Soft-Timeout Distributed Key Generation for Digital Signature based on Elliptic Curve D-log for Low-Power Devices

Caimu Tang[†], Anthony T. Chronopoulos [*] and Cauligi S. Raghavendra[‡]
[†] [‡] *Dept. of Computer Science, Univ. of Southern California, Los Angeles, CA 90089*
[*] *Dept. of Computer Science, Univ. of Texas, San Antonio, TX 78249*
[‡] *Dept. of Electrical Engineering, Univ. of Southern California, Los Angeles, CA 90089*
{*caimut@cs.usc.edu, atc@cs.utsa.edu, raghu@usc.edu*}

Abstract

Group based transactions are becoming common via handhelds. Single key based systems may not be able to meet various security requirements. In this paper, we propose a threshold signature scheme based on Pedersen distributed key generation principle which is suitable for handheld devices and ad-hoc networks. Existing distributed key generation protocols use either cryptosystems based on the hardness of discrete logarithm over a finite field or integer factorization. Elliptic curve cryptosystems provide a promising alternative with efficiency which is suitable for low-power devices in terms of memory and processing overhead. In the proposed scheme, the public key from the key generation protocol follows a uniform distribution in the elliptic curve additive group, and the signature can be generated and verified efficiently. We evaluated the proposed key generation protocol and signature scheme using PARI/GP, and the key generation time takes a fraction of a second and the signature signing and verifying can be finished in a few milliseconds on the LINUX Intel PXA 255 processor.

Key Words and Phases: *Elliptic Curve Cryptosystems (ECC), Threshold Cryptography, Distributed Key Generation (DKG), Elliptic Curve Discrete Logarithm Problem (ECDLP), Discrete Logarithm Problem over a Finite Field (DLP).*

1. Introduction

As collaboration via ad-hoc networks 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 over this information. Such cases occur in many emerging applications in e-Commerce through Blackberry type of devices [1], including online auctions, portfolio management where decisions regarding funds or auction items have to be jointly made. As collaboration in society has shown great successes in many areas, this engagement in cyberspace via handheld devices in an ad-hoc fashion is certainly a trend which is expected to grow further. The security basis to facilitate such collaborations is critical. Distributing trust and sharing secrets among a group of parties provides a promising foundation for above problems, and this has attracted research interest for more than a decade e.g.[38] [11]. Recently, a model on how to distribute trust has been proposed in [5] to provide a trust service model over a group of non-trustable entities (e.g. Internet).

Keys generated by distributed key generation protocols (DKG) [14] can be used for a multi-party digital signature. Existing DKG protocols are based on either a discrete logarithm problem (DLP) over a finite field or an 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 recently developed subexponential algorithms on IFP and DLP over a finite field. Elliptic curve cryptosystems (ECC) (cf. Appendix A), on the other hand, are safe against some common DLP algorithmic techniques, e.g. index-calculus. 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. It is expected that a shorter key length can provide similar level of secrecy compared to cryptosystems based on DLP or IFP. In fact, ANSI X9.62 [3] proposes to use ECC keys as short as 163 bits while subexponential algorithms for IFP and DLP render that any key from cryptosystems based on these problems is not secure for key length less than 1024 bits [19, 4].

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.

Efficiency issues on software implementations of ECC have being studied from the aspect of fast arithmetic operation algorithms over GF(2^n) [46] and over GF(p^n) [25]. ECC has also been applied to smart card applications [39, 47] where key generation and verification takes time only in milliseconds. 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 which allow a low processing overhead.
3) Smaller key size for a similar level of secrecy which saves space and communications.

In [45], a hard-timeout distributed key generation protocol based on ECC for a causal ordering broadcast systems is proposed. Experimental results of [45] shows that such protocol built upon ECC can be made very efficient in measures of both the key length and the processing time. A practical implementation of DKG is also presented in [6] where secure sockets are used to create the private channels out of Internet and that implementation demonstrates that practical applications of DKG are possible.

In [11], the verifiable secret sharing (VSS) scheme is presented in which the *dealer* (a dealer is defined as the 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 this VSS. To combat a dynamic adversary, this scheme forms identification numbers for players using permutation of $\lceil \log_2(l) \rceil$ bit long numbers (where $\lceil . \rceil$ is the integer ceiling function and l is the key length) instead of direct use of the indices. The adversary simulation is based on zero-knowledge proof [16]. The first distributed VSS version (cf. Appendix B) is presented in [31] (we call it *Pedersen Distributed VSS*), and it is based on Feldman VSS with each player acting as a dealer. It specifies n parallel runs of all the players, and each player selects a random secret $z_i \in$ GF(q) and shares it among other players. The player collaboratively constructs a non-disqualified set Q (a non-disqualified set is a set of players who conforms to the protocol and passes all tests) in which the secret is shared.

In [14], an improved (in terms of its secrecy) version of Pedersen distributed VSS is presented and it is called DKG. This protocol can tolerate the attack where the adversary can force the secret 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 $n \geq 3t + 1$. One static attack example is presented in [14] in which two faulty players collude to make a bias on a given bit of the public key. Pedersen distributed VSS fails to this attack, and the generated public key does not follow a uniform distribution over the given field. We call this attack the *GJKR attack*. This attack can be prevented by forcing all players in the non-disqualified set to extract the public key at the same time. However, this timeout method requires synchronization among all players which could be costly. Here we overcome this problem by introducing a handshake scheme using nounces.

Ever since the first proposal of using elliptic curves for cryptographic protocols [29], [21, 23], attempts of using existing techniques or new methods to solve ECDLP in subexponential time [18] [19] [26] are still ongoing. So far, only generic exponential algorithms, e.g. square-root [44] type algorithms, are available for a broad class of ECDLP. Only in some restricted cases (cf. Appendix A), do such subexponential algorithms exist [27], [40]. Cryptosystems based on ECDLP can use smaller key size (than that is needed by DLP or IFP based systems) to provide satisfactory level of secrecy. Small key size means savings on storage, processing time and communication bandwidth. This makes it especially appealing for applications with resource constraints.

In this paper, we first propose a distributed key generation protocol called elliptic curve distributed key generation (ECDKG) which is invulnerable to the GJKR attack. It is based on the soft timeout idea. In this protocol, a player can reveal its public key share only when all other players in their local non-disqualified sets have agreed not to change their local non-disqualified sets. Nonces are used to emulate the timeout, and this way no synchronization is required among all players. The nonce is simply a timestamp, and it serves as a confirmation to other players that this player is ready to extract its public key. Compared to DKG, this new protocol enjoys all advantages of DKG besides short key length and efficiency.

We next propose to allow multiple players to sign a document using their respective key shares which is motivated by [14]. We call this scheme MultI-party Digital Signature (MiDS). In MiDS, as long as $t+1$ players have contributed to the signature, the verifier can correctly perform the verification using the group public key, and less than $t+1$ players can not forge the signature. This scheme is based on the Schnorr digital signature algorithm [39] which makes the proposed signature scheme more suitable for low power devices as compared to ECDSA [20] since the Schnorr signature enjoys shorter key length. The proposed scheme has each player contribute a partial signature and any $t + 1$ of them when combined properly can yield a valid signature which is provably non-forgeable.

In Section 2, the system models are presented and the proposed key generation protocol and signature scheme are shown in Section 3 and Section 4, respectively. Implementation issues and results are presented in Section 5. Section 6 concludes this paper.

2. The System Models

In this section, we present the communication model and the adversary model on which the proposed protocol and scheme are based.

2.1. Communication Model

We assume that there are two kinds of channels available, a broadcast channel and a private channel, and any two players can communicate via their respective private channels. This private channel is assumed to be at least as secure as the building block cryptosystems. Since private channels are de-facto in nowadays computing environment, the scalability of the proposed protocol is not limited by this requirement. Broadcasting uses a flooding mechanism (e.g., scoped flooding with time-to-live constraint). Messages from ECDKG follow different semantics. A broadcast message, it either reaches all recipients or none. Furthermore, if it reaches all recipients, the reception order by these players is random. We denote this message broadcasting semantics as (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 no message loss 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.

2.2. Adversary Model

The adversaries can be categorized into two broad types: static and adaptive. For a static type adversary, decision on which player to break into is made before the run of the protocol. Adaptive adversaries, on the other hand, can change its scheme once new run-time information is made available. For both types of adversaries, they 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. The message processing time by these adversaries is ignored under the assumption that the adversary has more computing power than any of the honest players. There are four types of specific adversaries we are dealing with in this paper: 1) *Halting adversary*. In a 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 by all necessary means. This decision can not be changed by exploiting the run-time information obtained during protocol execution. All players are taken as the same to this adversary. 4) *Replay adversary*. A replay adversary buffers messages and sends these out whenever necessary to impersonate a honest player. This type of attacks is commonly applicable to a multi-stage protocol and signature schemes.

The design of the proposed protocol has taken into account these adversaries in the communication systems with MOS and MBS semantics.

3. The Proposed Key Generation Protocol

The proposed distributed key generation protocol under the models is based on the improved version of Pedersen Distributed VSS [14] focusing on efficiency. We use field GF(q) as the base 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 GF(q) and players know these numbers of each other. (There exist algorithms to generate such random numbers in a distributed setting [30] and the session initiator can be used to facilitate this process.)

Notation: GF$^+(q)$ denotes the induced additive group of GF(q) and GF$^*(q)$ denotes the induced multiplicative group of 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 $p_i \in$ GF$^*(q)$. We use "*player i*" or p_i interchangeably in this paper. Let $E/$GF(q) be an additive group based on a properly preselected elliptic curve E as explained in Appendix A. The cardinality of $E/$GF(q) is a prime number or has a large prime factor for the cryptographical purpose. Let T be the point in $E/$GF(q) of a large prime order and denote this order by p henceforth.

In this paper, we assume that point multiplication (a point multiplied by a scalar in GF(q)) is performed in $\mathcal{G}$, all other arithmetic operations are performed in finite field GF(q) unless otherwise specified. To evaluate $Q(x)T$, we first evaluate $Q(x)$ using field arithmetic in GF(q), then we take modulo p to the result of $Q(x)$, and $(Q(x) \bmod p)$ is the scalar multiplier 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. ECDKG consists of five algorithms : the key distribution (KD), the key verification (KV), the key check (KC), the key nonces collection (KN), and the key generation (KG). The protocol at a player p_i is given as follows:

```
Protocol 1  ECDKG($p_i, Q = \{p_1, p_2, \cdots, p_n\}$)
  Given:
  Q: a set of non-disqualified players.
  Execute:
  set $Q_i = Q$
  KD($p_i, t, T$)
  KV($p_i, t, T$)
  while(1) {
      KC($p_i, t, Q_i$)
      if(KN($p_i, t, Q_i$)) exit-loop
  }
  KG($p_i, Q_i$) □
```

Algorithm 1 KD(p_i, t, T)

1. *Initialization: pick $(2t + 2)$ random numbers uniformly, $a_{ik} \in$ GF(q) and $b_{ik} \in$ GF(q) $(0 \le k \le t)$, 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 $s_{ij} = f_i(p_j) \bmod p$, and $s'_{ij} = f'_i(p_j) \bmod p$ $(j \in \mathcal{S}_i)$.*
   - *compute $(t + 1)$ public values:*

$$P_{ik} = (a_{ik}T) \oplus (b_{ik}T') \qquad (0 \le k \le t)$$

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 KV(p_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*

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

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 KC(p_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 KN(p_i, t, Q_i)

1. *if $|Q_i| \le t$, p_i is excluded and exit and return false.*
2. *broadcast a nonce containing p_i and freeze Q_i.*
3. *receive nonces.*
4. *if nonces from all players in Q_i are received, exit and return true.* □

Algorithm 5 KG(p_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*

$$y_i = \sum_{j \in Q_i}^{\oplus} A_{j0} \qquad \square$$

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 the 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 nonce mechanism is used in ECDKG to countermeasure the GJKR attack. Because of the use of soft-timeout via timestamps and replay of public messages not being useful, ECDKG is also invulnerable to the replay adversary. 4) T' can be pre-computed in a distributed fashion based on the standard DKG with all the n players involved. Let these players form an ordered list, a simple scheme is given as follows: i) all players run DKG to generate a uniform random number $r \in$ GF(q) and no single player knows r and each has a shared piece of it. ii) each players 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 T' is precomputed.

5) The information dissemination order is private information first followed by the public information. 6) 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 $n \geq 3t+1$.*

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 message broadcasting semantics (MBS), the same public information should also be available to p_j by this time. By message ordering semantics (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 public. 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 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 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 henceforth.

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

This is the direct result of 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 ($n \leq t{+}1$), 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 invulnerable to the GJKR attack.*

Proof: By the simulation done in [14], we know that the oracle would have been able to construct the public key without a biased distribution in the base field before the execution of KG when the KN algorithm collects all the nonces and the size of Q_i is greater than t.

Since the honest players will follow the protocol, and their respective non-disqualified set are complete before the execution of KG algorithm by the soft timeout mechanism at the end of KN. Due to the MOS, when the public key shares $\{A_{i0}\}$ $(i \in Q)$ are broadcast, all nonces are received. Therefore, the non-disqualified set after KG algorithm should remain the same since the only messages allowed in KG algorithm are the public values of public key shares. These messages can not change the internal states of the protocol. □

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 5τ, where τ equals half of the longest roundtrip time between any two players.*

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

The KN algorithm requires one τ in the worst case, and 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 and the rest are the uncorrupted 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. The proposition follows by noticing that ECDKG will generate a valid public key when $n \geq 3t+1$. □

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

Proof: 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.

By using the simulator presented in [14], we can construct a new simulator with two parts, and the first part is an exact copy of the portion in *SIM* (2a) and (2b) of [14] which includes the KD, KV and KC algorithms. To incorporate the KN and KG algorithms into the simulator, by noticing that the honest players follow the protocol, and the common non-disqualified set is fixed before the execution of KG algorithm. The simulator can proceed to broadcast public key shares $A_{i0}, \forall i \in Q$ and collect these shares to generate the public key. Since in the KG algorithm all messages are public messages, the second part of the simulator is also indistinguishable from this based on public information. □

Note that in Proposition 7 we used an important fact of the soft timeout which is that the player after it sends its nonce, will disregard any further message except two types of messages: 1) another nonce and 2) public key shares A_{j0}. The protocol states can not be changed by the KG algorithm. This acts as if there is a timout right before the KG algorithm. This is the soft timeout.

The adaptive adversary is not explicitly dealt with in this paper. However, when combined with information erasure technique [18] and an elliptic curve zero-knowledge proof on the secret share from each player in the non-disqualified set before public key extraction, ECDKG is also invulnerable to this type of adversaries. The use of non-interactive zero-knowledge proof before public key extraction will render useless all gathered information by an adaptive adversary. A similar technique using discrete logarithm zero-knowledge proof has been developed in [7].

4. Multi-party Digital Signature Scheme Using ECDKG

In this section, we show how to sign documents using this soft timeout ECDKG. The approach is to have a group of individual players generate a partial signature each and the final signature is a combined version of these partial signatures. We call the proposed scheme MiDS. We will show that less than $t+1$ players can not forge a signature. MiDS is the elliptic curve version of the Schnorr signature scheme [39] and the corresponding threshold Schnorr signature scheme [15] under the proposed soft-timeout ECDKG protocol.

Denote the message by m, and the partial signature from p_i by sig_i and the final complete signature by sig. Algorithm 6 and Algorithm 7 show that $t+1$ players can sign a document which can only be verified using the group public key y generated by ECDKG where we denote MiDS-S and MiDS-V as the signing algorithm and verifying algorithm, respectively. We denote $\{T\}_x$ as the x-coordinate of point T. Note here that a public key in ECC may not necessary represented directly by a point as a point compression algorithm may be used to reduce the key length without affecting the underlying secrecy. We assume that there are n players and p_i has a secret share s_i and the public key y is known to all. By the protocol of ECDKG, the public share y_i of p_i can be computed using only public information and it equals to s_iT $(y_i = s_iT)$. *Hash* in Algorithm 6 is the one-way hash function which takes two arguments of the same bit length as its input.

Algorithm 6 *MiDS-S*(p_i, m, s_i, y)

1. *run ECDKG to generate a one-time secret share* k_i*, the corresponding one-time public key* $r = kT$*, and the one-time public share* $r_i = k_iT$*. (Note that* k *is unkown to any player by the properties of ECDKG and* r_i *can be also computed from public information only.)*
2. *compute* $c = Hash(m, \{r\}_x)$*.*
3. *broadcast* sig_i *where,*

$$sig_i = k_i - c\prod_{t \neq i} \left(p_t(p_i - p_t)^{-1}\right) s_i \qquad (2)$$

4. *receive partial signatures: if the following received partial signature set* SIG_i *defined in (3) has cardinality less than t, return failure.*

$$SIG_i = \left\{sig_j \,\middle|\, (sig_j)T = r_j - cQ(j)y_j\right\} \qquad (3)$$

where,

$$Q(j) = \prod_{t \neq j} p_t (p_j - p_t)^{-1}$$

5. *generate signature: pick t partial signatures from SIG$_i$ and add them to sig$_i$ to generate the complete signature (c, sig).* □

Algorithm 7 *MiDS-V(c, sig, m)*

1. *compute $r = ((sig)\,T) \oplus ((-c)y)$.*
2. *test $c = H(m, \{r\}_x)$, accept the signature if it passes the test.* □

Note that one of the reasons to use a random one-time secret upon which all player are agreed (via ECDKG) is to ensure that a nontrivial one-time secret is used and no information regarding the secret shares can be revealed. Also note that this one-time secret should not be used more than once.

Compared to the ECDSA [20], this signature can have a shorter length. Signature generation is also simple. The verification of message m given signature (c, sig) simply follows the verification of Schnorr signature scheme as shown in Algorithm 7.

Proposition 8 *The correctness of the verification.*

We notice that for any $(t+1)$ shares, we have (which can not be computed directly)

$$\text{private key} = \sum_{i=1}^{t+1} \left(\prod_{j=1, j\neq i}^{t+1} \left(p_j (p_i - p_j)^{-1}\right) s_i \right) \bmod p$$

the correctness then follows from the same argument in [39].

Due to the intractability of ECDLP and elliptic curve Diffie-Hellman problem (ECDHP) (i.e. compute rsT from point sT and rT, note that ECDHP can be reduced in polynomial time to ECDLP), the signature has the non-forgery property against any adversary. Proposition 9 shows the threshold non-forgery property of MiDS.

Proposition 9 *The non-forgeability of MiDS: less than $t+1$ players can not forge the signature.*

Proof: We first show that a reduction from MiDS to the Schnorr scheme is possible in polynomial time with a finite number of messages. Then the non-forgery property under the chosen message attack follows from that of Schnorr scheme of which the existential forgery can lead to a non-negligible probability of success of finding the discrete logarithm of ECDLP in polynomial time.

By the secrecy of ECDKG, k is unknown to all players and r follows a uniform distribution in $\mathcal{G}$. Therefore c follows a uniform distribution in the base field. Due to the lack of the required number of degree of freedoms, simultaneously solving systems of equations in the form of (2) is not possible even with two additional constraints, i.e. $k = k_1 + k_2 + \cdots + k_n$ and $t+1$ secret shares when combined can reveal the private key.

Without loss of generality, when the first t players collude, to generate a valid signature, we need to obtain the unknowns k_{t+1} and s_{t+1} which subject to (2). Due to the fact that k is unknown, and the shared secret is unknown. This system still lacks of one degree of freedom. Since the simulator can not access more than t players during one course of execution, it is not able to directly generate the valid sig for a given c which can pass the test in Algorithm 7.

With the random oracle, the simulator can initiate $t+1$ parallel runs with randomly selected $t+1$ signers. For each run, the simulator randomly feed messages to Algorithm 6 to hope for two partial signatures $\{\text{sig}'_i, \text{sig}''_i\}$. When these $\{\text{sig}'_i\}$ and $\{\text{sig}''_i\}$ $(1 \leq i \leq t+1)$ are combined as in Algorithm 6, it can yield two instances with two valid signatures (c', sig') and (c'', sig''). Let

$$r = ((\text{sig}')T) \oplus ((-c')y) = ((\text{sig}'')T) \oplus ((-c'')y) \quad (4)$$

From Lemma 2 in [34], we know this probability is non-negligible. From (4), following the same argument in the proof of Theorem 3 in [34] in the base field GF(q) instead of $Z/_pZ$, the ECDLP can be solved with non-negligible probability of success. The solution to $\log(y)$ is then

$$\log(y) = (\text{sig}' - \text{sig}'')(c' - c'')^{-1}$$

Therefore, the intractability of ECDLP guarantees the non-forgeability of MiDS. □

5. Implementation Issues and Results

There are many implementations of ECC in various platforms. The performance is promising for its use in resource constrained devices. One good practical implementation in these 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 constraints. In general, ECC has advantages over other cryptosystems, but the choice of parameters including the curve used also significantly affect the overall performance. Next we will examine various options and give comments on selecting parameters for ECDKG and MiDS which are suitable for applications using embedded devices.

One salient advantage of ECC is that there are many curves to choose from. Some curve may give better performance while some other may be easier to implement. Coupled with the application domain, a proper curve can be selected. There are three types of curves available for use based on application characteristics, namely, random curves, Complex Multiplication (CM) curves, and Anomalous Binary Curves (ABC) or Koblitz curves. First, for the use of random curves, applications pick a random curve,

Table 1: Computation Overhead of ECDKG

Algorithm	Multiplication	Addition
KD	$(t+1, (t+1)\log_2(t+1))$	(t, t)
KV	$(n(t+1)\log_2(t)+n-1, nt\log_2(t))$	$((n-1)(t+1), 2t+n-1)$
KC	$(0,0)$	$(0, n-1)$
KN	$(0,0)$	$(0,0)$
KG	$(1,0)$	$(n,0)$
MiDS-S	$(2t, n+C_{hm})$	$(t, 1+C_{ha})$
MiDS-V	$(2, C_{hm})$	$(1, C_{ha})$

then they check with an excluding list with known subexponential time attack. This approach suits for digital signature or long lasting keys. This is due to the fact that the probability is low to select a particular curve which happens to be vulnerable to a potential subexponential algorithms developed after the signature is established. Second, for the use of CM curves, the advantage is that the order of the group can be easily computed by a simple 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 the Schoof algorithms [37] and this otherwise is needed as in the random curve method. 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 in the field. Third, for the use of ABC curves, the order of the group always has a large prime factor. The point multiplication can be efficiently performed using the τ-adic non-adjacent form (NAF) or one of its variants. This gives an easy protocol setup, and it suits for applications in networks with resource constrained devices.

We use PARI/GP [2] to evaluate the proposed protocols. We assume that ABC curves are used for ECDKG and MiDS to derive the experimental data. 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 its coefficients 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 for an ABC curve. 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 $E/$GF(2^m).

We have an initial implementation of ECDKG procedures in PARI/GP [2]. We first itemize the overhead of ECDKG including computation and communication. Table 1 shows these estimates, where the elliptic curve multiplication is point multiplication. Each cell in Tab. 1 consists of two expressions in (x, y) format, and x is the cost on $\mathcal{G}$ – the elliptic curve main subgroup, and y is the cost on GF(q). Table 2 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. The hash cost is represented by two constants C_{hm} for field scalar multiplication and field modular operations, respectively.

Note that in Tab. 1, the precomputation cost is excluded for MiDS-S. Note also that in Tab. 1, the evaluation of an exponent in GF(q) uses repeated squaring and the evaluation of point multiplication in $\mathcal{G}$ uses repeated doubling.

Table 2: Communication Overhead of ECDKG

Algorithm	Reception	Transmission
KD	0	$(n-1, 1)$
KV	$n-1$	$(0, 3t)$
KC	$2t+n-1$	$(0,0)$
KN	$n-1$	$(0,1)$
KG	$n-1$	$(0,1)$
MiDS-S	1	$(0, n-1)$
MiDS-V	0	$(0,0)$

Note that in Tab. 2 for the reception overhead in the KC algorithm, although the KC algorithm is included in a loop, the number of received messages must be bounded above by the total number of messages actually transmitted.

Table 3 shows the worst-case memory requirement for each algorithm of ECDKG, where C is the worst-case memory requirement for performing one individual point multiplication. Proposition 1 is used in deriving these bounds. For the adversaries, the eavesdropper can only affect the KV, KN and KG algorithms where broadcast is involved. Due to the static malicious adversary, the number of players in KC has to be greater than $3t$.

We used the Intel PXA 255 200 MHz processor as the platform running PARI/GP to simulate the protocol. Table 4

Table 3: Worst-Case Memory Requirement of ECDKG

Algorithm	Worst-Case Memory	Group Size n
KD	$(t+1)\log_2(p)+C$	$n>t$
KV	$(n-1)\log_2(p)+C$	$n>t$
KC	$\frac{2}{3}(n-1)\log_2(p)+C$	$n>3t$
KN	$(n-1)\log_2(p)$	$n>t$
KG	$(n+1)\log_2(p)+C$	$n>t$
MiDS-S	$2\log_2(p)+C$	$n>t$
MiDS-V	$2\log_2(p)+C$	$n>t$

Table 4: Key Generation Timing (n = 5, t = 3)

Curve	$\log_2(p)$	RT-20	RT-100	RT-500
K-163	163	93 ms	198 ms	800 s
K-233	232	156 ms	261 ms	860 s
K-283	281	209 ms	314 ms	910 s

shows the key generation times for 5 players under varying key bit lengths and message roundtrip times where "RT-x" is for the roundtrip time at x ms (i.e. $\tau = x/2$) and "K-x" is for the NIST ABC curves [3] in the field GF(2^x). Table 5 shows the key generation times for 10 players under varying key bit lengths and message roundtrip times. In both cases, the threshold is set at 3.

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), where in this case, q is a prime power of 2. We generally believe that in a large network with a relative long message roundtrip time, it is the network latency that determines the key generation time, not the key processing time.

We next give the timing of MiDS based on the PXA 255 processor, the times we produced is the actual signing and verifying time and we exclude the precomputation cost involved. We still use the same set of curves for signature although the key generation protocol may use a longer key as compared to the signature scheme. Table 6 shows the timing on 5 players and 4 signers and Table 7 shows the timing on 10 players and 4 signers. We use roundtrip time of 20 ms in this set of experiments and the hash function cost is derived from SHA-1, SHA-256 algorithms with appropriate paddings. This timing makes MiDS suitable for real-time low-power devices.

Table 5: Key Generation Timing (n = 10, t = 3)

Curve	$\log_2(p)$	RT-20	RT-100	RT-500
K-163	163	124 ms	239 ms	840 ms
K-233	232	242 ms	357 ms	960 ms
K-283	281	341 ms	456 ms	1060 ms

Table 6: Signature Timing (n = 5, t = 3)

Curve	$\log_2(p)$	Signing Time	Verifying Time
K-163	163	19450 μ s	3000 μ s
K-233	232	32430 μ s	3500 μ s
K-283	281	43430 μ s	3900 μ s

Table 7: Signature Timing (n = 10, t = 3)

Curve	$\log_2(p)$	Signing Time	Verifying Time
K-163	163	20950 μ s	14250 μ s
K-233	231	35930 μ s	20300 μ s
K-283	281	48600 μ s	25520 μ s

6. Conclusions

In this paper, we proposed a distributed key generation protocol based on elliptic curve discrete logarithm problem and a multi-party signature scheme. They are well suited for applications where the devices are resource constrained. These schemes provide high level of secrecy with smaller key size as compared to the protocols based on discrete logarithm over finite field. Sub-second key generation with practical key length is possible for low-power devices, and the signature signing and verifying processes takes time in milliseconds.

Acknowledgment

Special thanks to Prof. Ming-deh Huang at the Computer Science Department of University of Southern California for many insightful discussions on elliptic curve cryptosystems and its use for threshold cryptography. Reviewer comments that helped enhance the quality of presentation of the article are gracefully acknowledged.

References

[1] Research in Motion Limited. Blackberry Wireless Solutions. [online]. Available: http://www.blackberry.com.

[2] The PARI/GP Group. PARI/GP, Version `2.1.5`. [online]. Available: http://pari.math.u-bordeaux.fr, Bordeaux, 2004.

[3] ANSI X9.63, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Key Agreement and Key Transport Protocols.

[4] D. Boneh, "Twenty Years of Attacks on the RSA Cryptosystem, " *Notices of the American Mathematical Society (AMS)*, 46(2):203 – 213, 1999.

[5] Christian Cachin, "Distributing Trust on the Internet", *Proc. Intl. Conference on Dependable Systems and Networks (DSN)*, June 2001.

[6] A. T. Chronopoulos, F. Balbi, D. Veljkovic, N. Kolani, "Implementation of Distributed Key Generation Algorithms using Secure Sockets", *Proc. of 3rd IEEE Intl. Symposium on Network Computing and Applications,* Aug. 2004.

[7] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, “Adaptive Security for Threshold Cryptosystems”, *Proc. Crypto’99.*, 1999.

[8] D. Chudnovsky, G. Chudnovsky, “Sequences of Numbers Generated by Addition in Formal Groups and New Primality and Factoring Tests”, *Advances in Applied Mathematics*, 7: 385 – 434, 1987.

[9] 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, Germany, 1998.

[10] P. Downey, B. Leong, R. Sethi, “Computing Sequences with Addition Chains”, *SIAM J. Computing*, 10:638-646, 1981.

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

[12] G. Frey, H. Rück, “A Remark Concerning m-divisibility and the Discrete Logarithm in the Divisor Class Group of Curves”, *Mathematics of Computation*, 62: 865 – 874, 1994.

[13] S. Galbraith, N. Smart, “A Cryptographic Application of Weil Descent”, *Codes and Cryptography, LNCS 1746*, Springer-Verlag, 191-200, 1999.

[14] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, “Secure Distributed Key Generation for Discrete-Log Based Cryptosystems”, *Proceeding Eurocrpt, 1999.*

[15] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, “Revisiting the Distributed Key Generation for Discrete-Log Based Cryptosystems”, *RSA Security’03*.

[16] 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 Kakobsson, Stanislaw Jarecki, Hugo Krawczyk, "Proactive Public Key and Signature Systems”, *Proc. Crypto’99.*, 1999.

[17] D. Hankerson, J. L. Hernandez, A. Menezes, “Software Implementation of Elliptic Curve Cryptography Over Binary Fields”, *Proc. of CHES 2000*, LNCS 1965, 1-24, 2000.

[18] Ming-Deh A. Huang, K. L. Kueh, and K. Tan, “Lifting Elliptic Curves and Solving the Elliptic Curve Discrete Logarithm Problem”, *LNCS 1838*, pp. 377-384, 2000.

[19] M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein, and E. Teske, “Analysis of the Xedni Calculus Attack”, *Design, Codes and Cryptography*, 2000.

[20] D. Johnson, A. Menezes, and S. Vanstone, “The elliptic curve digital signature algorithm (ECDSA)”, *International Journal on Information Security*, 1, 2001.

[21] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, 48: 203 – 209, 1987.

[22] N. Koblitz, “CM-curves with Good Cryptographic Properties,” *Advances in Cryptology, CRYPTO’ 91*, LNCS, 576, Springer-Verlag, 279-287, 1992.

[23] N. Koblitz, “The State of Elliptic Curve Cryptography,” *Designs, Codes and Cryptography*, Kluwer Academic Publishers, Boston, 19: 173 – 193, 2000.

[24] N. Koblitz, “Which Curve to Use?” *Presentation: UCLA IPAM Cryptography Workshop,* Jan. 11, 2002.

[25] 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, pp. 405-421, 2000.

[26] A. Menezes, M. Jacobson, and A. Stein, “Solving elliptic curve discrete logarithm problems using Weil descent”, *Journal of the Ramanujan Mathematical Society,* 16: 231 – 260, 2001.

[27] A. Menezes, T. Okamoto, and S.A. Vanstone, “Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field”, *IEEE Trans. Inform. Theory*, 39(5): 1639 – 1646, Sep. 1993.

[28] 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, 308-318, 2001.

[29] V. Miller, “The use of elliptic curves in cryptography”, *Advances in Cryptography* (Ed. H.C. Williams), Springer-Verlag, pp. 417-426, 1986.

[30] R. Nakano, and S. Olariu, “Randomized Initialization Protocols for Ad Hoc Networks”, *IEEE Transactions on Parallel and Distributed Systems*, 11(7): 749 – 759, July 2000.

[31] T. Pedersen, “A Threshold Cryptosystem Without a trusted Party”, *Advances in Cryptology – Eurocrypt ’91.* LNCS 547, Springer-Verlag, 1991.

[32] T. Pedersen, “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”, *Advances in Cryptology – Crypto’91*, LNCS 576, Springer-Verlag, 1991.

[33] S. Pohlig, M. Hellman, “An Improved Algorithm for Computing Logarithm over GF(p) and its Cryptographic Significance”, *IEEE Transactions on Information Theory*, 24(1): 106-110, Jan. 1978.

[34] D. Pointcheval, J. Stern, “Security Proofs for Signature Schemes”, *Advances in Cryptology – Proc. of EUROCRYPT’96*, LNCS 1070, Spinger-Verlag, May, 1996.

[35] 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): 1717 – 1719, 1999.

[36] T. Satoh, K. Araki, “Fermat Quotients and the Polynomial Time Discrete Log Algorithms for Anomalous Elliptic Curves”, *Commentarii Mathematici Universitatis Sancti Pauli*, 47: 81 – 92, 1998.

[37] R. Schoof, “Counting Points on Elliptic Curves over Finite Fields”, *J. Théor. des Nombres de Bordeaux* 7:483-494, 1998.

[38] A. Shamir, “How to Share a Secret”, *Communications of the ACM*, 22(11): 612 – 613 , Nov. 1979.

[39] C. P. Schnorr, “Efficient Signature Generation by Smart Cards”, *Journal of Cryptology*, 4: 161 – 174, 1991.

[40] I. A. Semaev, “Evaluation of Discrete Logarithms in a Group of p-Torsion Points of an Elliptic Curve in Characteristic p”, *Mathematics of Computation*, 67(221): 353 – 356, Jan. 1998.

[41] N. Smart, “The Discrete Logarithm Problem on Elliptic Curves of Trace One”, *Journal of Cryptology*, 12: 193-196, 1999.

[42] 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.

[43] J.A. Solinas, "Efficient Arithmetic on Koblitz Curves", *Designs, Codes and Cryptography*, 19(2):195 – 249, Mar. 2000.

[44] E. Teske, "Square-Root Algorithms for the Discrete Logarithm Problem", *Public-Key Cryptography and Computational Number Theory*, Walter de Gruyter, Berlin - New York, pages 283-301, 2001.

[45] C. Tang, and A. T. Chronopoulos, "An Efficient Group Key Generation Protocol in Causal Ordering Broadcast Systems", *Proc. of Intl. Workshop on Security in Networks and Distributed Systems* (to apear), in conjunction with The 11th IEEE Intl. Conf. on Parallel and Distributed Systems, 2005.

[46] 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, pp. 65-76, 1996.

[47] 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, Bristol, UK, 2000.

Appendix A: The 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 on F. For practical purposes, two types of fields are often considered: GF(p) for p a prime and GF(2^p) for p a prime. For 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 $a, b \in$ GF(p) and $4a^3 + 27b^2 \neq 0$. For 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 $a, b \in$ GF(2^p), and $b \neq 0$. Group E/F is appealing for cryptographic purpose 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, and there are a 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 x.*

There are the following known attacks specific to ECDLP.

1) Elliptic curves with T of a smooth order. An attack presented in [33] reduces the problem of finding the secret x to the problem of finding x modulo each of the prime factor of $p - 1$, then use Chinese Remainder Theorem to solve for x. This algorithm can solve this type of ECDLP in $O(poly(log(p)))$.

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 the prime field GF(p), we need to avoid the curve whose cardinality divides $p^k - 1$ for a small k ($k \leq 20$), since Weil pairing [26] (MOV attack) or Tate pairing [35, 12] can reduce ECDLP to DLP in a multiplicative group of some extension field of GF(p) where the DLP can be solved in subexponential time when k is small. For cryptographic purposes, we can check up to 20 and it is sufficient since the discrete logarithm over GF(p^C) for $C > 20$ is intractable.

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, $\mid E/\text{GF}(p) \mid = p$, and ECDLP can be reduced to DLP in an additive group [40] [41] [36].

4) Curves over field GF(2^m), m is composite. Weil descent [13] GHS attack might be used to solve the ECDLP over a binary field. Weil descent reduces ECDLP in GF(2^m) to a DLP in an abelian variety over a proper subfield of 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 [24]. However, it has been shown in [28] that it is infeasible for $E/\text{GF}(2^n)$ when n is a prime and $n \in [160, 600]$.

Denote by q the order of the base field GF(p) or 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 $E/\text{GF}(q)$ [37]. These can be checked before selecting the curve. From efficiency point of view, there are some efficient implementations of ECC systems based on ABC curves [22] [42, 43]. The base field is 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. The inverse in $E/\text{GF}(q)$ is cheap, and Non-Adjacent Form (NAF) has been particularly useful for point multiplication. 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 a group with complex multiplication property, an analogy of integer NAF has been developed. For some expansion, the non-zero terms are few and the point multiplication can be efficient performed. For an 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 a small number of point additions and no need of point doublings. For ABC curves, there exists a reduction of an elliptic curve with complex multiplication by a complex multiplication ring $Q[\sqrt{-7}]$. This leads to a simple point counting method and this can also significantly reduce the point multiplication [42] using the τ-adic NAF with

$$\tau = \frac{-(-1)^a + \sqrt{-7}}{2}$$

which is the root of the characteristic equation of the Frobenius automorphism over 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 point 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 [10]. This type of algorithms is practical since the inverse is almost "free", i.e. $-(x, y) = (x, -y)$ in the prime field GF(p), or $-(x, y) = (x, x + y)$ (because $-(x, x+y) = (x, x+y+x) = (x, y)$) in GF$(2^m)$. There are two group representations that can be used, i.e. with polynomial basis and with normal basis. Implementations have to choose a proper basis in order to be efficient. An element in GF(2^n) can be represented as a polynomial: $\sum_{i=0}^{n-1} c_i \alpha^i$, where $c_i \in$ GF(2). It can also be represented as a vector of dimension n using following basis:

$$\theta, \theta^2, \cdots, \theta^{2^{n-1}}$$

where $\theta \in$ GF(2^n) and the elements in the vector are in GF(2). For a polynomial representation, an irreducible trinomial or an irreducible pentanomial is selected as the reduction polynomial in practice. Normal basis allows simple point doubling but complex point multiplication in general. One type of normal bases, the so called Gaussian normal bases (GNB), allows simple group arithmetic on both point doubling and multiplication. There are also two kinds of point coordinates 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 [9]. Using the projective coordinate system, inverse can be avoided using multiplication. Especially, Jacobian projective coordinates [8], the projective point

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

and its variants yied superior performance for field arithmetic [17].

Appendix B: The Pedersen Distributed VSS

1. Each player p_i chooses a random polynomial $f_i(z)$ over Z_q of degree t:

   $$f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{it}z^t$$

   then p_i broadcasts

   $$A_{ik} = g^{a_{ik}} \bmod p$$

   for $k = 0, 1, \cdots, t$. Each p_i computes the shares $s_{ij} = f_i(j) \bmod q$, for $j = 1, 2, \cdots, n$ 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 $i = 1, 2, \cdots, n$:

   $$g^{s_{ij}} = \prod_{k=0}^{t} (A_{ik})^{j^k} \mod p \qquad (5)$$

   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 (5) 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. item 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 $k = 1, 2, \cdots, t$.

   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

   $$x = \sum_{i \in Q} a_{i0} \bmod q$$