Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-019-01289-z

ORIGINAL RESEARCH

A trust model based batch verification of digital signatures in IoT

Apurva S. Kittur[1] · Alwyn Roshan Pais[1]

Received: 6 September 2018 / Accepted: 2 April 2019

Abstract
In the modern day world, the Internet of things (IoT) is not a new concept. IoT is getting deployed in various applications and fields. Hence with this fast-growing trend, it is essential to maintain the security in the IoT network. Digital Signature is one of the important ways to authenticate an electronic document or a message during communication. Multiple digital signatures are verified at once through the concept of batch verification. Batch verification of multiple digital signatures reduces the computation load and time. Hence this concept is beneficial in IoT environment where nodes have low computation power and operate in a real-time environment. In this paper, we have developed a Trust Model for IoT which helps the Gateway node to identify the trusted sensor nodes which perform batch verification. The sensor nodes receive a batch of signatures from the Gateway node and verify signatures through batch verification and accordingly send back the results. The trust model that we have developed in this paper significantly reduces the probability of selecting unreliable nodes for verification and also reduces the computation load at Gateway node. We have implemented our trust model and presented the results for batch verification of digital signatures.

Keywords Batch verification · Digital signature · Trust model

1 Introduction

Digital Signature is a unique way to identify a signer. Every signer generates a unique signature using his/her private key. The verifier possesses the public key of the signer and verifies the signature using the same. In IoT applications, there are thousands of sensor nodes generating a massive amount of data which needs to be verified before considering for further processing. There are various Digital Signature algorithms available to verify the authenticity of the signer (Kalra and Sood 2015; Wu et al. 2017b).

There are many batch verification techniques introduced for verifying RSA, Digital Signature Standard (DSS) and Elliptic Curve Digital Signature Algorithm (ECDSA) digital signatures (Kittur and Pais 2017). In our study, we are considering ECDSA* signatures for authentication. ECDSA* signatures are the variation of ECDSA signatures (Antipaet al. 2005) which provide 40% more efficiency in verification without compromise in security. ECDSA signatures are considered deemed since they have a smaller key size. There are many batch verification techniques available for ECDSA signatures which verify a batch of these signatures at the same time.

We know that the Gateway node has many responsibilities in an IoT network. Gateway node acts as a bridge between the sensor nodes and the external cloud storage (Kittur et al. 2017). In our scheme, we have identified the parameters to check the availability of the node. Sensor nodes send their data at regular intervals to the Gateway node which verifies the data before processing. Hence we can easily say that the Gateway node has heavy computation to do which creates a bottleneck. Therefore in this paper, we are introducing a model to minimize the workload on the Gateway node. Our model chooses a few trusted sensor nodes carefully and shares the verification workload with these nodes. These nodes, in turn, verify the signatures through batch verification technique and send back the results to the Gateway node which is responsible for further processing.

The contributions of this paper are as follows

✉ Apurva S. Kittur
apurva.kittur@gmail.com

[1] Information Security Research Lab, Department of Computer Science and Engineering, National Institute of Technology, Surathkal, Karnataka 575025, India

1. Proposed a model for selection of sensor nodes for batch verification of digital signatures based on physical and security parameters of sensor nodes.
2. Introduced a novel reputation-based trust model to select the trusted nodes among the available nodes.
3. Parallel implementation of ECDSA* batch signature verification algorithm.

The paper is organized as follows: In Sect. 2 we provide groundwork for the paper. Section 3 introduces the proposed model and Sects. 4 and 5 describe the details of the algorithms used in the model. Then Sect. 6 discusses and makes a comparison of existing trust models with the proposed model. Section 7 provides experimental results and the paper is concluded in Sect. 8.

1.1 Literature review

There are various batch verification algorithms available for different digital signature schemes (Kittur and Pais 2017). The batch verification schemes for RSA (Bao et al. 2006; Changchien et al. 2002), DSS (Naccache et al. 1994; Lim and Lee 1994) and ECDSA (Miller 2004; Karati et al. 2012) digital signatures already exist. Depending upon the application and security requirements, the choice of digital signature and its batch verification scheme is made. As per our knowledge, there is no batch verification scheme implemented in the IoT environment other than our previous work (Kittur et al. 2017).

IoT has been implemented in various applications. One such application is the Secure Smart Healthcare (SSH) system (Zhang et al. 2018). The authors have proposed certificateless aggregate signature scheme to secure the healthcare system which provides secure access to the healthcare records to the users along with the privacy of identity maintained. We are providing a few distinctions between Batch verification scheme and Aggregate signatures scheme:

- In the aggregate scheme, an extra step of signature aggregation is needed at the signer, to group signatures before sending to verifier, whereas it is not needed in batch verification.
- Aggregate Verification scheme defined in Zhang et al. (2018), needs three pairing operations which are 8–10 times costlier than scalar multiplication needed for batch verification.
- The number of operations during aggregate verification in Zhang et al. (2018) does not depend on the number of signatures to be verified, whereas in batch verification the number of scalar multiplications differ for signatures signed by various signers.
- In the aggregate scheme defined in Boneh et al. (2003), the messages have to be different for generating the aggregate signature. The scheme is vulnerable if more than one signature is generated for the same message.

We surveyed various digital signature schemes and the batch verification schemes for the same in our previous work (Kittur and Pais 2017). We found that batch verification of ECDSA* signatures to be more efficient and secure among existing. In ECDSA, the signature verification is more time consuming compared to signature generation. Hence batch verification has the advantage in such cases.

There are various trust models available which prefer the selection of entities based on the trust value of the entity in the cloud platform (Manuel 2015; Xiong and Liu 2004). There are other reputation based models which consider the reputation of the entity with their neighbours and also examines the entity's performance in the past (Kalra and Sood 2015; Selcuk et al. 2004; Vu et al. 2012) before considering as trusted. Most of the reputation based models are designed for Peer-to-Peer (P2P) network (Kamvar et al. 2003; Xiong and Liu 2004; Zhou and Hwang 2007). The nodes are trusted based on the feedback received from its peers. The trust can also be computed using the entity's behaviour, its belief and other parameters as shown in Zhiwei et al. (2015). Buzzanca et al. (2017) consider aging also as a metric for trust computation in their study. Aging refers to the pattern of trust growth for the entity, which also takes into account the recent change in behaviour of the entity.

Implementing authentication schemes in IoT have challenges such as low computing power and memory in sensor nodes. The sensor nodes also have the restriction of low battery capacity. Hence there are various research works on the energy harvesting in sensor nodes and efficient usage of energy (Fisher et al. 2015; Escolar et al. 2014; Wu et al. 2017a) as well as developing lightweight schemes (Wu et al. 2018). Some researchers are working on how efficient public key cryptographic schemes are in IoT and WSN environment (Wander et al. 2005; Kayalvizhi et al. 2010). Since individual signature verification is a costly affair in IoT, the batch signature verification reduces the computation load significantly. In the model by Kittur et al. (2017), the Gateway node reduces the burden of computation by distributing its load to other nodes. The authors show that the load is distributed among other Gateway nodes to reduce the burden of computation at a single node. But the disadvantage with this scheme is that the scheme assumes that the Gateway nodes are idle and are available most of the times.

2 Preliminaries

In this section, we provide some of the preliminary details to understand the paper in a better way. First, we provide the specifications of a few popular sensor nodes followed by the specification for one of the famous Gateway nodes. We also brief the algorithm for key generation, signature generation and signature verification for ECDSA* signatures.

2.1 IoT network nodes

IoT network has sensor nodes and Gateway nodes. Multiple sensor nodes send their data at regular intervals to Gateway node. The sensor nodes have low computation capability and memory. We highlight the specifications of two popular sensor nodes:

1. LOTUS (Maurya and Shukla 2013)
   - *Processor* Cortex M3 CPU 32-bit processor with 10–100MHz
   - *Memory* 64kB SRAM, 512 kB FLASH, 64MB Serial FLASH
   - *Radio* integrated 802.15.4 Radio with on-board 2.4 GHz Antenna
   - *Radio throughput* 250 kbps, high data rate radio
   - *Battery* 2.7–3.3 V, 2× AA battery
2. Waspmote (Pham 2014)
   - *Processor* 8-Bit Microcontroller with 14.7456MHz
   - *Memory* 8kB SRAM, 128 kB FLASH
   - *Radio* XBee Pro 802.15.4 Radio with 2.4 GHz Antenna
   - *Radio throughput* 100 kbps
   - *Battery* 3.3–4.2 V, 6600 mAh Li-Ion rechargeable // 52000 mAh non-rechargeable

The Gateway nodes have higher computation power as well as memory when compared to sensor nodes. Raspberry Pi is one of the famous Gateway nodes used in IoT network. Hence we are providing the specifications of Raspberry Pi3 Model B+ (Aspernäs and Simonsson 2015)

Raspberry Pi

- *Processor* ARM Cortex-A53 64-bit quad core processor SoC @ 1.4 GHz
- *Memory* 1 GB LPDDR2 SDRAM, 512 KB L2 Cache
- *Connectivity* 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
  Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)
- *Power* 5 V/2.5 A DC via micro USB connector
  Power over Ethernet (PoE)—enabled (requires separate PoE HAT)

2.2 ECDSA* batch verification algorithm overview

In this subsection, we brief the ECDSA* signature generation and verification algorithms. ECDSA* is a variation of the ECDSA signature scheme which is about 40% efficient than ECDSA scheme in verification. The notations followed in the rest of paper are briefed in Table 1. We also brief on the batch verification equation for the ECDSA* signatures for both single and multiple signers.

Algorithm 1: ECDSA* Key Generation Algorithm

```
Input: Standard domain parameters
Output: Key Pairs: Public Key- $Q$, Private Key- $d$
 1. For an elliptic curve $E(\mathbb{F}_p)$, choose $P$ of order $n$, $P \in E(\mathbb{F}_p)$
 2. Choose a random integer $d$ such that $2 \leq d \leq n-2$.
 3. Compute $Q = dP$.
```

Algorithm 2: ECDSA* Signature Generation Algorithm

```
Input: Message to be signed $m$, Private Key $d$, Elliptic Curve Domain
       Parameters
Output: ECDSA* Signature $(m, R, s)$
 1. Signer chooses a random integer $k$, such that $2 \leq k \leq n-2$
 2. Compute $R = kP$
 3. $r = x(R) (\text{mod } n)$ \\$x(R)$ is $x$- coordinate of $R$.
 4. Compute $s = k^{-1}(h(m) + dr)$ (mod $n$)
```

Table 1 Notations followed in the paper

Symbol	Reference to
p	Order of the prime field $\mathbb{F}_p$
$E(\mathbb{F}_p)$	Elliptic Curve defined over a prime field $\mathbb{F}_p$
P	Random base point of order n in $E(\mathbb{F}_p)$
n	Prime order of P
$h(m)$	Hash value of the message m
(m, r, s)	Signature of ECDSA algorithm
t	Batch size of the signatures
h	The cofactor $\frac{\lvert E(\mathbb{F}_p)\rvert}{n}$

3 Proposed model

In IoT applications, the Gateway nodes and Sensor nodes have low computation power, low energy and they work in a real-time environment. Therefore batch verification of Digital Signatures in such an application is very efficient. Batch Verification can reduce the time spent on verifying individual signatures sent by thousands of sensor nodes. The Gateway nodes as specified in Sect. 2.1 have higher computational power as compared to sensor nodes and they also handle many responsibilities such as data aggregation, data preprocessing, authenticity and security of underlying nodes etc. Therefore it is very important to minimize load on the Gateway nodes. Our aim is to minimize the workload on the Gateway node by distributing signature verification task to other nodes without compromising the security.

Algorithm 3: ECDSA* Signature Verification Algorithm

```
Input: ECDSA* signature $(m, R, s)$, Public Key $Q$
Output: Signature Accept or Reject
 1. The verifier first computes $r$, $r = x(R)$
 2. The verifier computes $w = s^{-1} \pmod n$
 3. Then compute $u = h(m)w \pmod n$
 4. Compute $v = rw \pmod n$
 5. Compute $R = uP + vQ$ , and accept the signature if and only if $x(R) = r$
    $\pmod n$\\$x(R)$ is $x$- coordinate of $R$.
```

The algorithm 1 is for key generation in ECDSA* signatures. It is the same as for ECDSA signatures. Even the algorithm for the signature generation (algorithm 2) is same as ECDSA signature generation algorithm. But the signature sent across to verifier is different for ECDSA and ECDSA* signature schemes. Hence the signature verification algorithms are little different since the signature size is different. The ECDSA* signature verification algorithm are shown in algorithm 3.

The batch verification equations for ECDSA* signatures is given below,

In case of a single signer,

$$\sum_{i=1}^{t} R_i = \left(\sum_{i=1}^{t} u_i\right)P + \left(\sum_{i=1}^{t} v_i\right)Q. \tag{1}$$

For multiple signers,

$$\sum_{i=1}^{t} R_i = \left(\sum_{i=1}^{t} u_i\right)P + \sum_{i=1}^{t} v_i Q_i. \tag{2}$$

Figure 1 gives an overview of our model. The model aims at reducing the workload at Gateway node by sharing their load with a few trusted sensor nodes. Figure 1 has two algorithms for filtering the sensor nodes and choosing the trusted ones. The first algorithm is for fetching the *Avail* nodes from the set of sensor nodes. The second algorithm is for selecting the set of *Trusted* nodes from the *Avail* node set.

After choosing the *Trusted* set of nodes at the end of the second algorithm, the next task is sharing of load among these nodes by Gateway node. The Gateway node shares a set of digital signatures with each of these nodes, which perform batch verification on their received batch of signatures. This batch verification operation can be performed simultaneously across all nodes in parallel. This results in significant reduction of verification time needed for these multiple signatures. The results for the same are shown later in Tables 4 and 5. These selected *Trusted* nodes perform batch verification on the received batch of signatures. And finally respond the result of the batch verification to the Gateway node.

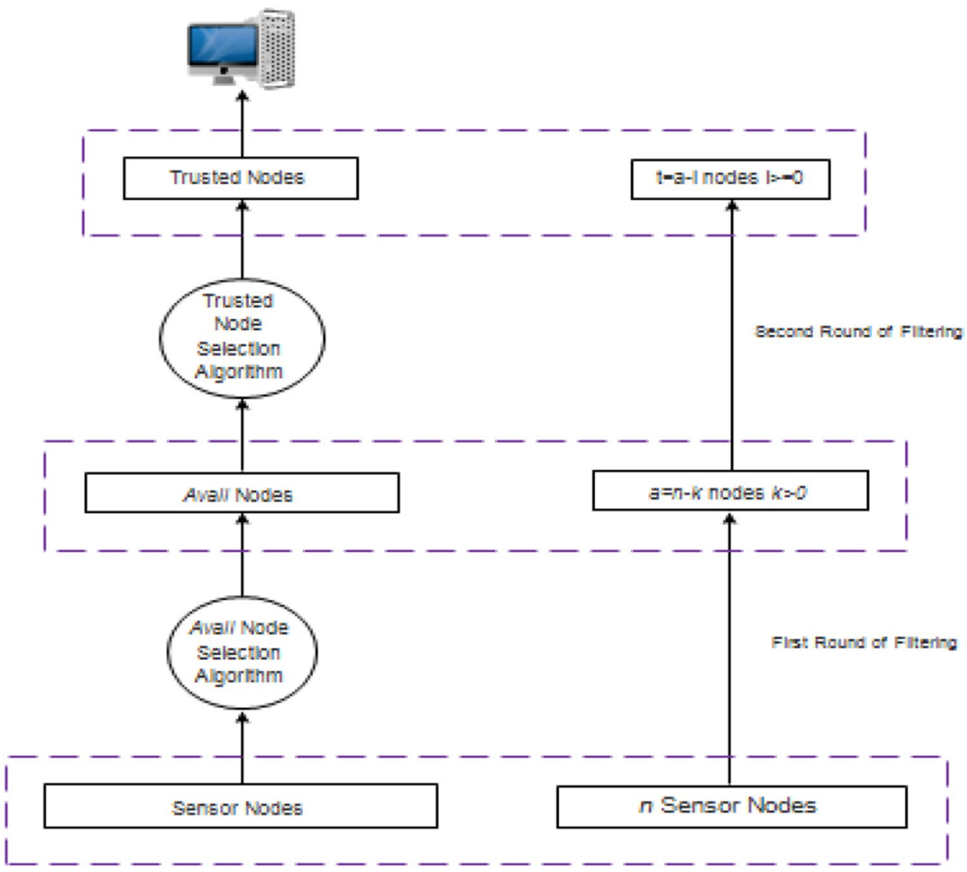

Fig. 1 Trust model for digital signature verification in IoT

Table 2 Current dissipation by a sensor node

Node state	Discharge current
Idle	4.8 mA
Transmitting	156.2 mA
Receiving	138.7 mA
Sleep	94.4 μ A

There are various parameters we have considered in our study to choose the trustworthy sensor nodes. We have designed two algorithms which assist in filtering the nodes. These algorithms are subsequently discussed in Sects. 4 and 5.

3.1 Various parameters for node selection

There are multiple parameters which help us to decide whether a given sensor node is suitable for executing our task. Few of the parameters are Battery level, the efficiency of execution, the type of task the node is already performing, the reputation of the node etc. We have divided the type of parameters into two categories: Physical Parameters and Security Parameters. The two algorithms in our model, filter the sensor nodes based on these parameters. The first algorithm considers the physical parameters and generates the *Avail* node set. The second algorithm uses security parameters to generate the *Trusted* node set which is used for load sharing with the Gateway node.

Physical parameters these are the parameters which define the physical state of the sensor node. In our study, we are considering three physical parameters.

- *Battery level* helps us decide whether the node is able to successfully complete the task assigned to it within the available battery.
- *Type of task* the node is currently running. If the node is currently running a heavy task like transmission or receiving, then it will end up taking more time to complete our task and also will be consuming more energy.

  Table 2 helps us analyze the amount of current discharged during various states of the node. Here we can observe that the node utilizes maximum battery during transmitting and receiving. Hence once a node performs batch verification, it has to send back the report of batch verification to the Gateway node. The result of any batch verification scheme is either 'True' or 'False'. Hence the sensor node has to just notify the Gateway node, whether the given batch of signatures passes the batch verification test or fails. Hence this will have minimum impact on the battery life of both the Gateway node and sensor nodes.

Table 3 Energy consumption by various signature algorithm

Algorithm	Signature generation energy (mJ)	Signature verification energy (mJ)
ECDSA-160	22.82	45.09
ECDSA-224	61.54	121.98

- *Charging status* of the node is one of the essential parameters. If the node's battery level is at the lower side and if it is charging, then the probability of the node running out of battery will be lower. Hence, even if the battery level of the node is at the lower side and if it is charging then it can be considered in *Avail* nodes set, provided it satisfies other conditions.

Security parameters these parameters help the Gateway node in building the trust on the sensor node, hence the name Trust Model. The various security parameters considered are:

- Availability.
- Reliability.
- Data integrity.
- Turnaround efficiency.

The security parameters are qualitative parameters. Hence we have made an effort to quantify these parameters based on the node's efficiency at completing the given task in history. These parameters are explained in Sect. 5.

As discussed in Liao and Hsiao (2014), for a sensor node with a 5 MHz processor, the time per Elliptic Curve Cryptography (ECC) scalar multiplication is approx. 0.06 s. Hence as per Eqs. 1 and 2, the number of scalar multiplications are minimum. Therefore our model does not overload the sensor nodes with batch verification. Hence for sensor nodes in the IoT network, Public Key Infrastructure (PKI) based authentications schemes such as ECC will not create a burden.

3.2 Implementation of the ECDSA* batch verification algorithm

This subsection briefs the implementation details of batch verification of ECDSA* Digital Signatures (Antipaet al. 2005). Our aim of reducing the verification time at the Gateway node is achieved through batch signature verification. In order to further reduce the load on the Gateway node, we are performing the batch verification task in parallel among the *Trusted* sensor nodes. We distribute a set of signatures to the *trusted* nodes which perform batch verification to reduce the verification time.

As we know that the sensor nodes have less computation energy, hence performing a complex mathematical operation is going to be a challenge. We are providing the energy consumption details for the signature generation and verification algorithms of the ECDSA* algorithm in Table 3.

We can observe that the energy consumed during verification is more, hence we use batch verification which significantly reduces the number of point additions and point multiplications operations of ECC. Hence batch verification reduces the power consumption, thus increasing the lifespan of the Gateway node. To further reduce the power consumption of the Gateway node, batch verification is distributed amongst the sensor nodes without compromising the security of the sensor network. This process indirectly increases the lifespan of the network. The selection of sensor nodes for the batch verification is done based on the available battery power of the node.

We have chosen ECDSA* algorithm for our experimentation since it is lightweight. Hence we have performed all our experiments with ECDSA* batch verification scheme. We have also provided the results of running the batch verification code on a different number of nodes for the single signer in Table 4. We can also observe from the table that batch verification has an advantage over individual verification in case of a higher number of signatures. Similarly, Table 5 shows the batch verification results for multiple signers.

For simulation of IoT environment in our system, the batch verification scheme for ECDSA* signatures have been implemented on a cluster system having seven machines, one is master, and other six are computing machines. The system is a Rock cluster 6.0 system. The processor is Intel® Xeon® E5-2650. Each machine has ten cores. And each core runs with 2.3 GHz processor. Based on the specification of the node provided in Sect. 2.1, we have simulated each machine as equivalent to 20 sensor nodes. Therefore in order to simulate the environment of multiple sensor nodes, we used the concept of multi-threading, where a single processor is divided into multiple threads. Hence each thread can be mapped to one sensor node. And we use the MPI library to communicate with other machines in the cluster system for parallel execution. The results show the gain in speedup as we increase the number of nodes. We have considered maximum 20 threads per machine.

The results of batch verification of ECDSA* signatures shown in Table 5 can be compared with the results of aggregate signature verification of Zhang et al. (2018). For verification of signatures using the proposed model, we achieve a huge gain in computation efficiency. The SSH model (Zhang et al. 2018) with aggregate signatures performs better than the proposed model, when the number of signatures is too high and when the Gateway node in our model has a very few trusted nodes to distribute task. But in such situation too, the aggregator in the SSH model, has to aggregate these

Table 4 Verification time (s) for a single signer

Batch size	Individual verification	No. of nodes					
		10	20	30	40	50	60
2^2	0.457	0.053	0.048	0.057	0.051	0.055	0.050
2^4	4.737	0.085	0.070	0.068	0.063	0.064	0.072
2^8	6.328	0.707	0.442	0.384	0.309	0.247	0.200
2^{12}	47.914	11.877	5.778	4.35	3.866	3.170	2.559
2^{16}	808.818	190.11	92.334	65.681	57.668	51.399	42.27

Table 5 Verification time (s) for multiple signers

Batch size	Individual verification	No. of nodes					
		10	20	30	40	50	60
2^2	0.457	0.064	0.060	0.058	0.061	0.063	0.062
2^4	4.737	0.096	0.089	0.102	0.114	0.101	0.097
2^8	6.328	1.546	0.688	0.552	0.472	0.470	0.460
2^{12}	47.914	25.574	12.470	9.362	7.866	6.82	5.386
2^{16}	808.818	421.88	199.93	150.16	128.26	89.362	65.271

large number of signatures which is extra time-consuming. Hence in IoT applications, where the sensor nodes do not have a sufficient capability to both generate signatures and then aggregate them, our model proves to be more efficient.

The Gateway node chooses the *Trusted* nodes based on the proposed Trust model for batch verification in IoT network. The results indicate that more the number of nodes, faster are the results. Our next sections brief the algorithms necessary for choosing *Trusted* nodes among the given sensor nodes by the Gateway node.

4 Node selection based on physical parameters

The sensor nodes have low computation power and battery. Therefore in order to share the workload of the Gateway node among these sensor nodes, it is very critical to check the availability and battery level of the sensor node. Therefore these parameters are considered as the physical parameters which define the physical state of the node.

We know the battery level is an important parameter in *Avail* node selection. Therefore understanding the behaviour of the battery charging and discharging is very important. The discharge time can be calculated with the following formula:

$$t = \left(\frac{C}{I}\right), \tag{3}$$

where t is the discharge time, C is the battery capacity and I is the discharge current. We can observe that the heavy tasks need more current discharge as shown in Table 2 which leads to faster battery discharge. Also, the increase in the rate of discharge decreases the battery capacity. Therefore the Eq. 3 holds true in ideal condition. Thus in practical situations, the time of battery discharge (Eq. 4), given by Peukert (1987), considers the decrease in battery capacity aspect also.

$$t = H\left(\frac{C}{IH}\right)^k, \tag{4}$$

where H is the hour rating, k is the Peukert's constant whose value mostly ranges between 1.1 and 1.3. The Peukert's constant value is usually specified by the vendor on the label of the battery. During our simulation, it is very important to know the battery level of the sensor node before assigning the task. Hence the equation to calculate the available battery capacity at a given time t is,

$$C = e^{\left(\frac{log\frac{t}{H}}{k}+log(IH)\right)}. \tag{5}$$

4.1 *Avail* node selection algorithm

After checking the battery level of the sensor node, it is important to check whether the node is connected to the power source or no and also to check the type of task the node is running. These parameters can be explained through the algorithm 4 in this subsection.

Algorithm 4: *Avail* node selection Algorithm

```
   Input: List of $n$ sensor nodes
          $n'$ nodes are not connected to a power source for charging
          $n''$ nodes are connected to a power source for charging
   Output: $Avail$ nodes
1  Case 1: Nodes whose battery is not connected to a power supply are $n'$
2  for $i \leftarrow 1$ to $n'$ do
       if $B_i > B_T$ then

           if $B_{status} \neq$ busy then
               add to $Avail$ list
           end if
       end if
3  end for
4  Case 2: Nodes whose battery is connected to Power supply are $n''$
5  for $j \leftarrow 1$ to $n''$ do
       if $B_{status} \neq$ busy then
           add to $Avail$ list
       end if
6  end for
```

Algorithm 4 explains how the *Avail* nodes are selected from the given set of sensor nodes. We first divide the algorithm into two cases, one is for nodes which are not charging, and other is for nodes which are charging.

The first case in the algorithm considers nodes which are not connected to the power supply for charging. We initially check the battery level of every node which is not charging. If the battery level is below threshold battery level B_T, then we will discard the node. The threshold level is set according to the type of battery as well as the application where it is deployed. In our experimentation, we consider a threshold battery level as 30% of the total battery capacity. If the battery level is more than the threshold, then we check the state of the node, else discard the node. If the node is either transmitting or receiving, then the node is considered busy, and it will not be considered further in the *Avail* node list. Otherwise, it will be added to the *Avail* list.

In the second case, where the nodes are connected to the power supply, we only verify the status of the node. Since the nodes are connected to the power supply, there is very less probability for the node running out of battery, since the rate of charging is faster than the rate of discharge. Hence if the node is not busy in either transmitting or receiving, then we include the node in *Avail* list of nodes.

5 Node selection based on security parameters

This section aims at choosing *Trusted* nodes by the Gateway node from *Avail* nodes for load sharing based on QoS (Quality of Service) value. We use the terms QoS value and trust value interchangeably, but both of them refer to the same value. The designed model aims at evaluating the QoS for every node which can be given as,

$$QoS = w1 * (AV) + w2 * (RL) + w3 * (DI) + w4 * (TE).$$

The QoS is determined by Availability (*AV*), Reliability (*RL*), Data Integrity (*DI*) and Turnaround efficiency (*TE*). The constants $w1$, $w2$, $w3$, and $w4$ are fixed at the beginning depending on the kind of application the model is being deployed in. The sum of $w1$, $w2$, $w3$, and $w4$ should be equal to one. Suppose in case of cloud computing applications, the choice of $w1 = 0.2, w2 = 0.2, w3 = 0.5$ and $w4 = 0.1$. In our application, the values of the constants are $w1 = 0.25, w2 = 0.4, w3 = 0.1$ and $w4 = 0.25$. The choice of constants can be explained as: since verification of the signature is done to avoid any kind of unreliability, we have assigned highest preference value to $w2$. Since IoT environment needs authentic results in real-time processing, Availability and Turnaround Efficiency are assigned next preference and at the end comes Data Integrity. The weights provided remain the same for a given application IoT network. Hence the choice of weights is an intelligent one. The variable (*AV*, *RL*, *DI*, *TE*) associated with every constant weight changes according to the performance of the nodes for the task given by the Gateway node. The weights help us decide the preferences to be given to various security parameters for the given application.

Availability availability in our case is the degree to which a node is operational or accessible when needed for service. The availability factor of a node is the fraction of the number of job requests accepted to the number of job requests received over a given period of time. Suppose if a node receives an *R* number of job requests and *A* number of requests are accepted for processing over a given time period *T*, then the Availability factor is given as,

$$\textit{Availability}\,(\mathbf{AV}) = \frac{A}{R}.$$

Reliability reliability is a measure of trust. It is the degree to which the node performs failure-free operations under given circumstances. It is the measure of the number of jobs successfully completed among the accepted jobs. Suppose among *A* jobs accepted, and only *C* are successfully completed over a period *T*, then Reliability can be given as,

$$\textit{Reliability}\,(\mathbf{RL}) = \frac{C}{A}.$$

Data integrity data Integrity refers to completeness, accuracy, and consistency of data. Data loss or data modification by any unauthorized node leads to loss of data integrity. Data Integrity aims at preventing unintentional and unauthorized changes to information. Suppose out of *C* successfully completed jobs, only for *D* jobs data integrity is maintained over a period *T*, then *DI* value can be given as,

$$\textit{DataIntegrity}\,(\mathbf{DI}) = \frac{D}{C}.$$

Turnaround efficiency turnaround time is the time gap between the submission of a job and successful completion of the job by the node. The expected turnaround time is the turnaround time specified for the node and actual turnaround time is usually different from the expected turnaround time. Actual turnaround time is the total time between the submission and successful completion of the job in a practical situation. *TE* for a given period *T* can be given as,

$$\textit{Turnaround Efficiency for a node}\,(\mathbf{TE}) = \frac{\textit{Expected turnaround time}}{\textit{Actual turnaround time}}.$$

5.1 *Trusted* node selection algorithm

Our trust model concentrates on choosing the *Trusted* nodes among the *Avail* nodes through their QoS value. Every *Trusted* sensor node and Gateway node update the QoS value for the *Trusted* node after efficient completion of verification job request. Therefore higher the QoS value, higher is the priority of the node getting chosen for load sharing. Thus depending on the need of the number of sensor nodes for load sharing in an application, we decide the size of the *Trusted* node set.

Suppose if there are 500 available nodes, and the signatures to be verified is 2^{15}, then according to our analysis, 100 nodes will provide a maximum speedup. Hence instead of choosing all 500 available nodes, the Gateway node chooses only 100 nodes from the available nodes to distribute the load. Therefore 100 nodes are selected based on their physical and security parameter values. The top 100 ranked QoS value *Avail* nodes are chosen. The choice of the number of *Trusted* nodes depends on the number of signatures received at the Gateway node. For our experimentation, we have decided the number of *Trusted* nodes in such a way that each trusted node gets a batch size maximum of 500 signatures. Hence with these conditions, we can achieve maximum speedup.

Algorithm 5: Scheduling Available Gateway Nodes

```
   Input: List of Avail nodes g
   Output: Trusted nodes c
1  Sort the Avail nodes based on QoS value
2  Verify the QoS value stored with the node with the value stored in Gateway
   node
3  for $j \leftarrow 1$ to Avail do
4      if(stored[j].QoS$\neq$j.QoS)
5         Discard the node from Avail list
6  end for
7  Choose the top $c$ nodes needed for execution
8  Send();   Distribute various batch of signatures to c nodes
9  Receive();   Receive the verification result from individual nodes
10 for $i \leftarrow 1$ to $c$ do
11     Increment AV for node i
12     if ( i.status=successfully completed in given turnaround time )
13        Increment RL, DI, TE for node i
14     else If (i.status=successfully completed with more than expected
       turnaround time)
15        Increment RL, DI for node i
16        Decrement TE for node i
17     else If (i.status=successfully completed with loss of data integrity)
18        Increment RL for node i
19        Decrement DI, TE for node i
20     else
21        Decrement RL, DI, TE for node i
22 end for
```

Algorithm 5 explains the criteria for choosing the nodes for the parallel batch verification by the Gateway node. The reason for sorting the nodes according to QoS value is to choose the most trusted nodes. The nodes with higher QoS value indicates that the nodes have a higher success rate for completion of verification. The unsuccessful completion of the task by the node might be due to many reasons. Either the node went down because of some hardware or software crash or natural calamities, or it may be due to sudden node compromise. Hence verifying the performance of the node in history is very important. The reputation of the node helps us minimize the vulnerable node selection probability.

When the trusted nodes are chosen, the Gateway node distributes the signature verification task to these nodes. As explained before, these trusted nodes perform batch verification over these signatures received from Gateway node. When the batch verification test fails, the next step is to identify the faulty signature/s. In the case of batch verification failure due to a faulty single or multiple signatures, the sensor node just sends the result back to the Gateway node. Now the Gateway node decides what it has to do with the failed batch of signatures. The Gateway node can either go ahead and process/verify the batch on its own or additional mechanism can be implemented to re-submit the batch back to the sensor node. There are many ways to identify the faulty signature in a given batch other than individual verification. Divide-and-Conquer (Pastuszak et al. 2000), Matrix Based Schemes by Li et al. (2010), Ren et al. (2015), help us identify the faulty signature/s. These schemes identify the faulty signature from the failed batch of signatures.

6 Existing models

As mentioned earlier in Sect. 1.1, there are many trust models developed for various type of networks. In the case of e-Bay transactions, the user feedback is very important. Therefore in Peer-to-Peer (P2P) network, it is important to trust the feedback received from various users. There are various trust models available for Wireless Sensor Networks (WSN) for finding the trusted path for the information to reach the central hub. Hence we have provided a comparison of our model with a few popular existing trust models.

Table 6 Trust models

Scheme	Type of network	Reputation	Feedback from	Save trust value of	Physical state of the node
Eigen (Kamvar et al. 2003)	P2P	Distributed	Peers	All peers	No
Peer Trust (Xiong and Liu 2004)	P2P	Distributed	Peers	Neighboring Peers	No
BTRM-WSN (Mármol and Pérez 2011)	WSN	Distributed	Peers	Neighboring Peers	No
Power Trust (Zhou and Hwang 2007)	P2P	Centralized	Reputation system	Only one	No
ATSN (Chen et al. 2007)	WSN	Centralized	Agent	Only one	Yes
Ganeriwal et al. (2008)	WSN	Distributed	Peers	All Peers	No
Our_Model	IoT	Centralized	Gateway	Only one	Yes

We have compared various trust models in Table 6 with the proposed trust model using certain parameters as explained below:

- *Type of network* type of network for which the model is developed, is an important parameter, which decides the way the trust model has to be developed efficiently. In the case of a P2P network, each node has sufficient computation capability and memory storage available to compute and store the trust value of other peer nodes.
- *Reputation* this parameter indicates whether the reputation of the node is decided at the single point of control or is decided collectively by the peers in the distributed network. We can observe in Table 6 that few of the models are centralized and few are distributed.
- *Feedback from* the third parameter indicates the source from where any node gets its trust value. But the trust value in a centralized network can also be calculated from using the feedback received from the other peer nodes. But only a single entity decides the final trust value for the node.
- *Save trust value* this parameter depends on the memory capacity of the node. In few of the models, nodes store the trust value of all the peers in the network or store only the trust value of the neighbouring peers. In centralized reputation models, the nodes need not have to store the trust value of other peers.
- *Physical state of the node* this parameter indicates whether the model considers the physical state of the node into account while choosing it. The physical state indicates whether the node is up and running, or whether the node is busy. This parameter is important in WSN and IoT environment.

The design of the trust model varies based on the application and network where the model needs to be deployed. Every application has its own security and privacy requirements, hence accordingly the models are designed. We can observe from Table 6 that our model is better suited for the IoT environments since it is centralised. Sensor nodes have low computation capability and hence a distributed model will not be suitable in such a case. Since we are considering a centralised model, the peers do not participate in the computation of trust value. Hence any sensor node gets its QoS value computed by the single entity, which saves the trust value of all the peers in the network. There is one more model, ATSN by Chen et al. (2007), which is also suitable for our network, with a difference that an extra entity Agent is required, that collects the feedback from the peer nodes in ATSN. Every node gets its QoS value finalised based on the final evaluation by the Agent.

The proposed trust model is most suited for IoT because it does not require an extra agent to store the trust values of each node as proposed in ATSN. Our model also considers the physical state of the node, which most of the models do not consider as one of the important parameters. Therefore if the node is busy or unavailable, then it is important not to assign extra load to it. Hence the Power Trust model is not suitable for IoT.

7 Results and discussion

In this section, we provide the results of implementing our model. We also compare our results in various scenarios with other models. We provide the probability of choosing nodes that can complete the job successfully in our proposed model and other models, through the normal probability distribution graph.

Our first set of results are taken by considering ideal conditions, (1) the battery is 100% efficient, (2) the node never goes down unless the battery level is zero, (3) the nodes with QoS value greater than 30% will never fail. We are considering four models for load distribution:

1. In the *Proposed model*, the nodes are filtered for batch verification based on the physical and security parameters. It is explained in detail in Sect. 3.

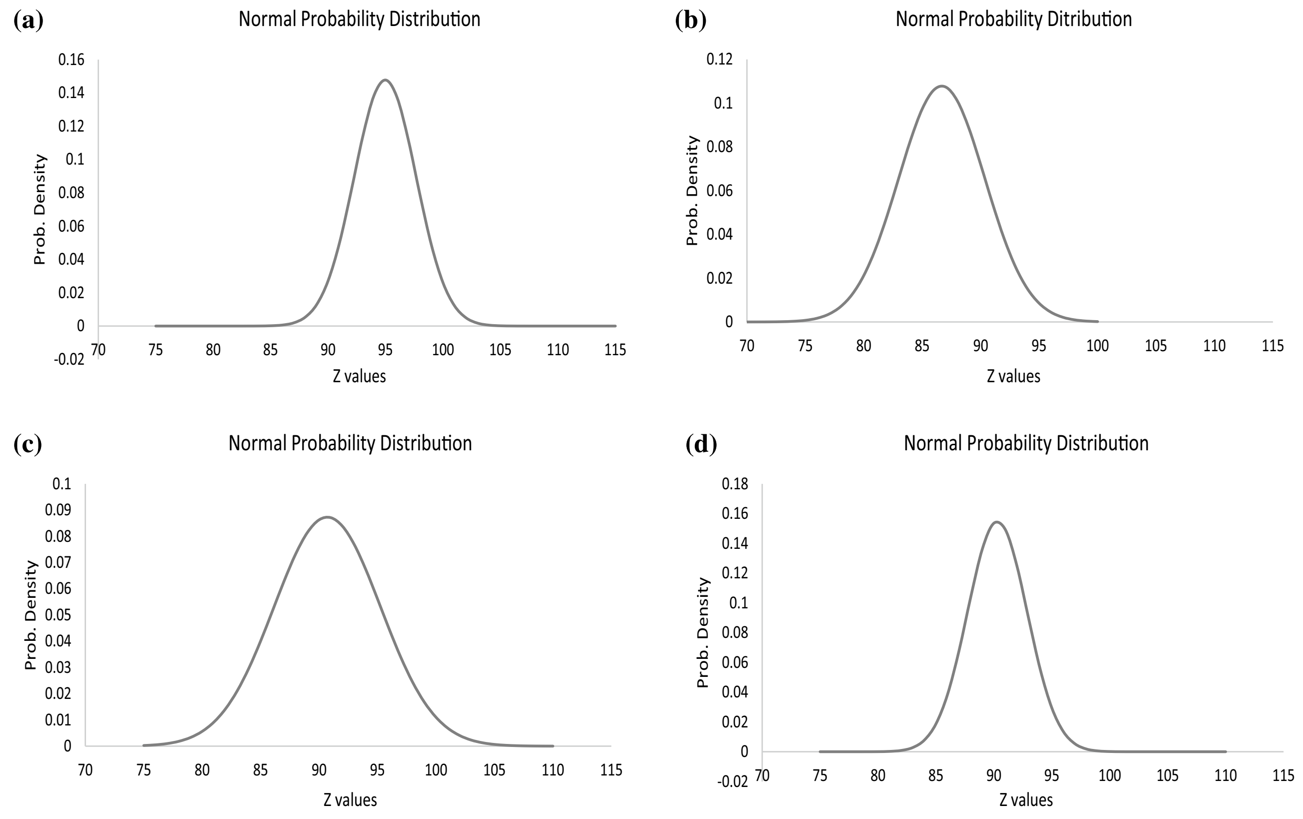

Fig. 2 Probability distribution for various models in ideal conditions

2. In the *Random selection model*, the nodes are considered randomly for batch verification. There is no filter on the selection of nodes. Hence this model has a higher probability of node failures.
3. In the *Physical parameter model* the nodes are filtered based on the physical parameters and not security. Hence the probability of node failure is reduced as compared to the Random Selection model.
4. In the *Security parameter model*, nodes are filtered based on the security parameters. The probability of node failure in this model is comparable to the Physical Parameter model.

We have implemented all the four models for the ideal condition as well as in practical conditions. We will go through the results in both the conditions in separate subsections.

7.1 Ideal condition results

As we know, Ideal condition is the state where the entire model is 100% efficient. In such a state node battery is fully efficient, and its efficiency does not decrease even if the battery level goes less than the threshold level.

We have experimented with 1000 nodes, and we are selecting 100 nodes as *Trusted*, i.e., 10% of the nodes for load distribution. We have already discussed the four models which have specific ways to choose the 10% *Trusted* nodes. The results show how efficiently the models pick the *Trusted* nodes among the given nodes.

We provide results for all the four models who choose 100 *Trusted* nodes from total 1000 nodes and we have randomized the node failure among the *Trusted* nodes due to hardware or software on the scale of ten in Fig. 2.

The graphs in Fig. 2 depict the Normal Probability Distribution for the four models. Figure 2a depicts the probability distribution for our proposed model. For our model, the mean number of nodes selected which are available without failure is almost 95%. Similarly, the graphs in Fig. 2b–d are results for the other three models in similar conditions. We can also interpret from the graphs that our model has the highest mean which implies that the proposed model has minimum node failures among the selected 100 *Trusted* nodes.

7.2 Practical condition results

We considered ideal conditions for node selection in the previous subsection, and now we consider the practical conditions. In practical conditions, the battery efficiency reduces when the battery level goes down below 30% (we assumed

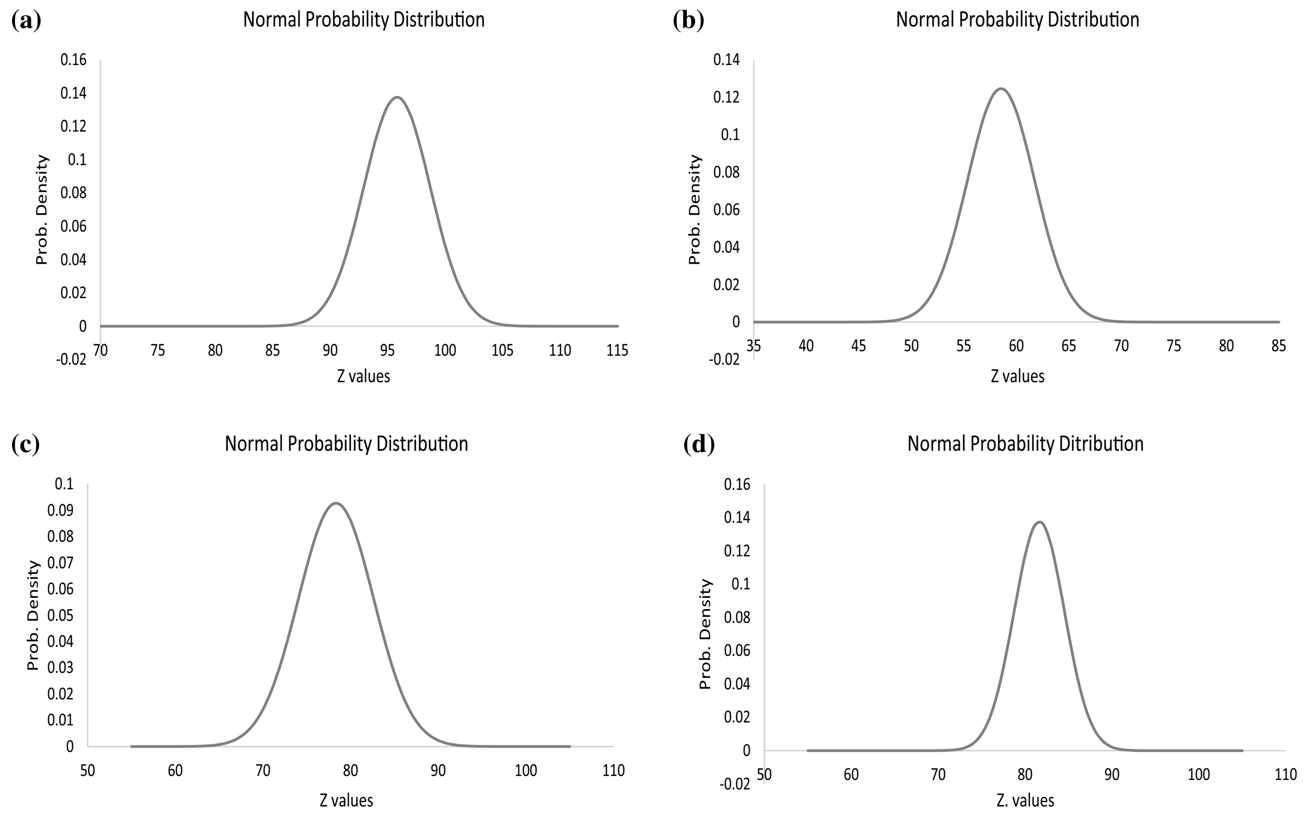

Fig. 3 Probability distribution for various models in practical conditions

that the efficiency decreases below 30%). Hence we have set the threshold Battery level to 30% in algorithm 4. And also the nodes with trust value less than 50%, fall under greater node failure category. Our proposed model checks for these conditions by default. Hence the performance of our model does not cease in realistic conditions too, whereas it affects the other models.

The probability distribution in practical conditions is shown in Fig. 3. The mean for the proposed model in Fig. 3a remains the same and for other models it differs. The random selection model in Fig. 3b performs the worst since it chooses the sensor nodes randomly without filters and hence has a higher node failure probability. The performance of physical parameter model in Fig. 3c and security parameter model in Fig. 3d are almost similar.

The proposed model performs better in most of the conditions. Since the model keeps a check on every parameter of the sensor node. Also, the performance does not degrade by varying conditions and by varying the number of sensor nodes.

7.3 Security analysis

We have used ECDSA* signatures (Antipaet al. 2005) for node authentication. The batch verification technique for ECDSA* is as secure as ECDSA signatures. The security for the trust model is measured through the trust value of the sensor node. Hence in algorithm 5, we can observe that the Gateway node prefers the nodes with higher QoS value.

Suppose in one of the scenarios, a node gets compromised and its QoS value is illegally incremented to get access to signatures, then the proposed model is efficient to detect it. In such an attack, the attacker tries to include the compromised node in the list of trusted nodes. This way the attacker tries to verify an illegal or bad signature as the legal signature by paving way for other attacks.

The unauthorized manipulation of the QoS value of the node can be detected by storing a copy of the QoS value for every node in the Gateway node. Hence, once the *Avail* nodes are derived from algorithm 4, our next task is to verify whether the QoS values of these nodes are same as the QoS values stored in the Gateway node. Therefore this way it will be easy for the Gateway node to identify the faulty node and discard it from the list. Hence the node will not be considered for further verification task by the Gateway node.

For the Data information spoofing during communication between the sensor node and the Gateway node, there are multiple Key Pre-distribution Schemes available (Du et al. 2005; Chan et al. 2003). Since sensor nodes have low bandwidth and computation power, it is very important to have lightweight protocols like Simple Object Access Protocol (SOAP) by Mitra et al. (2003), Constrained Application Protocol (CoAP) etc. There are other protocols for secure communication such as IPSec (Frankel and Krishnan 2011), Datagram Transport Layer Security (DTLS) (Rescorla and Modadugu 2012) etc. These protocols are not suitable for communication in sensor nodes since the protocols have a high bandwidth delay product, high packet loss.

There is always a trade-off between the security and the computation time. Many industries come up with technologies which compromise the security in a contest to reduce the computation time. Hence in our trust model, we are trying to reduce the computation time and load at the Gateway node by distributing load across *Trusted* nodes, which reduce the probability of node failure. Our scheme does not increase the latency since the sensor nodes need not have to communicate back the entire batch of signatures to the Gateway node. The sensor nodes just verify the signatures through batch verification and respond to the Gateway node whether the batch verification is a *success* or a *failure*.

The proposed model is agnostic of the underlying signature scheme used. The model can be implemented over other signature schemes too such as RSA or DSS. The security of the model depends on the underlying signature scheme but also depends on how efficiently the Gateway node chooses the trusted nodes, else the compromised node may verify an invalid signature as valid. Hence the security of the model depends on how carefully the trusted nodes are chosen as well as the signature scheme and the batch verification scheme.

8 Conclusion and future work

IoT network has thousands of sensor nodes and actuators. These are connected to the Gateway node. The Gateway node acts as the bridge between the sensor nodes and the cloud storage. To reduce the heavy workload of multiple signature verification at the Gateway node, we have proposed a model to share workload among the sensor nodes. Our proposed model is a trust model for the Gateway node where the sensor nodes selection for parallel batch verification, depends on the node's reputation in history. The selection criteria depend on the physical and security parameters. The performance of the proposed model is better than the other discussed models.

We have proposed two algorithms at two different stages in node selection. The final nodes are *Trusted* set of nodes that are used for load sharing. We have observed that parallel batch verification on the *Trusted* sensor nodes significantly reduces the bottleneck at the Gateway node. Our future aim is to reduce further bottleneck at the Gateway node efficiently as well as securely.

References

Antipa A, Brown D, Gallant R, Lambert R, Struik R, Vanstone S (2005) Accelerated verification of ecdsa signatures. In: International workshop on selected areas in cryptography, Springer, pp 307–318

Aspernäs A, Simonsson T (2015) Ids on raspberry pi: a performance evaluation

Bao F, Lee CC, Hwang MS (2006) Cryptanalysis and improvement on batch verifying multiple rsa digital signatures. Appl Math Comput 172(2):1195–1200

Boneh D, Gentry C, Lynn B, Shacham H (2003) Aggregate and verifiably encrypted signatures from bilinear maps. In: International conference on the theory and applications of cryptographic techniques, Springer, pp 416–432

Buzzanca M, Carchiolo V, Longheu A, Malgeri M, Mangioni G (2017) Direct trust assignment using social reputation and aging. J Ambient Intell Hum Comput 8(2):167–175

Chan H, Perrig A, Song D (2003) Random key predistribution schemes for sensor networks. In: Security and privacy, 2003. Proceedings. 2003 Symposium on, IEEE, pp 197–213

Changchien SW, Hwang MS, Hwang KF (2002) A batch verifying and detecting multiple rsa digital signatures. Int J Comput Numer Anal Appl 2(3):303–307

Chen H, Wu H, Zhou X, Gao C (2007) Agent-based trust model in wireless sensor networks. In: Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing (SNPD 2007), vol 3. IEEE, Qingdao, pp 119–124. https://doi.org/10.1109/SNPD.2007.122

Du W, Deng J, Han YS, Varshney PK, Katz J, Khalili A (2005) A pairwise key predistribution scheme for wireless sensor networks. ACM Trans Inf Syst Secur (TISSEC) 8(2):228–258

Escolar S, Chessa S, Carretero J (2014) Energy management in solar cells powered wireless sensor networks for quality of service optimization. Pers Ubiquitous Comput 18(2):449–464

Fisher R, Ledwaba L, Hancke G, Kruger C (2015) Open hardware: a role to play in wireless sensor networks? Sensors 15(3):6818–6844

Frankel S, Krishnan S (2011) Ip security (ipsec) and internet key exchange (ike) document roadmap (No. RFC 6071)

Ganeriwal S, Balzano LK, Srivastava MB (2008) Reputation-based framework for high integrity sensor networks. ACM Trans Sens Netw (TOSN) 4(3):15

Kalra S, Sood SK (2015) Secure authentication scheme for iot and cloud servers. Pervasive Mob Comput 24:210–223

Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web, ACM, pp 640–651

Karati S, Das A, Roychowdhury D, Bellur B, Bhattacharya D, Iyer A (2012) Batch verification of ecdsa signatures. In: International conference on cryptology in Africa, Springer, pp 1–18

Kayalvizhi R, Vijayalakshmi M, Vaidehi V (2010) Energy analysis of rsa and elgamal algorithms for wireless sensor networks. In: International conference on network security and applications, Springer, pp 172–180

Kittur AS, Pais AR (2017) Batch verification of digital signatures: approaches and challenges. J Inf Secur Appl 37:15–27

Kittur AS, Jain A, Pais AR (2017) Fast verification of digital signatures in iot. In: International symposium on security in computing and communication, Springer, pp 16–27

Li CT, Hwang MS, Chen S (2010) A batch verifying and detecting the illegal signatures. Int J Innov Comput Inf Control 6(12):5311–5320

Liao YP, Hsiao CM (2014) A secure ecc-based rfid authentication scheme integrated with id-verifier transfer protocol. Ad Hoc Netw 18:133–146

Lim CH, Lee PJ (1994) Security of interactive dsa batch verification. Electron Lett 30(19):1592–1592

Manuel P (2015) A trust model of cloud computing based on quality of service. Ann Oper Res 233(1):281–292

Mármol FG, Pérez GM (2011) Providing trust in wireless sensor networks using a bio-inspired technique. Telecommun Syst 46(2):163–180

Maurya M, Shukla SR (2013) Current wireless sensor nodes (motes): performance metrics and constraints. Int J Adv Res Electron Commun Eng 2(1):045

Miller VS (2004) The weil pairing, and its efficient calculation. J Cryptol 17(4):235–261

Mitra N, Lafon Y et al (2003) Soap version 1.2 part 0: primer. W3C Recomm 24:12

Naccache D, M'RaÏhi D, Vaudenay S, Raphaeli D (1994) Can dsa be improved?—complexity trade-offs with the digital signature standard. In: Workshop on the theory and application of of cryptographic techniques, Springer, pp 77–85

Pastuszak J, Michałek D, Pieprzyk J, Seberry J (2000) Identification of bad signatures in batches. In: International workshop on public key cryptography, Springer, pp 28–45

Peukert D (1987) Die Weimarer Republik, vol 9. VEB Deutscher Verlag für Musik, Leipzeg

Pham C (2014) Communication performances of ieee 802.15. 4 wireless sensor motes for data-intensive applications: a comparison of waspmote, arduino mega, telosb, micaz and imote2 for image surveillance. J Netw Comput Appl 46:48–59

Ren Y, Wang S, Zhang X, Hwang MS (2015) An efficient batch verifying scheme for detecting illegal signatures. IJ Netw Secur 17(4):463–470

Rescorla E, Modadugu N (2012) Rfc 6347, datagram transport layer security version 1.2. Internet Eng Task Force 13:101

Selcuk AA, Uzun E, Pariente MR (2004) A reputation-based trust management system for p2p networks. In: ccgrid, IEEE, pp 251–258

Vu QAN, Canal R, Gaudou B, Hassas S, Armetta F (2012) Trustsets: using trust to detect deceitful agents in a distributed information collecting system. J Ambient Intell Hum Comput 3(4):251–263

Wander AS, Gura N, Eberle H, Gupta V, Shantz SC (2005) Energy analysis of public-key cryptography for wireless sensor networks. In: Pervasive computing and communications, 2005. PerCom 2005. Third IEEE international conference on, IEEE, pp 324–328

Wu F, Rüdiger C, Yuce MR (2017a) Real-time performance of a self-powered environmental iot sensor network system. Sensors 17(2):282

Wu F, Xu L, Kumari S, Li X (2017b) A privacy-preserving and provable user authentication scheme for wireless sensor networks based on internet of things security. J Ambient Intell Hum Comput 8(1):101–116

Wu F, Xu L, Kumari S, Li X, Das AK, Shen J (2018) A lightweight and anonymous rfid tag authentication protocol with cloud assistance for e-healthcare applications. J Ambient Intell Hum Comput 9(4):919–930

Xiong L, Liu L (2004) Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. IEEE Trans Knowl Data Eng 16(7):843–857

Zhang Y, Deng RH, Han G, Zheng D (2018) Secure smart health with privacy-aware aggregate authentication and access control in internet of things. J Netw Comput Appl 123:89–100

Zhiwei G, Yingxin H, Kai L (2015) Cptias: a new fast pki authentication scheme based on certificate path trust index. J Ambient Intell Hum Comput 6(6):721–731

Zhou R, Hwang K (2007) Powertrust: a robust and scalable reputation system for trusted peer-to-peer computing. IEEE Trans Parallel Distrib Syst 4:460–473