IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 31, 2017, accepted September 10, 2017, date of publication September 19, 2017,
date of current version December 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2753778

TJET: Ternary Join-Exit-Tree Based Dynamic Key Management for Vehicle Platooning

CHANG XU[1,2], RONGXING LU[3], (Senior Member, IEEE), HUAXIONG WANG[2], LIEHUANG ZHU[1], (Member, IEEE), AND CHENG HUANG[4]

[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[2]Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798
[3]Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada
[4]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Corresponding author: Liehuang Zhu (liehuangz@bit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402037, Grant 61272512, Grant 61300177, and Grant 61572027, in part by the National Key Research and Development Program under Grant 2016YFB0800301, and in part by the Guangxi Cooperative Innovation Center of Cloud Computing and Big Data under Grant YD16E14.

ABSTRACT Vehicle platooning, which is formed by a group of vehicles traveling in close proximity to one another, nose-to-tail, at highway speeds, has received considerable attention in recent years. However, though it brings many opportunities in self-driving, the security of vehicle platooning is still challenging. In this paper, to secure vehicle platooning, particularly to secure communication and efficient key updating for vehicles in a platoon, we first present the notion of ternary join exit tree. A ternary join exit tree consists of main tree, join tree, and exit tree. The users join in the join tree and leave from exit tree. Then, we propose a new dynamic ternary join-exit tree-based dynamic key management scheme, called TJET, for vehicle platooning, which is characterized by providing efficient key updating for not only vehicle joining and leaving, but also platoon merging and splitting. Specifically, based on the structure of ternary join-exit tree, we devise concrete algorithms for vehicle joining, vehicle exiting, platoon merging, and splitting. Moreover, we also analyze the capacities and activation conditions of join tree and exit tree based on strict mathematical proofs. Detailed security analysis show that our proposed TJET holds desirable security properties including forward security, backward security, and resistance to key control. In addition, performance evaluations via extensive simulations demonstrate the efficiency of TJET.

INDEX TERMS Vehicle platooning, security, dynamic key management, ternary join exit tree.

I. INTRODUCTION

As the world's population grows, the number of vehicles on the road is continuously increasing by a wide margin, and it has been estimated that there exist more than 1 billion vehicles worldwide. Accordingly, some critical issues such as fuel waste, air pollution, and traffic congestion have become even more serious. In order to solve these issues, platoon-based driving pattern is proposed. A vehicle platoon (also known as road train) [1]–[3] is a collection of vehicles which consists of a leader vehicle and member vehicles. The leader vehicle is driven manually, and each member vehicle follows the leader automatically according to the instructions from the leader vehicle. In addition, to avoid interference from other vehicles, each member vehicle keeps a small gap to the vehicle in front. Vehicle platooning can not only help achieve fuel saving and reduction of air pollution by reducing the aerodynamic drag in a platoon, but also decrease traffic congestion by increasing roadway capacity, since the distance between two member vehicles is maintained at a small size. The experiments in PATH project [4] showed that if the vehicles were driven in ten-car platoon, the car lane capacity can be increased by a factor of two to three. Besides the above mentioned, vehicle platooning has numerous other expected advantages. For instance, platoon-based driving can help increase driver convenience and comfort. The drivers of member vehicles in a platoon can even read books, see movies and make phone calls and others.

However, designing a platoon-based vehicular cyber-physical system still face many security challenges. For instance, if a maliciously controlled vehicle joins a platoon, it can cause the platoon to oscillate at a resonant frequency which can result in serious injury. Additionally, how to design detailed platooning operations to achieve vehicle entry and vehicle leaving and what related information should be sent

inside the platoon should also be considered. For these reasons, some solutions have been presented to solve these challenges [5]–[7]. Nevertheless, another critical issue about platooning has not been well concerned, i.e., secure communication and efficient key update for vehicle platoon. Specifically, in order to achieve secure communication, group keys should be established and updated efficiently when vehicles join or leave, vehicles are moving at high speeds.

Aiming at the above challenge, in this paper, we propose a new ternary join-exit-tree (TJET) based key management scheme for vehicle platooning. Specifically, the main contributions of this paper are three-fold.

- We propose the notion of ternary join exit tree, and exploit 3-party Diffie-Hellman [8] and 2-party Diffie-Hellman protocols [9] to compute the group key. We present concrete group key updating algorithms to achieve secure communication inside the platoon.
- The proposed TJET scheme can efficiently achieve group key updating for dynamic platoon. In TJET, the computation complexity and communication round can be reduced to $\mathcal{O}(\log_3 \log_3(n))$ by utilizing ternary join exit tree where n is the number of group members. Though some other ternary tree based key management schemes were proposed [10], [11], join tree and exit tree are not utilized in these schemes. If join tree and exit tree are used, there are at least two issues which should be considered—how to compute the activation conditions and capacities of ternary join tree and exit tree. In TJET scheme, we give the concrete analysis of the activation conditions and capacities based on strict mathematical proofs. In TJET scheme, as soon as the number of vehicles n exceeds 5, the computation complexity and communication round of group key updating for vehicle join is reduced to $\mathcal{O}(\log_3 \log_3(n))$. In addition, if $n > 38$ holds, the computation complexity and communication round of group key updating for vehicle exit is also reduced to $\mathcal{O}(\log_3 \log_3(n))$. Though Mao *et al.* [12] proposed a binary join-exit-tree based key management scheme (JET). In contrast, in JET, if the group contains fewer than 256 users, the computation complexity and communication round of group key updating for user exit will be $\mathcal{O}(\log_2(n))$.
- To validate the efficiency of key updating for vehicle join and leaving, we also develop a custom simulator built in Java. We compare the efficiency of JET and the proposed TJET with different parameters. Simulation results show that the proposed TJET scheme can efficiently achieve the group key updating comparing with JET scheme.

The remainder of this paper is organized as follows. In Section II, we formalize the system model, security model and identify the design goal. In Section III, we recall bilinear pairing, related complexity assumptions, the two-party and three-party key agreement as preliminaries. The proposed TJET scheme is introduced in Section IV, followed by the security analysis in Section V and the performance evaluation in Section VI. We give the related work in Section VII, and conclude this work in Section VIII.

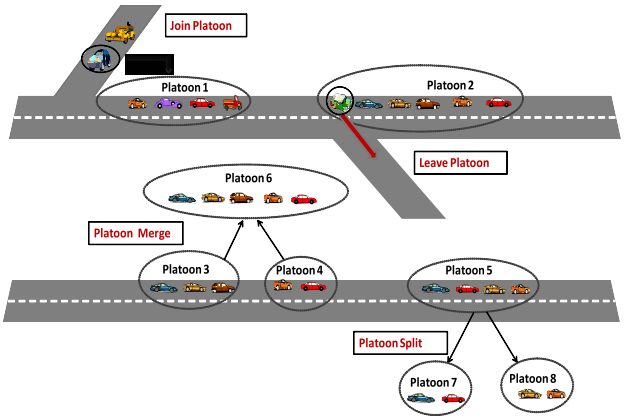


FIGURE 1. System model of dynamic platoon under consideration.

II. MODELS AND DESIGN GOAL

In this section, we formulate the system model, the security model and identify the design goal.

A. SYSTEM MODEL

Our focus is on how to present a secret group session key to achieve the secure communication for a platoon. Particularly, the key should be efficiently updated when the size of the platoon dynamically changes, as shown in Fig 1. The system consists of four kinds of entities: platoons, platoon leaders, followers, and individual vehicles.

- Platoon: a platoon is a group of close-following vehicles, which includes a platoon leader and followers. We assume that the length of a platoon is no more that 300 metres.
- Platoon leader: also known as head vehicle which controls the whole platoon. When the session key of the platoon needs to be updated, the platoon leader will broadcast instructions to the followers.
- Followers: also known as member vehicles which follow the leader automatically. Followers get instructions from the leader and send requests to the leader both by Vehicle-to-Vehicle (V2V) communications. If a follower approaches its destination, it will request to the leader and apply to leave the platoon.
- Individual vehicles: Individual vehicles do not belong to any platoon and could apply to join platoons. When an individual vehicle V wants to join a platoon, V has to get the leader's permit.
- TA: A Trusted Authority (TA) generates a private key and a public key for each vehicle. A revocation certificate list (RCL) is maintained by TA. If a vehicle is corrupted, its public key will be added in to RCL.

There are two basic platoon maneuvers: merge and split. In merge, two platoons traveling on the same lane merge into one big platoon. In split, one platoon separates to form two smaller platoons. When platoon A and platoon B are

driving to the same destination, they can choose to merge into one platoon C. If A wants to drive to another position, platoon C will separate. Accordingly, the platoon session key needs to be updated when aforementioned vehicle joining, follower exiting, platoon merging or platoon splitting event takes place. In a platoon, if the leader is unavailable, another vehicle (e.g.the vehicle behind the leader) in the platoon will be chosen as the new leader. For platoons, the leadership renewal is usually achieved by platoon management protocols, e.g. the scheme presented in [5]. Therefore, the details about how to capture leadership renewal are not discussed in details.

Generally speaking, vehicles can estimate when they will leave the group. The leader maintains a list according to when the vehicles leave. The list is also held by the vehicle members. If a vehicle joins or leaves, the leader will inform the members. Each vehicle in the platoon can improve the tree status according to TJET algorithms since all the algorithms and activation conditions are public. In addition, we assume the vehicles join/leave one by one.

B. SECURITY MODEL

To achieve secure communication inside a platoon, a group key for the platoon needs to be established. On one hand, the interference factors from the vehicles outside the platoon could be excluded. On the other hand, real-time information in the platoon can only be gained by current member vehicles. In our security model, outside attackers are considered. Similar to most previous key management schemes, the secure key management scheme for platoons is expected to hold the following secure properties.

- Authentication. Authentication guarantees the origin of data. Thus, attackers cannot impersonate the legitimate vehicles to execute the scheme.
- Integrity. Data Integrity ensures that data sent by vehicles has not been altered by attackers.
- Forward Secrecy: After a vehicle member leaves a platoon, it cannot compute any subsequent platoon group key. This property guarantees that as soon as a vehicle exits, it cannot communicate with the platoon any longer.
- Backward Secrecy: After a vehicle member joins a platoon, it cannot compute any previous platoon group key. This property can guarantee that the new member cannot get earlier information inside the platoon.
- Resistance to key control: Resistance to key control should be achieved to ensure any vehicle in a platoon is unable to choose or influence the value of the platoon group key.

If a key management scheme holds resistance to key control [13], no participant can control the key. Specifically, the group session key is sufficiently random if at least one of participants is able to generate sufficiently random inputs. In addition, the participants can be sure that the session key is new by ensuring that their own input is new. That is, no user can force use of an old key.

Under the aforementioned system model, our design goal is to construct an efficient and secure dynamic key management scheme for a platoon. Particularly, the following two objectives should be captured.

- Security. The proposed scheme should exhibit authentication, integrity, forward secrecy and backward secrecy. Thus, any member vehicle who leaves cannot learn any new group keys, and new group members will not be able to learn old keys as well. Besides, the scheme should resist key control to guarantee that no vehicle can control the key of platoon.
- Efficiency. As we have discussed early, a platoon key management scheme has stringent performance requirements. Specifically, the amount of time needed to recompute a new group key reflects the latency in vehicle join and exit. A good solution should achieve low time cost associated with key updates.

In VANET, how to find out inside attackers is a challenging topic. In order to identify inside attackers, some solutions have been proposed e.g., the schemes presented in [14]–[16]. In addition, packet loss brings in influences and limitation of the communications. Accordingly, how to improve the VANET performance has been addressed [17], [18]. In this paper, we mainly focus on constructing a dynamic key agreement for platoons to more efficiently establish and update shared group keys when vehicles join/leave. Therefore, inside attackers and packet loss are not considered.

III. PRELIMINARIES

In this section, we will introduce bilinear pairing, related complexity assumptions, and two-party key agreement and three-party agreement techniques, which will serve as a basis of our scheme.

A. BILINEAR PAIRING AND COMPLEXITY ASSUMPTIONS

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative groups of order p for some large prime p, and g be a generator of $\mathbb{G}$. A bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, which satisfies the following properties:

- Bilinearity: $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p^*$.
- Non-degeneracy: $\hat{e}(g, g) \neq 1$.
- Computability: $\hat{e}(x, y)$ can be computed efficiently.

Computational Diffie-Hellman (CDH) assumption: Given (g, g^a, g^b), there exists no probabilistic, polynomial time algorithm to compute g^{ab} with non-negligible probability of success.

Bilinear Diffie-Hellman (BDH)assumption [19]: Given (g, g^a, g^b, g^c), there exists no probabilistic, polynomial time algorithm to compute $\hat{e}(g, g)^{abc}$ with non-negligible probability of success.

Hash Diffie-Hellman (HDH) assumption [20]: given $g^a, g^b, H(g^{ab})$ and T, there exists no probabilistic, polynomial time algorithm to distinguish $H(g^{ab})$ and T with non-negligible probability of success, where Hčž$\{0, 1\}^* \to \mathcal{S}$ is a cryptographic hash function, and T is a random choice from $\mathcal{S}$.

Decision Hash Bilinear Diffie-Hellman (DHBDH) assumption [21]: Given g, g^a, g^b, g^c, $H(\hat{e}(g,g)^{abc}$ and T, there exists no probabilistic, polynomial time algorithm to distinguish $H(\hat{e}(g,g)^{abc})$ and T with non-negligible probability of success, where Hčž$\{0,1\}^* \rightarrow \mathcal{S}$ is a cryptographic hash function, and T is a random choice from $\mathcal{S}$.

B. TWO-PARTY AND THREE-PARTY KEY AGREEMENT

Two participants establish a common session key by executing a 2-party Diffie-Hellman (DH) key agreement protocol [9]. A calculates g^a and B calculates g^b and they exchange these values. After that, they compute the shared secret $K_{AB} = H(g^{ab})$ (as shown in Fig. 2 (a)), where H is a cryptographic hash function and $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.

Three participants will compute a shared session key by executing a 3-party DH key agreement protocol [8]. A, B and C compute g^a, g^b, and g^c respectively . After they exchange these values, they can compute shared secret $H(\hat{e}(g^b, g^c)^a) = H(\hat{e}(g^a, g^c)^b) = H(\hat{e}(g^a, g^b)^c = \hat{e}(g,g)^{abc})$ (as shown in Fig. 2(b)).

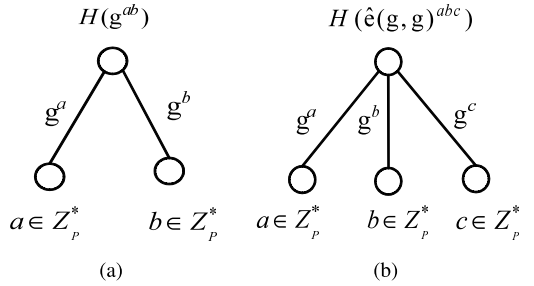


FIGURE 2. 2-party and 3-party key agreement. (a) 2-party DH. (b) 3-party DH.

C. TIME EFFICIENCY ISSUE IN KEY MANAGEMENT

Similar to [12], in this paper, we exploit ''simple round'' to evaluate the efficiency of key agreement. The notation of ''simple round'' is proposed in [22], where each participant can send and receive at most one message in each round. In each round, each participant performs at most a two-party DH operation or a pair operation. The following definitions in [12] are also used in this paper.

- Average Join Time T_J: The number of rounds to process key updates for a vehicle join event.

$$T_J = \frac{R_J}{N_J}$$

  where R_J is the total number of rounds executed for N_J join events.
- Average Departure Time T_E: The number of rounds for a vehicle exit event.

$$T_E = \frac{R_E}{N_E}$$

  where R_E is the total number of rounds performed for N_E exit events.
- Overall Average Processing Time T is defined as $T = \frac{R}{N}$, where $N = N_J + N_E$ and $R = R_J + R_E$.

D. NOTATIONS FOR TJET

We list the notations used in the following sections in Table 1 as follows:

IV. PROPOSED TJET SCHEME FOR VEHICLE PLATOONING

In this section, we first present the ternary join exit tree; introduce the ternary join tree algorithm and ternary exit tree algorithm. After that, we will analyze the capacities of join tree and exit tree, and their activation conditions. Finally, we will describe the detailed key establishment procedure for dynamic platoons.

A. TERNARY KEY TREE IN TJET

There are three types of nodes in a ternary key tree: leaf node, intermediate node, and root node.

- Leaf node: correspond with the private key of each group member. We also use a leaf node to denote a vehicle.
- Intermediate node: denote one key which is shared by all members of the subtree rooted at the intermediate node.
- Root node: represent the key shared by the whole group.

Assume that there are n vehicles. A ternary key tree is constructed from bottom to top. The vehicles are first divided into triples. Then, each triple executes a 3-party DH operation and forms a subgroup. After that, the subgroups are divided into triples again (as shown in Fig. 3 (a)). Continuing in this vein, the group key is established. If some subgroup consists of 2 vehicles (as shown in Fig. 3 (b)), the 2 vehicles will execute 2-party DH operation. If only one vehicle is contained in some subgroup (Fig. 3 (c)), no DH-operation is performed inside the subgroup. Take Fig. 3 (a) as an example, we will show how to compute the group key in detail. In order to make TJET scheme achieve authentication and integrity, a vehicle will send a message m together with the signature of m. For instances, a vehicle U_1 sends g^{a_1} along with $sig_{U_1}(g^{a_1})$. In order to simply the description of TJET, $sig_{U_1}(g^{a_1})$ is omitted below.

Round 1: Each set of three vehicles forms a subgroup, i.e., $G_1 = \{U_1, U_2, U_3\}$, $G_2 = \{U_4, U_5, U_6\}$, and $G_3 = \{U_7, U_8, U_9\}$. In subgroup G_1, U_1 sends $g^{a_1} \mod p$ to U_2 and U_3, U_2 sends $g^{a_2} \mod p$ to U_1 and U_3, and U_3 sends $g^{a_3} \mod p$ to U_1 and U_2. U_1, U_2, and U_3 compute the shared subgroup key $K_1 = H(e(g,g)^{a_1 a_2 a_3}))$. Similarly, U_4, U_5, and U_6 compute $K_2 = H(e(g,g)^{a_4 a_5 a_6}))$, U_7, U_8, and U_9 compute $K_3 = H(e(g,g)^{a_7 a_8 a_9}))$.

Round 2: Three subgroups (node) G_1, G_2 and G_3 compute the new group key. Assume the leftmost member of a subgroup can be considered as a sponsor of the group, i.e., U_1, U_4, and U_7 are the sponsors of G_1, G_2 and G_3 respectively. U_1 broadcasts $g^{H(e(g,g)^{a_1 a_2 a_3}))} \mod p$ to all the vehicles in G_2 and G_3, U_4 broadcasts $g^{H(e(g,g)^{a_4 a_5 a_6}))} \mod p$ to all the vehicles in G_1 and G_3, and U_7 broadcasts $g^{H(e(g,g)^{a_7 a_8 a_9}))} \mod p$ to all the vehicles in G_1 and G_2.

TABLE 1. Notations for TJET.

J-root	Root node of J-tree
vehicle_number(x)	Number of leaf nodes under the node x
k	Nonnegative integer
leftchild(x)	Left child node of node x
midchild(x)	Middle child node of node x
rightchild(x)	Right child node of node x
insertion-node	Insertion node
T_J	Average vehicle join cost
$R(i)$	Join latency for the ith vehicle
N_1	Number of vehicles(leaf nodes) in M-tree
N_2	Number of vehicles(leaf nodes) in J-tree
L_J	Capacity of J-tree
N_p	Number of vehicles (leaf nodes) in E-tree after latest batch Relocation
N_c	Number of current vehicles (leaf nodes) in E-tree
β	E-tree residual rate where $\beta \in [0, 1)$
L_E	Capacity of E-tree
T_E	Average departure time if E-tree is exploited
T_S	Average departure time if E-tree is not exploited
R_J	Total number of rounds for N_J join events
N_J	Number of join events
N_E	Number of exit events
R_E	Total number of rounds for N_E exit events
T	Average processing cost for all join events and leave events
β	E-tree residual rate where $\beta \in [0, 1)$

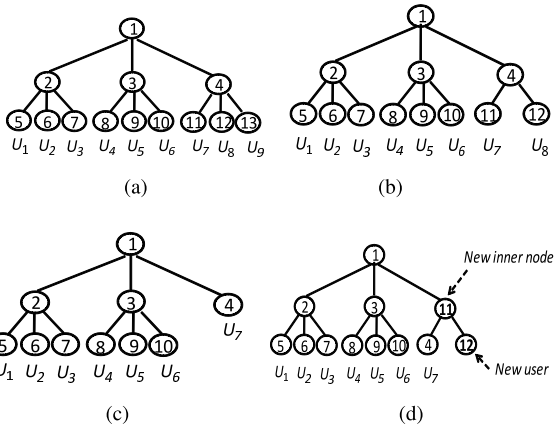


FIGURE 3. Ternary key trees with $n = 9, 8, 7$ users and new user join. (a) $n = 9 \equiv 0 \mod 3$. (b) $n = 8 \equiv 2 \mod 3$. (c) $n = 7 \equiv 1 \mod 3$. (d) New user join.

Then all the vehicles in G_1, G_2 and G_3 compute the common group key $H(\hat{e}(g, g)^{H(e(g,g)^{a_1a_2a_3}))H(e(g,g)^{a_4a_5a_6}))H(e(g,g)^{a_7a_8a_9}))})$.

Key Updating: After a ternary key tree is built, if a vehicle U leaves or joins the group, the keys on the path from U to the root node will be updated. In Fig. 3 (c), if U_1 leaves, both the keys on node 2 and node 1 require to be updated. If a vehicle joins the group, take Fig. 3 (d) as an example, the vehicle node 12 and new inner node 11 are added, both the keys on node 11 and node 1 need to be recomputed. Let the time efficiency be measured by number of rounds required to execute the (2 or 3-party) DH protocol. As the number of vehicles increases, the height of the key tree is expanded. As a result, more rounds are needed to update the subgroup keys when users joining/leaving.

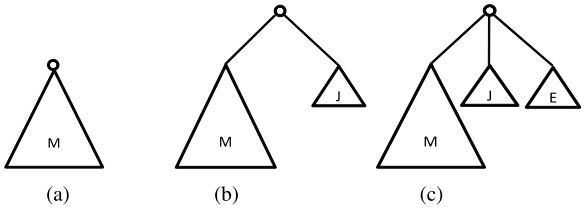


FIGURE 4. Topology of ternary join exit key tree. (a) M-tree. (b) M-tree and J-tree. (c) M-, J-, and E-tree.

B. STRUCTURE OF TERNARY JOIN EXIT KEY TREE

According to Fig.3, for a vehicle exit/join event, fewer rounds are required to update the group key if the height of a tree is smaller than that of other trees. Therefore, we construct ternary join exit key tree by exploring join tree and exit tree which are much smaller than the M-tree to reduce the computation and communication costs.

We define that a ternary join exit key tree consists of three parts: the join tree (J-tree), the exit tree (E-tree), and the main tree (M-tree). The topology of a ternary join exit key tree is showed as Fig. 4 (c).

The capacities of J-tree and E-tree vary with the number of vehicles in M-tree. For instance, if there are only a few vehicles in M-tree, the J-tree and E-tree are inactivated (as shown in Fig. 4 (a)), or only E-tree is set empty (as shown in Fig. 4 (b)).

C. OVERVIEW OF VEHICLE JOINING/LEAVING GROUPS

1) VEHICLE JOINING THE GROUP

If J-tree is inactivated, the joining vehicles will be inserted into the M-tree. Otherwise, the following steps will be executed:

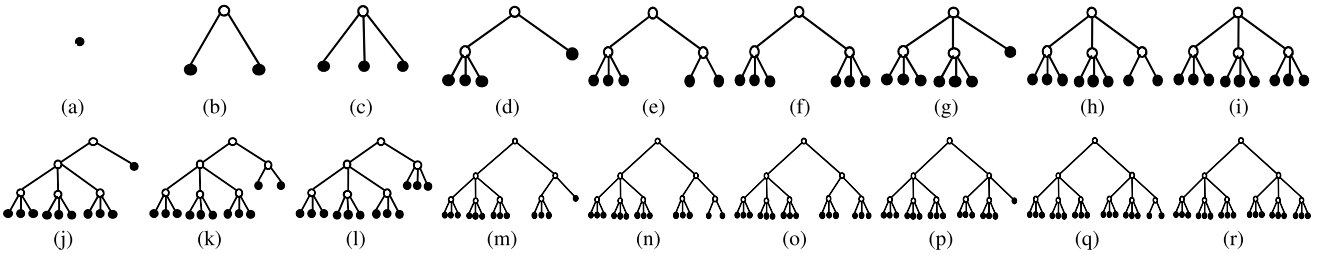


FIGURE 5. Expansion procedure of J-tree from 1 node to 18 nodes. (a) 1 user. (b) 2 users. (c) 3 users. (d) 4 users. (e) 5 users. (f) 6 users.. (g) 7 users. (h) 8 users. (i) 9 users. (j) 10 users. (k) 11 users. (l) 12 users. (m) 13 users. (n) 14 users. (o) 15 users. (p) 16 users. (q) 17 users. (r) 18 users.

- The joining vehicle will be **inserted into the J-tree**.
- If J-tree is full, all the vehicles in J-tree will **be moved in batch into M-tree**.

2) VEHICLE LEAVING THE GROUP

If E-tree is inactivated, the vehicles will leave from the M-tree. Otherwise, the following steps will be executed:

- The vehicles which are most likely to leave in a short time will be first relocated together into the E-tree,
- The vehicles will leave from E-tree.

Join latency. We consider the waiting time for the joining vehicles as join latency. Join latency does not include the time spent for the vehicle movement from J-tree to the M-tree, since as soon as a vehicle joins the J-tree, it can communicate with other members. Therefore, join latency is the amount of time spent in computing a new group key when a vehicle joins the J-tree.

Exit latency. Exit latency does not include the time spent for the relocation from M-tree to E-tree. Exit latency is the number of rounds required to update the group key when a vehicle leaves the E-tree.

D. TERNARY J-TREE ALGORITHM

Ternary J-tree algorithm includes two parts: vehicle joining group and J-tree parameters analysis. Vehicle joining group consists of two procedures: the vehicle insertion into the J-tree, and node relocation from J-tree to M-tree. The J-tree parameters analysis includes J-tree capacity and J-tree activation condition analysis.

1) USER JOINING GROUP

Assume that a vehicle U wants to join the group. Then, vehicle U joining group includes two procedures: **vehicle insertion into J-tree** and **nodes batch movement form J-tree to M-tree**.

a: VEHICLE INSERTION INTO J-TREE

In order to insert the joining vehicle, an insertion node should be chosen in J-tree.

Insertion node. The insertion node is only used to find the position of the new node. After the vehicle U is added in the J-tree, the insertion node will be the parent node or sibling node of U.

The detailed insertion procedure is showed as follows:

- If J-tree is empty, the root node of M-tree is chosen as the insertion node and the insertion node is set as sibling node of the new node. U is added as the root node of the J-tree (as shown in Fig.5 (a)) . Additionally, U executes a 2-party DH operation with the insertion node to compute the group key.
- If J-tree is not empty, Algorithm 1 will be executed to choose the insertion node. Accordingly, the notations in Algorithm 1 are listed in Table 1.
  - If the insertion node has no child node or has three child nodes, it becomes the sibling node of U. U will perform a 2-party DH operation with the insertion node (as shown in Fig. 5 (b) and Fig. 5 (d) respectively).
  - If the insertion node has two child nodes, the insertion node becomes the parent node of U. Accordingly, U will execute a 3-party DH operation with insertion node's two child nodes (as shown in Fig. 5 (c)).

  After that, the keys on the path from the insertion node to the root node of the whole key tree are recomputed.

Specifically, Fig. 5 shows the detailed expansion procedure of J-tree from 1 vehicle to 18 vehicles.

Algorithm 1 Finding the Insertion Node in J-Tree

$x \leftarrow J - root$;
while $usernumber(x) \neq 3^k$ for some integer k **do**
  while $usernumber(leftchild(x)) \neq 3^k$ **do**
    $x \leftarrow leftchild(x)$
  end while
  while $usernumber(midchild(x)) < usernumber(leftchild(x))$ **do**
    $x \leftarrow midchild(x)$
  end while
  while $usernumber(rightchild(x)) < usernumber(midchild(x))$ **do**
    $x \leftarrow rightchild(x)$
  end while
  $insertion - node \leftarrow x$
end while

b: VEHICLES BATCH MOVEMENT FROM J-TREE TO M-TREE

Assume that there are N_2 vehicles in J-tree. As the vehicles join the J-tree, if N_2 exceeds a threshold value, all of the vehicles in the J-tree will be moved into the M-tree in batch. If we consider the J-tree as a logical user and insert the logical user into the shortest branch leaf node of M-tree, the subgroup keys in the J-tree will remain unchanged. Thus, only the keys on the path from the shortest branch leaf node to the root node of M-tree will be recomputed. However, since M-tree becomes a skewed tree, if a vehicle leaves from the longest branch, much more time will be spent in updating the keys on this branch.

Therefore, we choose more insertion nodes in M-tree to maintain the balance of the M-tree (as shown in Fig. 6). Accordingly, the insertion nodes are chosen according to Algorithm 2. Then, the keys on the paths from each insertion node to the root node are computed in parallel.

Algorithm 2 Finding the Insertion Nodes in M-Tree

Set $i = 0$ {i denotes the number of the vehicles have been inserted into M-tree}
for each node from the top level to bottom level of M-tree **do**
  if $i < x$ {x represents the number of vehicles in J-tree} **then**
    if w has 2 child nodes{w is a node.} **then**
      w is chosen as an insertion node
      insert one vehicle node to w
      i=i+1
    end if
    if $i = x - 1$ **then**
      if node w has no child nodes **then**
        w is also chosen as an insertion node,
        insert one vehicle node to w
        i=i+1
      end if
    end if
    if $i < x$ **then**
      if node w has no child nodes **then**
        w is also chosen as a insertion node
        insert 2 vehicle nodes to w
        i=i+2
      end if
    end if
  else
    return
  end if
end for
Recompute the keys on the paths from insertions nodes to the root in parallel.

2) J-TREE PARAMETERS ANALYSIS

As we have mentioned above, when J-tree reaches its capacity, vehicles in the J-tree will be moved into the M-tree.

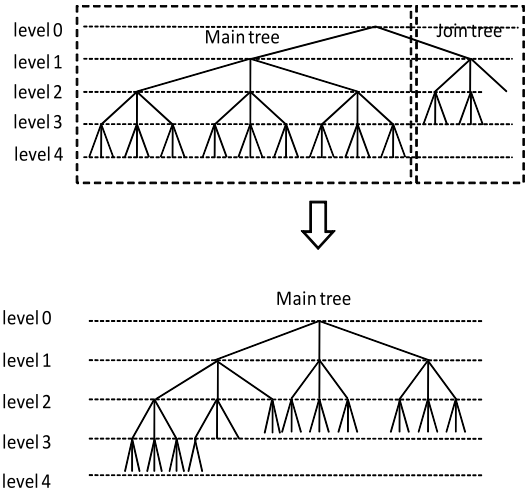


FIGURE 6. Nodes batch Movement.

TABLE 2. Joining latency.

R(1)	=	1	R(2)	=	2	R(3)	=	2
R(4)	=	2	R(5)	=	3	R(6)	=	3
R(7)	=	2	R(8)	=	3	R(9)	=	3
R(10)	=	2	R(11)	=	3	R(12)	=	3
R(13)	=	3	R(14)	=	4	R(15)	=	4
R(16)	=	3	R(17)	=	4	R(18)	=	4
R(19)	=	2	R(20)	=	3	R(21)	=	3
R(22)	=	3	R(23)	=	4	R(24)	=	4
R(25)	=	3	R(26)	=	4	R(27)	=	4
	...			...			...	

Besides, if there are only a few nodes in M-tree, the J-tree is not activated (The detailed analysis is given in Page 7 and Page 8). Both the capacity of J-tree and J-tree activation condition will be analyzed in this section. Accordingly, the notations used in this section are listed in Table 1.

a: J-TREE CAPACITY ANALYSIS

In this section, we will discuss how to adjust the J-tree capacity to minimize average vehicle join cost T_J. In the proposed TJET, the join time is measured by round. When U is moved to J-tree, let the time cost (measured as round) be t_1. When U is moved to M-tree, let the time cost (measured by round) be t_2. Then, the join time of U is $t_1 + t_2$. Let $R(i)$ represent the number of rounds spent by i-th car in joining the platoon. For n vehicles, they need $R(1) + \ldots + R(n)$ rounds to join the J-tree and $\log_3 N_1$ rounds to be moved to M-tree in batch. Therefore, we have

$$T_J = (\sum_{i=1}^{n} R(i) + \log_3(N_1))/n$$

Fig.5 shows the expanding process. According to Fig. 5, $R(i)$ is showed in Table 2.

According to Table 2, we can conclude that about sequence $R(i)$, the following property holds

$$\begin{cases} R(3^p + q) = 1 + R(q), & p \geq 0,\ 1 \leq q \leq 3^p \\ R(3^p + 3^p + q) = 1 + R(q), & p \geq 0,\ 1 \leq q \leq 3^p \end{cases} \quad (1)$$

Here p is a nonnegative integer and q is a positive integer.

Theorem 1: Assume p is a nonnegative integer. When $n = 3^p$, the following equation holds

$$\sum_{i=1}^{n} R(i) = n(\frac{2}{3}\log_3 n + 1) \tag{2}$$

Proof of Theorem 1: See Appendix A.

When $n = 3^p$, according to Theorem 1, we will obtain

$$\frac{\sum_{i=1}^{n} R(i)}{n} = \frac{2}{3}\log_3 n + 1 \tag{3}$$

Theorem 2: Let $r_i = R(i)$. When p is a non-negative integer, $n = 3^p$, and $1 \le k < n$, $\frac{\sum_{i=1}^{k} r_i}{k} < \frac{\sum_{i=1}^{n} r_i}{n}$ holds.

Proof of Theorem 2: See Appendix B.

Let p be a non-negative integer. We consider that x vehicles ($3^p \le x < 3^{p+1}$) are added into the J-tree one by one. We compute the average join time starting from the J-tree is empty. If the size of J-tree is up to the threshold value, all the users are concurrently moved into the M-tree with $\log_3 N_1$ rounds, where $\log_3 N_1$ is the height of M-tree, and N_1 is the number of vehicles (leaf nodes) in M-tree. Here the height of the tree is relaxed to a continuous value to simplify the analysis. If the batch movement time is considered, the join time for the x vehicles is $\sum_{i=1}^{x} r_i + \log_3 N_1$. Then, for x vehicles, the **average join cost** is

$$T_J = \frac{\sum_{i=1}^{x} r_i}{x} + \frac{\log_3 N_1}{x}$$

According to Theorem 2, we can obtain $\frac{\sum_{i=1}^{x} r_i}{x} + \frac{\log_3 N_1}{x} < \frac{\sum_{i=1}^{3^{p+1}} r_i}{3^{p+1}} + \frac{\log_3 N_1}{x}$. According to Theorem 1, we have $\frac{\sum_{i=1}^{3^{p+1}} r_i}{3^{p+1}} + \frac{\log_3 N_1}{x} = \frac{2}{3}\log_3 3^p + \frac{\log_3 N_1}{x} + \frac{5}{3}$. Since $3^p \le x < 3^{p+1}$, then $\frac{2}{3}\log_3 3^p + \frac{\log_3 N_1}{x} + \frac{5}{3} \le \frac{2}{3}\log_3 x + \frac{\log_3 N_1}{x} + \frac{5}{3}$. Therefore, we conclude that $T_J < \frac{2}{3}\log_3 x + \frac{\log_3 N_1}{x} + \frac{5}{3}$. To minimize the upper bound of T_J, we will compute the optimal J-tree capacity L_J. Let $f(x) = \frac{2}{3}\log_3 x + \frac{\log_3 N_1}{x} + \frac{5}{3}$. Then, we have $\frac{d(f(x))}{dx} = \frac{2}{3\ln 3}\cdot\frac{1}{x} - \frac{\ln N_1}{\ln 3}\cdot\frac{1}{x^2}$. When $x = \frac{3}{2}\ln N_1$, $\frac{d(f(x))}{dx} = 0$. Therefore, we get the optimal **J-tree capacity**

$$L_J = \frac{3}{2}\ln N_1$$

Moreover, we could obtain $f(L_J) = \frac{2}{3}\log_3(\log_3 N_1) - \frac{2}{3}\log_3(\log_3 e) + \frac{7}{3} + \frac{2}{3\ln 3} - \frac{2\ln 2}{3\ln 3}$. Thus, the equation $T_J < \frac{2}{3}\log_3(\log_3 N_1) - \frac{2}{3}\log_3(\log_3 e) + \frac{7\ln 3 + 2 - 2\ln 2}{3\ln 3}$ holds. We conclude that each vehicle only requires at most $\mathcal{O}(\log_3 \log_3 n)$ rounds to join the J-tree and to be moved into the M-tree, where n is the size of the group.

As soon as the vehicles are added into the J-tree, they can communicate with other group members, so their waiting time W_J does not include the batch relocation time. Since $x = \frac{3}{2}\ln N_1$, the inequality

$$W_J < \frac{2}{3}\log_3(\log_3 N_1) - \frac{2}{3}\log_3(\log_3 e) + \frac{7\ln 3 - 2\ln 2}{3\ln 3}$$

is satisfied. Therefore, their waiting cost is at most $\mathcal{O}(\log_3 \log_3 n)$ rounds.

b: J-TREE ACTIVATION CONDITION ANALYSIS

As we have aforementioned, the J-tree will not be activated unless the number of vehicles in the group exceeds a threshold value. The threshold value is considered as J-tree activation condition. We will show that the J-tree activation condition is 5.

Assume that the capacity of J-tree is L_J. For L_J users, they are parallelly moved into the M-tree with $\log_3 N_1$ rounds. Besides, each user spends at most $\log_3 L_J$ rounds in joining the J-tree. Therefore, for the average join time T_J of L_J users, the following inequality holds:

$$T_J \le \log_3 L_J + \frac{\log_3 N_1}{L_J}$$

If there are only a few leaf nodes in the M-tree, the J-tree is not activated. Specifically, the J-tree will not be activated unless the inequality $\log_3 L_J + \frac{\log_3 N_1}{L_J} \le \log_3 N_1$ is satisfied. That is, $\log_3 N_1 \ge \frac{L_J}{L_J - 1}\log_3 L_J$ holds. Let $f(N_1) = \frac{L_J}{L_J-1}\log_3 L_J - \log_3 N_1 \le 0$. Since optimal capacity of J-tree is $L_J = \frac{3}{2}\ln N_1$, when $L_J = \frac{3}{2}\ln N_1$, we can obtain $f(N_1) = \frac{\frac{3}{2}\ln N_1}{\frac{3}{2}\ln N_1 - 1}\log_3 \frac{3}{2}\ln N_1 - \log_3 N_1 \le 0$, Let $x = \frac{3}{2}\ln N_1$, then $f(x) = \frac{x}{x-1}\cdot\frac{\ln x}{\ln 3} - \frac{2x}{3\ln 3} \le 0$, or equivalently $f(x) = \frac{\ln x}{x-1} - \frac{2}{3} \le 0$ When $x > 1$, $f(x)' < 0$. That is, when $x > 1$, $f(x)$ is a decreasing function. We find that when $N_1 > 5$, $f(N_1) \le 0$. Therefore, when the number of vehicles in the group exceeds 5, the J-tree is activated.

E. TERNARY E-TREE ALGORITHM

We construct the ternary E-tree algorithm which includes two parts: vehicle leaving group and E-tree parameters analysis.

1) VEHICLE LEAVING GROUP

Vehicle leaving group includes two procedures: the vehicle nodes are batch relocated from M-tree to E-tree and a vehicle leaves from E-tree. The notations used in the following section are listed in Table 1.

a: BATCH RELOCATION INTO E-TREE

The batch relocation will be performed when the batch relocation condition holds. According to [12], the batch relocation condition is set as $N_c \le \beta N_p$. When $N_c \le \beta N_p$ is satisfied, the batch relocation is executed. The insertion algorithm for E-tree is similar to the insertion algorithm for M-tree, i.e., Algorithm 2, except that x and i denote the number of nodes which will be relocated in batch to E-tree, and the number of the nodes which have been inserted into E-tree.

b: VEHICLE LEAVING FROM E-TREE

After vehicles are moved into E-tree, they will leave from E-tree but not M-tree. A vehicle U_c leaves E-tree by executing the following algorithm:

Algorithm 3 Vehicle Leaving Algorithm

1: **if** U_c's parent node has 2 child nodes **then**
2: delete the vehicle node U_c,
3: replace U_c's parent node by U_c's sibling node.
4: **end if**
5: **if** U_c's parent node has 3 child nodes **then**
6: delete the vehicle node U_c.
7: **end if**
8: Recompute all the keys which are on the path from U_c's parent node to the root node.

2) E-TREE PARAMETERS ANALYSIS

As the vehicles in M-tree are moved into E-tree, the number of vehicles in E-tree increases. In this section, we will discuss the capacity of E-tree in this section. Besides, if the M-tree is small, the E-tree is not activated. Therefore, the E-tree activation condition will also be discussed in this section.

a: E-TREE CAPACITY ANALYSIS

Assume that the first D vehicles who most likely leave the group are moved into E-tree when E-tree is empty. After that, D vehicles are moved into E-tree in batch every time, similar to those in [12]. Then, after nth batch movement, the number of vehicles in E-tree will be $D+D\cdot\beta+D\cdot\beta^2+\cdots+D\cdot\beta^{n-1} = \frac{D(1-\beta^n)}{1-\beta}$. Since $\beta \in [0, 1)$, we have $(1-\beta^n) \to 1$. Therefore, when the residual rate is $\beta \in [0, 1)$, the capacity of E-tree is $L_E = \frac{D}{1-\beta}$.

For D nodes, batch relocation needs at most $1+\log_3 N_1$ rounds. Note that $\log_3 N_1$ is the average height of the M-tree. The addition of one refers to the additional one level above the M-tree and E-tree (as shown in Fig.4 (b)). Suppose that the E-tree capacity is x, then every vehicle leaving from the E-tree will requires at most $1 + \log_3 x$ rounds. When U is moved to E-tree, let the time cost (measured as round) be t_1. When U leaves E-tree, let the time cost (measured by round) be t_2. Then, the departure time of U is $t_1 + t_2$. Therefore, for the average departure time T_E of D vehicles the inequality $T_E \le \frac{1}{D}(1 + \log_3 N_1) + (1 + \log_3 x)$ holds, or equivalently,

$$T_E \le \frac{1}{x(1-\beta)}(1 + \log_3 N_1) + (1 + \log_3 x) \tag{4}$$

To minimize the right side of (4), we have

$$x = \frac{\ln N_1 + \ln 3}{1-\beta}$$

Therefore, the capacity of E-tree is $\frac{\ln N_1+\ln 3}{1-\beta}$.

When $x = \frac{\ln N_1+\ln 3}{1-\beta}$, the average exit time is

$$T_E \le \log_3(\log_3 N_1) - \log_3\log_3 e - \log_3(1-\beta) + 1 + \log_3 e$$

We can see the average exit time is bounded by $\mathcal{O}(\log_3\log_3 n)$ where n is the group size. According to $L_E = \frac{D}{1-\beta}$ and $x = \frac{\ln N_1+\ln 3}{1-\beta}$, we obtain

$$D = \ln N_1 + \ln 3$$

That is, in order to minimize T_E, $\ln N_1 + \ln 3$ vehicles are moved from M-tree to E-tree in batch every time.

When β is set as 0, the batch movement will not be executed, unless the E-tree becomes empty. Otherwise, if β is close to 1, the batch relocation will be performed frequently.

b: E-TREE ACTIVATION CONDITION

If the vehicles leave the group from M-tree, the average departure time is $T_S = \log_3 N_1$. If E-tree is explored, the average departure time is $T_E = \frac{1}{L_E(1-\beta)}(\log_3 N_1+1)+(\log_3 L_E+1)$. If $T_E \le T_S$ is satisfied, E-tree will be activated. That is, the following equation should hold:

$$\frac{1}{L_E(1-\beta)}(\log_3 N_1 + 1) + (\log_3 L_E + 1) \le \log_3 N_1 \tag{5}$$

According to $x = \frac{\ln N_1+\ln 3}{1-\beta}$ and (5), we can obtain $\ln(\ln N_1 + \ln 3) - \ln N_1 + \ln 3 + 1 < 0$. Let $y = \ln N_1$, then $f(y) = \ln(y + \ln 3) - y + \ln 3 + 1 < 0$. When $y > 0$, we have $f(y)' = \frac{1}{y+\ln 3} - 1 < 0$. Therefore, When $y > 0$, $f(y)$ is a decreasing function. That is, when $\beta = 0$, for any $N_1 > 38$, the inequality (5) holds. Besides, we compute different activation conditions for different β. For instance, for $\beta = 0.1, 0.2, 0.3, 0.4$, and 0.5, the activation conditions are 45, 52, 61, 74, and 92, respectively. According to the results of the experiment, we suggest that β is set around 0. Then, if the group size is larger than 38 and the E-tree is activated.

F. TJET IN PLATOONS

In this section, we will introduce five protocols in vehicle platooning, i.e., the platoon group key establishment, vehicle join, vehicle exit, platoon merge and platoon split protocols.

1) PLATOON KEY ESTABLISHMENT

If multiple vehicles want to form a platoon, the corresponding ternary key tree will be constructed and the subgroup keys will be parallelly computed in bottom-up pattern. The details about building a ternary key tree and computing the group key have been introduced in Section IV-A. After the platoon P is established, when a single vehicle wishes to join P, the key update will be realized by executing the vehicle join protocol which will be described in next section.

2) VEHICLE JOIN

Let N denote the number of vehicles in the platoon, N_1 and N_2 represent the numbers of nodes in M-tree and J-tree, respectively. Assume that the vehicle V wants to join the platoon.

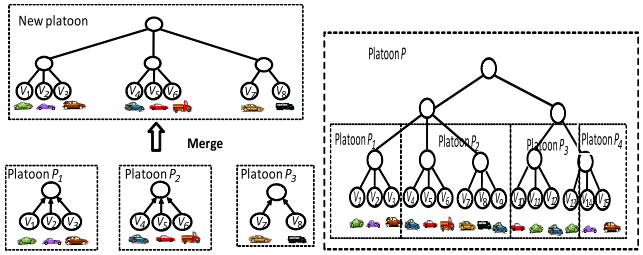
- If $N = 1$ or 2, V executes a 2- or 3-party DH protocol with the platoon members.
- When $N = 3$ or 4, V executes a 2-party DH protocol.
- If $N \ge 5$, J-tree is activated. The new user insert into J-tree procedure (as shown in IV-D.1) will be executed.
- As N_2 increases, if J-tree is full, execute Algorithm 2.
- Update the capacity of J-tree.

3) VEHICLE EXIT

If $N_1 > 38$ and $\beta = 0$, the equation 5 is satisfied. Thus, the E-tree is activated. When a vehicle leaves from the group, Algorithm 3 is performed. If E-tree batch relocation threshold is achieved, Algorithm 2 is executed. With the increase/decrease of the users in M-tree, the capacity of E-tree will be recomputed.

4) PLATOON MERGE

When multiple platoons want to merge into one big platoon, the group key for the big platoon needs to be established. We consider each small platoon as a logic user, construct a ternary key tree for the logic users, and then compute the session key for the logic users. The details to construct the key tree and compute the group key have been showed in Section IV-A.



(a) Platoon merge (b) Platoon split

FIGURE 7. Platoon merge and split. (a) Platoon merge. (b) Platoon split.

Assume that Platoon P_1 consists of vehicles V_1, V_2 and V_3; P_2 consists of vehicles V_4, V_5 and V_6; P_3 consists of vehicles V_7 and V_8; As Fig. 7 (a) shows, platoons P_1, P_2 and P_3 will merge into a new platoon. Let a_i represent the private key of V_i. Then, the session keys for platoon P_1, P_2 and P_3 are $K_1 = H(\hat{e}(g, g)^{a_1 a_2 a_3})$, $K_2 = H(\hat{e}(g, g)^{a_4 a_5 a_6})$, and $K_3 = H(g^{a_7 a_8})$, respectively. To establish a new group key, the messages g^{K_1}, g^{K_2}, and g^{K_3} will be sent among P_1, P_2 and P_3. Finally, the new group key $H(\hat{e}(g, g)^{K_1 K_2 K_3})$ is computed by all the vehicles in the platoon.

When two or more platoons merge into one platoon, all the nodes of the platoons form a main tree. According to the size of the main tree, the capacities of exit tree and join tree are computed, respectively. Then, new nodes join in join tree and nodes leave from exit tree.

5) PLATOON SPLIT

When a platoon want to split in several small platoons, a new group key needs to be established inside each small platoon for further communication. Assume that the platoon P will split in platoons $P_1,\ldots, P_n$. In a ternary key tree, each inner node bounds a key which is shared by all members of the subtrees rooted at the inner node. Thus, all these member nodes of the subtree form a subgroup.

- If such a subgroup forms a small platoon P_i (as shown in Fig. 7 (b) Platoon P_1), then it is not necessary to generate a new group key for P_i.
- If a platoon P_i consists of several subgroups (as shown in Fig. 7 (b) Platoon P_2), then each subgroup will be considered as a logic user. A ternary key tree will be constructed for these logic users to compute the shared session key.
- If there is no subgroup in P_i (as shown in Fig. 7 (b) Platoon P_4), each vehicle in P_i is considered as a node. A ternary tree will be built for the nodes and then the group key is computed.
- If P_i consists of subgroups and single vehicles (as shown in Fig. 7 (b) Platoon P_3), the subgroup will be considered as a logic user. A ternary tree will be constructed for the logic users and vehicle nodes.

V. SECURITY ANALYSIS

In this section, we discuss the security properties of the proposed TJET scheme. Particularly, according to the security requirements discussed earlier, our analysis will focus on how the proposed TJET scheme can capture the group key's forward security and backward security, authentication, integrity and resist key control. Since only active outside adversary is considered in our scheme, other attacks launched by the inside adversary, e.g., the collusion among joined vehicles and departed vehicles are out of the scope of this paper.

- Authentication. The TJET holds authentication. Thus, the attackers cannot impersonate the vehicles to execute the TJET scheme.
- Integrity. The TJET holds data integrity which ensures that data sent by vehicles has not been altered by attackers.
- Forward security. The proposed TJET scheme holds forward security. That is, as soon as a vehicle member V leaves the platoon, V cannot communicate with the vehicle members any more. Specifically, after a vehicle member V leaves a platoon, the platoon members will update the group key.
- Backward security. The proposed TJET scheme holds backward secrecy. In other words, after a vehicle member V joins a platoon, it cannot compute any previous platoon group key. That is, the new member cannot get earlier information inside the platoon. Before V joins the platoon, it did not participate the execution of the TJET scheme. Specifically, none of the messages used to compute the session key is generated by V.
- Resistance to key control. For proposed TJET scheme, any vehicle in a platoon cannot choose or influence the value of the platoon group key. The reason is the group key is computed by the random numbers generated by each vehicle in the platoon. Thus, the proposed TJET scheme can resist key control.

A. PARTICIPANTS AND NOTATIONS

We use Π_i^s to denote the instance s of player U_i involved with his/her partner players $U_1,\ldots, U_{i-1}, U_{i+1},\ldots, U_n$. Each instance Π_i^s possesses the variables sid_i^s, pid_i^s, kid_i^s, k_i^s, state_i^s and ms_i^s, which are described as follows:

- sid_i^s denotes the session identifier of instance Π_i^s. In our protocol, sid_i^s is set as concatenation of all public inputs and all messages sent and received by Π_i^s.
- pid_i^s is a set which contains U_i's identifier and the identifiers of the players with whom Π_i^s wants to share a session key. Besides, we assume the identifiers in pid_i^s are ordered according to which are in the lexicographic order.
- U_i has a unique key identifier kid_i^s.
- k_i^s represents the session key established by Π_i^s.
- state_i^s denotes the current state of Π_i^s. We say that instance Π_i^s has accepted if it possesses $\text{kid}_i^s (\neq null)$, $\text{k}_i^s (\neq null)$, pid_i^s and sid_i^s.

Definition 1: We say the instances Π_i^s and Π_j^t (where $i \neq j$) are partnered iff (a) they have accepted; (b) $\text{pid}_i^s = \text{pid}_j^t$; and (c) $\text{sid}_i^s = \text{sid}_j^t$.

Definition 2: A group key agreement protocol is said to be correct if for any instance Π_i^s and any of its partners Π_j^t, whenever Π_i^s has accepted, it holds that $k_j^t = k_i^s$.

Similar to the adversarial model of [23], we present a new adversarial model to prove our scheme achieves key secrecy, forward secrecy and backward secrecy.

B. ADVERSARIAL MODEL

We define security by using a game run between a challenger $\mathcal{C}$ playing n participants, and an adversary $\mathcal{A}$. We want to prove that $\mathcal{A}$ cannot distinguish a common group session key of any fresh instance (see **Definition 3**) from a random string. In our model, after $\mathcal{A}$ has received the challenge, he can continue starting and revealing instances except of the tested instance or any instance which is partnered with it.

$\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

- `Send`(Π_i^s, Δ): Send messages Δ to instance Π_i^s, and the answer generated by this instance.
- `k.Reveal`(Π_i^s): Return the common group key and subgroup keys.
- `Curupt`(U_i): Return the private key of U_i. (This query is used to show that even all the private keys are leaked, the previous keys will not be computed.)
- `Test`(Π_i^s): This query is allowed to issued only once. $\mathcal{A}$ submits the query, $\mathcal{C}$ picks a random bit $b \in \{0, 1\}$ uniformly, then returns k_b to $\mathcal{A}$. It is required that $\Pi_{\mathcal{U}_i}^{\pi}$ be fresh.

Finally, $\mathcal{A}$ outputs a bit b'. The advantage with which $\mathcal{A}$ wins above game is defined as

$$\text{Adv}(\mathcal{A}) = |2 \cdot \Pr[b = b'] - 1|.$$

Definition 3: An instance Π_i^s is fresh, if none of the following happens:

- *At some point, $\mathcal{A}$ issued* `k.Reveal`(Π_i^s) *or* `k.Reveal`(Π_j^t)*, where Π_i^s is partnered with Π_j^t.*
- *A query* Corrupt(U_i) *was asked before a query of the form* `Send`(Π_i^s, Δ).

TABLE 3. User average join cost (round).

Schemes	$1 \leq n \leq 5$	$5 < n \leq 8$	$n > 8$
JET	$\mathcal{O}(\log_2(n))$	$\mathcal{O}(\log_2(n))$	$\mathcal{O}(\log_2(\log_2(n)))$
TJET	$\mathcal{O}(\log_3(n))$	$\mathcal{O}(\log_3(\log_3(n)))$	$\mathcal{O}(\log_3(\log_3(n)))$

TABLE 4. User average exit cost (round).

Schemes	$1 \leq n \leq 38$	$38 < n \leq 256$	$n > 256$
JET[1]	$\mathcal{O}(\log_2(n))$	$\mathcal{O}(\log_2(n))$	$\mathcal{O}(\log_2(\log_2(n)))$
TJET	$\mathcal{O}(\log_3(n))$	$\mathcal{O}(\log_3(\log_3(n)))$	$\mathcal{O}(\log_3(\log_3(n)))$

[1] In JET, β is set around 0.5.

Definition 4: A group key protocol is said to hold key secrecy, if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage Adv$(\mathcal{A})$ *is negligible in above game.*

Theorem 3: Suppose there is an adversary $\mathcal{A}$ wins above game with non-negligible advantage Adv$(\mathcal{A})$*. Then there exists an algorithm to break DHBDH assumption or HDH assumption with with non-negligible probability of success.*

Proof of Theorem 3: See Appendix C.

VI. PERFORMANCE EVALUATION

In this section, we will compare the time efficiency of our TJET scheme and JET scheme proposed in [12]. The notations used in this section are listed in Table 1.

A. PERFORMANCE ANALYSIS

The time efficiency is measured by the numbers of rounds required to execute the DH protocol. In each round, a user executes one 2-party/3-party DH operation. In order to facilitate the comparison and analysis, we take the same definitions of average join/exit cost presented in [12]. Specifically, the average join cost T_J is represented by $T_J = \frac{R_J}{N_J}$, the average exit cost T_E is represented by $T_E = \frac{R_E}{N_E}$, and the average processing cost is $T = \frac{R_J + R_E}{N_J + N_E}$.

Assume that there have been n vehicles in the platoon. In Table 3, we compare the average user join cost of proposed TJET scheme and JET scheme. In proposed TJET scheme, when $n \leq 5$, the new user will be added into the M-tree. Therefore, the average join cost is $\mathcal{O}(\log_3(n))$ rounds. When $n > 5$, the J-tree is activated, the average join cost is reduced to $\mathcal{O}(\log_3(\log_3(n)))$ rounds. Comparatively, the join tree activation condition of JET is $n > 8$. When $n \leq 8$, in JET scheme the average join cost is $\mathcal{O}(\log_2(n))$ rounds. When $n > 8$, the average join cost is $\mathcal{O}(\log_2(\log_2(n)))$ rounds.

Table 4 shows the average user exit cost of proposed TJET scheme and JET scheme. We can see in JET when $38 < n \leq 256$, the average user exit cost is $\mathcal{O}(\log_2(n))$ rounds. In the proposed TJET, the average user exit cost has been reduced to $\mathcal{O}(\log_3(\log_3(n)))$ rounds.

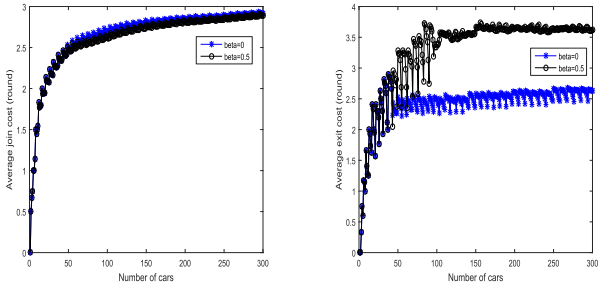
B. PERFORMANCE EVALUATION

1) EXPERIMENTAL SETTINGS

Fig. 8 (a) shows the average join cost T_J for the proposed TJET scheme varying with N_J from 0 to 300 when the

E-tree activation is 38 ($\beta = 0$) and 92 ($\beta = 0.5$) and the initial platoon size is 0. In order to indicate the variations of T_J with N_J, we let the vehicles stay enough time in the platoon, which means no vehicles leave the platoon during this period.

Fig. 8 (b) shows the average exit cost T_E for the proposed TJET scheme varying with N_E from 1 to 300 and the initial platoon size is 0. In order to indicate the change of T_E over N_E, we suppose 1) no vehicles join the platoon during this period; 2) the vehicles follow the principle of first-in-first-out (FIFO).



(a) (b)

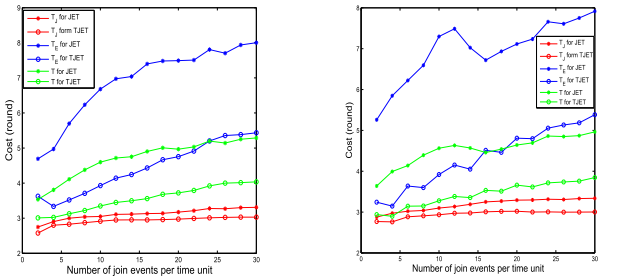
FIGURE 8. The average join/exit cost for TJET. (a) The average join cost T_J. (b) The average exit cost T_E.

In Fig. 9 and Fig. 10, we compare the average join/exit /join & exit cost with all join events and exit events occurs in the whole simulation. The initial platoon size is 0. Let each time unit contains 10 minutes, the duration of simulation is 48 time units. In the simulations, vehicles' arrival time is modeled as a Poisson process with mean arrival rate λ_i, and vehicles' staying time follows an exponential distribution with mean value m_i. Fig. 9 and Fig. 10 show the average join/exit cost varies with the average arrival rate as 2 cars per time unit, 4 cars per time unit, ..., 30 cars per time unit. In Fig. 9 (a) and Fig. 10 (a), the average time per vehicle staying in the platoon is 2 hours. In Fig. 9 (b) and Fig. 10 (b), the average time per vehicle staying in the platoon for is 4 hours. Additionally, in Fig. 9 and Fig. 10, β is set as 0.5 for JET scheme, since Mao *et al.* [12] suggested that setting β to around 0.5 for JET. In Fig. 9 and Fig. 10, β is set as 0 and 0.5 for TJET, respectively.

2) EXPERIMENTAL RESULTS

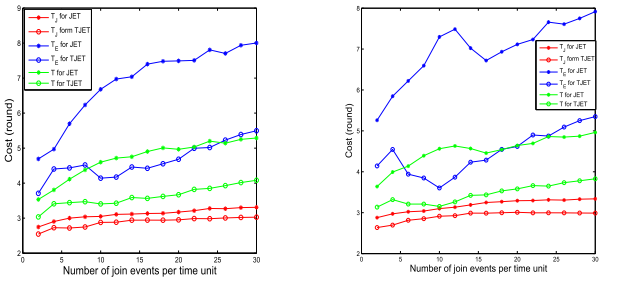
In Fig. 8 (a), we can see as N_J goes up from 1 to 300, T_J has an whole upward trend. However, in some points, T_J declines. The reason is when the J-tree is empty, the new node join cost is only 1, which makes T_J decreases. In contrast, if a new node is added and the J-tree is full, all the nodes in the J-tree are moved into the M-tree, the new node join cost will be $\log_3 N_1 + \log_3 N_2$, which makes T_J grows dramatically.

In Fig. 8 (a), as N_J rises, T_J grows slowly. That is because both the levels of M-tree and J-tree increase more slowly than the number of vehicles. Moreover, the slow growth of join cost is burdened by more vehicles. Therefore, T_J increases



(a) (b)

FIGURE 9. T_J, T_E and T for JET (Beta=0.5) and TJET (Beta=0). (a) Average staying 2 hours. (b) Average staying 4 hours.



(a) (b)

FIGURE 10. T_J, T_E and T for JET (Beta=0.5) and TJET (Beta=0.5). (a) Average staying 2 hours. (b) Average staying 4 hours.

slowly with N_J increasing. In addition, we can observe that in this case β has little effect on T_J.

In Fig. 8 (b), when $\beta = 0$ and $N_E \leq 38$, T_E increases/declines significantly at some points. For example, when $N_E = 31$, T_E are 1.9355 rounds. In contrast, T_E rises up to 2.8125 rounds when $N_E = 32$. The reason is 1) the E-tree has not been activated, all the vehicles will leave the platoon from M-tree; 2) when there are 32 nodes in the tree, the join tree contains one node. Thus, the level of each node in M-tree increases 1, which makes T_E goes up dramatically. When N_E is 33, T_E goes down to 2.7576 rounds. That is because 1) the level of each node in M-tree remains unchanged; 2) the new node spends less cost leaving the group than the current nodes in M-tree.

When $N_E > 38$, the E-tree has been activated. Thus, adding a new node will not result in the level of each node in M-tree increasing. However, as N_E increases, T_E still declines at some points. The reason is the exit cost of these new nodes is less than the old nodes. When the exit cost of new nodes is more than that of old nodes, T_E increases. Similarly, we can see when $\beta = 0.5$ (E-tree activation condition is 92) and $N_E \leq 92$, T_E increases/declines dramatically at some points. We can conclude that in this case β should be set as 0 to achieve key updating more efficiently.

From Fig. 9, we can observe that the average join/leave/ join & exit cost of the proposed TJET scheme remains less than that of JET for any arrival rate, and any staying time. Generally speaking, the smaller staying time results in fewer vehicles in the platoon and smaller sizes of M-tree, J-tree and E-tree. However, though the vehicles stay longer in Fig. 9 (b)

than that of Fig. 9 (a), the average join cost of the proposed TJET increases little. That is because in the proposed TJET, even though the number of vehicles increases a lot, the size of J-tree and M-tree only increase a little. For instance, when N_1 increases from 100 to 400, the capacity of J-tree increases from $\frac{3}{2}\ln(100) \approx 6.9$ to $\frac{3}{2}\ln(400) \approx 8.99$.

Additionally, the vehicles spend less time for key updating when new vehicles join the platoon in the proposed TJET scheme. Moreover, as the average arrival rate rises and vehicles' average staying time increases rapidly, the average join cost rises more slowly for TJET, since comparing with JET the key updating cost of TJET is $\mathcal{O}(\log_3(\log_3(n)))$ rounds.

In Fig. 9 and Fig. 10, the overall appearance of the average leave cost is that: as the average arrival rate grows, the average exit cost increases. It is because the size of the tree grows as the average arrival rate grows. However, in some point, when the arrival rate rises, T_E declines in TJET. The reason is that as the arrival rate rises, the number of vehicles increases. After the number of vehicles rises up to the E-tree activation condition, the E-tree is activated. Besides, after the E-tree is activated, the exit cost of some new nodes may be less than that of old nodes. Therefore, T_E decreases at some points. From Fig. 9 and Fig. 10, we can see that the vehicles spend less time updating the group key for user joining or leaving in TJET than that of JET both for $\beta = 0$ and $\beta = 0.5$.

VII. RELATED WORKS

The related work about key management for vehicular platooning is not introduced in this section, since there is no dynamic key management scheme for platoon proposed before. In this section, we will mainly explore some of the existing platoon management protocols and tree-based key management schemes.

In [24], the authors considered how to merge two platoons into one big platoon. They defined the data shared among the platoon members and described how to utilize the data to realize the merger operation. In [25], the authors introduced the design of an on-board supervisor to achieve new user joining, vehicle exit with an automated lane change, and making a platoon stop at the end of a highway. Hall *et al.* [26] considered the platoon as a multi-agent system to capture inter-vehicle coordination. They assigned roles to the vehicles. For instance, the vehicle which will exit is named as splitter, the vehicle behind it is gap creator and other vehicles which monitor the situation are safety observers. Jia *et al.* [1] proposed a novel disturbance-adaptive platoon architecture. They discussed the traffic dynamics inside the platoons in disturbance scenario and optimize the distance between two platoon members and the platoon size. In order to distinguish the messages sent by the vehicles from different platoons, platoon-ID should be allocated. Filter-multicast [27] is proposed to realize distributing dynamic platoon-ID. Segata *et al.* [28] proposed a vehicle-platoon protocol to identify the platoon formation and predict platoon splits. A data management component is also designed to inform the platoon members to replicate the data, since a vehicle contributes parts of their buffers to other platoon members and shared data with them. To maximize the platoon size and the life time of a platoon, Hall et al. [29] proposed a platoon management strategy. Dao *et al.* [30] proposed a distribute-control strategy, by which cars could be sorted to appropriate platoons and lanes, and the balance of lane traffic flows and decreasing the vehicle travel time could be achieved. Though numerous work have been undertaken for platoon management, they never take the secure communication inside the platoon into consideration yet do they consider the necessity of secure key establishment for platoons.

There have been some simple and efficient key management schemes proposed for group communication. Kim *et al.* [31] proposed a secure and efficient key management protocol based on binary tree. In their scheme, the leaf nodes host members and every parent of two nodes blinds a key. In Kim *et al.*'s scheme, the computation cost is $\mathcal{O}(\log_2(n))$ and DH key agreement is explored to compute and update the group key. In order to improve the time efficiency of key update, Lee *et al.* [10] proposed a novel tree based group key agreement based on ternary tree using bilinear map. They reduced the computation cost to $\mathcal{O}(\log_3(n))$. Dutta and Barua [11] proposed an authenticated dynamic group key agreement scheme in tree setting based on [32] and [33]. In [11], $\mathcal{O}(\log_3(n))$ key updates are required in the execution of user insertion and deletion. Mao *et al.* [12] presented a join-exit-tree based key management scheme. In their scheme, the computation cost of key update is reduced to $\mathcal{O}(\log_2(\log_2(n))$. Besides, some other excellent constant-round key agreement protocols are proposed, e.g., the schemes in [25] and [36] only require one round to establish a group key. However, the protocol is not constructed for dynamic groups. That is, all the group members have to re-execute the whole protocol for any join/exit event.

Nam *et al.* [35] proposed a key management protocol. However, Tseng [36] pointed out that Nam *et al.*'s protocol was a key distribution scheme, but not a real key agreement protocol. In a key distribution scheme, a secret key between the key distribution centre and user needs to be established before the key distribution scheme is executed. Though Tseng [37] proposed a more efficient key agreement protocol. However, it is static, and not suitable in mobile networks. In [38], Dutta and Barua presented a group key agreement protocol (called Dutta-Barua protocol) in dynamic setting with which users can efficiently join or leave a group. Then, they proposed a formal security model and provided security proofs in [39] to evidence the security of Dutta-Barua protocol. Teo *et al.* [40] showed that the protocol in [38] fails to achieve backward secrecy. Besides, Tan and Yang [41] showed that the Dutta-Barua protocol is insecure against an outsider adversary defined by the adversarial model in [39]. Junbeom Hur and Younho Lee [42] proposed a key management scheme. However, it is still a distribution scheme. When a member joins or leaves the group, secret group keys are updated and delivered to the valid group members using the key update protocol. Teng and Wu [43] proposed a key

agreement protocol for low-power mobile devices. In their protocol, a powerful node (also called server) will generate the messages and send them to all the users to establish the group key. In addition, authenticated private channel between the key server and the users are assumed during system setup and key distribution for a newly joining member. Seo *et al.* [44] has established a certificateless effective key management scheme in dynamic wireless sensor networks. In their scheme, only the cluster head can update the cluster key when a node leaves or joins the cluster, after updating the group key, the cluster head will send the key to the cluster members. Tang *et al.* [45] presented a novel group key management designed with a group controller (GC). In their approach, a hyper-sphere is constructed for a group and each member in the group corresponds to the member's private point on the hyper-sphere. The GC computes the central point of the hyper-sphere, and publishes it such that each member can compute a common group key, using a function by taking members' private points and the central point of the hyper-sphere as the input. Thus, the members cannot establish a common group key without GC. Büttner *et al.* [46] has proposed an anonymous authenticated key agreement protocol for vehicular Ad-Hoc Networks. In their scheme, the key security and privacy preservation of the vehicles are achieved. However, only two participants are involved in their scheme. Yang *et al.* [47] proposed an efficient group key agreement scheme for mobile Ad-Hoc Networks based on Identity Based Broadcast Encryption methodology to enable secure group communications. Their scheme only requires constant rounds. When a new Ad-hoc network is constructed, the suggested scheme requires no message exchange to establish a group key if the receivers' identities are known to the broadcaster. However, the session key is not computed by the group members together, but generated by broadcaster. That is, their scheme cannot resist key control. Shanthi and Murugan [48] proposed hop by hop group key agreement protocol where each node generates pairwise key for encryption of data. After that, group key agreement protocol is executed to provide complete authentication for those nodes. However, the user join and exit is not considered in this paper.

VIII. CONCLUSIONS

In this paper, we have proposed an efficient key management scheme (TJET) which achieves dynamic group key update for vehicle platooning. To realize TJET, we have designed ternary join-exit-tree based dynamic join and leave algorithms. TJET's efficacy has been evaluated with theoretical analysis and experiments, and detailed analysis shows it holds forward security, backward security and can resist key control. Compared with JET scheme, the proposed TJET scheme has been identified to significantly reduce the cost of group key updating for the vehicle joining and leaving by strict mathematical proof. In addition, through extensive performance evaluations, we have further demonstrated that the proposed TJET scheme can achieve much better efficiency in terms of the average join time and average leave time during key update procedure.

APPENDIX A
PROOF OF THEOREM 1

Proof: This theorem is proven with mathematical induction. As Table 2 shows, when $p = 0, 1, 2$ the equation (2) holds. Assume that the equation (2) holds for $n = 3^p$, i.e., $\sum_{i=1}^{3^p} R(i) = 3^p(\frac{2}{3}p + 1)$ is satisfied. When $n = 3^{p+1}$, we obtain

$$\begin{aligned}\sum_{i=1}^{3^{p+1}} R(i) &= \sum_{i=1}^{3^p} R(i) + \sum_{i=3^p+1}^{3^p+3^p} R(i) + \sum_{i=3^p+3^p+1}^{3^{p+1}} R(i) \\ &= \sum_{i=1}^{3^p} R(i) + \sum_{i=1}^{3^p} R(i+3^p) + \sum_{i=1}^{3^p} R(3^p+3^p+i)\end{aligned}$$

According to the equation (1), we have

$$\begin{aligned}&\sum_{i=1}^{3^p} R(i) + \sum_{i=1}^{3^p} R(i+3^p) + \sum_{i=1}^{3^p} R(3^p+3^p+i) \\ &= \sum_{i=1}^{3^p} R(i) + (3^p + \sum_{i=1}^{3^p} R(i)) + (3^p + \sum_{i=1}^{3^p} R(i)) \\ &= 3^{p+1} + 3^p \cdot 2p + 2 \cdot 3^P \\ &= 3^{p+1}(\frac{2}{3}(p+1) + 1)\end{aligned}$$

That is, when $n = 3^{p+1}$, $\sum_{i=1}^{n} R(i) = n(\frac{2}{3}\log_3 n + 1)$ holds. □

APPENDIX B
PROOF OF THEOREM 2

Proof: This theorem is also proven with mathematical induction. When $n = 3$ and $n = 9$, according to Table 2 the inequality $\frac{\sum_{i=1}^{k} r_i}{k} < \frac{\sum_{i=1}^{n} r_i}{n}$ holds. Assume for $n = 3^p$ and $1 \le k < n$, the inequality $\frac{\sum_{i=1}^{k} r_i}{k} < \frac{\sum_{i=1}^{n} r_i}{n}$ is satisfied. When $n = 3^p$, assume that $r_1 = q_1$, $r_2 = q_2, \ldots, r_n = q_n$, then according to equation (1), we have $r_{n+1} = 1 + q_1$, $r_{n+2} = 1 + q_2, \ldots, r_{2n} = 1 + q_n$, and $r_{2n+1} = 1 + q_1$, $r_{2n+2} = 1 + q_2, \ldots, r_{3n} = 1 + q_n$. Let $a_n = \frac{\sum_{i=1}^{n} r_i}{n}$. When $n = 3^{p+1}$, for $3^p \le k < 3^{p+1}$, we obtain

$$\begin{aligned}\sum_{i=1}^{k} r_i &= r_1 + r_2 + \cdots + r_k \\ &= q_1 + q_2 + \ldots + q_{n/3} + q_{n/3+1} + q_{n/3+2} + \ldots + q_k \\ &= a_{n/3} \cdot \frac{n}{3} + q_{n/3+1} + q_{n/3+2} + \ldots + q_k \\ &< a_{n/3} \cdot \frac{n}{3} + (k - \frac{n}{3})(a_{n/3} + 1) \\ &= k a_{n/3} + k - \frac{n}{3}\end{aligned}$$

and $\frac{\sum_{i=1}^{k} r_i}{k} = a_{n/3} + 1 - \frac{n}{3k}$. Since $k < n$, we have $a_{n/3} + 1 - \frac{n}{3k} < a_{n/3} + \frac{2}{3}$. When $n = 3^{p+1}$, according to the equation (3), we have $\frac{\sum_{i=1}^{n} r_i}{n} = a_{n/3} + \frac{2}{3}$. Therefore, when $n = 3^{p+1}$, for

$3^p \leq k < 3^{p+1}$, we obtain $\frac{\sum_{i=1}^{k} r_i}{k} < \frac{\sum_{i=1}^{n} r_i}{n}$. Thus, when p is a non-negative integer, for $n = 3^p$, and $1 \leq k < n$, $\frac{\sum_{i=1}^{k} r_i}{k} < \frac{\sum_{i=1}^{n} r_i}{n}$ holds. □

APPENDIX C
PROOF OF THEOREM 3

Proof: Let $\mathcal{C}$ be a challenger. Suppose $\mathcal{C}$ receives an instance of the DHBDH problem, i.e., given $g, g^{\alpha}, g^{\beta}, g^{\gamma}$, $H(\hat{e}(g,g)^{\alpha\beta\gamma})$ and T to distinguish $H(\hat{e}(g,g)^{\alpha\beta\gamma})$ and T, and an instance of the HDH problem, i.e., given $(g, g^{\alpha}, g^{\beta})$, $H(g^{\alpha\beta})$ and T' to distinguish $H(g^{\alpha\beta})$ and T'.

Let $\mathcal{A}$ be the adversary who wants to break the protocol. We will show how to break DHBDH assumption or HDH assumption by using $\mathcal{A}$. $\mathcal{C}$ passes the system parameters to $\mathcal{A}$. H is treated as a random oracle.

Assume that the adversary executes q_e Send queries. $\mathcal{C}$ picks uniformly l from $[1, q_e]$, and set $c_{\mathrm{sid}_i^l} = 1$, and $c_{\mathrm{sid}_i^{l'}} = 0$, where $l' \in [1, q_e]$, $i \neq l$. After $\mathcal{C}$ completes the initialization process, answers them as follows:

H queries: After $\mathcal{A}$ submits H queries, $\mathcal{C}$ passes the queries to H hash function. However, if $\mathcal{A}$ queries a secret value of some inner node, abort (**Event 1**).

k.Reveal(Π_i^s): $\mathcal{C}$ recovers $c_{\mathrm{sid}_i^s}$ corresponding to sid_i^s, then $\mathcal{C}$ preforms the follow steps:

- If $c_{\mathrm{sid}_i^s} = 0$, abort (**Event 2**).
- Else if $\mathrm{state}_i^s \neq accepted$, $\mathcal{C}$ outputs *null*;
- Else, $\mathcal{C}$ outputs the group session key by executing the protocol.

Send(Π_i^s, Δ): $\mathcal{C}$ performs the following steps:

- If $c_{\mathrm{sid}_i^s} = 0$ and there are three nodes in the level 1, $\mathcal{C}$ sets the messages sent by the nodes in level 1 as g^{α}, g^{β} and g^{γ} respectively, and generates the messages in other levels by running the protocol. If the adversary generates messages and generate a valid forged signature, abort (**Event 3**).
- Else if $c_{\mathrm{sid}_i^s} = 0$ and there are two nodes in the level 1, $\mathcal{C}$ sets the messages sent by the nodes in level 1 as g^{α} and g^{β} respectively, and generates the messages in other levels by running the protocol. If the adversary generates messages and generate a valid forged signature, abort (**Event 4**).
- Else if $c_{\mathrm{sid}_i^s} = 1$, $\mathcal{C}$ executes the scheme normally.

Corrupt(U_i): Return the private key of U_i to the adversary.

Test(Π_i^s): A fresh ęijinstance Π_i^s is chosen by $\mathcal{A}$. If $c_{\mathrm{sid}_i^s} = 0$, $\mathcal{C}$ preforms the follow steps:

1. If there are three nodes in Level 1 of the key tree, $\mathcal{C}$ sends $b_0 = T$ and $b_1 = H(\hat{e}(g,g)^{\alpha\beta\gamma})$ to $\mathcal{A}$. After $\mathcal{A}$ outputs its guess b_i, $i \in \{0, 1\}$ as the group key, $\mathcal{C}$ outputs b_i as the answer that g^{α}, g^{β}, g^{γ} and b_i is a BDH tuple.
2. Else if there are two nodes in Level 1 of the key tree, $\mathcal{C}$ sends $b_0 = T'$ and $b_1 = H(g^{\alpha\beta})$ to $\mathcal{A}$. After $\mathcal{A}$ outputs its guess, $\mathcal{C}$ answers the HDH challenge.

$\mathcal{A}$ cannot find inconsistence between the real world and the simulation, since all oracles are simulated validly and the messages output by the oracles are distributed uniformly in the message space. Thus, we have $\Pr[b = b'] \geq \mathrm{Adv}(\mathcal{A})$. We can see that if **Event 1** or **Event 2** or **Event 3** or **Event 4** happens, $\mathcal{C}$ will abort. Suppose $\mathcal{A}$ asks at most q_H H queries, q_e **send queries**, q_k k.Reveal queries and the inner nodes in all the instances are m. We have $\Pr[\neg\textbf{Event 1}] \geq 1 - mq_H \max\{\mathrm{Adv}_{BDH}, \mathrm{Adv}_{CDH}, \frac{1}{P}\}$, $\Pr[\neg\textbf{Event 2}] \geq 1 - \frac{q_r}{q_e}$, $\Pr[\neg\textbf{Event 3}] \geq 1 - q_{e_1} n\mathrm{Adv}_{sig}$, and $\Pr[\neg\textbf{Event 4}] \geq 1 - q_{e_2} n\mathrm{Adv}_{sig}$ where n is the number of participants. Since Event 1, Event 2, Event 3, and Event 4 are independent, the probability with which $\mathcal{C}$ does not abort is $(1 - mq_H \max\{\mathrm{Adv}_{BDH}, \mathrm{Adv}_{CDH}, \frac{1}{P}\})(1 - \frac{q_r}{q_e})\ (1 - q_{e_2} n\mathrm{Adv}_s ig)$ $(1 - q_{e_1} n\mathrm{Adv}_s ig)$, which is non-negligible. Assume there are q_{e_1} Send queries in which there exist three nodes in Level 1, and q_{e_2} Send queries in which there exist two nodes in Level 1, where $q_{e_1} + q_{e_2} = q_e$.

To summarize, the probability with which $\mathcal{C}$ can break the DHBDH is $(1 - mq_H \max\{\mathrm{Adv}_{BDH}, \mathrm{Adv}_{CDH}, \frac{1}{P}\})(1 - \frac{q_r}{q_e})\frac{q_{e_1}}{q_e}(1 - q_{e_1} n\mathrm{Adv}_{sig})\mathrm{Adv}(\mathcal{A})$, and the probability with which $\mathcal{C}$ can break the HDH is $(1 - mq_H \max\{\mathrm{Adv}_{BDH}, \mathrm{Adv}_{CDH}, \frac{1}{P}\})(1 - \frac{q_r}{q_e})\frac{q_{e_2}}{q_e}\ (1 - q_{e_2} n\mathrm{Adv}_{sig})\ \mathrm{Adv}(\mathcal{A})$. Therefore, $\mathrm{Adv}(\mathcal{A})$ is negligible.

AVAILABILITY

The Java implementation and source codes of the proposed TJET scheme are available at http://www.cs.unb.ca/~rlu1/project/index.htm#trans-xu.

REFERENCES

[1] D. Jia, K. Lu, and J. Wang, "A disturbance-adaptive design for VANET-enabled vehicle platoon," *IEEE Trans. Veh. Technol.*, vol. 63, no. 2, pp. 527–539, Feb. 2014.

[2] H. Hu, R. Lu, Z. Zhang, and J. Shao, "REPLACE: A reliable trust-based platoon service recommendation scheme in VANET," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1786–1797, Feb. 2017.

[3] H. Hu, R. Lu, and Z. Zhang, "TPSQ: Trust-based platoon service query via vehicular communications," *Peer Peer Netw. Appl.*, vol. 10, no. 1, pp. 262–277, 2017.

[4] *About PATH*. Accessed: Jun. 13, 2017. [Online]. Available: http://www.path.berkeley.edu/

[5] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by VANET," *Veh. Commun.*, vol. 2, no. 2, pp. 110–123, 2015.

[6] L. Xu, L. Y. Wang, G. Yin, and H. Zhang, "Communication information structures and contents for enhanced safety of highway vehicle platoons," *IEEE Trans. Veh. Technol.*, vol. 63, no. 9, pp. 4206–4220, Nov. 2014.

[7] S. Dadras, R. M. Gerdes, and R. Sharma, "Vehicular platooning in an adversarial environment," in *Proc. 10th ACM Symp. Inf., Comput. Commun. Security*, 2015, pp. 167–178.

[8] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in *Algorithmic Number Theory*. Berlin, Germany: Springer, 2000, pp. 385–393.

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

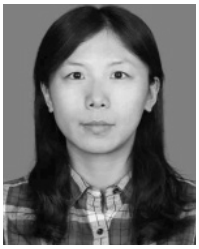
[10] S. Lee, Y. Kim, K. Kim, and D.-H. Ryu, "An efficient tree-based group key agreement using bilinear map," in *Applied Cryptography and Network Security*. Berlin, Germany: Springer, 2003, pp. 357–371.

[11] R. Dutta and R. Barua, "Dynamic group key agreement in tree-based setting," in *Information Security and Privacy*. Berlin, Germany: Springer, 2005, pp. 101–112.

[12] Y. Mao, Y. Sun, M. Wu, and K. J. R. Liu, "JET: Dynamic join-exit-tree amortization and scheduling for contributory key management," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 1128–1140, Oct. 2006.

[13] C. Boyd and A. Mathuria, *Protocols for Authentication Key Establishment* (Information Security and Cryptography). Berlin, Germany: Springer, 2003.

[14] T. Leinmuller, R. K. Schmidt, E. Schoch, A. Held, and G. Schafer, "Modeling roadside attacker behavior in VANETs," in *Proc. IEEE Globecom Workshops*, Dec. 2008, pp. 1–10.
[15] F. A. Ghaleb, A. Zainal, and M. A. Rassam, "Data verification and misbehavior detection in vehicular ad-hoc networks," *J. Teknologi*, vol. 73, no. 2, pp. 37–44, 2015.
[16] S. Dietzel, J. Petit, G. Heijenk, and F. Kargl, "Graph-based metrics for insider attack detection in VANET multihop data dissemination protocols," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1505–1518, May 2013.
[17] J. M.-Y. Lim, Y. C. Chang, J. Loo, and M. Y. Alias, "Improving VANET performance with heuristic and adaptive fuzzy logic scheme," *Wireless Pers. Commun.*, vol. 83, no. 3, pp. 1779–1800, 2015.
[18] H. Jiang, Y. Yang, J. Xu, and L. Wang, "Estimation of packet error rate at wireless link of VANET," in *Advances in Wireless Sensors and Sensor Networks*. Berlin, Germany: Springer, 2010, pp. 329–359.
[19] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.
[20] M. Abdalla, M. Bellare, and P. Rogaway, "The oracle diffie-hellman assumptions and an analysis of DHIES," in *Proc. Topics Cryptol. Cryptographer's Track Conf. (CT-RSA)*, San Francisco, CA, USA, Apr. 2001, pp. 143–158.
[21] R. Barua, R. Dutta, and P. Sarkar, "Extending Joux's protocol to multi party key agreement," in *Proc. 4th Int. Conf. Cryptol. Prog. Cryptol. (INDOCRYPT)*, 2003, pp. 205–217.
[22] K. Becker and U. Wille, "Communication complexity of group key distribution," in *Proc. 5th ACM Conf. Comput. Commun. Security*, 1998, pp. 1–6.
[23] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, "Asymmetric group key agreement," in *Advances in Cryptology-EUROCRYPT*. Berlin, Germany: Springer, 2009, pp. 153–170.
[24] R. Horowitz and P. Varaiya, "Control design of an automated highway system," *Proc. IEEE*, vol. 88, no. 7, pp. 913–925, Jul. 2000.
[25] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 4, pp. 695–708, Jul. 2000.
[26] S. Hallé, J. Laumonier, and B. Chaib-Draa, "A decentralized approach to collaborative driving coordination," in *Proc. 7th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2004, pp. 453–458.
[27] A. Uchikawa, R. Hatori, T. Kuroki, and H. Shigeno, "Filter multicast: A dynamic platooning management method," in *Proc. 7th IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2010, pp. 1–5.
[28] M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. Lo Cigno, "Supporting platooning maneuvers through IVC: An initial protocol analysis for the join maneuver," in *Proc. 11th Annu. Conf. Wireless Demand Netw. Syst. Services (WONS)*, Apr. 2014, pp. 130–137.
[29] R. Hall and C. Chin, "Vehicle sorting for platoon formation: Impacts on highway entry and throughput," *Transp. Res. C, Emerg. Technol.*, vol. 13, no. 5, pp. 405–420, 2005.
[30] T.-S. Dao, C. M. Clark, and J. P. Huissoon, "Distributed platoon assignment and lane selection for traffic flow optimization," in *Proc. Intell. Veh. Symp.*, 2008, pp. 739–744.
[31] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Trans. Inf. Syst. Security*, vol. 7, no. 1, pp. 60–96, 2004.
[32] R. Dutta, R. Barua, and P. Sarkar, "Provably secure authenticated tree based group key agreement," in *Information and Communications Security*. Berlin, Germany: Springer, 2004, pp. 92–104.
[33] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group diffie-hellman key exchange under standard assumptions," in *Advances in Cryptology-EUROCRYPT*. Berlin, Germany: Springer, 2002, pp. 321–336.
[34] C. Xu, Z. Li, Y. Mu, H. Guo, and T. Guo, "Affiliation-hiding authenticated asymmetric group key agreement," *Comput. J.*, vol. 55, no. 10, pp. 1180–1191, 2012.
[35] J. Nam, J. Lee, S. Kim, and D. Won, "DDH-based group key agreement in a mobile environment," *J. Syst. Softw.*, vol. 78, no. 1, pp. 73–83, 2005.
[36] Y. Tseng, "On the security of two group key agreement protocols for mobile devices," in *Proc. 7th Int. Conf. Mobile Data Manage. (MDM)*, Nara, Japan, May 2006, p. 97.
[37] Y. Tseng, "A secure authenticated group key agreement protocol for resource-limited mobile devices," *Comput. J.*, vol. 50, no. 1, pp. 41–52, 2007.
[38] R. Dutta and R. Barua, "Constant round dynamic group key agreement," in *Proc. 8th Int. Conf. Inf. Security (ISC)*, Singapore, Sep. 2005, pp. 74–88.
[39] R. Dutta and R. Barua, "Provably secure constant round contributory group key agreement in dynamic setting," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2007–2025, May 2008.
[40] J. C. M. Teo, C. H. Tan, and J. M. Ng, "Security analysis of provably secure constant round dynamic group key agreement," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 89-A, no. 11, pp. 3348–3350, 2006.
[41] C. H. Tan and G. Yang, "Comments on 'provably secure constant round contributory group key agreement in dynamic setting,'" *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5887–5888, Nov. 2010.
[42] J. Hur and Y. Lee, "A reliable group key management scheme for broadcast encryption," *J. Commun. Netw.*, vol. 18, no. 2, pp. 246–260, Apr. 2016.
[43] J. Teng and C. Wu, "An identity-based group key agreement protocol for low-power mobile devices," *Chin. J. Electron.*, vol. 25, no. 4, pp. 726–733, 2016.
[44] S. H. Seo, J. Won, S. Sultana, and E. Bertino, "Effective key management in dynamic wireless sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 371–383, Feb. 2015.
[45] S. Tang, L. Xu, N. Liu, X. Huang, J. Ding, and Z. Yang, "Provably secure group key management approach based upon hyper-sphere," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3253–3263, Dec. 2014.
[46] C. Büttner, F. Bartels, and S. A. Huss, "Real-world evaluation of an anonymous authenticated key agreement protocol for vehicular ad-hoc networks," in *Proc. 11th IEEE Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2015, pp. 651–658.
[47] Y. Yang, Y. Hu, C. Sun, C. Lv, and L. Zhang, "An efficient group key agreement scheme for mobile ad-hoc networks," *Int. Arab J. Inf. Technol.*, vol. 10, no. 1, pp. 10–17, 2013.
[48] K. Shanthi and D. Murugan, "Pair-wise key agreement and hop-by-hop authentication protocol for MANET," *Wireless Netw.*, vol. 23, no. 4, pp. 1025–1033, 2017.



CHANG XU received the bachelor's and master's degrees from the School of Computer Science and Technology, Jilin University, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from Beihang University in 2013. She is currently an Assistant Professor with the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests include security and privacy in VANET and big data security.

RONGXING LU received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2006, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012. From 2012 to 2013, he was a Post-Doctoral Fellow with the University of Waterloo. From 2013 to 2016, he was an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Since 2016, he has been an Assistant Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. His research interests include computer network security, mobile and wireless communication security, and applied cryptography. He was a recipient of the Canada Governor General Gold Medal.

HUAXIONG WANG received the Ph.D. degree in mathematics from the University of Haifa, Israel, in 1996, and the Ph.D. degree in computer science from the University of Wollongong, Australia, in 2001. Since 2006, he has been an Associate Professor with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. His research interests include cryptography, information Security, coding theory, and theoretical computer Science.

LIEHUANG ZHU received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2004. He is currently a Professor with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests include security protocol analysis and design, group key exchange protocol, wireless sensor network, and cloud computing.

CHENG HUANG received the B.Eng. and M.Eng. degrees from Xidian University, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada. He was a Project Officer with the INFINITUS Laboratory, School of Electrical and Electronic Engineering, Nanyang Technological University, until 2016. His research interests are in the areas of applied cryptography, cyber security, and privacy.