# Empirical Evaluation of ECC Batch Verification Algorithms of Digital Signature in Wireless Sensor Network

Pankaj Kumar, Saurabh Kumar Sharma, and Kaveri Umesh Kadam

***Abstract*— Wireless sensor network (WSN) is an emerging topic in wireless communication where various sensor nodes are connected over the internet for data exchange. Exchanging the information over the WSN is very critical; therefore, it requires authentication and security mechanisms to ensure integrity and confidentiality. To avoid unauthorized use of WSN, the authentication through a digital signature turns into an essential task in WSN. Verification of individual signature is taking too much time and it is not suitable for wireless sensor network. Digital signature verification time can be reduced by using batch verification techniques in WSN. Therefore, the goal of this research is to examine and analyze the different type of batch verification algorithm based on Elliptic Curve Cryptography (ECC). With respect to security, the entire algorithm is identical to the standard Elliptic Curve Digital Signature Algorithm (ECDSA) verification algorithm. These algorithms are effectively limited to small batch sizes.**

***Index Terms*—Digital Signature, Elliptic Curve Cryptography, Batch Signature Verification, ECDSA, Wireless Sensor Network**

*Original Research Paper*
*DOI: 10.53314/ELS2226053K*

## I. Introduction

A digital signature is desired in many applications in today's modern world. Batch verification techniques of digital signatures verify multiple signatures simultaneously. These verification techniques can be used in wireless sensor networks (WSN) due to their low computing cost and less verification time at the verifier end. The batch verification process has certified that the signatures in the batch are true or valid. The batch contains at least one or many more invalid signatures if its verification returns false. Digital signatures are frequently used in network-based applications like air traffic control, e-marketing, e-hospital management systems, and the processing of financial transactions. As a result, using batch verification in certain cases speeds up communication. Batch verification's benefits can be used in WSN, which has computation time and energy constraints. As a result, batch verification saves time and energy.

Batch signature systems have been proposed for a variety of digital signature schemes. The Elliptic Curve Digital Signature Algorithm (ECDSA) is one such digital signature scheme. ECDSA is referred to as a lightweight digital signature method due to its minimal key size. The signature verification process is slower than the signature generation process in the majority of ECDSA signature schemes. Hence, verifying such signatures in a batch saves time over sequential verification. A variety of batch verification approaches can be used for verifying several ECDSA signatures simultaneously. It minimises the verification time and CPU consumption. Verifying several ECDSA signatures in a batch makes the verification more relevant in diverse real-time applications. Despite the fact that existing strategies, as assessed by Kittur et al. [19], attempt to reduce computing time, the majority of them are inefficient for larger batch sizes. ECDSA signatures are being considered for authentication in our research. ECDSA signatures are a variant of ECDSA signatures that enables a 40% increase in verification time efficiency.

Individual signature verification takes too much time, making it unsuitable for wireless sensor networks. Using batch verification techniques in WSN helps to accelerate digital signature verification. Consequently, this study's objective is to evaluate and analyze several batch verification algorithms based on Elliptic Curve Cryptography (ECC). The entire technique is the same as the common ECDSA verification algorithm in terms of security. Effectively, these methods are constrained to small batch sizes. Fig. 1 shows a hierarchical order of batch signature verification techniques.

The layout of the paper is structured as: Section 2 explains the brief overview of wireless sensor networks. Section 3 introduces the major security threats to WSN. Section 4 explains the related work of batch verification of digital signatures. Section 5 discusses the use of digital signature in WSN and its batch verification techniques. Section 6 explains the ECDSA batch verification algorithms. Section 7 describes the experimental setup of this research. Section 8 explains the experimental results and their analysis, and Section 9 concludes the solution and discusses future scope.

Manuscript received 10 May 2022. Received in revised form 4 August 2022. Accepted for publication 6 August 2022.

Pankaj Kumar is a PhD student at School of Computer and Systems Sciences, Jawaharlal Nehru University (JNU), New Delhi, India. (phone: +91-90-1542-7741; e-mail: pankajkumar.scss.jnu@gmail.com).

Saurabh Kumar Sharma is an Assistant Professor at School of Computer and Systems Sciences, Jawaharlal Nehru University (JNU), New Delhi, India. (e-mail: saurabhsharma.lpu@gmail.com).

Kaveri Umesh Kadam is an Assistant Professor at the Electronics and Communication Engineering Department, Jamia Millia Islamia Central University (JMI), New Delhi, India. (e-mail: gf.kkadam@jmi.ac.in).

S.K.S. is financially supported by the University Grants Commission (UGC) (UGC, File no. F.30-553/2021(BSR)) New Delhi, India

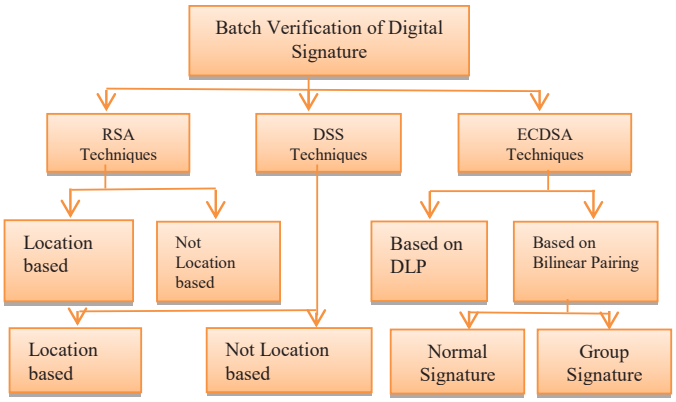


Fig. 1. Classification of batch verification

## II. An overview of wireless sensor networks

A wireless sensor network is made up of autonomous objects called sensor nodes that are dispersed around ongoing activities and work together to detect the physical state of the environment, such as vibration, motion, noise, temperature, stress, impurity, etc. In order to acquire specific information and send it to the base station for analysis via a multi-hop communication link, a specific sensor can detect events and objects that are within its sensing range. The base station then uses the internet to deliver the network data to the application server for further analysis. There are typically two types of wireless sensor networks: structured and unstructured. Structured WSNs typically have a limited number of nodes and are very simple to manage due to the predetermined placement of each node. It is extremely difficult to handle the unstructured WSN sensors because they are installed in an adhoc manner. In the WSN, each sensor has a sensing zone where it can detect an event or an object. A sensor, however, has very little processing power, the smallest buffer capacity, and limited processing ability. A huge number of nodes are working cooperatively, and managing all those nodes requires scalable and efficient algorithms. The dynamic nature of WSNs may have an impact on the network routing strategy, delay, localization, Quality of Service (QoS), coverage, connection quality, cross-layer design, fault detection, etc.

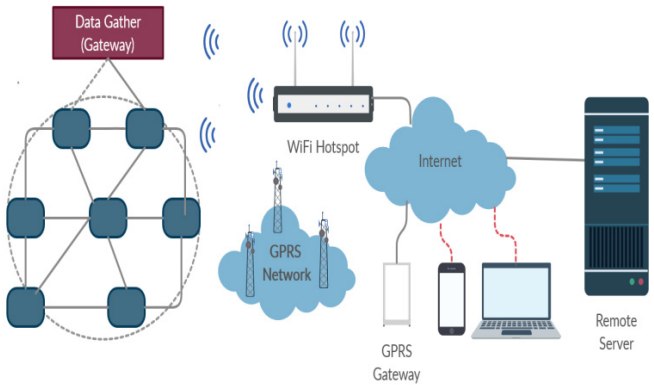


Fig. 2. Wireless sensor networks: basic architecture

WSNs have two key components with higher capabilities than other sensors: “aggregate points” and “base station.” Basically, aggregation points gather the data from the nearby sensors and transfer it to the “base stations” for processing. The basic architecture of WSN is shown in Fig. 2. Three key responsibilities of a sensor node are: (i) sensing the environment; (ii) data processing; and (iii) interacting with other sensor nodes via the base station/sink node.

## III. Major Security Threats in WSN

Confidentiality, integrity, authentication, non-repudiation, and anti-playback are all characteristics of security in WSN. WSN mostly consists of tiny sensors, which essentially have insufficient battery life, processing speed, and memory. Any of these security measures requires additional transmission of data, which requires additional processor, battery, and memory. Data encryption in a WSN could cause more jitter, delay, and packet loss. When applying security mechanisms to WSN, some critical questions arise, such as how keys are generated, disseminated, supervised, revoked, and allocated to new sensor nodes included in the network. The two main categories of security threats in WSN are described below. One is a threat to a routing mechanism, and the other is a threat to a security mechanism. Some of the major threats to WSN are:

### *A. Denial of Service (DOS)*

A Denial of Service (DOS) attack tries to consume the resources of the victim node by sending out illegal junk packets and prevents legitimate network nodes from accessing the resources. DoS attacks are possible across a wide range of WSN layers. At the physical layer, DoS threats can be jammed and tempered with the network. The data link layer is susceptible to exhaustion, collision, and unfairness. At the network layer, there may be misdirection, homing, black holes, garbage, and greed. De-synchronization and malicious flooding are both possible at the transport layer. To stop DoS attacks, the network requires strong authentication, pushback, and traffic identification.

### *B. Attack on information during transit*

A sensor node’s job is to keep track of, record, and report on various factors to a base station, such as temperature, sound, humidity, etc. In transit, information can be changed, replayed, and deleted. A hacker can eavesdrop on WSN traffic while it is being monitored and then intervene to alter, obstruct, or falsify data. Thus, inaccurate data could be transmitted to the base station.

### *C. Sybil Attack*

A node can spoof the identity of other nodes in a Sybil attack. In some circumstances, sensors in a WSN must collaborate to perform a task; thus, dissemination of subtasks or redundant data over sensor nodes may be possible. In this case, a node can impersonate many nodes by acquiring the identities of other valid nodes. A Sybil attack may target distributed storage mechanisms, routing mechanisms, data aggregations, voting mechanisms, fair resource distribution and techniques for

detecting bad behavior. The Sybil attack on a wireless sensor network is depicted in Fig. 3.

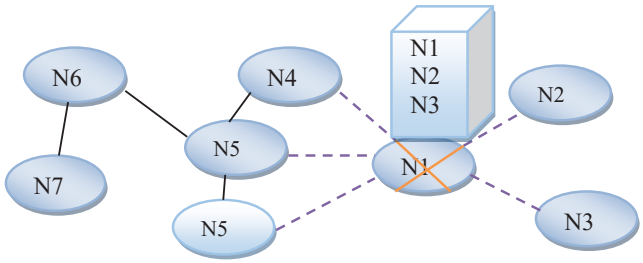


Fig. 3. Sybil attack

Detection of Sybil nodes in wireless networks is too difficult. Radio resource testing was used by Newsome et al. to find out if there is a sybil node ($S$) in WSN. A Sybil node can be detected with a probability (Pr) is shown in eq. 1.

$$\Pr = 1 - \left( \sum_{allM,S,G} \frac{\frac{s!}{S!(s-S)!}\frac{m!}{M!(m-M)!}\frac{g!}{G!(g-G)!}}{\frac{n!}{c!(n-c)!}} \frac{S-(m-M)}{c} \right)^{r} \quad (1)$$

In this model, $n$ indicates the number of neighbors, $s$ indicates the number of Sybil nodes, $m$ indicates the number of malicious nodes, $g$ indicates the number of good nodes, $c$ indicates the number of nodes that any one node can test at one time, $S$ indicates the number of Sybil nodes, $M$ indicates the number of malicious (faulty) nodes, and $r$ indicates the number of test rounds.

*D. Black hole/Sinkhole Attack*

As a result of this attack, malicious nodes become black holes, attracting all traffic to themselves. The attacking node responds to the target node by taking the quickest route to the sink node when flooding-based protocols are in operation. The rogue node can manipulate the packets moving between the communication nodes if it is able to position itself between them. Fig. 4 is a conceptual representation of a black hole/ sinkhole attack.

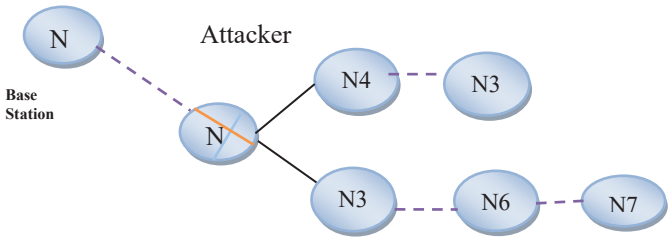


Fig. 4. Black hole attack

*E. Hello flood attack*

Hello packets are used by the attacking node to mislead the WSN sensors into thinking it is a neighboring node. Using a long radio broadcast range and significant computational power, the attacker uses the WSN to transmit hello packets to sensor nodes dispersed across a large area. The attacking node poses as a nearby node. In this instance, the attacker spoofed the information when the victim node attempted to interact with the base station through the attacker node.

*F. Wormhole attack*

In this crucial attack, the attacker tunnels packets from one network point to another. The compromising of a WSN sensor node is not necessary for this kind of attack. When sensors start to find local information, it starts to work. A wormhole attack scenario is depicted in fig. 5. The attacker duplicates the routing request packets to its neighbor nodes after receiving them from node N4. Node N4 gains parent status since it is in its range when a neighbor node receives a replay packet. Even though N4 was several hops away, the attacker convinced the victim nodes that it was only one hop away.

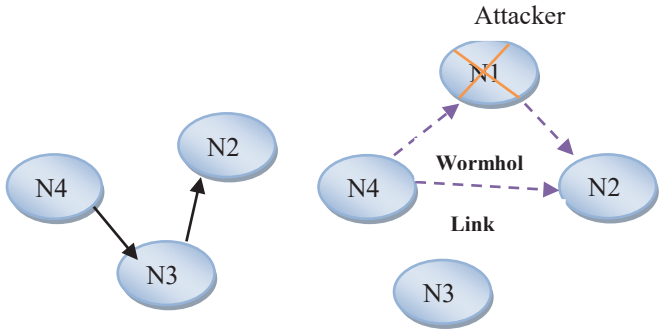


Fig. 5. Wormhole attack

## IV. Related work

L. Harn et al. [1] introduced the Rivest Shamir Adelman (RSA) batch verification procedure, which recognizes the existing invalid signature. It has been suggested that many more strategies [2]-[5] can be used to detect invalid signatures in a batch of RSA signatures. It was proposed by Harn and Shao [2] that batch verification techniques can be used to detect whether the signatures in a batch of Digital Signature Standard (DSS) signatures contain an invalid signature. It has been proposed by Lin et al. [6] and Pastuskzak et al. [7] that identifies the location of faulty signatures in a specific batch of DSS signatures. The performance of an RSA-based batch verification scheme was analyzed by D. J. Guan et al. [8]. This approach is based on a randomized numbering test that randomizes signature order and validates it in batches, where k denotes the number of signatures. Individually, Koblitz [9] and Miller [10] presented ECC-based digital signature techniques for individual verification. To speed up ECDSA signature verification [11]-[12] several batch verification techniques have been developed. As part of the Automatic Dependent Surveillance-Broadcast (ADS-B) System, A. Yang et al. [13] proposed a method ba-

sed on hierarchical identity-based signatures with batch verification. By using batch verification, n-signatures can be verified more efficiently than independently. Cheon et al. [14] introduced a batch verification technique which requires *w*-Non-Adjacent-Form (NAF) computation, where *w* denotes the window-width. The *w*-NAF method is used for calculating the square root computation. This scheme is more efficient in calculating point *R* of signature than traditional approaches. J. Liu et al. [15] designed an aggregate signature-based monitoring batch verification scheme for the Internet of Vehicles (IoV). This scheme lowers the communication and computation costs to enhance the survivability of the IoV framework. The traditional approach for batch verification of the ECDSA algorithm is based upon the idea of taking square roots in the underlying field. S. Karati et al. [16] proposed a novel approach for batch signature verification which replaces the square root computation by symbolic representation. Therefore, it reduces the time consumption taken by the square root method. However, it is only effective for lower batch sizes. As the batch size increases, the efficiency of these signature algorithms decreases. For an anonymous authentication and batch verification scheme for Mobile Healthcare Crowd Sensing (MHCS), J. Liu et al. [17] proposed a bio-information batch verification scheme. In this scheme, the computational Diffie-Hellman Problem (CDHP) determines the hardness, and there is a lower computation overhead than in earlier schemes. However, it is only suitable for the MHCS system. Based on dynamic group management and certificate-less public-key cryptography, L. Jiang et al. [18] proposed a fast and secure batch verification system. Following the establishment of two-way authentication between the proxy vehicle and the roadside unit (RSU), batch verification approaches relying on RSU-assisted vehicle management are provided. While boosting robustness in dynamic environments, this technique decreases transmission overhead and computational time. An analysis of alternative RSA, DSS, and ECDSA signature batch verification strategies is presented by A. S. Kittur et al. [19]. This study also highlights significant research concerns in other areas of RSA, DSS, and ECDSA. An automated system was introduced by KN. Sridharan Namboodiri et al. [20] for authenticating key agreements with vehicles and roadside units (RSUs), resulting in a significant reduction in data exchange traffic as well as the ease of batch verification of transactions. A. S. Kittur et al. [21] created an Internet of Things (IoT) Trust Model that assists the gateway node in identifying trusted sensor nodes that execute batch verification. The sensor nodes receive a batch of signatures from the gateway node, verify them with batch verification, and send back the results to the gateway node. It minimizes the likelihood of selecting unreliable nodes during verification and reduces the computational load on the gateway node at the same time. N.B. Gayathri et al. [22] proposed a certificate-less signature (CLS) system that uses pairings to facilitate batch verification. Under the Computational Diffie-Hellman (CDH) assumption, the given approach is unforgeable. Furthermore, the scheme is proved without the use of the Forking lemma, which results in a very secure system. X. Hu et al. [23] proposed an identity-based batch verification (IBBV) scheme. It doesn't employ bilinear pairing because it's insecure and privacy-invading. For single and multiple messages, the overall calculation cost for signing and verifying is the same: 1.2 milliseconds. R. Jagadeesha et al. [24] present a new energy-saving strategy for mobile receivers. The Mobile Sender adjusts the batch size according to the amount of energy available at the receiver. When the mobile receiver's energy level hits a specific threshold, the sender is notified to cut the batch size in half, lowering the amount of energy needed by the mobile receiver for batch signature verification. J. Shen et al. [25] present a secure real-time traffic data aggregation technique for vehicular clouds in vehicular adhoc networks (VANETs) by utilizing the data recovery property of the message recovery signature (MRS). The legitimacy of vehicle signatures is confirmed in this system, after which the original traffic data is reconstructed from signatures. When processing signatures from several vehicles, batch verification makes the scheme more efficient. H. Zhong et al. [26] proposed an improved elliptic curve cryptography digital signature system by optimizing the multiplicative inverse module of ECDSA. The ECC lightweight cryptography is used in this approach. This system was implemented using the MicaZ sensor network platform. M. S. I. Mamun et al. [27] proposed a group signature (GS) technique for Wireless Sensor Networks that is application-friendly. As a single short GS method, it incorporates linking, direct opening, message-dependent opening (MDO), revocation, and batch verification. P. Thorncharoensri et al. [28] suggested an "Aggregatable Certificate Less Designated Verifier Signature" (ACLDVS) method. ACLDVS is an ideal solution for IoT and vehicle ad hoc networks that require fast, privacy-preserving authentication methods. C. Meshram et al. [29] proposed an online/offline subtree-based short signature technique with a high security level. Chebyshev chaotic maps were used to share WSN fuzzy user data over a Galois field.

As a result of this architecture, offline storage can be reused during polynomial time rather than being created from scratch. By combining the advantages of Certificate Less Public Key Cryptography (CLPKC) and Aggregate Signature (AS), H. Du et al. [30] created a Certificate Less Aggregate Signature (CLAS) scheme without pairings. In a healthcare wireless sensor network, this technique ensures the privacy and integrity of healthcare information for many patients. By evaluating certain WSN reliability indicators such as data delivery resiliency, power consumption, coverage, and network capacity, the heuristic approach technique [31] may be used to assess the reliability of the suggested batch verification algorithms. In order to facilitate the individual verification of ECDSA signatures, the ECDSA* [32] scheme has been proposed. For message m and public key Q, the signature parameters in ECDSA* [32] are (R, s). ECDSA includes some additional information that is appended by the signer. However, it reduces verification time by 40% while increasing the signature size.

Table I
Representation of Terminology through Symbols

| Symbol | Description |
|---|---|
| Q | The order of elements in a prime field Fq |
| E(Fq) | Over a prime field Fq, an elliptic curve is defined |
| P | Base point over curve E(Fq) |
| N | The prime order of point P |
| h(m) | Message(m) hash code |
| (m,r,s) | Signature parameters of ECDSA |
| (m,R,s) | Signature parameters of modified ECDSA |
| T | Signature verification batch size |
| H | The cofactor $\frac{\lvert E(F_q)\rvert}{n}$ |
| Gen | Key generation function |
| Sign | Signature generation function |
| Verify | Signature verification function |
| L | Security parameter |
| Pk | Public key |
| Sk | Private key |

## V. The use of digital signatures in WSN

Researchers have devoted a great deal of attention to security in WSNs since data security has been the primary concern in WSNs over the last few years. It is possible for intruders to take hold of sensor nodes and upload illegitimate data. Authentication plays a vital role in prohibiting unauthorized access of data in WSN. For safety and security purposes, authentication of sensor nodes is required by digital signature algorithms. These signature schemes are represented symbolically in Table I.

Assume that (*Gen*, *Sign*, *Verify*) is a digital signature technique with security parameter l, batch size *t* and $n \in poly(l)$. An input security parameter $1^n$ is passed to Gen function and a result is generated as (*pk*,*sk*). In this case, pk stands for a public key, and *sk* stands for a private key of length *n*. Sign is a signature generation algorithm which accepts private key sk and message m as input and produce signature s as an output (i.e., $s \leftarrow Sign_{sk}(m)$). *Verify* is a signature verifier function, which takes public key *pk*, signature s and message m as input and produces *v* as an output. (i.e., $v \leftarrow Verify_{pk}(m,s)$ ). the signature s is examined to be valid if and only if $v = 1$. In a batch with a size of *t*, the public key *PK*= (*pk1*, *pk2*, *pk3*…. *pkt*) and private key *SK*= (*sk1*, *sk2*, *sk3*…...*skt*) are generated by generator function *Gen*. The following conditions must be met by the batch verification algorithm to identify valid or invalid signatures.

If $pk_i \in PK$ and $Verify_{pk_i}(m_i, s_i) = 1$ for all $i \in (1,n)$

then $Batch((pk_1,m_1,s_1),(pk_2,m_2,s_2),,,,,(pk_n,m_n,s_n)) = 1$ and considered to be valid.

If $pk_i \in PK$ for all $i \in (1,n)$ and $Verify_{pk_i}(m_i, s_i) = 0$

for some $i \in (1,n)$ then

$Batch((pk_1,m_1,s_1),(pk_2,m_2,s_2),,,,,(pk_n,m_n,s_n)) = 0$ and considered to be invalid.

The batch verification schemes verify multiple signatures at once. According to Karati et al. [14], ECDSA signatures can be verified in batches. This technique, however, is vulnerable to forgery attacks. The basic approach for verifying ECDSA signatures issued by single and multiple signers is defined by eq. 2 and 3.

The batch verification algorithms verify the signed signature using the following three types:

- Type-1: A single signer generates signatures for multiple messages (*m1*, *m2*, *m3*...*mt*) using his private key (*sk*). These signatures are all validated at the same time in a batch of t signatures (*s1*, *s2*...*st*).
- Type-2: Multiple message (*m1*, *m2*, *m3* …*mt*) have been signed by multiple signers using their private key. The batch verification algorithm is being used for verification of signatures (*s1*, *s2*…*st*) at once, where signatures are generated by n different signers $(2 \le n \le t)$.
- Type-3: Signatures that do not fit into either type-1 or type-2 can be classified as Type-3.

## VI. ECDSA Batch Verification

This study considered multiple signatures signed by the ECDSA algorithm. Various types of ECDSA signature verification algorithms have been proposed for batch verification of signatures. This research tried to examine three variants of the ECDSA batch verification algorithm to verify the existence of invalid signatures. If batch verification is found to be invalid, then entire signatures in a batch need to be verified individually to find the location of those invalid signatures. Table II explains the steps of algorithm-A1. The algorithm-A1 is a standard ECDSA batch verification algorithm. For *t*-signed messages $(M_i, r_i, s_i)\forall_{i=1,2,3,.....t}$ must need to satisfy the eq. 2.

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

If the entire signatures in a batch are signed by the same signer, then *Q1*= *Q2*= *Q3*= *Q4*= *Q5*=…….. *Qt*= *Q*

$$\sum_{i=1}^{t} R_i = (\sum_{i=1}^{t} u_i)P + (\sum_{i=1}^{t} v_i)Q \tag{3}$$

The number of scalar multiplications from 2t to [2, t+1] is minimized by computing two sides of eq. 2 and eq. 3, where 2 represents the state where the signature is signed by the same signer and *t*+1 represents the state where the signature is signed by a *t*-different signer. From the signatures, only the *x*-coordinate of *Ri* can be determined. In general, a given *x*-coordinate equates to two *y*-coordinates. Finding these *y*-coordinates requires a time-consuming square root modulo *q* calculation.

There are generally *2t* solutions of $\sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3....t}$. If any of these solutions satisfy eq. 2, the algorithm accepts the batch signatures; otherwise, it rejects the entire signatures. Step-7 of the algorithm-A1 executes *2t* times to check *2t* possible combinations for batch

verification, which is a very costly operation unless the value of t is small.

Table II
Algorithm-A1: Signature Verification

Input: Message *M1*, *M2*, *M3*……..*Mt*, thier signature parameters (*r1*, *s1*), (*r2*, *s2*), (*r3*, *s3*) ……..(*rt*,*st*) and public key *Q1*,*Q2*,*Q3*,……*Qt* accordingly
Output: Accepts/Rejects entire signatures
Begin:

Step-1: Find $w_i = s_i^{-1} \bmod n \forall_{i=1,2,3.....t}$

Step-2: Find $u_i = H(M_i)w_i \bmod n \forall_{i=1,2,3.....t}$

Step-3: Find $v_i = r_i w_i \bmod n \forall_{i=1,2,3.....t}$

Step-4: Find $R' = (\sum_{i=0}^{t} u_i)P + \sum_{i=0}^{t} v_i Q_i \in E(F_q) = R' = (\sum_{i=0}^{t} u_i)P + \sum_{i=0}^{t} v_i Q \in E(F_q)$ if all the signatures belong to same signer

Step-5: If $r_i^3 + ar_i + b, \forall_{i=1,2,3....t}$ is not equal to zero and if the i-th signature is not a quadratic residue modulo q, reject it and remove it from the batch.

Step-6: Find $\sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3....t}$

Step-7: For each $\sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3....t}$

check if $R' = \sum_{i=1}^{t} (r_i y_i)$ accepts entire signature
Step-8: Otherwise rejects entire signatures
End;

*A. Modified ECDSA batch verification*

A batch verification algorithm (algorithm-A2) has been developed by concerting eq. 2 or eq. 3 to a form which eliminates the problem associated with *y*-coordinates of *Ri*. Table III shows the steps of the modified batch verification algorithm-A2. The right side of these equations has been computed efficiently. The left side of these equations cannot be solved explicitly, but symbolically with unknown values of *y1*, *y2*, *y3*... *yt* (*y*-coordinates of *Ri*). This paper finds sufficient information for verification of t signatures simultaneously by solving the system of linear equations. This new algorithm (algorithm-A2) is faster than algorithm-A1 for smaller batch sizes ($t \leq 8$).

Suppose, *Ri*=(*xi*,*yi*), *xi*=x*(Ri)=ri=ri+n*, the *y*-coordinate *yi=y(Ri)* is not known , Therefore it is required to find the value of *yi* from the eq. 4.

$$y_i = \sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3,...t} \tag{4}$$

Consider the point *Ri* = (*ri*, *yi*) on the elliptic curve *E*. Two non-zero points on the elliptic curve *E* are *P1* = (*w1*, *z1*) and *P2* = (*w2*, *z2*).

Where $P1 \neq \pm P2$, the point *P3=P1+P2* = (*w*, *z*) is the third point on curve *E* computed as:

$$\lambda = (z_2 - z_1)/(w_2 - w_1) \tag{5}$$

$$w = \lambda^2 - w_1 - w_2 \tag{6}$$

$$z = \lambda(w_1 - w) - z_1 \tag{7}$$

By using the above formula repeatedly, the point R can be represented as: $R = \sum_{i=1}^{t} R_i$

$$R = \left(\frac{g_x(y_1, y_2, ....y_t)}{h_x(y_1, y_2, ...y_t)}, \frac{g_y(y_1, y_2, ....y_t)}{h_y(y_1, y_2, ...y_t)}\right) \tag{8}$$

Where *gx*, *hx*, *gy* and *hy* are polynomials in Fq[*y1*, *y2*,…..,*yt*]. Assume that the *yi*-degree of these polynomials is less than or equal to 1 for all *i*=1, 2…t. The denominator is represented as $h_x(y) = u(y_2, y_3, ...y_t)y_1 + v(y_2, y_3, ...y_t)$. Now multiply *gx* and *hx* by the denominator. As a result, *y1* can be removed using eq. 3, and the same process may be repeated for *y2*, *y3*, *yt*, allowing the point *R* to be represented as a pair of polynomials.

$$R = (R_x(y_1, y_2, ..y_t), R_y(y_1, y_2, ..y_t)) \tag{9}$$

The x and y co-ordinates of R can be computed using the right side of eq. 2 and eq. 3 as $R = (\alpha, \beta) \exists \alpha, \beta \in F_q$. It produces two multivariable equations as shown below:

$$R_x(y_1, y_2, ..y_t) = \alpha \tag{10}$$

$$R_y(y_1, y_2, ..y_t) = \beta \tag{11}$$

The algorithm-A2, uniquely find the value of *Ri* with *x(Ri)=ri*. This combination does not find the value of $\sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3....t}$ in Fq and it also eliminates the need for calculating $R' = u_i P + v_i Q$ which is necessary for individual verification. The step-13 of the algorithm-A2 assures that the reconstructed point must lie on the curve. The point reconstruction process works efficiently for small batch sizes, but it is very difficult to find individual invalid signatures. Whenever the algorithm fails in step-13, then the algorithm restarts from the beginning and repeats the same on sub-batches or rearranges for individual verification.

The value of $R = (R_x, R_y) = \sum_{i=0}^{t} R_i$ is symbolically calculated. The multivariate equation $\phi = R_x - \alpha = 0$ (linear individual in *yi*) has been considered for eliminating the value of *yi*. The value of *y1* is eliminated first and substituting the value in the equation $y_i = \sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3....t}$. This paper obtained a multivariable equation in *y1*, *y2*, *yt* and it is linear for every variable. A multivariate equation in *y3*, *y4*... *yt* is obtained by eliminating the value of *y2*. This operation is repeated until all of the variables *y1*, *y2*, *y3*... *yt* are gone. If the polynomial reduces to zero after all variables are removed, then the original equation $R_x = \alpha$ is consistent with respect to $y_i = \sqrt{r_i^3 + ar_i + b}, \forall_{i=1,2,3....t}$. Similarly, *y1*, *y2*, *y3*….*yt* are eliminated from $\phi = R_x - \beta = 0$ also, but this is not necessary because *yi* uniquely up to multiplication by $\pm 1$. The digital signature algorithm is divided into three stages: key generation,

signature generation, and signature verification. As a result, parallelism has been implemented in the signature verification stage. A single signer or multiple signers sign the signatures created for various communications.

The algorithm-A2 is suitable for smaller batch sizes (i.e., $t$<9) and its complexity increases with an increase in batch size, so it is not feasible for batches of size ≥9. Therefore, a relatively more efficient algorithm has been constructed as shown in table IV (algorithm-A3).

Table III
Algorithm-A2: Signature Verification

Input: Message *M1*, *M2*, *M3*……...*Mt*, thier signature parameters (*r1*, *s1*), (*r2*, *s2*), (*r3*, *s3*) ……...(*rt*,*st*) and public key *Q1*,*Q2*,*Q3*,……*Qt* accordingly
Output: Accepts/Rejects entire signatures
Begin:

Step-1: Find $w_i = s_i^{-1} \bmod n \forall_{i=1,2,3.....t}$

Step-2: Find $u_i = H(M_i)w_i \bmod n \forall_{i=1,2,3.....t}$

Step-3: Find $v_i = r_i w_i \bmod n \forall_{i=1,2,3.....t}$

Step-4: Find $R' = (\sum_{i=0}^{t} u_i)P + \sum_{i=0}^{t} v_i Q_i \in E(F_q) = R' = (\sum_{i=0}^{t} u_i)P + \sum_{i=0}^{t} v_i Q \in E(F_q)$
if all the signatures belong to same signer
Step-5: Rearrange to individual verification if *R'* is a point at infinity or *y(R')* =0, otherwise continue.
Step-6: Let *Ri*=(*xi*,*yi*) for all i=1,2,3,…t

Step-7: Calculate $R = (R_x, R_y) = \sum_{i=0}^{t} R_i$ symbolically using the sub-solution $y_i = \sqrt{r_i^3 + ar_i + b}$ , polynomial Rx and Ry are linear equation to each yi.
Step-8: Formulate the equations $R_x = \alpha$ and $R_y = \beta$ where $R' = (\alpha, \beta)$
Step-9: Create the equation $\mu - 1 = 2^{t-1} - 2$ and more equations are generated by repeated squaring of $R_x = \alpha$ or by repeated multiplication of even degree monomials (*y1*, *y2*, *y3* …*yt*) and substitute the value of $y_i = \sqrt{r_i^3 + ar_i + b}$
Step-10: Resolve the linear system of equation to find the value of all even degree monomials of *y1*, *y2*, *y3*….*yt*. When the equation is unsolvable, re-arrange the batch of signature to individual verification

Step-11: Resolve the equation $R_y = \beta$ for *y1* by multiplying both sides by *y1*.
Step-12: Find *yi*=*y1yi*/*y1* from the monomials *y1yi* for all i=1,2,3….t (If the value of *y1*=0 solve for *y2*, *y3*….*yt* in step 11).

Step-13: Accepts all signatures only if $y_i = \sqrt{(r_i^3 + ar_i + b) \bmod q}, \forall_{i=1,2,3,...t}$

End;

The method for obtaining x and y values from multivariate equations has been updated to make it easier to identify *y's* sign. An equation $u_i^2(r_i^3 + ar_i + b) - v_i^2$ is used to eliminate the values of *y1*, *y2*...*yt* from the first multivariate equation. In the case of elliptic curve cryptography, if the equation is reduced to zero, the original equation can still be used (i.e., $y^2 = x^3 + ax + b$ ,where $a, b \in Z_p$ and $4a^3 + 27b^2 = 0$ ).

The algorithm-A3 works efficiently for batch sizes up to 8 (i.e., t≤8).

Table IV
Algorithm-A3: Signature Verification

Input: Message *M1*, *M2*, *M3*……...*Mt*, thier signature parameters (*r1*, *s1*), (*r2*, *s2*), (*r3*, *s3*) ……...(*rt*,*st*) and public key *Q1*,*Q2*,*Q3*,……*Qt* accordingly
Output: Accepts/Rejects entire signatures
Begin:

Step-1: Find $w_i = s_i^{-1} \bmod n \forall_{i=1,2,3.....t}$

Step-2: Find $u_i = H(M_i)w_i \bmod n \forall_{i=1,2,3.....t}$

Step-3: Find $v_i = r_i w_i \bmod n \forall_{i=1,2,3.....t}$

Step-4:Find $R' = (\sum_{i=0}^{t} u_i)P + \sum_{i=0}^{t} v_i Q_i \in E(F_q) = R' = (\sum_{i=0}^{t} u_i)P + \sum_{i=0}^{t} v_i Q \in E(F_q)$
if all the signatures belong to same signer
Step-5: Rearrange to individual verification if *R'* is a point at infinity or *y(R')* =0, otherwise continue.

Step-6: If $r_i^3 + ar_i + b, \forall_{i=1,2,3....t}$ is not equal to zero and the residue modulo *q* is not quadratic, reject the *i*-th signature and delete it from the batch
Step-7: Assuming *Ri*=(*xi*,*yi*) for all i=1,2,3,…t

Step-8: Symbolically calculate $R = (R_x, R_y) = \sum_{i=0}^{t} R_i$ using the sub-solution $y_i = \sqrt{r_i^3 + ar_i + b}$

Step-9: Assuming $\phi = R_x - \alpha$

Step-10: Eliminate *yi* from $\phi$ for *i*=1,2, 3...t-1, using below steps

Step-10a: Rearrange for individual verification if $\phi$ is equivalent to zero, otherwise write $\phi = uy_i + v$ where *u* and *v* are polynomials in $y_{i+1}, ...y_t$
Step-10b: Rearrange for individual verification if u is equivalent to zero polynomial. Set $\phi = u^2(r_i^3 + ar_i + b) - v^2$ and substitute $y_j^2$ in $\phi$ by $r_j^3 + ar_j + b, \forall j = i+1, ....t$

Step-11: Accepts entire signatures only if $\phi$ =0
End;

## VII. Experimental Setup

The experiment has been done on a rock cluster Linux-based open-source distribution. The cluster comprises seven workstations, each with two CPUs and 20 cores. As a result, a 7-node cluster with 140 cores has been used for simulation. Each core has a processor that runs at 2.3 GHz. One of the workstations acts as a master among the seven workstations, and the remaining work as slaves. The master workstation distributes the load over the six workstations using the MPI library standard. The computational results of all parallel workstations are accumulated, and the master workstation calculates the final results. The battery power of WSN nodes is limited, performance is limited, communication bandwidth is narrow, memory is limited, and computation capacity is low. One of the primary goals of this research is to minimize the ECDSA computation load

on a single node during signature verification. Through parallel configuration, the computational load is distributed among the available nodes, reducing the calculation overhead. The proposed batch verification system is designed and built using a 7-node cluster setup. The cluster nodes have large computing power than gateway node. Fig. 6. depicts a load distribution scenario in a WSN. When gateway nodes receive batches of digital signatures for verification, they identify other freely available gateway nodes that have enough computing power. It distributes a set of signatures to every node after the identification process. As a result, the available nodes are slaves while the distributing node is a master. In fig. 7, the master node distributes the load over the entire slave node. Each slave node gets a set of signatures for verification, and the verification results are sent back to the master node. Public key information is available on all the nodes. If an invalid signature or signatures appear on any slave node, the batch verification procedure at that node fails. It has an advantage over serial processing, which requires individual verification of the entire batch to find the incorrect signature when an invalid signature occurs.

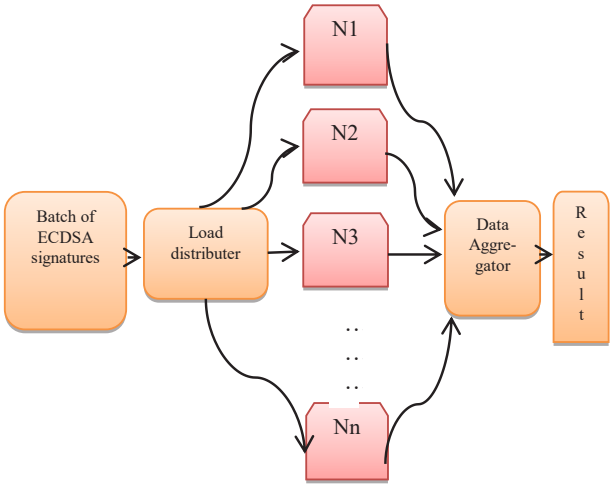


Fig. 6. Load distribution workflow

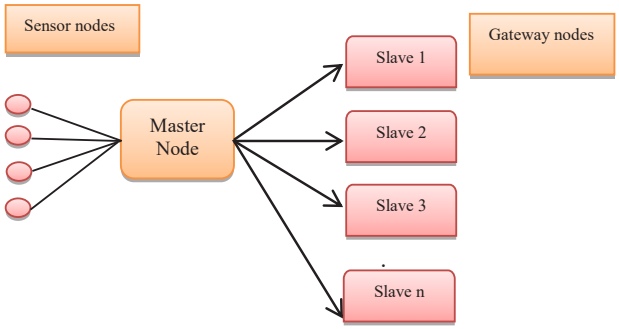


Fig. 7. Batch signature verification model

## VIII. Experimental result analysis

This paper has taken batch sizes of 2, $2^2$, $2^3$, $2^4$, $2^5$, and $2^6$. The algorithms are executed in parallel on a cluster of seven nodes. Table V provides the running time of the algorithm-A1 on seven sensor nodes. The verification time remains almost the same for all seven nodes for batch sizes up to $2^4$. Table V shows that as batch size increases beyond $2^4$, running time increases exponentially. However, algorithm-A1 is faster than the individual verification of the ECDSA algorithm.

Table V
Verification Time of The Algorithm-A1 on a Cluster of 7-Nodes (in ms)

| Block Size | Individual verification | N1 | N2 | N3 | N4 | N5 | N6 | N7 |
|---|---|---|---|---|---|---|---|---|
| 2 | 12.05 | 1.23 | 1.23 | 1.23 | 1.23 | 1.23 | 1.23 | 1.23 |
| $2^2$ | 16.11 | 1.65 | 1.65 | 1.65 | 1.65 | 1.65 | 1.65 | 1.65 |
| $2^3$ | 20.17 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 |
| $2^4$ | 22.23 | 2.62 | 2.62 | 2.62 | 2.62 | 2.62 | 2.62 | 2.62 |
| $2^5$ | 65.73 | 46.11 | 428.63 | 396.35 | 367.12 | 335.33 | 292.41 | 230.1 |
| $2^6$ | 98.73 | 1245.27 | 1200.16 | 1128.47 | 968.38 | 763.72 | 562.35 | 396.1 |

Table VI
Verification Time of The Algorithm-A2 on a Cluster of 7-Nodes (in ms)

| Block Size | Individual verification | N1 | N2 | N3 | N4 | N5 | N6 | N7 |
|---|---|---|---|---|---|---|---|---|
| 2 | 13.01 | 1.21 | 1.21 | 1.21 | 1.21 | 1.21 | 1.21 | 1.21 |
| $2^2$ | 16.20 | 1.64 | 1.64 | 1.64 | 1.64 | 1.64 | 1.64 | 1.64 |
| $2^3$ | 20.31 | 1.97 | 1.97 | 1.97 | 1.97 | 1.97 | 1.97 | 1.97 |
| $2^4$ | 22.81 | 2.98 | 2.98 | 2.98 | 2.98 | 2.98 | 2.98 | 2.98 |
| $2^5$ | 63.41 | 48.42 | 432.14 | 400.12 | 371.21 | 339.21 | 295.15 | 233.1 |
| $2^6$ | 97.09 | 1249.31 | 1204.11 | 1133.30 | 973.11 | 767.23 | 566.12 | 406.1 |

Table VII
Verification Time of The Algorithm-A3 on a Cluster of 7-Nodes (in *ms*)

| Block Size | Individual verification | N1 | N2 | N3 | N4 | N5 | N6 | N7 |
|---|---|---|---|---|---|---|---|---|
| 2 | 12.21 | 1.10 | 1.10 | 1.10 | 1.10 | 1.10 | 1.10 | 1.10 |
| $2^2$ | 15.10 | 1.53 | 1.53 | 1.53 | 1.53 | 1.53 | 1.53 | 1.53 |
| $2^3$ | 17.13 | 1.93 | 1.93 | 1.93 | 1.93 | 1.93 | 1.93 | 1.93 |
| $2^4$ | 23.41 | 2.71 | 2.71 | 2.71 | 2.71 | 2.71 | 2.71 | 2.71 |
| $2^5$ | 64.30 | 46.44 | 430.23 | 398.32 | 369.50 | 337.32 | 293.01 | 231.5 |
| $2^6$ | 97.51 | 1247.12 | 1202.10 | 1131.82 | 970.45 | 765.47 | 564.30 | 398.7 |

Table VI indicates the verification time of algorithm-A2 on seven nodes of WSN. As the batch size grows, the verification time grows as well. For batch sizes up to $2^4$, the verification time remains almost the same for all seven nodes. It is clearly illustrated that the performance gain for seven machines is almost six-six times more efficient than the individual verification for the batch size up to $2^4$. Table VII shows the verification time of algorithm-A3 on seven node configurations of WSN. The performance of algorithm-A3 is better than the performance of algorithm-A1 and algorithm-A2, because of the exponentiation operation of these algorithms. The algorithm-A3 is more secure than A1 and A2.

*A. Computational Time of Batch Verification Algorithm*

In this paper, a comparative study of the computation time of all existing batch verification algorithms has been performed on a cluster of seven nodes in WSN. All of the parameters used in this scenario are NIST (National Institute of Standards and Technology)-recommended standard parameters. For the experiment, three NIST primary curves have been used: (P-192), (P-224), and (P-256). On elliptic curves (P-192), (P-224), and (P-256), the verification time of these techniques is investigated using single and multiple signers.

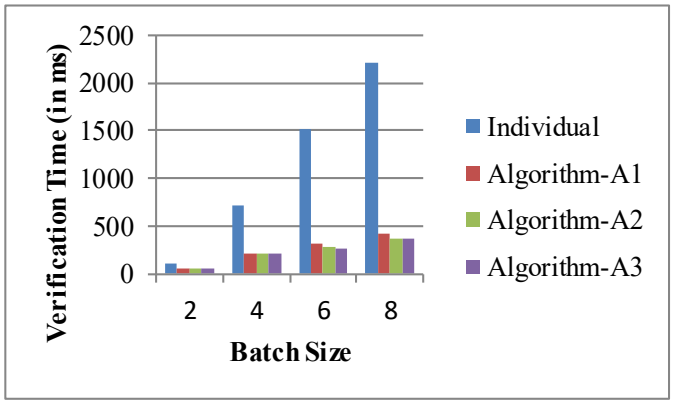


Fig. 8. Single signer verification time for curve (P-192)

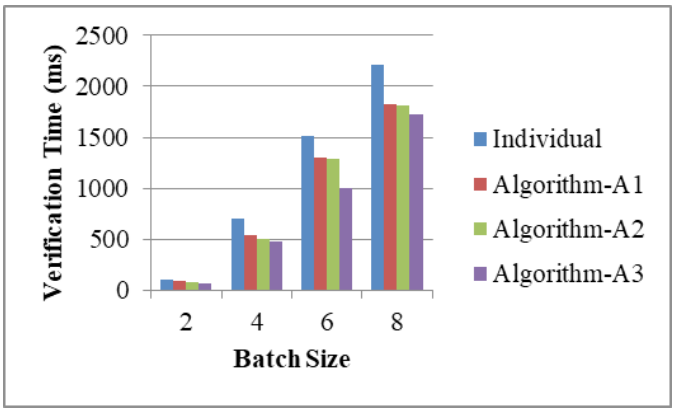


Fig. 9. Multiple signer verification time for curve (P-192)

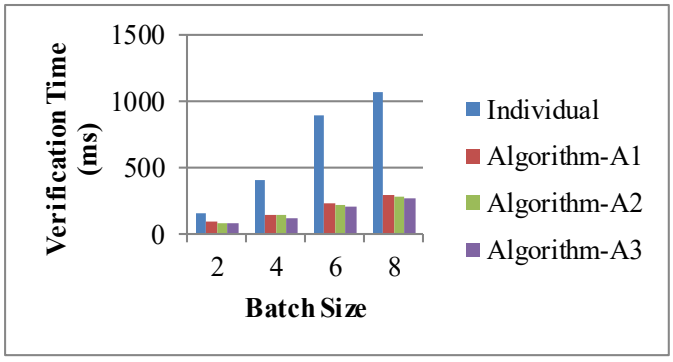


Fig. 10. Single signer verification time for curve (P-224)

Figs. 8-9 show the time taken by these algorithms to verify batches of signatures on an elliptic curve (P-192). These algorithms work more efficiently as compared to individual verification. However, the performance of algorithms-A1, A2, and A3 decreases drastically and becomes incomparable for larger batch sizes (8). The increase in the number of scalar and modular multiplication operations increases the overhead of these methods. These algorithms, on the other hand, maintain their performance throughout a wide range of batch sizes. It can be observed that these algorithms perform better for single signers. The performance of these algorithms has been decreased for multiple signers because of an increase in execution time.

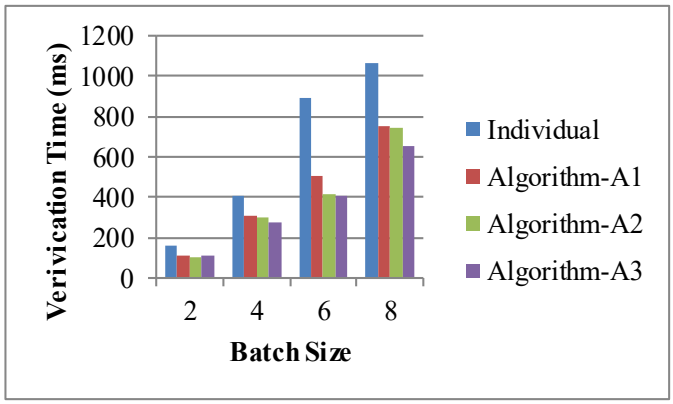


Fig. 11. Multiple signer verification time for curve (P-224)

Figs. 10-11 show the outcome of curve (P-224). The performance of these algorithms is better than the performance of individual verification algorithms. However, these algorithms work efficiently for small batch sizes (8) and beyond that, the performance decreases drastically. It is discovered that algorithm-A3 has the greatest speedup, followed by algorithms-A2 and A1.

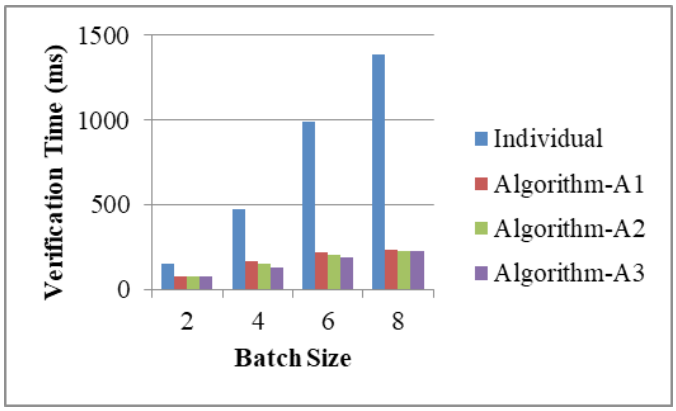


Fig. 12. Single signer verification time for curve (P-256)

Compared to other curves, the curve (P-256) produces similar results. The verification time for single and multiple signers of these algorithms is illustrated in figs. 12-13. The relative performance of these algorithms remains the same as that of curves (P-192) and (P-224).

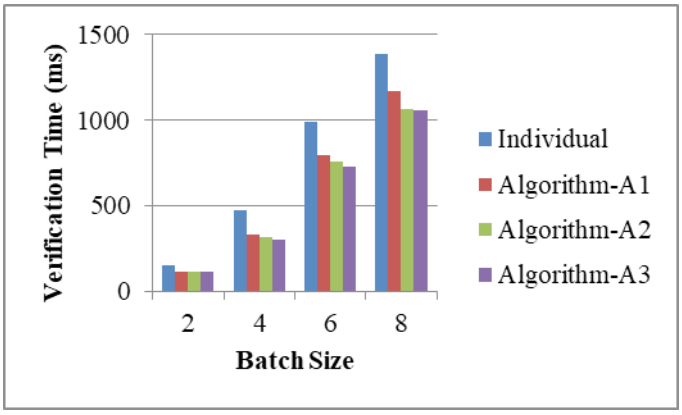


Fig. 13. Multiple signer verification time for curve (P-256)

TABLE VIII
ADVANTAGES AND DISADVANTAGES OF BATCH VERIFICATION ALGORITHMS

| Algorithms | Efficient for type of signer | Efficient for batch size(t) | Advantages | Disadvantages |
|---|---|---|---|---|
| A1 | Multiple and single signer | t ≤ 7 | It is efficient for small batches of single and multiple signers | As the batch size grows larger, efficiency drops abruptly. |
| A2 | Single signer | t ≤ 7 | It is efficient for small batches of single and multiple signers | Increase in batch size leads to a sudden decrease in efficiency |
| A3 | Single signer | 6 ≤ t ≤ 8 | Efficient for single signer for small batch size | As the batch size grows larger, efficiency drops abruptly. |

Table VIII shows the behavior of these batch verification schemes. This table also discusses the advantages and disadvantages of these algorithms.

*B. Time and Space Complexity of Batch Verification*

The computational time complexity of these strategies is calculated using the cost of computing relevant points and the cost of signature verification time. If an ECDSA signature algorithm generates $\tau$ acceptable candidate points, then the algorithm's es timated time complexity is $\frac{(\tau+1)}{2} ECDSA * Verifier$. By taking cofactor value $h$=1, $\tau$ =2 and non-availability of side information, the average time is calculated as $\frac{3}{2} ECDSA * Verifier$. The average time complexity of these algorithms is less than that of the traditional verification algorithm. If the side information is known, then $\tau$ =1 is used, and just one ECDSA verification is needed. It can be observed that the *R*-value can be chosen from a maximum of 2 ($h$ + 1) options. Therefore, the most favorable case is where $\tau$ =1. Algorithm-A1 requires calculating t-modular square roots in the field Fq. Each square root computation involves exponential time complexity. Subsequently this algorithm requires verifying at most $m = 2^t = 2(t+1)$ conditions. This algorithm requires computing the sum of *t*-points on the curve to verify every condition. Therefore, the entire time complexity of algorithm-A1 is $O((\sigma+m)t)$ where $\sigma$ denotes the time complexity of calculating one square root in the field Fq. The exponential time complexity with *t*-batch size has been taken for generation and solution stage of the equation $\mu - 1 = 2^{t-1} - 2$. The algorithm-A1 needs *t*-1 symbolic additions having a rational function with at most $\theta(m)$ non-zero terms to solve the eq. 10 and eq. 11. Each symbolic addition requires at most *t*-times uses of the eq. 4. Therefore $O(mt^2)$ operation is required for symbolic derivation of *R* in the field Fq. The successive generation of $\mu \times \mu$ linear system requires $O(m^2 t)$ operation in the field Fq. The $\mu \times \mu$ system requires $\theta(m^3)$ field operations for Gaussian elimination. $O(mt^2)$ field operations are required for individual value of *yi*. The execution time of algorithm-A2 is dominated by solving stage of linear systems. Therefore, algorithm-A2 is not practical for the larger batch size (*t*). The storage of coefficient matrix of algorithm-A2 consumes the majority of memory. The coefficient matrix dimension is $\mu \times \mu$. Therefore, the space complexity is $\theta(\mu^2) = \theta(2^{2t})$. The algorithm-A3 requires $O(mt^2)$ field operations as require in algorithm-A2 for the symbolic computation of (*Rx*, *Ry*). Consequently, to solve polynomial $\phi = R_x - \alpha$ require at most $\mu + 1 = \frac{m}{2} + 1$ non-zero terms. Calculation of *u2* and *v2* requires solving the equation $\phi = u^2(r_i^3 + ar_i + b) - v^2$ and $t - i$ substitutions of *y2* by $r_j^3 + ar_j + b, \forall j = i+1, \ldots t$ for elimination of *yi*. Reduction of $\phi$ requires $O(mt^2)$ operations, which is substantially better than $O(m^3)$ (as in algorithm-A2). The major storage requirement of the algorithm-A3 is for multivariate polynomial $\phi$ and it has *m/2* non-zero elements. So, the space complexity of the algorithm-A3 is $\theta(2^t)$.

*C. Security Analysis of Batch verification*

Suppose an attacker able to satisfy one of the *m=2t* conditions in algorithm-A1 with known value of *r1*, *r2*, *r3*…*rt*. The right choice of *yi* of $\sqrt{r_i^3 + ar_i + b}, \forall_{i=1,2,3,\ldots t}$ constitutes a case to fulfil the ECDSA batch verification criteria. An attacker can reconstruct point *Ri* in algorithm-A2 with *x*-coordinates of *x(Ri)=ri* by satisfying the condition $R = \sum_{i=1}^{t} R_i = \sum_{i=1}^{t} R_i^{'} = R^{'}$ where $R^{'} = u_i P + v_i Q$. Suppose an attacker force the condition $R = R^{'}$ then there exists a singular solution for the respective value of the *y*-coordinates *y1*, *y2*, *y3*…..*yt* of *R1*, *R2*, *R3*….. *Rt*. If the value of batch size (*t*) is very small (i.e., *t*≤4) then an attacker requires only moderate computing resources to find the value of *y1*, *y2*, *y3*…..*yt* uniquely. As a result, an intruder simply needs to give the *x*-coordinates (*ri*). An adversary essentially requires knowing the full points of *Ri*. This satisfies the normal batch verification of ECDSA algorithm; therefore, an attacker can make a fool of algorithm-A2. In the case of algorithm-A3, assume an attacker discovers *x*-coordinates (*r1*, *r2*, *r3*...*rt*)

of ECDSA and moves it to batch verification. Step-9 of algorithm-A3 states that given $\phi = 0$ $y_i = \sqrt{r_i^3 + ar_i + b}, \forall_{i=1,2,..t}$, there are exactly two solutions (*y1*, *y2*, *y3 ...yt*) and (*-y1*, *-y2*, *-y3*.....*-yt*). One of these values may correspond to a standard ECDSA signatures algorithm based on known value of *r1*, *r2*, *r3…rt*. This solution would satisfy the equation $R_y = \beta$. To conclude an attacker can falsify the standard batch verification of ECDSA. This falsifying process requires only a moderate amount of computing resources as long as the value of batch size (t) is small.

## IX. Conclusions and Future Work

Authentication and security are the most important features of sensor nodes to ensure data integrity and privacy. A sensor node must digitally sign all critical information before sending it to a sink node. The individual verification of a digital signature by the sink node is a time-consuming process. Therefore, the idea of batch signature verification techniques has come into the picture to reduce the time requirement for verification of millions of signatures. It is very important to have a secure and fast signature verification scheme in real-time wireless sensor networks. This study concludes that the use of batch signature verification algorithms can lead to some sort of speedup gained as compared to individual verification. The computational time and space complexity of these algorithms (A1, A2, and A3) have been compared and analyzed. This paper analyses the execution time of three different types of ECDSA-based batch verification algorithms on seven-node cluster systems in WSNs. A comparison of the verification times of these algorithms has been conducted on three NIST primary curves (P-192), (P-224), and (P-256) for both single and multiple signers. It has been observed that these algorithms work efficiently for batches of a maximum size of 8. Moreover, the security level of these algorithms. Sensor nodes with low batteries consume more energy when running a digital signature algorithm. Multi-hop WSNs using digital signatures incur additional overhead due to network interface costs. The security protocol should therefore be designed to minimize delay times as much as possible. In this paper, an extensive range of batch digital signature algorithms is evaluated and compared invariably. An appropriate security protocol can be designed and evaluated with the help of this comparison. As a part of WSN research themes, researchers can examine how to secure and enhance performance. In the future, some real-time applications of wireless sensor networks can be simulated using these batch verification techniques. The execution time of algorithm-A3 can be reduced from today to in the future. The symbolic computation algorithm runs with time complexity. The design of the algorithm would be useful for getting higher speedup. In summary, all the algorithms proposed in this paper are suitable for WSN applications with limited computation cost, storage capacity, and bandwidth, such as healthcare sensor networks, terrestrial wireless sensor networks, and military surveillance based on WSN. The trade-off between WSN's resource limitations, such as its small battery energy, limited memory, and less bandwidth, and the use of digital signatures within WSN may be resolved in the future. Although the research is only limited to batches of 8, it could be expanded in the future to larger batches. A cluster of 7 sensors is taken into account for simulation, but it could be extended to larger number of sensors in the future.

## References


[1] L. Harn, "Batch verifying multiple rsa digital signatures," *Electron Lett*, vol. 34(12), pp. 1219-1220, 1998.

[2] S.W. Changchien, M-S. Hwang, K-F. Hwang, "A batch verifying and detecting multiple rsa digital signatures," *International Journal Comput Numer Anal Appl*, vol. 2(3) pp. 303–307, 2002.

[3] C-T. Li, M-S. Hwang, S. Chen, "A batch verifying and detecting the illegal signa- tures," *International Journal Innovative Comput Inf Control,* vol. 6(12)*,* pp.5311–5320, 2010.

[4] Y. Ren, S. Wang, X. Zhang, M-S. Hwang, "An efficient batch verifying scheme for detecting illegal signatures," *International Journal of Network Security,* vol. 17(4), pp. 463–470, 2015.

[5] L. Seungwon, C. Seongje, C. Yookun, "Efficient identification of bad signatures in rsa-type batch signature," *IEICE Transaction Fundamental Electronic Communication Computer Science,* vol. 89(1), pp. 74–80, 2006.

[6] C-H. Lin, R-H. Hsu, L. Harn, "Improved dsa variant for batch verification," *Applied Mathematics and Computation,* vol. 169(1), pp. 75–81, 2005.

[7] J. Pastuszak, D. Michałek, J. Pieprzyk, J. Seberry, "dentification of bad signatures in batches," *In International Workshop on Public Key Cryptography*, pp. 28-45, 2000, Springer, Berlin, Heidelberg.

[8] DJ. Guan, ES. Zhuang, IC. Chung, YS. Lin, "Performance analysis of some batch verification methods of digital signatures," *In 12th Asia Joint Conference on Information Security (AsiaJCIS)*, pp. 10-14, 2017, IEEE.

[9] N. Koblitz, "Elliptic curve cryptosystems," *Mathematical Computation*, vol. 48(177), pp. 203-209, 1987.

[10] VS. Miller, "Use of elliptic curves in cryptography," *In Conference on the theory and application of cryptographic techniques*, pp. 417-426, 1985, Springer, Berlin, Heidelberg.

[11] JH. Cheon, JH. Yi, "Fast batch verification of multiple signatures," *In: International workshop on public key cryptography*," *Springer*, pp. 442-457, 2007, Springer, Berlin, Heidelberg

[12] S. Karati, A. Das, D. Roychowdhury,B. Bellur, D. Bhattacharya, A. Iyer, "Batch verification of ecdsa signatures," *In International conference on cryptology in Africa, Springer,* pp. 1–18, 2012.

[13] A. Yang, X. Tan, J. Baek, and D. S. Wong, "A new ADS-B authentication framework based on efficient hierarchical identity-based signature with batch verification," *IEEE Transactions on Services Computing* vol.10 (2), pp. 165-175, 2015.

[14] J. H. Cheon. J. H. Yi, "Fast batch verification of multiple signatures," *In Proceedings of the International Workshop on Public Key Cryptography*, pp. 442–457, 2007, Springer, Berlin, Heidelberg.

[15] J. Liu, Q. Li, H. Cao, R. Sun, X. Du, and M. Guizani, "MDBV: Monitoring data batch verification for survivability of Internet of Vehicles," *IEEE Access*, vol. 6, pp. 50974-50983, 2018.

[16] S. Karati, A. Das, D. Roychowdhury,B. Bellur, D. Bhattacharya, A. Iyer, "New algorithms for batch verification of standard ECDSA signatures," *Journal of Cryptography Engineering*, vol. 4 pp. 237–258, 2014.

[17] J. Liu, H. Cao, Q. Li, F. Cai, X. Du and M. Guizani, "A large-scale concurrent data anonymous batch verification scheme for mobile healthcare crowd sensing," *IEEE Internet of things Journal,* vol. 6(2), pp. 1321-1330, 2018.

[18] L. Jiang, Y. Wang, J. Tian, F. Jiang, N, Patterson and R. Doss, "An Efficient Dynamic Group-Based Batch Verification Scheme for Vehicular Sensor Networks," *In International Conference on Future Network Systems and Security*, Springer, Cham, pp. 52-64, 2019, Springer, Cham.

[19] A. S. Kittur, A. R. Pais, "Batch verification of digital signatures: approaches and challenges, *"Journal of Information Security and Applications,* no: 37, pp. 15-27, 2017.

[20] KN. Sridharan Namboodiri and I. Praveen,"An Efficient Batch Verification

Scheme for Secure Vehicular Communication Using Bilinear Pairings," *In International Conference on Soft Computing and Signal Processing*, Springer, Singapore, pp. 711-720, 2019, Springer, Singapore

[21] A. S. Kittur and A.R. Pais,"A trust model-based batch verification of digital signatures in IoT," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11(1), pp. 313-327,2020.

[22] N.B. Gayathri and P. V. Reddy, "Secure Certificateless Signature Scheme with Batch Verification from Bilinear Pairings," *In International Symposium on Security in Computing and Communication*, pp. 225-235, 2016, Springer, Singapore.

[23] X. Hu, J. Wang, H. Xu, Y. Liu and X. Zhang, "Secure and pairing-free Identity-based batch verification scheme in vehicle ad-hoc networks," *In International Conference on Intelligent Computing*, pp. 11-20, 2016, Springer, Cham.

[24] R. Jagadeesha and K. Thippeswamy, "Energy-Efficient Mobile Node in Multicast Batch Verification," *SN Computer Science*, vol. 1(2), pp. 1-6, 2020.

[25] J. Shen, D. Liu, X. Chen, J. Li, N. Kumar and P.V. Kumar, "Secure real-time traffic data aggregation with batch verification for vehicular cloud in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 69(1), pp. 807-817, 2019.

[26] H. Zhong, R. Zhao, J. Cui, X. Jiang, "An Improved ECDSA Scheme for Wireless Sensor Network," *International Journal of Future Generation Communication and Networking,* vol. 9(15), pp. 73-82, 2016.

[27] M. S. I. Mamun, A. Miyaji, "Secure VANET applications with a refined group signature," *Twelfth Annual International Conference on Privacy, Security and Trust,* 2014, IEEE.

[28] P. Thorncharoensri, W. Susilo, J. Baek, "Aggregatable Certificateless Designated Verifier Signature," *IEEE Access,* vol. 4, 2016.

[29] C. Meshram, C-C. Lee, S. G. Meshram, A. K. Meshram, "OOS-SSS: An Efficient Online/Offline Subtree-Based Short Signature Scheme Using Chebyshev Chaotic Maps for Wireless Sensor Network," *IEEE Access,* vol. *8*, pp.80063-80073, 2020.

[30] H. Du, Q. Wen, S. Zhang, "An Efficient Certificateless Aggregate Signature Scheme without pairings for Healthcare Wireless Sensor Network," *IEEE Access,* vol. *7*, pp.42683-42693, 2019.

[31] A. Testa, M. Cinque, A. Coronato, G. D. Pietro and J. C. Augusto, "Heuristic strategies for assessing wireless sensor network resiliency: an event-based formal approach," *Journal of Heuristics, Springer,* vol. *21*(2), pp.145-175, 2015.

[32] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik, and S. Vanstone, "Accelerated veri-fication of ECDSA signatures," *SAC 2005*, *LNCS*, vol. 3897,pp. 307–318, 2006.