An Elliptic Curve On-line\Off-line Digital Signature Scheme for Internet of Things

Hisham Dahshan
hdahshan1@gmail.com

Abstract—The communication model of Internet of Things (IOT) includes networks infrastructure (e.g. ultra-wideband networks, 3G and 4G networks). It also includes the adoption of IPv6 in order to provide a unique IP address to any entity involved in the network. It also comprises technologies that allow the location and identification of physical objects (e.g. RFID). There are also some other technologies that influence on the successful development of IoT applications. These technologies are computer vision, biometric systems, robotics, and others. In this paper, an elliptic curve on-line/off-line threshold digital signature scheme for Internet of Things is presented. In the proposed scheme, there are two phases for computing the digital signature of a message: off-line phase and on-line phase. The majority of the digital signature computation is performed off-line which saves power and time. We also prove that our proposed schemes have achieved the desired security requirements. Extensive simulations show that the proposed scheme is more efficient and robust compared with the one-phase elliptic curve threshold digital signature scheme.

Index Term—IoT, Digital Signature, Elliptic Curve Cryptography

I. INTRODUCTION

THE Internet of Things (IOT) is the interconnection of heterogeneous network entities, such as laser scanners, radio frequency identification (RFID), global positioning systems, and information sensing devices and so on [1]. With the development of communication technology, IOT has been widely used in various fields like military surveillance, medical care, industrial control and so on [2]. However, because of the significant resource constraints of IOT entities, they are vulnerable to numerous kinds of attacks. Providing security services such as authentication, integrity, and non-repudiation for IOT is challenging because of the lack of infrastructure, the lack of a priori trust among network entities and, the limited computing power. Due to the wireless nature of IOT, malicious nodes can easily inject fake messages or alter the legitimate messages during multi-hop forwarding. In such cases, IOT applications need to rely on authentication mechanisms to assert the integrity of the legitimate data messages. Thus, an efficient and robust digital signature mechanism is essential to provide IOT with the required security services. The idea of threshold cryptography [3] is to distribute a secret and the required computations among a group of users in order to prevent a single point of failure or misbehaviour. In the distributed environment of IOT, threshold-multi-signature [4] gives a promising solution for the above problems. In threshold-multi-signature scheme, a threshold t or more entities are required to cooperatively generate a digital signature of a message, instead of generating a digital signature by a single entity. Because of the absence of a trusted administration authority in the distributed environment of IOT, a distributed key generation protocol (DKG) for discrete logarithm-based threshold cryptosystems is required. In Pedersen DKG protocol [5], a group of n players run parallel executions of Feldman’s verifiable secret sharing (VSS) protocol [6] in which each player Pi acts as a dealer of a random secret zi. The DKG protocol proposed by Gennaro et al in [7] guarantees a uniform distribution of the generated keys which was a major drawback of Pedersen’s scheme [5]. Distributed key generation protocols DKG 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) [8, 9], 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.

Table I gives approximate key sizes for symmetric cryptosystems, ECC cryptosystems, and DH/DSA/RSA cryptosystems. These estimates are based on the running times of the best algorithms known today. Thus, for example, when a 192-bit symmetric key security is required, we can use either 409-bit ECC or 7680-bit DH/DSA/RSA.

The on-line/off-line digital signature scheme was firstly introduced by Even et al in [10, 11]. The basic idea of their scheme is to split the signature generation algorithm into two phases: the off-line phase and the on-line phase. For efficient performance, the majority of signature computation is performed off-line independently of the message to be signed. The results of the off-line computation are stored to be used
during the on-line phase. The generation of the required signature is performed efficiently during the on-line phase by using the stored computation of the off-line phase. Based on Even et al scheme, Shamir and Tauman [12] proposed an improved on-line/off-line signature scheme by introducing the hash-sign-switch paradigm. Their scheme is based on an ordinary digital signature scheme, in which the size of the signature is reduced, compared with Even et al scheme. Shamir and Tauman in [12] suggest that some signature schemes such as Schnorr, Fiat-Shamir, and ElGamal signature schemes [13, 14, 15] can be efficiently partitioned into on-line/off-line schemes. A generic threshold on-line/off-line digital signature scheme was introduced by Crutchfield et al in [16] which largely reduced the size of the generated signature by using the chameleon hash function but it needs 2t + 1 out of n players to cooperate in generating a valid signature. In [17], the authors present a certificateless Online/Offline Signature (CLOOS) Scheme and give a tight security reduction to the Gap Diffie-Hellman problem in the random oracle model. This scheme suffers from the high computation overhead compared to other schemes. In [18], J. Kar proposes a secure and efficient online/off-line signature scheme for WSN. The scheme is secure against existential forgery on chosen message attack in random oracle model under the assumption of Computational Diffie-Hellman Problem (CDH) is hard. However, their paper lack comparisons with other schemes regarding the computation overhead. In this paper an elliptic curve threshold on-line/off-line signature scheme for IOT is presented. To generate a valid signature in our scheme, it needs a collaboration of only t + 1 out of n players. The scheme is suitable for distributed networks such as IOT networks where there is no trusted administration authority available. Simulation results show that our proposed scheme is more efficient compared with the one phase elliptic curve threshold signature scheme in terms of memory size and timing. The rest of the paper is organized as follows: The system model and definitions are presented in Section II. Section III provides the scheme description of our proposed scheme. The security analysis of the proposed scheme is presented in Section IV. Section V provides the efficiency analysis of our proposed scheme and then the simulation results of the proposed scheme is presented in Section VI. Finally Section VII concludes the paper.

II. SYSTEM MODEL

In this section, the communication model and some definitions used in our proposed scheme are presented.

A. Communication Model

We assume that there are two kinds of communication channels available for each player in our proposed scheme, a point to point private channel and a broadcast channel. Private channels can be implemented with standard cryptographic techniques and any two players can communicate via their respective private channel. Broadcast channel enables each player to send an identical message to all players, without violating the consistency requirement. All messages sent on both channel are sent with time to live constraints, i.e. they are received by their recipients within a fixed time bound.

B. Adversary Model

We assume that the group of players P that participate in the security protocol have a cardinality equal to n and each player $p_i \in P$ is identified by its unique identity $p_i$, where $p_i \in GF^*(q)$. We assume that there is a static adversary A that can corrupt up to t of the players in P before beginning the threshold signature scheme. Adversaries can be categorized into two main types: static or adaptive. Static adversary decides which player to break into at the beginning of the protocol. Adaptive adversaries choose the corrupted players during the protocol run time. Three types of adversaries are considered in our proposed scheme:

- Eavesdropping adversary: passively listen to the channel and can hear all broadcasted messages.
- Halting adversary: can cause corrupted players to stop sending messages or prevent them from responding to received messages during a protocol run time.
- Static adversary: is another type of the eavesdropping adversary that can cause corrupted players to deviate from the specified protocol.

C. Chameleon Hash Function

A chameleon hashing function $CH_{HK}(m, r)$ [19], is a trapdoor collision resistant hash function, which is associated with a public/private key pair $(HK, TK)$, and has the following properties:

- Collision Resistance: Given any probabilistic polynomial time algorithm A that only knows the public key HK, the function is collision resistant, if it is infeasible to find two pairs $(m_1, r_1), (m_2, r_2)$ such that $m_1 \neq m_2$ and $CH_{HK}(m_1, r_1) = CH_{HK}(m_2, r_2)$.
- There exists a polynomial time algorithm A such that when input the pair $(m_1, r_1)$, and a message $m_2$, and with the knowledge of the trapdoor information TK, then A outputs $r_2$ such that $CH_{HK}(m_1, r_1) = CH_{HK}(m_2, r_2)$
- Uniform probability distribution: If the inputs of $CH_{HK}(m, r)$ are distributed uniformly, then the value computed using the private key TK is computationally indistinguishable from uniform distribution.

III. SCHEME DESCRIPTION

Our proposed scheme consists of the following algorithms: elliptic curve distributed key generation algorithm, elliptic curve threshold signature generation algorithm, on-line/off-line signing algorithm, and signature verification algorithm.

A. Elliptic Curve Distributed Key Generation Algorithm (ECDKG)

In this algorithm, a group of IOT entities $p_i$ ($i = 1, 2, \ldots, n$) cooperate to generate a master public/private key pair $PK/SK$. The public key PK is made known to all entities that take part
in the distributed key generation algorithm, while the private key SK is not known by any single entity. Each entity gets a private key share SK. No less than t + 1 entities can reconstruct the master private key SK. The DKG protocol is fully distributed, where no trusted dealer is required, and is based on the modified Pedersen dlog-based DKG protocol [2].

The algorithm includes the following steps:

- **step 1**: each entity $p_i$ chooses two random polynomials $f_i(z), f_j(z)$ of degree $t$ as follows:
  
  $f_i(z) = a_0 + a_1 z + \cdots + a_t z^t$, 
  $f_j(z) = b_0 + b_1 z + \cdots + b_t z^t$

  where $a_i, b_i \in GF(q)$, and $(0 \leq l \leq t)$.

  Let $d_{il} = f_i(0)$ is the private key of the entity $p_i$.

- **step 2**: compute the shares $s_{ij} = f_i(p_j) mod p$.

- **step 3**: compute and broadcast the following public values:

  \[
  PV_{ij} = (a_i \odot G) \oplus (b_i \odot G) \quad \text{where} \quad \odot \text{ is the point multiplication by a scalar operator.}
  \]

- **step 4**: each entity $p_i$ verifies the shares it received from other entities. $p_i$ checks for $i = 1, \cdots, n$ if

  \[
  s_{ij} \oplus (s_{ij} \odot G) = \sum_{l=0}^{t} (p_i l \odot PV_{il})
  \]

  where $\sum \oplus$ is the point summation under the point add operation $\oplus$. If the verification fails, $p_i$ broadcasts a complaint against $p_i$.

- **step 5**: each entity $p_i$ that received a complaint from another entity $p_j$ broadcasts the values $s_{ij}, s_{ij}'$ that satisfy (1).

- **step 6**: entity $p_i$ removes entity $p_j$ from its trusted group if it received more that $t$ complaints against $p_j$ in step 4, or if the reply of the entity $p_j$ in step 5 does not satisfy (1).

- **step 7**: each entity $p_t$ computes its public key $A_{k_{i0}} = a_{k_{i0}} \odot G$ and broadcasts $A_{k_{i0}}$.

- **step 8**: each entity $p_t$ broadcasts the public share $w_t = s_{ij} \odot G$, where $s_{ij} = \sum_{l=1}^{t} s_{ij}$ is the master secret polynomial share for the entity $p_t$.

- **step 9**: each entity $p_t$ receives $A_{k_{i0}} (j = 1, \cdots, n)$ and computes the master public key:

  \[
  PK = \sum_{j=1}^{n} A_{k_{i0}}
  \]

Only a group of $t + 1$ entities can deduce the master secret polynomial by doing the following Lagrange interpolation:

\[
 f(x) = \sum_{j=0}^{t} s_{ij} \prod_{1 \leq j \leq t, j \neq i} \frac{x - j}{j - i} \mod p
\]

and by substituting $x = 0$ in the secret polynomial, the master secret key SK be reconstructed by the following formula:

\[
 SK = \sum_{j=0}^{t} s_{ij} \prod_{1 \leq j \leq t, j \neq i} \frac{x - j}{j - i} \mod p
\]

**B. Elliptic Curve Threshold Signature Generation Algorithm (ECTSig(m, PK, p_i, s_j))**

In this algorithm, a group of $t + 1$ entities cooperate in generating a signature of a message $m$. We call the $t + 1$ entities that involved in the generation of the threshold signature as the signing group. This algorithm is based on the threshold version of Schnorr’s signature scheme presented in [2]. The generated signature can be verified using the master public key $PK$. No subset of $t$ entities can generate a valid master signature. The algorithm can be summarized in the following steps:

- **step 1**: each of the $t + 1$ entities $p_j \in P$, where $P$ is the signing group, performs an instance of the distributed key generation algorithm mentioned in subsection III-A to generate a one-time secret share $k_j$ of the secret $k$, the corresponding one-time public share $r_j = k_j \odot G$, and the one-time public value $r$. 

- **step 2**: $p_t$ computes $c = \text{Hash}(m, X)$, where $X$ is the $x$-coordinate of the one-time public value $r$.

- **step 3**: $p_t$ generates and broadcasts $sig_t$, where

  \[
  sig_t = k_t + c \prod_{j=1}^{t+1} (p_j(p_j - p_i))s_i
  \]

  where $s_i = \sum_{i=1}^{n} s_{ij}$, and $n$ is the total number of entities that hold a share of the master shared secret.

- **step 4**: $p_t$ verifies the partial signature received from $p_i$ by checking if

  \[
  sig_t \odot G = r_j - c \prod_{j=1}^{t+1} (p_j(p_j - p_i))w_j
  \]

  where $w_j = s_{ij} \odot G$ is a public value.

- **step 5**: $p_t$ collects another $t$ partial signature from $t$ entities $\in P$ and add them together to form the master signature $(c, sig)$, where $sig = \sum (t+1) sig_i$.

- **step 6**: any node in the network can verify the generated master signature by using the master public key $PK$ as follows:

  1. compute $r = (sig \odot G) \odot ((-c) \odot PK) mod p$.
  2. check if $c = \text{Hash}(m, X)$, then the signature is correct.

**C. On-line/Off-line Signing Algorithm**

Our on-line/off-line signing algorithm includes the following schemes:

1) **Threshold Key Generation**

1. Run the ECDKG algorithm to generate the master public/private key pair $PK/SK$ of the IOT network. The master public key $PK$ is known by all entities in the IOT network while SK is not known by any single entity, but on the other hand each entity $p_i$ has a share $SK_i$ of the master private key $SK$. 

[1] IJEN S Vol:16 No:03 44
2. Run an instance of the ECDKKG algorithm to generate a one-time secret $d$, and the corresponding one-time public value $z$ (where $z = d \mathcal{O}(G)$). The one-time public value $z$ is known to all entities in the network, while each entity acquires only a share $d_i$ of the one-time secret $d$. Each entity $p_i$ broadcasts its public share $d_i \mathcal{O}(G)$.

2) Off-line Phase (Done per message)

The off-line phase can be implemented either during IOT entities manufacturing process or as a background computation whenever the entities is connected to power before network deployment or during low traffic in distributed systems which have bursty traffic. The off-line phase can be summarized as follows:

a) Chameleon Hash Function Generation:

1. Run the ECDKKG algorithm to generate a secret $\mu$ of degree $t$ polynomial $f(x)$ and the corresponding public value $\mu \mathcal{O}(G)$, where $\mu \in \mathbb{Z}_p$ and $f(0) = \mu$. Each entity $p_i$ acquires only a share $\mu_i$ of the shared secret $\mu$ and broadcasts its public share $\mu_i \mathcal{O}(G)$.

2. Run the ECDKKG algorithm to generate a secret $v$ of degree $t$ polynomial $f(x)$ and the corresponding public value $v \mathcal{O}(G)$, where $z = d \mathcal{O}(G)$ is the public value generated in the off-line threshold key generation algorithm, $v \in \mathbb{Z}_p$, and $f(0) = v$. Each entity $p_i$ acquires only a share $v_i$ of the shared secret $v$ and broadcasts its public share $v_i \mathcal{O}(G)$.

3. Run the ECDKKG algorithm to generate the shared secret $\gamma$ of degree $t$ polynomial $f(x)$, and the corresponding public value $\gamma \mathcal{O}(G)$, where $\gamma \in \mathbb{Z}_p$, and $f(0) = 0$. Each entity $p_i$ acquires only a share $\gamma_i$.

4. Calculate the chameleon hash function as follows:

5. \( CH_{HK}(v, \mu) = (\mu \mathcal{O}(G)) \oplus (v \mathcal{O}(G)) \)

b) Off-line Signature Generation:

1. Use the ECTSig algorithm to generate a threshold signature for the chameleon hash value $CH_{HK}(\mu, v)$ generated in step 4 of the chameleon hash function generation algorithm $ECTSig(CH_{HK}(\mu, v), PK, p, s_i)$.

2. Each entity $p_i$ broadcasts its generated partial signature $ECTSig(CH_{HK}(\mu, v), PK, p, s_i)$.

3. On-line phase (Done per message): The on-line phase includes the on-line signature generation algorithm and the on-line signature verification algorithm.

a) On-line Signature Generation Algorithm:

In this algorithm a group of $t + 1$ entities \( \{p_r, p_{r+1}, \ldots, p_{r+t+1} \} \) cooperate to generate a signature for a message $m \in \mathbb{Z}_p$, where $p_r \in P_r$ and $P \subset P$ as follows:

1) Each $p_i \in P$ generates and broadcasts three trapdoor collision shares $\beta_i, \alpha_i, \delta_i$ as follows:

\( \beta_i = \mu - md_i \)

\( \alpha_i = d_i + \gamma_i \)

\( \delta_i = v_i + \gamma_i \)

2) Each $p_i \in P'$ receives the trapdoor collision shares $\beta_i, \alpha_i$, and $\delta_i$ from other $t$ entities in $P$ and do the following verifications for the received trapdoor collision shares from each $p_j \in P'$:

a) $p_i$ checks if

\( \beta_j \mathcal{O}(G) = (\mu \mathcal{O}(G)) \oplus (d_j \mathcal{O}(G) \ominus m) \)

where $\mu \mathcal{O}(G)$ and $d_j \mathcal{O}(G)$ are public values and $\ominus$ is the point subtract operation over $E$.

b) $p_i$ checks if

\( \alpha_j \mathcal{O}(G) = (d_j \mathcal{O}(G)) \oplus (\gamma_j \mathcal{O}(G)) \)

where $d_j \mathcal{O}(G)$ and $\gamma_j \mathcal{O}(G)$ are public values.

c) $p_i$ checks if

\( \delta_j \mathcal{O}(G) = (v_j \mathcal{O}(G)) \oplus (\gamma_j \mathcal{O}(G)) \)

where $v_j \mathcal{O}(G)$ and $\gamma_j \mathcal{O}(G)$ are public values.

If any of formulas 2, 3, or 4 fails, $p_i$ broadcasts a complains against $p_j$. If there are at least $t + 1$ complaints against $p_j$, $p_j$ must be compromised since with the existence of $t$ malicious entities in the network, there can be at most $t$ faked complaints. In this case $p_i$ is removed from $P$ and replaced with a new entity.

3) Each $p_j \in P'$ collects another $t$ trapdoor collision shares $\beta_j, \alpha_j, \delta_j$ from other $t$ entities in $P$ to compute the trapdoor collision $\mu$ as follows:

\( \mu' = \left( \sum_{p_j \in P'} \beta_j \nu_1 \right) + \left( \sum_{p_j \in P'} \alpha_j \nu_1 \right) \left( \sum_{p_j \in P'} \delta_j \nu_1 \right) \)

\( \equiv \mu + dv - md \pmod{p} \)

is the Lagrange interpolation coefficient of the entity $p_i$.

The generated on-line signature of the message $m$ is:

\( (c, sig, m, \mu) \), where

\( sig = (ECTSig(CH_{HK}(\mu, v), PK, p, s_i)) \)

b) On-line Signature Verification Algorithm (Done per message): Any entity in the network can verify the received signature \( (c, sig, m, \mu) \) for the message $m$ as follows:

1) Compute the chameleon hash function by using $\mu$ and the message $m$ as follows:

\( CH_{HK}(m, \mu) = (\mu \mathcal{O}(G)) \oplus (m \mathcal{O}(G)) \)

2) Compute \( r = (\text{sig} \mathcal{O}(G)) \oplus ((-c) \mathcal{O}(PK)) \mod p \).

3) Check if $c = \text{Hash}(CH_{HK}(m, \mu), X_1)$, then accept the signature.

IV. SECURITY ANALYSIS

In this section, a discussion on the security of the proposed scheme is given. We prove the security of the On-line/Off-line threshold signature algorithm.

A. Completeness

The proposed on-line/off-line threshold signature scheme is complete.
In this section, we present an efficiency analysis of our proposed scheme with that of the underlying threshold signature scheme ECTSig. We denote by $M$ the computation cost of point multiplication by a scalar in $G$, by D a point add operation in $G$, by $H$ the modular multiplication in $GF(q)$, by T the add operation in $GF(q)$. The subtraction operation is considered the same computation cost as the add operation in this analysis. The computation cost of the one-way hash function $c = \text{Hash}(m, X_i)$ is omitted in this analysis. The computation complexity of the ECDKG protocol is referred to as DKG in table II.

A. Efficiency of ECTSig

In this Subsection, we introduce a brief analysis of the elliptic curve threshold signature scheme ECTSig which is the underlying threshold signature algorithm of our on-line/off-line threshold signature scheme. A summary of the complexity analysis is shown in Table II.

a) Threshold Key Generation: In ECTSig, the ECDKG is run only once to get the network private/public key pairs $SK_i/PK_i$.

b) Off-line Phase: There is no off-line phase in ECTSig.

c) On-line Phase: In the on-line phase of ECTSig, the signing group runs ECDKG once to generate the one-time secret $k$ and the corresponding one-time public value $r$. The on-line phase of ECTSig includes also the computation of Lagrange interpolation which is $2t$ multiplication and $2t$ subtraction in $GF(q)$ and another multiplication by the share $s_i$, so we have $(2t+1)H$. There are also $(2t+1)T$ add operation in $GF(q)$ in the on-line phase of ECTSig. The total computation cost of the on-line phase of ECTSig algorithm is $ECDKG+(2t+1)(H+T)$.

d) On-line Reconstruction: In the on-line Reconstruction of ECTSig, there are $2M$ point multiplication operation for each entity, so we have $2(t+1)M$ for the signing group plus $(t+1)D$ point add operation for the signing group, we have also $2t$ modular multiplication and $2t$ subtraction in $GF(q)$ for Lagrange interpolation for each entity, so we have a total computation cost of $(t+1)[2M + D + 2(tH + T)]$ in the one-line reconstruction process of the ECTSig algorithm.

e) On-line Verification: The on-line verification of ECTSig includes two point multiplication by a scalar operations $2M$, and one point add operation D in G.

f) Signature Share Verification: The signature share verification process includes two point multiplication by a scalar operation $2M$, one point add operation D, $2t$ multiplication and $2t$ subtraction in $GF(q)$ for each entity, so we have a total of $(t+1)[2M + D + 2t(H+T)]$.
Table II  
A comparison between ECTSig and our proposed scheme

<table>
<thead>
<tr>
<th>Threshold Sig. Scheme</th>
<th>ECTSig Scheme</th>
<th>Our On-line/Off-line Signature Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold Key Generation</td>
<td>ECDKG</td>
<td>2ECDKG</td>
</tr>
<tr>
<td>Off-line Phase</td>
<td>None</td>
<td>3ECDKG + 2M + D + ECTSig</td>
</tr>
<tr>
<td>On-line Phase</td>
<td>ECDKG + (2t + 1)H + T</td>
<td>(H + 3 T)</td>
</tr>
<tr>
<td>On-line Reconstruction</td>
<td>(t + 1)[2M + D + 2n(H + T)]</td>
<td>(4r^2 + 6r^2 + 2j)(H + T) + 2M + D</td>
</tr>
<tr>
<td>On-line Verification</td>
<td>2M + D</td>
<td>2M + D</td>
</tr>
<tr>
<td>Signature Share Verification</td>
<td>(t + 1)[2M + D + 2n(H + T)]</td>
<td>10(M + 3jD)</td>
</tr>
</tbody>
</table>

B. Efficiency of our Proposed Scheme

a) Threshold Key Generation: The proposed on-line/off-line threshold signature scheme runs the elliptic curve distributed key generation protocol once to generate the public/private key pair PK/SK, and another time to generate the one-time secret d, and the corresponding one-time public value z.

b) Off-line Phase: In the off-line phase, the proposed scheme runs ECDKG protocol three times to generate the shared secrets μ, ν, and γ which are required for the generation of the chameleon hash function. The generation of the chameleon hash function requires 2 point multiplication by a scalar operations and one point add operation in G. Finally, the proposed scheme runs ECTSig algorithm to generate a threshold signature for the chameleon hash function. The total computation cost of the off-line phase is 3ECDKG + 2M + D + ECTSig.

c) On-line Phase: In the on-line phase of our proposed scheme, we have one modular multiplication H and three addition operations T in GF(q) for the computations of αi, βi, and δi for each entity with a total cost of (H + 3 T).

d) On-line Reconstruction: The computation of μ requires the calculation of the sum of Lagrange interpolation multiplied by βi and the sum of Lagrange interpolation multiplied by ai and this sum is multiplied by the sum of Lagrange interpolation multiplied by δi which is 2n(H + T)(t + 1) + 4r(H + T)(t + 1) = (4r^2 + 6r^2 + 2j)(H + T) + 2M + D. The computation of the chameleon hash function CHsk(m, μ) requires 2M + D operations in G, so the total computation cost of on-line reconstruction is (4r^2 + 6r^2 + 2j)(H + T) + 2M + D.

e) On-line Verification: The on-line verification process requires two point multiplication by a scalar in G, and one point add operation in G for the calculation of the chameleon hash function On-line Verification CHsk(m, μ).

f) Signature Share Verification: The signature share verification process requires four point multiplication by a scalar operation and one point subtraction operation in G for the computation of formula 2 with a total cost of 4M + D. The computation of formula 3 requires three point multiplication by a scalar operation and one point addition operation in G with a total computation cost of 3M + D. The computation of formula 4 requires three point multiplication by a scalar operation and one point addition operation in G with a total computation cost of 3M + D. We conclude from above that the total computation cost of the signature share verification for each entity shares is 10M + 3D, and it is performed for t entity shares, so the total computation cost for the signature verification process is 10tM + 3tD. It is obvious from the analysis of the signature share verification process that it is very expensive in terms of computation cost as it requires 10r point multiplication by a scalar operations and 3t point additions which are the most costly operations in G. Consequently, we limit the execution of the signature share verification process to the case when the on-line verification fails.

VI. SIMULATION RESULTS

In this section, we evaluate the performance of our proposed elliptic curve on-line/off-line threshold digital signature scheme for IOT. The growing difference in key bit length for equivalent security levels as shown in table (I) accounts for the performance advantages to be obtained from using ECC. ECC also gives promising results for substituting RSA/DH/DSS in public key cryptographic protocols. One of the major factors that affect the performance of ECDLP based cryptographic protocols is the domain parameters and the elliptic curve selected for implementation. The fundamental operation underlying ECC is point multiplication, which is defined over finite field operations. Elliptic curves are defined over either prime fields GF(p) or binary fields GF(2^n). In the performance evaluation of our proposed scheme, we consider only prime fields GF(p) since binary field arithmetic, is insufficiently supported in PARI/GP [21] and would thus lead to lower performance. On a laptop with an Intel core i3 2.13GHz processor and 3GB memory, PARI/GP [21] is used to evaluate the performance of our proposed scheme. The prime elliptic curve over GF(p) is defined by the equation y^2 = (x^3 + ax + b) mod p where a and b ∈ GF(p) for p a prime and satisfy 4a^3 + 27b^2 = 0 (mod p).

The domain parameters used in our implementation are as follows:

A field size q which defines the underlying finite field GF(q), where q > 3 is a prime number; two field elements a and b in GF(q) which define the equation of the elliptic curve E: y^2 = (x^3 + ax + b) mod p; two field elements xG and yG in GF(q), which define a point G = (xG, yG) of prime order on E; the order p of the point G (it must be the case that p > 2^160); and the cofactor h = (#E(GF(q))/p). The performance evaluation of the proposed scheme will be given in terms of the minimum threshold t + 1 nodes (out of the total n entities or session nodes) required to collaboratively recover the CA or the session secret keys or sign an arbitrary message m. All results are the average of 10 runs and the total number of mobile nodes in the network is set to 50 mobile nodes. The performance of the proposed scheme is evaluated for three different key sizes: 192 bits, 239 bits, and 256 bits [22]. Values that remain constant between different scheme runs...
(for example, the inner parts of the Lagrange coefficients) can be precomputed and are therefore not included in the evaluation. The end-to-end delay caused by buffering during route discovery latency, queueing at the interface queue, retransmission delays at the MAC, and propagation and transfer times are not included in the timing analysis and will be considered in our future work. For simplicity reason, we present simulation results in three tables for each key size: one table for the key generation timing, one table for the off-line phase timing, and one table table for the on-line phase timing. The on-line phase timing includes also the on-line reconstruction and the on-line verification timing. Tables III, IV, and V show the key generation timing for the 192-bits, 239-bits, and 256-bits prime curves respectively. They show that the key generation timing in our proposed scheme exceeds that of ECTSig because our scheme executes the ECDKG algorithm two times in this phase while ECTSig does only one run of ECDKG in the key generation phase. It is obvious also that the timing increase with increasing the threshold due to increasing the computation with increasing the number of entities involved in the signing group. Tables VI, VII, and VIII show the timing results of the off-line phase in our proposed on-line/off-line threshold signature scheme only as the ECTSig does not have an off-line phase. It is obvious from the timing results of the off-line phase of our proposed scheme that the computation time increases with increasing the threshold \((t+1)\). The increase in timing with increasing the threshold \((t+1)\) in the off-line phase occurs because the off-line phase executes the ECDKG algorithm three times and the computation of ECDKG depends on the threshold \((t+1)\). The timing results of the off-line phase show also and increase in timing with increasing the key size. Tables IX, X, and XI show the timing results of the on-line phase in ECTSig versus that of our proposed scheme. The on-line phase timing results of our proposed scheme is very low compared with that of ECTSig which reflects the gained efficiency due to partitioning the signature algorithm into off-line phase and on-line phase. The results of the off-line phase show that the timing increases with increasing the number of entities taking part in the signing procedures. They show also that the timing increases with increasing the key size as a result of increasing the complexity.

### Table III

<table>
<thead>
<tr>
<th>Threshold ((t+1))</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>479.6</td>
<td>560</td>
</tr>
<tr>
<td>10</td>
<td>687.17</td>
<td>874</td>
</tr>
<tr>
<td>20</td>
<td>1391.32</td>
<td>1976</td>
</tr>
<tr>
<td>30</td>
<td>2477.06</td>
<td>3883</td>
</tr>
<tr>
<td>40</td>
<td>3894.89</td>
<td>6997</td>
</tr>
<tr>
<td>50</td>
<td>5781.53</td>
<td>11498</td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Threshold ((t+1))</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>480.24</td>
<td>846</td>
</tr>
<tr>
<td>10</td>
<td>690.9</td>
<td>1262.6</td>
</tr>
<tr>
<td>20</td>
<td>1393.07</td>
<td>2679.3</td>
</tr>
<tr>
<td>30</td>
<td>2447.39</td>
<td>5179.37</td>
</tr>
<tr>
<td>40</td>
<td>3881.86</td>
<td>9227.04</td>
</tr>
<tr>
<td>50</td>
<td>5714.76</td>
<td>15852.02</td>
</tr>
</tbody>
</table>

### Table V

<table>
<thead>
<tr>
<th>Threshold ((t+1))</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>481.16</td>
<td>926.34</td>
</tr>
<tr>
<td>10</td>
<td>688.73</td>
<td>1373.83</td>
</tr>
<tr>
<td>20</td>
<td>1386.21</td>
<td>2838.83</td>
</tr>
<tr>
<td>30</td>
<td>2442.03</td>
<td>5678.67</td>
</tr>
<tr>
<td>40</td>
<td>3884.32</td>
<td>10104.04</td>
</tr>
<tr>
<td>50</td>
<td>5800.67</td>
<td>16553.56</td>
</tr>
</tbody>
</table>

### Table VI

<table>
<thead>
<tr>
<th>Threshold ((t+1))</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-</td>
<td>1992</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>2935</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>6132</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>11223</td>
</tr>
<tr>
<td>40</td>
<td>-</td>
<td>18668</td>
</tr>
<tr>
<td>50</td>
<td>-</td>
<td>29088</td>
</tr>
</tbody>
</table>

### Table VII

<table>
<thead>
<tr>
<th>Threshold ((t+1))</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-</td>
<td>2494</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>3721</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>7732.45</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>14279.61</td>
</tr>
<tr>
<td>40</td>
<td>-</td>
<td>23666.96</td>
</tr>
<tr>
<td>50</td>
<td>-</td>
<td>39135.87</td>
</tr>
</tbody>
</table>

### Table VIII

<table>
<thead>
<tr>
<th>Threshold ((t+1))</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-</td>
<td>2587.07</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>3803.84</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>7948.75</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>15775.54</td>
</tr>
<tr>
<td>40</td>
<td>-</td>
<td>25988.29</td>
</tr>
<tr>
<td>50</td>
<td>-</td>
<td>39992.72</td>
</tr>
</tbody>
</table>
Table IX

<table>
<thead>
<tr>
<th>Threshold (t+1)</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>125.7</td>
<td>14.65</td>
</tr>
<tr>
<td>10</td>
<td>265.07</td>
<td>14.7</td>
</tr>
<tr>
<td>20</td>
<td>768.8</td>
<td>15.4</td>
</tr>
<tr>
<td>30</td>
<td>1887.7</td>
<td>18.36</td>
</tr>
<tr>
<td>40</td>
<td>3978.64</td>
<td>19.58</td>
</tr>
<tr>
<td>50</td>
<td>8085.42</td>
<td>21.44</td>
</tr>
</tbody>
</table>

Table X

<table>
<thead>
<tr>
<th>Threshold (t+1)</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>137.87</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>293.26</td>
<td>18.12</td>
</tr>
<tr>
<td>20</td>
<td>835.35</td>
<td>19.25</td>
</tr>
<tr>
<td>30</td>
<td>2122.17</td>
<td>21.45</td>
</tr>
<tr>
<td>40</td>
<td>4408.58</td>
<td>43.28</td>
</tr>
<tr>
<td>50</td>
<td>8290.1</td>
<td>58.79</td>
</tr>
</tbody>
</table>

Table XI

<table>
<thead>
<tr>
<th>Threshold (t+1)</th>
<th>ECTSig</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>125.7</td>
<td>14.65</td>
</tr>
<tr>
<td>10</td>
<td>265.07</td>
<td>14.7</td>
</tr>
<tr>
<td>20</td>
<td>768.8</td>
<td>15.4</td>
</tr>
<tr>
<td>30</td>
<td>1887.7</td>
<td>18.36</td>
</tr>
<tr>
<td>40</td>
<td>3978.64</td>
<td>19.58</td>
</tr>
<tr>
<td>50</td>
<td>8085.42</td>
<td>21.44</td>
</tr>
</tbody>
</table>

VII. CONCLUSIONS

In this paper, we present a practical implementation of an elliptic curve on-line/off-line threshold signature scheme. We propose an elliptic curve trapdoor chameleon hash function and apply the “hash–sign–switch” paradigm to implement an efficient elliptic curve on-line/off-line threshold signature scheme. Efficiency analysis and Extensive simulations show that the proposed scheme is more efficient and robust compared with the one-phase elliptic curve threshold digital signature scheme especially in the on-line phase. We prove that our proposed scheme has achieved the desired security requirements.

REFERENCES