific to ECDLP.

- 1) Elliptic curves with T with a smooth order. An attack presented in [33] reduces the problem to find the secret key x to the problem of finding x modulo each of the prime factor of n , then use Chinese Remainder Theorem to solve for x . This algorithm can solve this type of ECDLP in $O(\text{poly}(\log(n)))$.
- 2) Supersingular elliptic curves. An elliptic curve E over F_q of q elements is called supersingular if the trace of the Frobenius map (i.e., the map $(x, y) \mapsto (x^q, y^q)$ and it takes *point at infinity* to itself.) is divisible by the characteristic of F_q . For prime field $\text{GF}(p)$, we need to avoid curve whose cardinality divides $p^k - 1$ for small k ($k \leq 20$), since Weil pairing [24] (MOV attack) or Tate pairing [34, 8] can reduce ECDLP to DLP in the multiplicative group of some extension field of $\text{GF}(p)$ where the DLP can be solved in subexponential time when k is small. For cryptographic purpose, we can check up to 20 and it is sufficient since the discrete logarithm over $\text{GF}(p^C)$ for $C > 20$ is computationally infeasible.

- 3) Prime-field anomalous curve. An elliptic curve is called anomalous if the trace of the Frobenius map is equal to 1. In this case, $|E/\text{GF}(p)| = p$, and ECDLP can be reduced to DLP in an additive group [39] [40] [35].
- 4) Curves over field $\text{GF}(2^m)$, m is composite. Weil descent [9] GHS attack might be used to solve the ECDLP over binary field. Weil descent reduces ECDLP in $\text{GF}(2^m)$ to a DLP in an abelian variety over a proper subfield of $\text{GF}(2^m)$, then one can use algorithms for the hyperelliptic curve DLP that are significantly faster than the best available ones for the ECDLP [20]. However, it has been shown in [26] that it is infeasible of GHS attack for $E/\text{GF}(2^n)$ when n is a prime and $n \in [160, 600]$.

With these known attacks, two types of elliptic curves are favorable for cryptography, they are $\text{GF}(p)$ and $\text{GF}(2^p)$ for prime p . Denote q as the order of the ground field $\text{GF}(p)$ or $\text{GF}(2^p)$, in order to avoid MOV attack, q should not divide $q^n - 1$ for small n 's. Also the cardinality of $E/\text{GF}(q)$ should not be q . There is polynomial time algorithm to count the points in $E/\text{GF}(q)$ [36]. These can be checked before select the curve.

From implementation efficiency point of view, one type of curve so called anomalous binary curve (ABC) of the form $y^2 + xy = x^3 + ax^2 + 1$, where $a \in \text{GF}(2)$. There are some efficient implementations of ECC systems based on ABC curves [18] [41]. The ground field is $\text{GF}(2^n)$, when n is prime, there exists a larger subgroup for cryptographic use, the cofactor is either 2 or 4 (very small) for $a = 1$ or 0 respectively. For elliptic curve, the inverse is cheap, Non-Adjacent Form (NAF) has been particularly interesting to scalar multiplication of elliptic curve points. For any number x , it can be represented as $\sum_{i=0}^{\infty} c_i 2^i$. The NAF of x is such a representation with $c_i c_{i+1} = 0$ for all $i \geq 0$. For group with complex multiplication property, an analogy of integer NAF has been developed. For some expansion, the non-zero terms are few and the scalar multiplication can be efficient performed. For ABC curve with the complex multiplication property, a new integer expansion, so called τ -adic NAF, can be used. It can significantly improve the point arithmetic. With it, the computation of kT requires small number of point addition and no need of point doublings. Since for ABC curve, there exists a reduction of an elliptic curve with complex multiplication by a CM (complex multiplication) ring $\mathcal{O}[\sqrt{-7}]$. This leads a very simple point counting method and can also significantly reduce the point scalar multiplication [41] using the τ -adic NAF with $\tau = \frac{-(-1)^a + \sqrt{-7}}{2}$ and it is the root of the characteristic equation of Frobenius automorphism over $\text{GF}(2^n)$ as $(x, y) \mapsto (x^2, y^2)$ and maps *the point at infinity* to itself. The computation of the map is almost “free” with only two cyclic shift operations if normal basis is used.

For curves other than ABC curves, a general scalar multiplication method, i.e. addition-subtraction method, can be used. The method uses a chain of numbers to generate the final point using doublings and addition. Shorter chain length gives better performance, however, the general problem to find the shortest addition chain is NP-complete [6]. This type of algorithms are very practical for elliptic curve since the inverse is almost “free”, i.e. $-(x, y) = (x, -y)$ in prime field $\text{GF}(p)$, or $-(x, y) = (x, x + y)$ (because $-(x, x + y) = (x, x + y + x) = (x, y)$) in $\text{GF}(2^m)$. There are two group representations can be used, i.e. with polynomial basis and with normal basis. Implementations have to choose a proper basis for it to be efficient. An element in $\text{GF}(2^n)$ can be represented as a polynomial:

$\sum_{i=0}^{n-1} c_i \alpha^i$, where $c_i \in \text{GF}(2)$. It can also be represented as a vector of dimension n using following basis:

$$\delta, \delta^2, \dots, \delta^{2^n-1}$$

where $\delta \in \text{GF}(2^n)$ and the elements in the vector are in $\text{GF}(2)$. For polynomial representation, irreducible trinomial or irreducible pentanomial is selected as the reduction polynomial in practice. Normal basis allows simple point doubling but complex multiplication in general. One type of normal bases, so called Gaussian normal bases (GNB), allows simple group arithmetic on both point doubling and multiplication. There are also two kinds of point coordinate representations, one is affine coordinate and the other is the projective coordinate. Some further improvement on group arithmetic can be achieved via selecting a proper coordinate system [5]. Using projective coordinate system, inverse can be avoided by multiplication. Especially, Jacobian projective coordinates [4], the projective point

$$(x : y : z) \mapsto \left(\frac{x}{z^2}, \frac{y}{z^3} \right)$$

and its variants gives superior performance for field arithmetic [13].

2. System Model

In this section, we present the communication model and adversary model which the protocol is based.

2.1. Communication Model

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. This private channel is assumed to be at least as secure as the building block cryptosystems. Broadcast has to use some form of flooding mechanism (e.g., scoped flooding with time-to-live constraint). Messages from ECDKG follow different semantics. For a broadcast message, it either reaches all recipients or none. Furthermore, if it reaches all recipients, the order for the recipients reach it is random, i.e., the arrival ordering of a message to the players from a given sender is arbitrary. We denote this semantics as (BS). 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 semantics as (OS). In this communication model, we also assume that no message loss can occur. Once a message is sent by a player (faulty or not), the message will reach its intended recipient(s) in finite time. 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.

2.2. Adversary Model

The adversary can be categorized into two kinds: 1) static, 2) dynamic and adaptive. For a static adversary, decision on which one 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, here we assume there is no information erasure in a player. Also 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 all the good players. There are six types of attackers:

- 1) Halting adversary. In the protocol run, a player may not response to a message either deliberately or due to processor stop-fail failure.
- 2) Eavesdropper. A adversary passively monitors the channel, and all public messages are also available to it.
- 3) Static malicious adversary. Before the protocol executes, this type of adversaries has already decided which player to corrupt during the execution of the protocol. In other words, he can not change to attack another player by exploiting the runtime information obtained during protocol execution.
- 4) Replay adversary. Replay adversary buffers message and sends these out whenever necessary to impersonate a good player. This type of attacks is commonly applicable to a multi-stage protocol.
- 5) Adaptive adversary. Adaptive adversary can select which player to attack based on its dynamic gathered information at runtime. In each round, he can perform necessary computation and generate messages and send them out before any honest players can. Compared to static adversaries, this type of adversaries is more difficult to deal with. Information erasure after each step during protocol execution turns out to be effective [3].

Proactive approach [18] with periodic share refreshing could be an effective counter-measure method to prevent on adversaries (static, adaptive or both) which can cumulate runtime information and eventually corrupt players.

One static attack example is presented in [11] in which two faulty players collude to make a bias on the public key. DF-VSS fails, i.e. the generated public key does not follow a uniform distribution over the given field, on this attack.

The design of our proposed protocol has taken into account these adversaries. It consists of ECDKG which is immune to the above static attack, and the adaptive version of ECDKG. Furthermore, a proactive protocol is proposed which can be used with them to prevent more sophisticated adversaries.

3. ECDKG

Our distributed key generation protocol is based on the improved version of DF-VSS [11] with enhancement to protect against adversaries enumerated in Section 2.2 . A description of this protocol is given below. We use field $\text{GF}(q)$ as the ground field where q is a prime or some proper power of a prime. We assume each player has a unique random identification number p_i in $\text{GF}(q)$ and players know these numbers of each other. (There exist algorithms

to generate these random numbers in a distributed setting [30] and the session initiator can be used to facilitate this random number generation.)

Let n be the total number of players who want to form a secure group, and they are identified as (p_1, p_2, \dots, p_n) , where $p_i \in \text{GF}^*(q)$ and these identification numbers are distinct, we use *player* i and p_i interchangeably herein. Let $E/\text{GF}(q)$ be an additive group based on a properly preselected elliptic curve E as explained in Section 1.2, and T be a point in $E/\text{GF}(q)$. The cardinality of $E/\text{GF}(q)$ is a prime or has a large prime factor, we denote this prime and the prime factor as p .

Notation: \mathcal{G} denotes the additive group derived from point T ; \oplus denotes the add operator over \mathcal{G} and \sum^\oplus denotes the point summation under \oplus , and $\mathcal{S}_i = \{p_j | j \neq i, 1 \leq j \leq n\}$.

In this paper, we assume that scalar and point multiplication is performed in \mathcal{G} , all other arithmetic operations are performed in finite field $\text{GF}(q)$ unless otherwise specified. To evaluate $Q(x)T$, we first compute $Q(x)$ using field arithmetic operations, then we take modular p to the result of $Q(x)$, and $Q(x) \bmod p$ is the scalar multiplier on point T . We also assume that there is another point T' on E of \mathcal{G} whose discrete logarithm with respect to T is unknown to none of these n players. ECDKG consists of four phases, namely, key distribution phase (KD), key verification phase (KV), key check phase (KC) and key generation phase (KG). The protocol at a player is given as follows:

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

Given:

τ : timeout value.

Q : a set of non-disqualified members.

Execute:

set $Q_i = Q$

KD(p_i, t, T)

KV(p_i, t, T)

while(1) {

 if(timeout with τ) {

 KG(p_i, Q_i)

 exit

 }

 KC(p_i, t, Q_i)

} \diamond

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

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

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

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

Algorithm 2: 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. verifies

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

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

Algorithm 3: 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). \diamond

Algorithm 4: 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 $y_i = \sum_{j \in Q_i}^{\oplus} A_{j0}$. \diamond

Notes:

1. Since at p_j ,

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

equation (1) should hold at p_i for $j \in \mathcal{S}_i$. This explains the necessity of Step 2 in 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. Timeout mechanism is introduced to ECDKG to countermeasure the GJKR attack [11] 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.

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 notice that the key generation is usually run once at the initialization stage of an application, it should not be considered as an issue in practice. This timeout value should be set 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 generated in a distributed fashion with all the n players involved. Let these players form an ordered list, a simple scheme is given as follows: 1) each player generates a uniform random number $r \in \text{GF}(q)$ (i.e. no other player knows r); 2) set $T' = \sum_{i=1}^n \oplus (r_i T)$. No single player knows the discrete logarithm of T' with respect to T .
6. The information dissemination order is private information first followed by the public information. A static malicious adversary is able to generate the public information before the public information is actually sent. Any tampering on the public information will be found when the actual public information is sent.
7. The security of ECDKG partially depends on the intractability of ECDLP. If ECDLP can be solved efficiently, the problem to find of the shared secret of ECDKG can be solved efficiently; however, the inverse does not hold in general. When only static adversary model is assumed, Proposition 6 states that the inverse does hold.

We assume corrupted players have all the information a player can have, including all public information, accessing of private channel and private information these players (otherwise honest players) have.

Proposition 1: Uniqueness of Q : When 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 $n \geq 3t + 1$.

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

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

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

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 message broadcasting semantics (BS), the same public information should also be available to p_j by this time. By message ordering semantics (OS), the private information s_{uj} and s'_{uj} should be already available to p_j before this time. After KD phase, all messages are broadcasted. If p_i receives, so does

p_j . If p_u is uncorrupted, s_{uj} and s'_{uj} are genuine at p_j , then,

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

This means that p_u will pass KV phase.

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 KC phase. Since p_u is always able to re-broadcast 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 phase. Since $n \geq 3t + 1$, the protocol exits with nonempty non-disqualified set of size at least $t + 1$. So, we have $Q_i \in Q_j$. ■

Note that since all non-disqualified set are the same, we denote this unique set by Q herein.

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

This is the direct result following Proposition 1. Herein, denote y as 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 of lose of generality, assume that there are n players in Q ($n \leq t + 1$), and we consider the first $t+1$ players (p_1, p_2, \dots, p_{t+1}). Consider the following polynomial:

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

where the coefficients are unknown. There are available 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: Immunity to GJKR attack: ECDKG is immune to the GJKR attack.

Proof: Denote the time to execute KG phase 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 generate in finite time. In particular, this duration is less than 4τ , where τ equals half of the longest roundtrip time between any two players.

Proof: At the beginning of Step 2 of KV phase, one τ is required for all public and private information to be delivered. There is 2τ duration needed for complaints to be sent and satisfied. 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 KV phase. ■

Lemma 1: In ECDKG, when the number of corrupted players is less than $t+1$, corrupted players can only complain against each other.

Proof: Without loss of generality, assume that p_1, p_2, \dots, p_t are the corrupted players and the rest are the uncorrupted players.

If p_i ($1 \leq i \leq t$) complains 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 KV phase; 2) there are at most t complaints against p_j , so p_j can pass Step 2 in KV phase. Therefore, any complaint from the first t players can not affect any uncorrupted player. Corrupted players can only complain against each other. ■

Proposition 6: Secrecy of ECDKG: ECDKG enables $(n, t, t+1)$ threshold secret sharing.

In order to show the secrecy, an oracle is created using 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 simulation. The actual proof is omitted here and it is similar to that of DKG in [11] with an exception on that the discrete logarithm is based on the additive field $E/\text{GF}(q)$ instead of $\text{GF}(q)$ used by the proof of DKG.

3.1. Adaptive ECDKG

In the heart of the proposed adaptive ECDKG, there lies the zero-knowledge proof on a player for his contribution to the shared secret through the ECDLP. Before proceed, we show the zero-knowledge proof technique based on ECDLP.

Let P be the prover and V be the verifier, and P tries to prove to V that P possesses a secret $1 \leq \omega \leq p-1$ with a public key $\hat{y} = \omega T$. The zero-knowledge proof of a secret ω proceeds as follows:

Algorithm 5: ECZKP(d, ω): Elliptic curve zero-knowledge proof

Given: a common number d agreed by P and V where $1 \leq d \leq p-1$

Executes:

(P1) P selects a random number $1 \leq r \leq p-1$ and sends $\tilde{T} = rT$ to V .

(P2) P computes $\Gamma = r + d\omega$ and sends Γ to V .

(V1) V verifies

$$\Gamma T = \tilde{T} + d\hat{y} \tag{2}$$

if the equation holds, V is confirmed that P possesses ω .

There are three properties we need to show that Algorithm 5 must have, namely, completeness, soundness and zero-knowledgeness. To show the completeness property, knowing the fact that (2) holds, V is convinced that only P which knows ω can generate the pair (\tilde{T}, Γ)

since no other pair of (r', ω') can satisfy (2). To show the property of soundness, any cheating P' which knows d, \hat{T} and \hat{y} is not able to generate a corresponding Γ' (without knowing ω) which can satisfy (2). For the zero-knowledgeness, an indistinguishable transcript can be generated without knowledge of ω : Item 5 only involves a random number and its associated elliptic curve point and Item 5 does not leak any information about ω since Γ is simply another random number. The transcript consists of only random numbers and public information in the forms of multiples of T .

Algorithm 5 provides zero-knowledge proof for two parties. This technique can be extended for a multiple-party zero knowledge proof by first setting up a common agreed upon d and individually verifying all other players.

To countermeasure the adaptive adversary, an ECZKP procedure is added at the beginning of the public key extract at each player in the non-disqualified set. This procedure is to prove to p_i that player p_j ($j \neq i$) does own s_{j0} . Any player which can not prove this fact is excluded from the non-disqualified set Q . The reason this modification works against an adaptive adversary is that a player corrupted before the timeout of ECDKG is excluded during the public key generation stage and at most one player can be corrupted at a time due to that player p_i erases s_{ji} at the end of the KV phase.

The pseudocode for adaptive ECDKG (A-ECDKG) is shown as follows:

Protocol 2: A-ECDKG($p_i, \tau, Q = \{p_1, p_2, \dots, p_n\}$)

Given:

τ : timeout value.

Q : a set of non-disqualified members.

Execute:

set $Q_i = Q$

KD(p_i, t, T)

KV(p_i, t, T)

Erase s_{ji} for $j \in \mathcal{S}_i$

while(1) {

 if(timeout with τ) {

 ECZKP(d, s_{j0})

 KG(p_i, Q_i)

 exit

 }

 KC(p_i, t, Q_i)

} \diamond

3.2. Proactive ECDKG

Share refreshing is an effective countermeasure towards an adaptive adversary who may cumulate partial results and eventually is possible to obtain more than $t + 1$ shares during the lifetime of these secret shares and the public key. Assume the duration of the protocol execution is splitted into slots of length σ . If an adaptive adversary can not take over t players in any slot, i.e. the adversary is computationally bounded, it can not obtain the group private key. The adversary can not incrementally attack the system since share refreshing renders partial results obtained before this refreshing period useless. One practi-

cal requirement of share refreshing is that the public key should not be changed. This also means that the shared secret remains intact after the share refreshing.

Algorithm 6: SRA: Share Refreshing Algorithm at p_i

Denote s'_i as the new secret share

Execute:

set $\hat{Q}_i = Q$

$\text{KRD}(p_i, t, T)$

$\text{KRV}(p_i, t, T)$

while(1) {

 if(timeout with τ) {

 if $Q_i \neq Q$

$\text{ECDKG}(p_i, \tau, Q)$

 exit

 }

$\text{KRC}(p_i, t, Q_i)$

} \diamond

Algorithm 7: $\text{KRD}(i, t, T)$

1. Initialization: pick t random numbers uniformly, $\hat{a}_{ik} \in \text{GF}(q)$ ($1 \leq k \leq t$), as polynomial coefficients to generate a polynomial of degree t as follows:

$$\hat{f}_i(z) = a_{i0} + \sum_{k=1}^t (a_{ik} + \hat{a}_{ik})z^k$$

- compute $\hat{s}_{ij} = \hat{f}_i(p_j) \bmod p$ ($j \in Q$).
 - compute t public values: $\hat{A}_{ik} = ((a_{ik} + \hat{a}_{ik})T)$ ($1 \leq k \leq t$).
2. Dissemination of private information: sends a message containing \hat{s}_{ij} to p_j using the private channel between p_i and p_j ($j \in Q$).
 3. Dissemination of public information: broadcasts a message containing $\{A_{i0}, \hat{A}_{ik} | 1 \leq k \leq t\}$. \diamond

Algorithm 8: $\text{KRV}(i, t, T)$

Receive \hat{s}_{ji} sent by p_j ($j \in Q$), then for $j \in Q$, do the following:

1. verifies

$$\hat{s}_{ji}T = A_{j0} + \sum_{k=1}^t \left(p_i^k \hat{A}_{jk} \right) \quad (3)$$

2. broadcasts a complaint against p_j , if (3) fails for p_j .
3. broadcasts \hat{s}_{ij} that satisfy (3), if p_i receives a complaint to him from p_j . \diamond

Algorithm 9: $\text{KRC}(i, t, \hat{Q}_i)$

Update share \hat{s}_i : p_j is removed from \hat{Q} and update $\hat{s}_i = \sum_{j \in \hat{Q}_i} \hat{s}_{ji}$, if one of the following two conditions holds:

1. received $t + 1$ or more distinct complaints against p_j .
2. received a re-broadcasted \hat{s}_{ji} , but the received \hat{s}_{ji} still falsifies (3). \diamond

Proposition 7: All \hat{Q}_i for $i \in Q$ are the same.

The proof is similar to that of Proposition 1 and it is omitted here. We denote the common non-disqualified set by \hat{Q} .

Proposition 8: The public key y is still valid unless $\hat{Q} \neq Q$.

Notes:

1. When $\hat{Q} \neq Q$, y is invalidated by SRA.
2. Replay of old A_{i0} is not possible since \hat{A}_{i0} ($\neq A_{i0}$) is used in SRA.
3. $\{s_{ji} | j \in Q\}$ at p_i can be removed after they are used in ECDKG as that in adaptive ECDKG since SRA does not require such information.

3.3. An Application Example of ECDKG: Shadow Computation and Exchange

It is not desirable to reveal the private key to any of the players during message decryption or signature signing process. We use the elliptic curve ElGamal encryption algorithm (ECElGamal) [10] as an example to show how ECDKG works. Assume there is a trusted information distribution source encrypts a message to send to a group which shares a secret s with the corresponding public key y .

The information center picks a random number $k \in \text{GF}(q)$, and the encrypted message by ECElGamal is in the form (\hat{T}, M') where $\hat{T} = kT$, and $M' = M + ky$. By ECDKG, $t + 1$ players can collaboratively decrypt this message. Without loss of generality, assume that the $t + 1$ players are p_1, p_2, \dots, p_{t+1} and the corresponding shares are s_1, s_2, \dots, s_{t+1} . If one had known the group private key s , the message could have been decrypted using $M = M' - s\hat{T}$. Let us define a shadow for p_i as follows:

$$w_i = \left(\prod_{j=1}^{t+1} p_j (p_i - p_j)^{-1} \right) s_i$$

Note that a shadow only depends on a share. All the players can compute their shadows simultaneously once the group, for which the message is shared, is formed. Assume there is an access key for this group η ¹. This example decryption algorithm is given as follows:

Algorithm 10: MDA: Message Decryption Algorithm at p_i

1. p_i computes $T_i = \eta w_i \hat{T}$ and broadcasts T_i .
2. p_i receives T_j ($j \neq i$ and $1 \leq j \leq t + 1$) and adds them to obtain $W = \sum_{k=1}^{t+1} T_k = (\eta \sum_{i=1}^{t+1} w_i) \hat{T}$.
3. p_i decrypts the message by $M = M' + W\eta^{-1}$. \diamond

¹ This key can be generated in a distributed manner via the private channels, it can be per-message based, and it should be hidden from any player out of the group.

The random number η is used to allow simultaneous decryption in multiple local groups of players in Q without exposing the secret shares. k and η together hides the secret shares from each other. Less than $t + 1$ players cannot decrypt a message since at least one piece of the complete W is hidden.

4. Implementation Issues

There are many implementations of ECC in various platforms. The performance is promising for its use in low-power devices. One good practice of implementation in low-power devices is to use table lookup method, i.e. precomputing the values and saving on-line computation. It is important for some applications with timing constraint. However, this table-lookup method in general have extra requirement on memory. Since arithmetic operation is not expensive once there is mathematical co-processor or SA-1100 like processor with that capability built in. In general, ECC has advantageous over other cryptosystems, but the choice of parameters including curve used also significantly affect the overall performance. Next we will examine various options and select a group of candidate parameters suitable for many applications .

One salient advantage of ECC is that there are many curves to choose from. Some curve may give better performance and some may be easier to implement. Coupled with the application domain, appropriate curve can be selected. There are three types of curves available for use based on application characteristics.

- 1) random curve, picking random curve and checking with an excluding list with known subexponential time attack. This approach suits for digit signature or long lasting security applications. This is due to the fact that the probability is low to select a particular curve which could be attacked by a potential subexponential algorithms which could be developed after the signature is established (*Assumption: a general subexponential algorithm for ECDLP is highly unlikely*).
- 2) CM-curve, for this type of curves, the order of the group can be computed as easily as that is needed by a reduction sequence (the reduction is defined as $V_{k+1} = \mu V_k - 2V_{k-1}$ with $V_0 = 2$ and $V_1 = \mu$). Therefore, there is no need to count points using Schoof type algorithms. To construct a CM-curve, one starts with a quadratic imaginary field and then constructs an elliptic curve over a finite field which is the reduction of an elliptic curve with complex multiplication by the field.
- 3) ABC curve, for this type of curve, the order of the group always has a large prime factor. The scalar multiplication can be efficiently performed using the τ -adic NAF or one of its variants.

For random curve approach, verification is needed to ensure a random curve is appropriate. When performance is the critical for an application and its devices, ABC curve is a good choice because efficient point arithmetic algorithms are available. Remarks on the use of ABC type of curves are given as follows:

With a given curve in the following form,

$$y^2 + xy = x^3 + ax^2 + 1,$$

where a can be 0 or 1. The domain parameters related to ECDKG are given as follows:

- 1) one field element for the coefficient a .
- 2) field representation type, for polynomial basis, the irreducible polynomial and the coefficients of it are required; for normal basis, the constant base number α is needed.
- 3) point representation, point compression type, coordinate system type and the selected point whose order must have a large prime factor p ($p > 2^{160}$).
- 4) cofactor of the p .

With current implementation with Koblitz curve, ECDKG can finish within 500 ms for key length of 283 bits and a network of PXA 255 200 MHz with 100 ms roundtrip time. Additional full experimental performance data will be generated.

5. Conclusions

In this report, we proposed a distributed key generation protocol based on ECDLP. It is well suited for applications with low-power devices due to its levels of security and key size. Further work includes how to address practical implementation issues and study the effect of the protocol when the players are spread in a large networks since long delays of communication could affect the generation time of valid keys.

References

- [1] M. Burrows, M. Abadi, R. Needham, "A Logic of Authentication", *ACM Transactions on Computer Systems*, Vol. 8, No. 1, Feb. 1990.
- [2] Christian Cachin, "Distributing Trust on the Internet", *Proc. Intl. Conference on Dependable Systems and Networks (DSN-2001)*, June 2001.
- [3] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, "Adaptive Security for Threshold Cryptosystems", *Proc. Crypto'99*.
- [4] D. Chudnovsky, G. Chudnovsky, "Sequences of Numbers Generated by Addition in Formal Groups and New Primality and Factoring Tests", *Advances in Applied Mathematics*, 7(1987), 385-434.
- [5] H. Cohen, A. Miyaji, T. Ono, "Efficient Elliptic Curve Exponentiation using Mixed Coordinates", *Advances in Cryptology - ASIACRYPT 98, LNCS, 1514*, pp. 51-65, Springer-Verlag, Berlin, Germany, 1998.
- [6] P. Downey, B. Leong, R. Sethi, "Computing Sequences with Addition Chains", *SIAM J. Computing*, 10:638-646, 1981.
- [7] P. Feldman, "A Practical Scheme for Non-Interactive Verifiable Secret Sharing", *Proc. 28th IEEE FOCS*, 1987.
- [8] G. Frey, H. Rück, "A Remark Concerning m -divisibility and the Discrete Logarithm in the Divisor Class Group of Curves", *Mathematics of Computation*, 62 (1994), 865-874.
- [9] S. Galbraith, N. Smart, "A Cryptographic Application of Weil Descent", *Codes and Cryptography, LNCS 1746 (1999)*, Springer-Verlag, 191-200.

- [10] T. El Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transactions on Information Theory*, 31, 1985, pp. 469-472.
- [11] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems", *Proceeding Eurocrypt, 1999*.
- [12] Oded Goldreich, Silvio Micali, Avi Wigderson, "Proofs that Yield Nothing But Their Validity and a Methodology of Cryptographic Protocol Design", *Proc. IEEE FOCS, 1986*.
- [18] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, "Proactive Public Key and Signature Systems", *Proc. Crypto'99*.
- [13] D. Hankerson, J. L. Hernandez, A. Menezes, "Software Implementation of Elliptic Curve Cryptography Over Binary Fields", *Proceedings of CHES 2000, Lecture Notes in Computer Science*, 1965 (2000), 1-24.
- [14] Ming-Deh A. 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.
- [15] M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein, and E. Teske, "Analysis of the Xedni Calculus Attack", *Design, Codes and Cryptography*, 2000.
- [16] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)", *International Journal on Information Security*, 1 (2001).
- [17] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.
- [18] N. Koblitz, "CM-curves with Good Cryptographic Properties," *Advances in Cryptology, CRYPTO' 91*, LNCS, 576 (1992), Springer-Verlag, 279-287.
- [19] N. Koblitz, "The State of Elliptic Curve Cryptography," *Designs, Codes and Cryptography*, 19, 173-193(2000), Kluwer Academic Publishers, Boston.
- [20] N. Koblitz, "Which Curve to Use?," *Presentation: UCLA IPAM Cryptography Workshop*, Jan. 11, 2002.
- [21] L. Lamport, "The Weak Byzantine General Problem", *The Journal of ACM*, Vol. 30, No. 3, July, 1983.
- [22] C. H. Lim, H. S. Hwang, "Fast Implementation of Elliptic Curve Arithmetic in $GF(p^n)$ ", *Public Key Cryptography-CRYPTO'2000*, LNCS 1751, Springer-Verlag, 2000, pp. 405-421.
- [23] Nancy A. Lynch, *Distributed Algorithms [Chapter 6 & 7]*, Morgan Kaufmann Publishers, Inc. 1997.
- [24] A. Menezes, M. Jacobson, and A. Stein, "Solving elliptic curve discrete logarithm problems using Weil descent", *Journal of the Ramanujan Mathematical Society*, 16 (2001), 231-260.
- [25] A. Menezes, T. Okamoto, and S.A. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field", *IEEE Transactions on Information Theory*, Vol. 39, No. 5, Sep. 1993, pp. 1639-1646.
- [26] A. Menezes, Minghua Qu, "Analysis of the Weil Descent Attack of Gaudry, Hess and Smart", *Topics in Cryptology - CT-RSA 2001*, Lecture Notes in Computer Science, 2020 (2001), 308-318.

- [27] V. Miller, "The use of elliptic curves in cryptography", *Advances in Cryptography* (Ed. H.C. Williams), Springer-Verlag, 1986, pp. 417-426.
- [28] R. Needham, M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", *Communication of the ACM*, Dec. 1978, Vol. 21, No. 12.
- [29] B. C. Neuman, T. Ts'o, "Kerberos: An Authentication Service for Computer Networks", *IEEE Communication Magazine*, Sept. 1994.
- [30] R. Nakano, and S. Olariu, "Randomized Initialization Protocols for Ad Hoc Networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 7, July 2000.
- [31] T. Pedersen, "A Threshold Cryptosystem Without a trusted Party", *Advances in Cryptology – Eurocrypt '91*. LNCS 547, Springer-Verlag.
- [32] T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing", *Advances in Cryptology – Crypto'91*, LNCS 576, Springer-Verlag.
- [33] S. Pohlig, M. Hellman, "An Improved Algorithm for Computing Logarithm over $GF(p)$ and its Cryptographic Significance", *IEEE Transactions on Information Theory*, 24 (1978) 106-110.
- [34] G. Frey, H.-G. Rück, M. Müller, "The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems", *IEEE Transactions on Information Theory*, 45, 5, 1999.
- [35] T. Satoh, K. Araki, "Fermat Quotients and the Polynomial Time Discrete Log Algorithms for Anomalous Elliptic Curves", *Commentarii Mathematici Universitatis Sancti Pauli*, 47 (1998) 81-92.
- [36] R. Schoof, "Counting Points on Elliptic Curves over Finite Fields", *J. Théor. des Nombres de Bordeaux* 7:483-494, 1998.
- [37] A. Shamir, "How to Share a Secret", *Communications of the ACM*, Vol. 22, No. 11, Nov. 1979.
- [38] C. P. Schnorr, "Efficient Signature Generation by Smart Cards", *Journal of Cryptology*, Vol. 4, pp. 161-174, 1991.
- [39] I. A. Semaev, "Evaluation of Discrete Logarithms in a Group of p -Torsion Points of an Elliptic Curve in Characteristic p ", *Mathematics of Computation*, Vol. 67, No. 221, Jan. 1998, pp. 353-356.
- [40] N. Smart, "The Discrete Logarithm Problem on Elliptic Curves of Trace One", *Journal of Cryptology*, 12 (1999), 193-196.
- [41] J.A. Solinas, "An Improved Algorithm for Arithmetic on a Family of Elliptic Curves", *Advances in Cryptology, CRYPTO 97*, pp. 357-371, Springer-Verlag, Berlin, 1997.
- [42] E. Teske, "Square-Root Algorithms for the Discrete Logarithm Problem", *Public-Key Cryptography and Computational Number Theory*, Walter de Gruyter, Berlin - New York 2001, pages 283-301.
- [43] E. D. Win, A. Bosselaers, and S. Vandenberghe, "A Fast Software Implementation for Arithmetic Operations in $GF(2^n)$ ", *Advances in Cryptology-ASIACRYPTO'96*, LNCS 1163, Springer-Verlag, 1996, pp. 65-76.
- [44] A. D. Woodbury, D. V. Bailey, C. Paar, "Elliptic Curve Cryptography on Smart Cards Without Coprocessors", *The Fourth Smart Card Research and Advanced Applications Conference*, Sep. 20-22, 2000, Bristol, UK.

[45] A. C.-C. Yao, "How to Generate and Exchange Secrets," *Proc. IEEE FOCS*, 1986.

6. Appendix: DF-VSS

As the first Distributed version of VSS, it can tolerate halting, eavesdropping adversary and fails for the other kind of adversary as listed in Section (2). The protocol is shown below.

DF-VSS

1. Each player p_i chooses a random polynomial $f_i(z)$ over Z_q of degree t : $f_i(z) = a_{i0} + a_{i1}z + \dots + a_{it}z^t$
 p_i broadcasts $A_{ik} = g^{a_{ik}} \bmod p$, for $k = 0, 1, \dots, t$. Each p_i computes the shares $s_{ij} = f_i(j) \bmod q$, for $j = 1, 2, \dots, n$ and sends s_{ij} secretly to player p_j .
2. Each p_j verifies the shares he received from the other players by checking for $i = 1, 2, \dots, n$:

$$g^{s_{ij}} = \prod_{k=0}^t (A_{ik})^{j^k} \bmod p \quad (4)$$

If the check fails for an index i , p_j broadcasts a complaint against p_i .

3. If more than t players complain against a player p_i , that player is clearly faulty and he is disqualified. Otherwise, p_i reveals the share s_{ij} matching (4) for each complaining player p_j . If any of the revealed shares fails this equation, p_i is disqualified. Set Q is defined to be the set of non-disqualified players.
4. The public value y is computed as $y = \prod_{i \in Q} A_{i0} \bmod p$. The public verification values are computed as $A_k = \prod_{i \in Q} A_{ik} \bmod p$ for $k = 1, 2, \dots, t$. Each player p_j sets his share of the secret as $x_j = \sum_{i \in Q} s_{ij} \bmod q$. The secret shared value x itself is not computed by any party, but it is equal to $x = \sum_{i \in Q} a_{i0} \bmod q$.