Research Article

Invalid Signatures Searching Bitwise Divisions-Based Algorithm for Vehicular Ad-Hoc Networks

Xin Ye ¹, Gengcheng Xu ², Xueli Cheng ², Jin Zhou,¹ and Zhiguang Qin¹

¹School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China

²School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

Correspondence should be addressed to Gengcheng Xu; xugengcheng@std.uestc.edu.cn

Received 4 March 2021; Accepted 12 May 2021; Published 24 May 2021

Academic Editor: Chi-Hua Chen

Copyright © 2021 Xin Ye et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Vehicular ad-hoc networks (VANETs) are the crucial part of intelligent transportation systems (ITS), which are brought to enhance the security, efficiency, and comfort of transportation. VANETs have aroused extensive attention in the world recently. One of the challenges in practice is real time and low delay, which strongly requires VANETs to be efficient. Existing schemes have properly solved the problem which is how to aggregate the signatures and verify the aggregated signature. However, few solutions are proposed to pinpoint all invalid signatures if existing. The algorithms that can find all invalid signatures are not efficient enough. Following consideration of the above deficiencies of existing approaches, this paper proposes a factorial bitwise divisions (FBD) algorithm and its optimized version and early-stopping factorial bitwise divisions (EFBD) algorithm. Both algorithms are parallel-friendly. Compared with the binary-based batch verification algorithm, the experimental results demonstrate that the proposed algorithms achieve better performance in both theory and practice at low invalid signatures' rate. Especially, in the parallel condition, when the number of invalid signatures is 1, the proposed algorithms cost only one aggregation-verification delay, while the comparison is more than $\log_2 n$ times.

1. Introduction

Vehicular ad-hoc networks (VANETs) were created by applying the principles of the mobile ad-hoc networks (MANETs), which is a spontaneous creation of wireless networks for data exchange in the vehicle domain. In the early 21st century, VANETs were considered direct implementation of the MANETs, and since then, because of the particularity of VANETs, they have developed into an independent field of study. By 2015, the focus was still on spontaneous networking, not to mention roadside units (RSUs) or cellular networks. The vehicle that joins the network becomes a wireless node or a wireless route. It allows vehicles spaced 100 to 300 meters from each other to form a decentralized and dynamic network automatically. When one vehicle is out of range and off the network, other vehicles can join and link to each other, creating new mobile

networks [1–3]. Nowadays, VANETs have become an important part of the Internet of things (IoT) and they have a broad application prospect.

Although it seems that VANETs own a lot of merits, there are still many problems to overcome to achieve a wide application. One of the obstacles is that VANETs require strong timeliness. Existing single verification methods can verify a signature one by one. However, in the area of heavy traffic, there are too many vehicles to be fully verified, which may lead to potential hazards. Batch verification and aggregate verification emerged at the right moment. In the batch verification mode, signatures are aggregated simultaneously at first; then, the aggregated signatures need to be verified. Compared with the validation process, the aggregation process takes very little time that it can be ignored. In this approach, if the aggregate signature can be verified, then all individual signatures are considered legal. However, all

the individual signatures will be rejected when there exists one or more illegal signature, which will cause a great waste of time and cannot take full advantage of batch verification [4–6].

In recent years, some researchers try to propose a few solutions to the aforementioned problem. Some just discard all signatures or verify them individually. Huang et al. [7, 8] use binary search to solve the problem. Indeed, it is easy to understand. Huang et al. [9] proposed a matrix-detection algorithm to reduce the escape probability of bad signatures, but it only applies to a batch of signatures which has less than four or odd number of bad signatures. However, the method still fails to achieve high efficiency. Ferng et al. [10] proposed a dynamic batch verification scheme. However, it cannot fully address the problem and has a relatively low efficiency. In addition, in the binary search algorithm, the number of layers is $\log_2 n$ which is not parallel-friendly. Therefore, binary search cannot have a relatively good performance in practice.

To solve the problem, this paper proposes a factorial bitwise divisions' algorithm (FBD) and an early-stopping factorial bitwise divisions' (EFBD) algorithm. According to the index of each signature, they can be divided into a few groups; the details of the grouping strategy will be covered in Section 4. Then, each group will be verified aggregately. If the aggregate validation passes, all signatures in this group are supposed to be legitimate. Therefore, in the later aggregate verification process, we can exclude all signatures in this group. In Section 4, we show that the size of each group is half the size of the input. Thus, half of the signatures are verified at least. Eventually, we will get a set including all illegitimate signatures and $n(n \geq 0)$ good signatures. The aforementioned algorithm is recursively executed until the input and output are completely same. Later, two algorithms execute differently. FBD will reorder each group, make them as the input, and recursively execute the algorithm, whereas EFBD will singly verify all signatures in the final output set when the size of the set is less than $2 \log_2 n$. Our main contributions in this paper are as follows:

- (i) Propose FBD and EFBD algorithms which are recursive and parallel-friendly for aggregate verification in VANETs and adopt a classical bitwise method
- (ii) Through simulation experiments, the proposed algorithms are effective in a real application scenario

The rest of this paper is organized as follows. In Section 2, we will introduce the related work, mainly about how the binary search algorithm works in batch verification. In Section 3, we describe some related concepts in VANETs and the problem definition of our work. Then, we present our proposed algorithms in Section 4 in detail. Time complexity analysis is described in Section 5. Experiments are performed and analyzed in Section 6. Finally, we will draw a conclusion of this paper in Section 7.

2. Related Works

In this section, we will cover several famous grouping strategies, especially the binary search algorithm.

Heidari et al. [11] propose a novel population-based, nature-inspired optimization paradigm, which is called

Harris Hawks Optimizer (HHO). Yang et al. [12] propose a general-purpose population-based optimization technique called Hunger Games Search (HGS). It has a simple structure, special stability features, and very competitive performance to realize the solutions of both constrained and unconstrained problems more effectively. Li et al. [13] propose a new stochastic optimizer called slime mould algorithm (SMA) which is based on the oscillation mode of slime mould in nature.

To detect all invalid requests, Huang et al. [8] proposed a binary divisions' detection (BDD) algorithm, which is based on the "divide-and-conquer" approach. It is shown in Algorithm 1. When BDD is applied in the signature verification process, it can easily and efficiently find all invalid signatures, and it is easy to understand. A full BDD tree is shown in Figure 1, where the dark red and dark purple grids represent the invalid signatures and the blue group of signatures indicates that all signatures in the group are valid, and the detecting branch should be terminated.

The algorithm is described as follows. Firstly, aggregate the input and verify the aggregated signature. If the verification passes, it reveals all are valid, and the function returns an empty set; if not, it will divide the input into two batches. Then, treat each batch as the new input of the algorithm and call the BDD recursively. Once the input contains only one signature, the algorithm should be terminated. Eventually, the algorithm will return a set which exactly includes all invalid signatures.

A randomly selected test was proposed by Guan et al. [14], the approach randomly chooses half of the signature and aggregates them into a batch. Bellare et al. [15] proposed a small exponent test, which also randomly selects 1, 2, 4, ... signatures in each verification process. It improves the efficiency a little. Later, Hwang et al. [16] simplified the approach. Both are inefficient when we care about the exact invalid signatures.

3. Preliminaries

This section presents the basics related to signature verifications in VANETs. Note that this paper is based on popular and efficient schemes: certificateless signature (CLS) scheme and certificateless aggregate signature (CL-AS) scheme [4, 17]. The notations are listed in Table 1.

3.1. Elliptic Curve Cryptography. As widely used in cryptographic, elliptic curve cryptography (ECC) is an excellent algorithm which has an extremely high efficiency and a relatively excellent security [18, 19]. It can use much fewer bits, less time, and memory to encrypt messages than RSA algorithm in the field of public key cryptography. Thanks to the excellent advantages, ECC can be perfectly applied to the application scenarios of VANETs [20]. We will make the following three definitions to describe ECC.

Definition 1 (elliptic curve definition). Assume that F_p is a finite field of the module p , where p is a large prime number. The elliptic curve over a finite field F_p can be defined as

```

Input: Nonempty Set A, start index i, end index j
Output: Set F
(1)  function BDD (A, i, j)
(2)    if Verify (A, i, j) then
(3)      return  $\emptyset$ 
(4)    end if
(5)    if  $i == j$  then
(6)      return  $A[i]$ 
(7)    end if
(8)     $F = F + \text{BDD}(A, i, (i + j)/2)$ 
(9)     $F = F + \text{BDD}(A, i, (i + j)/2 + 1, j)$ 
(10)   return F
(11) end function

```

ALGORITHM 1: Binary divisions' detection algorithm.

FIGURE 1: Binary-based aggregate verification process of Huang et al. [8].

TABLE 1: List of notations.

Notation	Description
TA	Trusted authority
KGC	Key generation center
RSU	Roadside unit
p	A secure prime numbers
E	An elliptic curve: $y^2 = x^3 + ax + b \pmod{p}$
P	A generator of the additive group
F_p	The finite field of the elliptic curve
(PK_{TA}, α)	The public key and private key of the TA
(PK_{KGC}, β)	The public key and private key of the KGC
(PK_{V_i}, ρ_i)	The public key and private key of the vehicle
PPK_i	Partial private key of the vehicle V_i
σ_i	The signature of vehicle i
PID_i	Pseudoidentity of the vehicle V_i

$E: y^2 = x^3 + ax + b \pmod{p}$, where $a, b, x, y \in F_p$ and $\Delta = 4a^3 + 27b^2 \neq 0 \pmod{p}$.

Definition 2 (addition of elliptic curves). Assume that $P = (x_1, y_1) \in E$, where P is a point of the elliptic curve E , and $-P = (x_1, -y_1) \pmod{p}$ is the negative point of P . Suppose $Q = (x_2, y_2) \in E$, $Q \neq -P$, and we can define a line l passes through P and Q and intersects the elliptic curve at a point $R' = (x_3, y_3)$. The symmetrical point about the x -axis with R' is $R = (x_3, -y_3)$; then, we can define $R = P + Q$.

Definition 3 (elliptic curve discrete logarithm problem). Firstly, the discrete logarithm problem should be defined. If

a and b are known, how to get k from the equation $b = a^k \pmod{p}$ is a question. Assume that P_1 is a point on the elliptic curve E on the finite field F_p , and select a random number $k \in Z_p^*$. Then, we can calculate $P_2 = k \cdot P_1$. In this case, it is straightforward to calculate P_2 according to Definition 2. However, given the description of elliptic curve discrete logarithm problem (ECDLP), it is scarcely possible to get k according the above equations.

3.2. System Model. In this section, we will try to describe the system model of VANETs in detail. As shown Figure 2, there are four main participants: trusted authority (TA), key generation center (KGC), roadside unit (RSU), and vehicle, which can be divided into two layers: the upper layer includes TA and KGC, both are of more powerful computation, and the lower layer consists of RSUs and vehicles. The demonstration of each participant is as follows:

TA: it is a fully trusted third party that is responsible for system initialization, user registration, system parameter generation, and system security implementation. If necessary, it can track malicious behavior and catch malicious nodes. In addition, it also has enough computing power and storage capacity.

KGC: it is a partially trusted party used for generating partial private key. It can help vehicle generate partial secret key which contributes to its privacy security. It also has sufficient memory, processing, and computing capabilities the same as TA.

FIGURE 2: System model of VANETs.

RSU: it is a smart application device installed on the roadside, which is able to transmit and submit information to TA, KGC, vehicles, or other RSUs in a secure wired connection and plays the role of an intermediary agent. Unlike TA or KGC, RSU usually has limited computing power and storage capacity.

Vehicle: it is the major and basic member in VANETs, which is generally equipped with a smart device which can perform the basic functions such as transmitting the vehicle's messages and performing a simple calculation. In addition, vehicles commonly have limited computing power and storage capacity. The most obvious feature of the vehicle is that the network connection between vehicles is always in dynamic change.

Note that TA and KGC are distinct entities in function and logical; however, the two usually are deployed on a single server in physics if the scale of VANETs is small.

3.3. Certificateless Aggregate Signature Scheme. Generally, a CLS scheme and a CL-AS scheme consist of the following seven algorithms [21, 22].

Setup: the KGC and TA will execute a probabilistic algorithm, which needs a security parameter λ . Then, it generates an elliptic curve E , the public key PK_{TA} and PK_{KGC} , and the master secret key α and β , respectively. After the security parameters for ECC are generated, a number of system parameters which is used for

ensuring the system in order are transmitted between each participant.

PartialPrivateKeyGeneration: in this algorithm, firstly, the vehicle V_i transmits a tuple, which consists of a real identity and a partial pseudoidentity, to TA. Then, the TA generates a complete pseudoidentity with the tuple and sends it to the KGC for calculation. Eventually, the KGC transmits the partial private key to the vehicle V_i in a secure channel [23].

VehicleKeyGeneration: the vehicle V_i selects a random $\rho_i \in \mathbb{Z}_q^*$ as its secret key and calculates its public key $PK_{V_i} = \rho_i \cdot P$.

IndividualSign: this algorithm is used by each vehicle V_i ; after generating a message m_i , the vehicle V_i tries to calculate a set of variables. Then, it sends the signature σ to the verifier.

IndividualVerify: this algorithm is executed by the verifier such as RSU. When the inputs, including signature σ_i , pseudoidentity PID_i , and current time T_{cur} , are received, the verifier will check the validity of the timestamp firstly. Then, the algorithm will output true if the signature is valid or false otherwise.

AggregateSign: in this algorithm, generally, the aggregate signature generator is RSU in the system. For an aggregating set V of n vehicles $V_0, V_1, \dots, V_{n-1}$, the pseudoidentity PID_i of each vehicle V_i is the list PID , the corresponding public key PK_{V_i} of V_i , and message signature tuples $((m_1, \sigma_1), (m_2, \sigma_2), \dots, (m_n, \sigma_n))$ of V_i ,

respectively. The aggregate signature generator will generate signature σ ; then, it will transmit the tuple including the signature, PID, and timestamp list T to the verifier.

AggregateVerify: in general, this algorithm is executed by another RSU. It takes an aggregated set V of n vehicles $\{V_0, V_1, \dots, V_{n-1}\}$, the pseudoidentity PID_i of each vehicle V_i . The verifier will check the timestamp validity for each vehicle firstly. Then, it will output true if the signature is valid or false otherwise.

To demonstrate the whole communication process, Figure 3 shows the certificateless aggregate signature scheme.

As shown in Figure 3, there are two main steps for CL-AS, aggregation and verification, which cause some delays in VANETs. For convenience, the delay brought by the aggregation of signatures is defined as T_a , while T_c is the total time in a cryptographic operation of the verification process. Without loss of generality, in the following algorithm, the tokens, rather than the actual number, will be used in the relevant calculation because different schemes adopt different algorithms and different time costs. T_a and T_c of different VENETs schemes are shown in Table 2 for reference. From the table, we can conclude that T_a is much smaller than T_c in whichever scheme. Even in some schemes, T_c is almost 1000 times of T_a . Therefore, in the following sections, T_a can be ignored and is not calculated.

3.4. Problem Definition. Batch verification is widely used in VANETs [5, 9, 26]. There are two major research aspects about the batch verification: one is about the correctness and security of the encryption algorithm and how to aggregate the signatures into one and verify the aggregated one correctly and the other is about the strategy of grouping. An appropriate strategy will benefit the aggregation and verification process with significant time savings.

This paper focuses on the latter one, the strategy of grouping, that is, how to divide all signatures into one or more batches to determine all invalid signatures as soon as possible. Here, the problem can be converted into a logical problem, given an input set whose size is n ($n \geq 1, n \in \mathbb{Z}^+$). In the set, all n elements are either true or false, and assume that they are distributed randomly and testing an element is true or false costs much more time than other operations, such as AND. The number of false elements is x , and the number of true elements is $n - x$. The problem is how can we design an algorithm to determine exactly all false elements as efficiently as possible.

In summary, the problem is as follows: given a signatures' set $A = \{\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n\}$ which includes a certain number of valid signatures and invalid signatures, find a set F only including all invalid signatures, i.e., $A - F$ is verified true and minimized to $|F|$.

4. The Proposed Bitwise Divisions' Algorithm

In this section, factorial bitwise divisions' (FBD) algorithm and early-stopping factorial bitwise divisions' (EFBD)

algorithm will be fully introduced. In addition, three theorems will be proved to support FBD and EFBD.

4.1. Description of FBD. FBD is a recursive algorithm, and the maximal recursion time is $\log_2 n$. Note that the output set F equals to the input set A in the beginning. First of all, it will check whether all signatures are valid or not. If so, it will return the full set F , which means all signatures in the input set A are valid. And if the size of the set A is 1, it means that the algorithm has gone into the final layer and the algorithm will return (Algorithm 2).

Figure 4 shows the FBD algorithm flow which selects a sample to demonstrate the entire process. Function DivideByBits is utilized to divide the set A to $2 \log_2 n$ groups. It is a bitwise operation. For example, a signature whose index is 12 and the binary representation is 001010. Therefore, it is expected to be divided into groups $S_1^0, S_2^1, S_3^0, S_4^1, S_5^0, S_6^0$. The superscript represents that the group is divided by 0 or 1, and the subscript represents which bit is used to divide the set. All groups in G are verified and the set X includes all invalid groups. Then, we can filter F and make it smaller. Recursive boundary is the size of F is 1 or 0. If no signatures are filtered, then the algorithm randomly selects a certain bit and gets two original groups S_i^0, S_i^1 and reorders them and makes them as input to recursively call the algorithm. If some signatures are filtered, then use the output set F as an input to recursively call the algorithm until the filter does not work.

A few proofs will be necessary to demonstrate to support FBD and EFBD algorithms. Theorems 1 and 2 prove the recursive boundary, and Theorem 3 proves that the result is correct. With the following four proofs, FBD and EFBD can work normally and efficiently.

Theorem 1. *If the output set F is an empty set, then the algorithm is supposed to be terminated.*

Proof. As described in the algorithm, if the output set F is an empty set, then all signatures are filtered and there exist no invalid signatures in set A . Therefore, the original proposition is correct. $\square$

Theorem 2. *If the output set F has just one signature, then the algorithm is supposed to be terminated.*

Proof. As the algorithm goes, if the output set F has only one invalid signature, it means that the number of both invalid and valid groups definitely is $\log_2 n$. According to FBD, one valid group can half the size of F and $\log_2 n$ valid groups can cut the size of F to 1. Therefore, the algorithm can return the output set F and the original proposition is true. $\square$

Theorem 3. *The output set F consists of all invalid signatures.*

Proof. Assume that the output set F does not consist of all invalid signatures, which means that the algorithm has omitted certain invalid signatures. According to the

FIGURE 3: Certificateless aggregate signature scheme.

TABLE 2: Time cost of different schemes.

Scheme	T_a (ms)	T_c (ms)
Cui et al. [4]	0.0018	1.3298
Kamil et al. [5]	0.0699	2.2899
Kumar et al. [24]	0.2457	4.5120
Malhi et al. [25]	5.2061	12.6330
Zhao et al. [26]	0.0424	17.7673

```

Input: nonempty Set A
Output: set F
(1) function FBD(A)
(2)   if A · size = 1 then
(3)     if Verify(A) then
(4)       return ∅
(5)     else
(6)       return A
(7)     end if
(8)   end if
(9)    $G\{S_0^0, S_0^1, S_1^0, S_1^1, \dots\} = \text{DivideByBits}(A)$ 
(10)   $X = \text{Verify}(G)$ 
(11)   $F = F - X$ 
(12)  if F · size = 1 then
(13)    return F
(14)  end if
(15)  if F = ∅ then
(16)    return ∅
(17)  end if
(18)  if F · size = A · size then
(19)    choose two groups  $S_i^1, S_i^0$  on random bit
(20)     $F = F - \text{FBD}(\text{Reorder}(S_i^0))$ 
(21)     $F = F - \text{FBD}(\text{Reorder}(S_i^1))$ 
(22)  else
(23)     $F = F - \text{FBD}(F)$ 
(24)  end if
(25)  return F
(26) end function
  
```

ALGORITHM 2: Factorial bitwise divisions' algorithm.

algorithm, the only way to filter from the set F is that the group is verified as true. However, when there exist invalid signatures in a certain group, it will definitely not pass the

verification, and the whole group will not be filtered which contradicts the assumption. Therefore, the original proposition is correct. $\square$

FIGURE 4: Proposed factorial bitwise divisions' algorithm flow.

Theorem 4. In the output set F , if the number of invalid signatures is greater than that of valid signatures, then the scale of F will not be reduced.

Proof. Assume that the number of input set A is $n, n = 2^k$, the number of invalid signatures is $x, x > n/2$, and the number of valid signatures is $n - x$ which is small than $n/2$. In the grouping of certain bit i , S_i^0 and S_i^1 can be seen as two boxes whose capacity is $n/2$. When the number of invalid signatures is greater than the capacity of one box, the other box surely has at least one invalid signature whatever the distribution is. Since both boxes have at least one invalid signature, then they will both be verified failed. Also, the inference is true for other bits. Eventually, all groups are false and no more signatures can be removed from the output set F . Therefore, the original proposition is correct.

Consider its contrapositive. In the output set F , if the scale of F can be reduced, then the number of valid signatures is greater than that of invalid signatures. Indeed it is true. However, it is converse. In the output set F , if the number of valid signatures is greater than that of invalid signatures, then the scale of F which can be reduced is obviously false. Unfortunately, it is exactly what FBD needs. It can easily find its counterexample which can be called repulsion number.

The filter will definitely be invalid and cannot filter any valid signatures. Then, the algorithm will reorder each group. The group can go back to FBD as the group A recursively. Eventually, the algorithm will go to the recursive boundary which can be seemed as a single verification. $\square$

4.2. Description of EFBD. In FBD, it is usual that the majority of the output set F are invalid signatures. As FBD goes, it will branch to the final layer which is a waste of time. As an optimized algorithm for FBD, EFBD adopts single

verification when the size of the set F is less than $2 \log_2 n$. The algorithm is designed specially for parallel calculation, and $2 \log_2 n$ verifications can be calculated at a time. The specific algorithm is shown in Algorithm 3.

Most part of EFBD is completely consistent with FBD. The difference is that when the loop terminates, it will singly check each signature in F which will filter all valid signatures. Eventually, it will get a final set F including exactly all invalid signatures.

Definition 4. For two or more binary codes, if both 0 and 1 appear in every bit, then they are called repulsion numbers.

If there exist false repulsion numbers, as EFBD algorithm goes, they will be divided into different groups no matter which bit is used to group. As a result, all groups will be verified false and the filter will be invalid.

5. Performance Analysis

In this section, the performance and time complexity will be analyzed. In addition, our discussion is based on a parallel calculation. For convenience, we define $N = 2 \log_2 n$ which equals to the number of groups in a bitwise partition. For BDD algorithm, it can calculate at most N nodes in the same layer at a time. However, it cannot execute cross-layer computing. For FBD and EFBD, they can also calculate N nodes at a time.

If there exist no invalid signatures, then both algorithms need only one verification. Then, one invalid signature circumstance can be idealized. Time complexity of binary search is reproduced below:

$$O(n) = \left\lceil \frac{1}{N} \right\rceil + \underbrace{\left\lceil \frac{2}{N} \right\rceil + \left\lceil \frac{2}{N} \right\rceil + \dots + \left\lceil \frac{2}{N} \right\rceil}_{\log_2 n \text{ items}} = 1 + \log_2 n \times \left\lceil \frac{2}{N} \right\rceil. \quad (1)$$

```

Input: nonempty Set A
Output: set F
(1)  function EFBD (A)
(2)    if A · size = 1 then
(3)      if Verify (A) then
(4)        return  $\emptyset$ 
(5)      else
(6)        return A
(7)      end if
(8)    end if
(9)     $G\{S_0^0, S_0^1, S_1^0, S_1^1, \dots\} = \text{DivideByBits}(A)$ 
(10)    $X = \text{Verify}(G)$ 
(11)    $F = F - X$ 
(12)   if F · size = 1 then
(13)     return F
(14)   end if
(15)   if F =  $\emptyset$  then
(16)     return  $\emptyset$ 
(17)   end if
(18)   if F · size  $\leq 2 \log_2 n$  then
(19)     return SingleVerify (F)
(20)   end if
(21)   if F · size = A · size then
(22)     choose two groups  $S_i^1, S_i^0$  on random bit
(23)      $F = F - \text{EFBD}(\text{Reorder}(S_i^0))$ 
(24)      $F = F - \text{EFBD}(\text{Reorder}(S_i^1))$ 
(25)   else
(26)      $F = F - \text{EFBD}(F)$ 
(27)   end if
(28)
(29)   return F
(30) end function

```

ALGORITHM 3: Early stopping bitwise divisions' algorithm.

In general, N is always greater than or equal to 2. Therefore, equation (1) can be further simplified:

$$O(n) = 1 + \log_2 n. \quad (2)$$

According to the algorithm, the time complexity of FBD is $O(n) = (2 \log_2 n/N) = 1$.

Considering the worst case, that is, all signatures are invalid; BDD algorithm will branch on each leaf. Its time complexity is as follows:

$$\begin{aligned}
 (n) &= \left\lceil \frac{1}{N} \right\rceil + \left\lceil \frac{2}{N} \right\rceil + \left\lceil \frac{4}{N} \right\rceil + \dots + \left\lceil \frac{n}{N} \right\rceil \\
 &\geq \frac{1}{N} + \frac{2}{N} + \frac{4}{N} + \dots + \frac{n}{N} \\
 &\geq \frac{2n-1}{N}.
 \end{aligned} \quad (3)$$

Time complexity of FBD is as follows:

$$\begin{aligned}
 O(n) &= \left\lceil \frac{2 \log_2 n}{N} \right\rceil + \left\lceil \frac{2^2 (\log_2 n - 1)}{N} \right\rceil + \left\lceil \frac{2^3 (\log_2 n - 2)}{N} \right\rceil \\
 &\quad + \dots + \left\lceil \frac{2^{\log_2 n} [\log_2 n - (\log_2 n - 1)]}{N} \right\rceil \\
 &= \left(\left\lceil \frac{2 \log_2 n}{N} \right\rceil + \left\lceil \frac{2^2 \log_2 n}{N} \right\rceil + \left\lceil \frac{2^3 \log_2 n}{N} \right\rceil + \dots + \left\lceil \frac{2^{\log_2 n} \log_2 n}{N} \right\rceil \right) \\
 &\quad - \left(\left\lceil \frac{2^2}{N} \right\rceil + \left\lceil \frac{2^3 \times 2}{N} \right\rceil + \left\lceil \frac{2^4 \times 3}{N} \right\rceil + \dots + \left\lceil \frac{2^{\log_2 n} (\log_2 n - 1)}{N} \right\rceil \right) \\
 &\leq (1 + 2 + 4 + \dots + 2^{\log_2 n - 1}) - \frac{2n \log_2 n - 4n + 4}{N} \\
 &= \frac{4n - 4 - 2 \log_2 n}{N}.
 \end{aligned} \quad (4)$$

TABLE 3: Number of T_c used in different algorithms.

	BDD					FBD					EFBD				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
4	3	3	3	3	3	1	3	3	4	4	1	1	1	1	1
8	4	4	4	4	4	1	3	4	5	6	1	2	2	3	3
16	5	5	5	5	5	1	4	5	6	8	1	2	2	3	3
32	6	6	6	6	6	1	4	6	8	10	1	2	4	4	5
64	7	7	7	7	7	1	5	8	9	11	1	3	4	6	7
128	8	8	8	8	8	1	5	8	11	13	1	3	5	7	8

TABLE 4: Number of T_a used in different algorithms.

	BDD					FBD					EFBD				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
4	2	2	2	2	2	1	1	1	1	1	3	3	3	3	3
8	3	3	3	3	3	3	4	4	4	4	3	6	7	8	8
16	4	4	4	4	4	7	9	11	12	13	7	14	16	19	20
32	7	7	8	8	8	15	19	25	30	32	15	31	34	37	42
64	11	12	13	13	14	31	38	56	66	72	31	59	66	79	87
128	20	23	24	26	27	63	77	108	140	155	63	115	123	155	177

(a)

(b)

FIGURE 5: Continued.

FIGURE 5: Total time cost of different number of vehicles. (a) $n = 4$. (b) $n = 8$. (c) $n = 16$. (d) $n = 32$. (e) $n = 64$. (f) $n = 128$.

6. Simulation Analysis

In the simulation experiment, BDD, FBD, and EFBD are simulated to count T_a and T_c , respectively, in different scales and invalid signatures. In addition, the parallel calculation is adopted and the scale $N = 2 \log_2 n$.

For different invalid signatures and scales, the performance of the three algorithms is shown in Tables 3 and 4. The number of signatures is from 4 (2^2) to 128 (2^7) and the invalid signatures are 1, 2, 3, 4, and 5, respectively. Because the distribution of invalid signatures greatly influences the result, the simulation is repeated 1000 times and the result is equilibrated to avoid randomness.

As shown in Table 3, when the number of signatures is 4, EFBD only uses $1T_c$ for the reason that it will use single verification when the size of the input set is less than or equal to $N = 2 \log_2 n$. In addition, when the number of false nodes

is 1, FBD and EFBD only need $1T_c$ because of their unique filtering structure. BDD has a relatively constant number because of its regular hierarchic execution. In comparison, FBD and EFBD have a high performance when the number of invalid signatures is less than 2 and 4, respectively. BDD has to go through $\log_2 n$ layers to find the invalid signatures. However, as described in Section 4, FBD and EFBD can be seen as a filter, and it can efficiently filter most of the valid signatures and get a set containing invalid signatures when there exist a few invalid signatures. In Table 4, although the number of T_a of our algorithms is bigger than that of BDD, T_a is defined as a much smaller time cost compared to T_c , which makes no difference in the circumstance.

Furthermore, Figure 5 shows the total time cost comparison among the three algorithms. It should be noted that the total time cost includes both aggregation time and verification time. In our work, T_a is 0.036 ms, and T_c is

2.2899 ms which is much bigger than T_a . Therefore, in Figure 5, the rank of different algorithms is the same as the rank in Table 3. From the figures, when the number of elements and the number of false elements are small, the two proposed algorithms are better than the BDD significantly. Actually, in a real scene, the number of elements and the number of false elements is small. Therefore, FBD and EFBD algorithm can achieve high efficiency in real application scenarios.

7. Conclusion

To solve the problem that batch verification has low efficiency in the process of message verification, FBD and EFBD are proposed, and they are featured by parallel calculation and filtering. They show their high efficiency after simulation experiments. However, FBD and EFBD have relatively low efficiency performance in serial computing and only have high efficiency in parallel calculation. And, they cannot outperform BDD when invalid signatures are more than a certain number. According to the experiments, when the number of invalid signatures is more than 4, FBD and EFBD have low efficiency. In the future, FBD and EFBD can be simulated in a more real scene and some factors can be considered. Furthermore, they can be implemented in the hardware that supports parallel computing. Due to repulsion number problem, they can also be optimized to attain higher performance.

Data Availability

The data used to support the findings of the study are included within the article.

Disclosure

This paper was presented in Journal of Physics: Conference Series, Volume 1856, 2021 International Conference on Computer Network Security and Software Engineering (CNSSE 2021), February 26–28 2021, Zhuhai, China.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by Key international cooperation projects of the National Natural Science Foundation of China (No. 61520106007).

References

- [1] M. M. Zanjireh and H. Larijani, "A survey on centralised and distributed clustering routing algorithms for WSNS," in *Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pp. 1–6, Glasgow, UK, May 2015.
- [2] C.-K. Toh, "Future application scenarios for MANET-based intelligent transportation systems," in *Proceedings of the Future Generation Communication and Networking (FGCN 2007)*, Jeju, People Republic of Korea, January 2008.
- [3] Y. Zhou, L. Tian, C. Zhu, X. Jin, and Y. Sun, "Video coding optimization for virtual reality 360-degree source," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 1, pp. 118–129, 2020.
- [4] J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu, "An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks," *Information Sciences*, vol. 451–452, pp. 1–15, 2018.
- [5] I. A. Kamil and S. O. Ogundoyin, "An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks," *Journal of Information Security and Applications*, vol. 44, pp. 184–200, 2019.
- [6] C.-H. Chen, F. Song, F.-J. Hwang, and L. Wu, "A probability density function generator based on neural networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 541, Article ID 123344, 2020.
- [7] C.-H. Chen, "An arrival time prediction method for bus system," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4231–4232, 2018.
- [8] J.-L. Huang, L.-Y. Yeh, and H.-Y. Chien, "Abaka: an anonymous batch authenticated and key agreement scheme for value-added services in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 1, pp. 248–262, 2011.
- [9] Y.-L. Huang, C.-H. Lin, and F.-Y. Leu, "Verification of a batch of bad signatures by using the matrix-detection algorithm," in *Proceedings of the 2011 First International Conference on Data Compression, Communications and Processing*, pp. 299–306, Palinuro, Italy, June 2011.
- [10] H. Ferng, J. Chen, M. Lotfolahi, Y. Tseng, and S. Zhang, "Messages classification and dynamic batch verification scheme for vanets," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1156–1172, 2021.
- [11] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.
- [12] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Systems with Applications*, vol. 177, Article ID 114864, 2021.
- [13] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.
- [14] D. J. Guan, E. S. Zhuang, I. C. Chung, and Y.-S. Lin, "Performance analysis of some batch verification methods of digital signatures," in *Proceedings of the 2017 12th Asia Joint Conference on Information Security (AsiaJCIS)*, pp. 10–14, Seoul, People Republic of Korea, August 2017.
- [15] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, vol. 1403, pp. 236–250, Espoo, Finland, May 1998.
- [16] J. Y. Hwang, B. Song, D. Choi, S.-H. Jin, H. S. Cho, and M.-K. Lee, "Simplified small exponent test for batch verification," *Theoretical Computer Science*, vol. 662, pp. 48–58, 2017.
- [17] X. Ye, G. Xu, X. Cheng, Y. Li, and Z. Qin, "Certificateless-based anonymous authentication and aggregate signature scheme for vehicular ad hoc networks," *Wireless*

- Communications and Mobile Computing*, vol. 2021, Article ID 6677137, 16 pages, 2021.
- [18] E. P. Quiroz, A. Cuno, W. R. Lovón, and E. Cruzado, "ECC usage on X.509 digital certificates," in *Proceedings of the 2020 IEEE Engineering International Research Conference (EIRCON)*, pp. 1–4, Lima, Peru, October 2020.
 - [19] R. Yadav, S. Srinivasan, and S. Gupta, "Security analysis of RSA and ECC in mobile wimax," in *Proceedings of the 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPEs)*, pp. 1725–1729, Paralakhemundi, India, October 2016.
 - [20] Y. Yang, D. He, H. Wang, and L. Zhou, "An efficient blockchain-based batch verification scheme for vehicular ad hoc networks," *Transactions on Emerging Telecommunications Technologies*, 2019.
 - [21] X. Hu, W. Tan, C. Ma, F. Chen, and C. Yu, "Study on security analysis and efficient improvement of certificateless aggregate signature scheme," in *Proceedings of the 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 343–346, Beijing, China, October 2020.
 - [22] C. Hu, S. Wangan, and Z. Bing, "Certificateless aggregate signature scheme," in *Proceedings of the 2010 International Conference on E-Business and E-Government*, pp. 3790–3793, Guangzhou, China, May 2010.
 - [23] M. Syfullah and J. M.-Y. Lim, "Data broadcasting on cloud-vanet for IEEE 802.11p and LTE hybrid vanet architectures," in *Proceedings of the 2017 3rd International Conference on Computational Intelligence Communication Technology (CICT)*, pp. 1–6, Ghaziabad, India, February 2017.
 - [24] P. Kumar, S. Kumari, V. Sharma, X. Li, A. K. Sangaiah, and S. H. Islam, "Secure cls and cl-as schemes designed for vanets," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3076–3098, 2019.
 - [25] A. K. Malhi and S. Batra, "An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks," *Discrete Mathematics & Theoretical Computer Ence*, vol. 17, no. 1, 2015.
 - [26] Y. Zhao, Y. Hou, L. Wang, S. Kumari, M. K. Khan, and H. Xiong, "An efficient certificateless aggregate signature scheme for the internet of vehicles," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, pp. 3708–3711, 2020.