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].
Efficiency issues on software implementations of ECC have been 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 $\lceil \log_2(t) \rceil$ bit long numbers (where $\lceil \cdot \rceil$ is the integer ceiling function and $t$ 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 others. 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 nonces.

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 cannot 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-forgable.

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). 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 executed, 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 corrupt 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 pi 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)). G denotes the main subgroup of order p which is derived from a point T, and used as the base group of ECDKG; ⊕ denotes the point add operator over G and ⊕ denotes the point summation under ⊕, and

\[ S_i = \{ p_j | j \neq i, 1 \leq j \leq n \} \]

is the set of peer players of pi.

Let n be the total number of players who want to form a secure group, and they are identified by the distinct IDs (p1, p2, ..., pn), where pi ∈ GF×(q). We use “player i” or pi 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 cryp-tographical 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 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 $(Q(x) \mod p)$ is the scalar multiplier on point $T$ to get the result point in $G$. 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 $G$ 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**($p_i, Q = \{p_1, p_2, \ldots, p_n\}$)

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($q$) $(0 \leq k \leq t)$, as polynomial coefficients to generate two polynomials of degree $t$ as follows:

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

- compute $s_{ij} = f_i(p_j) \mod p$, and $s'_{ij} = f'_i(p_j) \mod p$ ($j \in S_i$).
- compute $(t+1)$ public values:

$$P_{ik} = (a_{ik}T) + (b_{ik}T') \quad (0 \leq k \leq t)$$

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 $p_j$ ($j \in S_i$).

3. Dissemination of public information: broadcasts a message containing $\{P_{ik} | 0 \leq k \leq t\}$.

**Algorithm 2 KV($p_i, t, T$)**

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

1. verify

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

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 $s_i = \sum_{j \in Q_i} s_{ij}$, if one of the following two conditions holds:

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

**Algorithm 4 KN($p_i, t, Q_i$)**

1. if $|Q_i| \leq t$, $p_i$ 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

$$y_i = \sum_{j \in Q_i} A_{j0}$$

Remarks: 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 $p_j$ for $j \in S_i$. 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 ≥ 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 $p_u \in Q_i$, since $p_u$ 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,

$$(s_{ui}T) \oplus (s_{ui}T') = \sum_{k=0}^{t} (p_{i}^{k}P_{uk})$$

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 (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 $p_j$ receives them, so does $p_j$. If $p_u$ is uncorrupted, $s_{uj}$ and $s_{uj}'$ are correct at $p_j$, then,

$$(s_{uj}T) \oplus (s_{uj}T') = \sum_{k=0}^{t} (p_{j}^{k}P_{uk})$$

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. $\square$

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:

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

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. $\square$

**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. $\square$

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

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

The KN algorithm requires one $\tau$ in the worst case, and in the end, one more $\tau$ 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

A new simulator with two parts, and the first part is

\[ \text{cast public key shares} \]

noticing that the honest players follow the protocol, and

which includes the KD, KV and KC algorithms. To incor-

porate 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 execu-

tion of KG algorithm. The simulator can proceed to broad-

cast 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 $A_{i0}$. The protocol states can not be changed by the KG algorithm. This acts as if there is a timeout 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 $\text{sig}_i$, and the final complete signature by $\text{sig}$. 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 necessarily represent 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 unknown to any player by the properties of ECDKG and $r_i$ can be also computed from public information only.)
2. compute $c = \text{Hash}(m, \{r\}_x)$.
3. broadcast $\text{sig}_i$ where,

\[ \text{sig}_i = k_i - \prod_{t \neq i} (p_t(p_t - p_i)^{-1}) s_i \] (2)

4. receive partial signatures: if the following received partial signature set $\text{SIG}_i$ defined in (3) has cardinality less than $t$, return failure.

\[ \text{SIG}_i = \{ \text{sig}_j | (\text{sig}_j)T = r_j - cQ(j)y_j \} \] (3)
where,
\[ Q(j) = \prod_{i \neq j} p_i(p_j - p_i)^{-1} \]

5. generate signature: pick \( t \) partial signatures from \( \text{SIG}_i \) and add them to \( \text{sig}_i \) to generate the complete signature \((c, \text{sig})\). 

Algorithm 7 MiDS-V\((c, \text{sig}, m)\)

1. compute \( r = ((\text{sig}) T) \oplus ((-c)y) \).
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 players 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 \((c, \text{sig})\) 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 cannot be computed directly)

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

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-forgery of MiDS: less than \( t + 1 \) players cannot 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 \( G \). 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 \( \text{sig} \) 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 \( \{\text{sig}, \text{sig}'\} \). These \( \{\text{sig}, \text{sig}'\} \) (1 \( \leq t \leq t + 1 \)) are combined as in Algorithm 6, it can yield two instances with two valid signatures \((c', \text{sig}')\) and \((c'', \text{sig}'')\). Let

\[ r = ((\text{sig}')T) \oplus ((-c')y) = ((\text{sig}'')T) \oplus ((-c'')y) \]

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-forgery 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,
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. Third, for the use of ABC curves, the advantage is that the order of the group is almost guaranteed by the total number of messages actually transmitted. 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 presents 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.

Table 1: Computation Overhead of ECDKG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Multiplication</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>((t + 1, (t + 1) \log_2(t + 1)))</td>
<td>((t, t))</td>
</tr>
<tr>
<td>KV</td>
<td>((n(t + 1) \log_2(t) + n - 1, nt \log_2(t)))</td>
<td>(((n - 1)(t + 1), 2t + n - 1))</td>
</tr>
<tr>
<td>KC</td>
<td>((0, 0))</td>
<td></td>
</tr>
<tr>
<td>KN</td>
<td>((0, 0))</td>
<td></td>
</tr>
<tr>
<td>KG</td>
<td>((1, 0))</td>
<td></td>
</tr>
<tr>
<td>MiDS-S</td>
<td>((2t, n + C_{hm}))</td>
<td>((t, 1 + C_{ha}))</td>
</tr>
<tr>
<td>MiDS-V</td>
<td>((2, C_{hm}))</td>
<td>((1, C_{ha}))</td>
</tr>
</tbody>
</table>

Table 2: Communication Overhead of ECDKG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reception</th>
<th>Transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>0</td>
<td>((n - 1, 1))</td>
</tr>
<tr>
<td>KV</td>
<td>(n - 1)</td>
<td>((0, 3t))</td>
</tr>
<tr>
<td>KC</td>
<td>(2t + n - 1)</td>
<td>((0, 0))</td>
</tr>
<tr>
<td>KN</td>
<td>(n - 1)</td>
<td>((0, 1))</td>
</tr>
<tr>
<td>KG</td>
<td>(n - 1)</td>
<td>((0, 1))</td>
</tr>
<tr>
<td>MiDS-S</td>
<td>1</td>
<td>((0, n - 1))</td>
</tr>
<tr>
<td>MiDS-V</td>
<td>0</td>
<td>((0, 0))</td>
</tr>
</tbody>
</table>

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: Worst-Case Memory Requirement of ECDKG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst-Case Memory</th>
<th>Group Size n</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>((t + 1) \log_2(p) + C)</td>
<td>(n &gt; t)</td>
</tr>
<tr>
<td>KV</td>
<td>((n - 1) \log_2(p) + C)</td>
<td>(n &gt; t)</td>
</tr>
<tr>
<td>KC</td>
<td>((n - 1) \log_2(p) + C)</td>
<td>(n &gt; 3t)</td>
</tr>
<tr>
<td>KN</td>
<td>((n - 1) \log_2(p))</td>
<td>(n &gt; t)</td>
</tr>
<tr>
<td>KG</td>
<td>((n + 1) \log_2(p) + C)</td>
<td>(n &gt; t)</td>
</tr>
<tr>
<td>MiDS-S</td>
<td>(2 \log_2(p) + C)</td>
<td>(n &gt; t)</td>
</tr>
<tr>
<td>MiDS-V</td>
<td>(2 \log_2(p) + C)</td>
<td>(n &gt; t)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Curve</th>
<th>(\log_2(p))</th>
<th>RT-20</th>
<th>RT-100</th>
<th>RT-500</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-163</td>
<td>163</td>
<td>93 ms</td>
<td>198 ms</td>
<td>800 s</td>
</tr>
<tr>
<td>K-233</td>
<td>232</td>
<td>156 ms</td>
<td>261 ms</td>
<td>860 s</td>
</tr>
<tr>
<td>K-283</td>
<td>281</td>
<td>209 ms</td>
<td>314 ms</td>
<td>910 s</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Curve</th>
<th>(\log_2(p))</th>
<th>Signing Time</th>
<th>Verifying Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-163</td>
<td>163</td>
<td>19450 (\mu s)</td>
<td>3000 (\mu s)</td>
</tr>
<tr>
<td>K-233</td>
<td>232</td>
<td>32430 (\mu s)</td>
<td>3500 (\mu s)</td>
</tr>
<tr>
<td>K-283</td>
<td>281</td>
<td>43430 (\mu s)</td>
<td>3900 (\mu s)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Curve</th>
<th>(\log_2(p))</th>
<th>Signing Time</th>
<th>Verifying Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-163</td>
<td>163</td>
<td>20950 (\mu s)</td>
<td>14250 (\mu s)</td>
</tr>
<tr>
<td>K-233</td>
<td>231</td>
<td>35930 (\mu s)</td>
<td>20300 (\mu s)</td>
</tr>
<tr>
<td>K-283</td>
<td>281</td>
<td>48600 (\mu s)</td>
<td>25520 (\mu s)</td>
</tr>
</tbody>
</table>

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


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: \( \text{GF}(p) \) for a prime and \( \text{GF}(2^n) \) for 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 \( \text{GF}(2^n) \), 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 \( a, b \in \text{GF}(2^n) \), 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 factors 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 \( \text{GF}(q) \) 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 \( \text{GF}(q) \). For the prime field \( \text{GF}(p) \), we need to avoid the curve whose cardinality divides \( p^k - 1 \) for a small \( k \) \((k \leq 20)\), since a Weil pairing [26] (MOV attack) or Tate pairing [35, 12] can reduce ECDLP to DLP in a 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 purposes, we can check up to 20 and it is sufficient since the discrete logarithm over \( \text{GF}(p^k) \) 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, \( |E/\text{GF}(p)| = p \), and ECDLP can be reduced to DLP in an additive group [40] [41] [36].

4) Curves over field \( \text{GF}(2^n) \), \( m \) is composite. Weil descent [13] GHS attack might be used to solve the ECDLP over a binary field. Weil descent reduces ECDLP in \( \text{GF}(2^n) \) 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 [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 \( \text{GF}(p) \) or \( \text{GF}(2^n) \), in order to avoid MOV attack, \( q \) should not divide \( q^a - 1 \) 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 \( \text{GF}(2^n) \), 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. 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 \(\tau\)-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 \(\tau\)-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., \(-x, -y\) in the prime field \(GF(p)\), or \(-x, -y = (x, x + y)\) (because \(-x, x + y = (x, x + y + x) = (y, y)\) in \(GF(2^n)\)). 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: \(\sum_{i=0}^{n-1} c_i \theta^i\), where \(c_i \in GF(2)\). It can also be represented as a vector of dimension \(n\) using following basis:
\[
\theta, \theta^2, \ldots, \theta^{2^{n-1}}
\]
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 \(t\):
\[
f_i(z) = a_{i0} + a_{i1} z + \ldots + a_{it} z^t
\]
then \(p_i\) broadcasts
\[
A_{ik} = g^{a_{ik}} \mod p
\]
for \(k = 0, 1, \ldots, t\). Each \(p_i\) computes the shares \(s_{ij} = f_i(j) \mod q\), for \(j = 1, 2, \ldots, n\) 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 \(i = 1, 2, \ldots, n\):
\[
g^{s_{ij}} = \prod_{k=0}^{t} (A_{ik})^{j_k} \mod p \quad (5)
\]
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
\[
y = \prod_{i \in Q} A_{i0} \mod p
\]
The public verification values are computed as
\[
A_k = \prod_{i \in Q} A_{ik} \mod p
\]
for \(k = 1, 2, \ldots, t\). Each player \(p_j\) sets his share of the secret as
\[
x_j = \sum_{i \in Q} s_{ij} \mod q
\]
The secret shared value \(x\) itself is not computed by any party, but it is equal to
\[
x = \sum_{i \in Q} a_{i0} \mod q