Group-based Secure Source Authentication Protocol for VANETs

You Lu*, Biao Zhou**, Fei Jia*, Mario Gerla**
*Beijing University of Posts and Telecommunications, Beijing 100876, P.R. China
**IAC Search & Media, Campbell, CA 95008, USA

Abstract—Recently the security in vehicular ad hoc networks (VANETs) draws attention because of increasing vehicular applications. One of the main challenges of secured communication in VANET is multicast source authentication which essentially guarantees that the received data in communication is genuinely sent from the source and not modified by impersonators. Although much research has been focused on message authentication, there are no general accepted solutions widely used for source authentication in VANET because of its dynamic network topology and complicated membership management. In this paper we proposed a Group-based Source Authentication protocol (GSA) to handle the message authenticity in VANETs.

Many VANET applications have natural group property and VANET nodes follow the similar moving pattern. GSA makes use of group attributes as dynamic group key to protect data transmission in intra-group communication, which is dynamic changing with real-time environment and consistently updates among group members. Then GSA deploys the promising TESLA scheme to perform source authentication in inter-group communication. The results from our implementation show that GSA can guarantee multicast source authenticity and significantly enhance the efficiency of authentication for multicast communication in VANETs, and our implementation testbed also shows that GSA scheme can be easily deployed in real VANET environment.

Index Terms—source authentication, multicast security

I. INTRODUCTION

Envisioned applications such as traffic accident prevention, battlefield control commands transmission and emergency coordination rescue arose a great deal of academic and industrial research on Vehicular Ad hoc Networks (VANETs). Vehicles are equipped with wireless communication devices known as On-Board Units (OBU) which enables them to communicate with other vehicles and Roadside Units. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) are two typical modes of vehicle communication in VANETs.

With the growth and commercialization of the VANETs' applications, simultaneous transmission of data to multiple receivers becomes a prevalent mode of communication. Securing multicast communication in VANETs introduces a number of difficulties that are not encountered when trying to secure unicast communication. Source authentication is one of the most important requirements in multicast transmissions for VANET security. There are a lot of challenges of VANETs’ multicast applications. The traditional point-to-point authentication mechanism (i.e. symmetric key) cannot provide secure source authentication in the case of multicast application in VANETs. The problem is that any receiver that has the shared key can forge data and impersonate the sender. Furthermore, the asymmetric cryptography (i.e. PKI and digital signature schemes) has high overhead, in terms of time to encoding and decoding computation, and in terms of bandwidth consumption [1]. This overhead is a critical problem for the embedded wireless communication device such as OBU. Several schemes were proposed to secure the source authentication in multicast applications based on TESLA (Times Efficient Stream Loss-tolerant Authentication [1]) as summarized in section 2. But none of them can be adopted in VANETs’ multicast applications because the long authentication time cannot adapt the vehicle’s dynamic topology. Therefore, a better and more efficient scheme for secure multicast source authentication in VANET must be designed.

Based on our observation, there are natural group attributes existing in many VANET applications. A kind of classical application is vehicle platoon and co-operative driving. There are many platoon applications where vehicles are often organized as a group with similar mobility patterns and similar view of the environment. For example, in military troops’ application, vehicles travel in the similar speed and keep in the limited distance. The similar scenario is criminal tracking. When police cars track the escaped criminals, they coordinated with each other and follow the similar movement trail. So it is a natural motivation to consider taking advantage of group attribute for VANETs’ source authentication protocol in multicast applications to reduce the long authentication time in the TESLA-based scheme.

In this paper, we developed a new security scheme called GSA (Group-based Secure Source Authentication Protocol for VANET), which is based on TESLA and combined group attribute to avoid long authentication delay generated by TESLA. GSA adopts different security schemes for inter-group and intra-group communication and aim to provide efficient secure protect for inter-group communication and instant source authentication for intra-group applications.

The rest of paper is organized as follows. Section 2 discusses the related work. Section 3 describes the system design of GSA. Section 4 demonstrates the implementation and evaluation of the GSA protocol. Section 5 concludes the paper.

II. RELATED WORK

A few papers have considered adopting Time Efficient security mechanism in VANET based on the TESLA scheme. TESLA is proposed in [1] to perform source authentication. It uses symmetric cryptographic primitives and depends on key disclosure delay to achieve asymmetric feature. Xiaodong Lin, etc in [2] proposed a scheme called TSVC to authenticate the source of the message in VANET. TSVC sets the key disclosure delay to be 100ms which is about 10 times of the communication latency. A modified version of TESLA, TESLA++, is advanced
The main contribution of TESLA++ is to provide resilience to memory-based DoS attacks. [4] provides a solution to do immediate authentication in TESLA receiver end. A sender in TESLA buffers packets for the duration of one disclosure delay and hashes the value of message chunk \(<i+d>\) in message \(i\). This scheme proved to be effective for DoS attack, but has not avoided the long disclosure delay. In TESLA-related schemes above, the relative long disclosure delay cannot be tolerant in many fast-response applications such as military common-executing application in VANETs.

To provide location privacy and user privacy, group concept is introduced in VANET security area [5] [6] in these years. Group concepts can be implemented intuitively in many VANET applications, especially for platoon-like application [8]. M. Raya [7] gives two general categories forming groups: preset group and on-the-fly group. GKMP (Group Key Management Protocol) [9] is a simple and efficient protocol to establish symmetric group key. In many VANET application scenarios, nodes often form a group to accomplish a task cooperatively. We are inspired to take advantage of natural group attributes of VANET scenarios and TESLA scheme to guarantee multicast source authentication.

III. SYSTEM DESIGN

3.1 Application Scenario

Group formation in our secure source authentication protocol (GSA) is flexible and variable according to different application scenarios. In this paper, we analyze one application scenario in military vehicle troops as an example to present the functionality and availability of GSA.

Referring to the group organization in [11], we arrange the military vehicles into several squads. As shown in Figure 1, a squad is consists of several military vehicles and a leader (The blue vehicle) is assigned beforehand and served as a trust authority (TA) to multicast commands to each squad member. Members in a squad have similar spatial and temporal characteristics. For instance, they have similar choice for new direction, new destination, travelling velocity and stop/active intervals. In GSA, the squads could be treated as mobility groups. The group leaders will multicast secure messages to group members, and also the group leader will communicate with other groups.

![Figure 1. Groups in Military Troops](image)

3.2 Group Concept

Many existed mobility model is complicate and consuming large computation resource [10] [11]. In GSA protocol, group attributes should be defined proactively, updated periodically and distinguish group membership efficiently. Therefore, we select a few easy-to-compute mobility characters to define a group. For example, in military troops’ scenario, it is reasonable to define a vehicle that can offer a valid group certificate and is within a certain range of the group leader as a valid group member. Physical distance is a rigid secure group attribute for group members because an enemy vehicle hardly accesses to our territory and fakes as a group leader to disseminate information in military environment. In criminal tracking scenario, all of the police cars follow the similar movement trail and have the same GPS. So the Passing Building ID can be another secure group attribute.

3.3 GSA Scheme

In the following we will present the proposed group-based source authentication scheme (GSA). On the assumption that every vehicle is equipped with GPS to gather position information, GSA synchronizes time of different vehicles by GPS.

3.3.1 Initialization Phase

Before sending data packets, the multicast server gathers receivers’ group attribute to authenticate their affiliation. If a receiver belongs to the same group as the server, the server will speed up key disclosure. We make an assumption that initial group members will not become attackers. The server needs to verify that the received group attribute originated from the purported receiver. Here, group authentication process use original TESLA scheme to perform source authentication. Figure 2 shows the communication initialization process and we will describe it in the following.

![Figure 2. Communication Initialization Process](image)

3.3.1.1 Sender Setup

In GSA protocol, a multicast server divides the time into uniform intervals \(T_{int}\) and commits packet \(P(i)\) which is sent in interval \(Ti(j)\) with the key \(K(i)\). \(T_{int}\) can be 100 milliseconds to 1 second [4] and should be larger than the upper bound of the network delay between sender and receivers. For intra-group communication, the sender discloses encryption \(K(i)\) with the current packet \(P(i)\). For inter-group communication, the sender discloses encryption \(K(i)\) with the future packet \(P(i+d)\). The sender needs to choose proper key disclosure interval \(d\) in setup step.

Just as TESLA did [1], the sender produces a one-way hash chain \(K(x)\) with a randomly selected seed \(S\) to commit data packets. In the hash chain, \(S=K(N)\) and \(K(i)=F^iK(N)\). Here, \(N\) is the length of hash chain. \(F\) is a pseudo-random function and \(K(i)\)
is the committed key to calculate MAC (Message Authentication code) of \( P(i) \). The choice of hash chain length is related with security level requirements and the length of data trunks need to be transmitted.

### 3.3.1.2 Group Membership authentication

In a GSA group, each group member is assigned with an anonymous group certificate \( GID \) (a Group Identification Certificate) which is a hashed value of the private maintained group identification \( gid \) for secure concerning. Here, we define that \( gid \) is 32 bits and begins with character ‘G’. We use MD5 as the hash function to keep \( gid \) private. \( GID = MD5(gid) \) with 128 bits output. In the beginning of a GSA session, the multicast server broadcast nonce messages to gather receivers’ group attributes (\( GID \) and mobility information) as shown in the first step of both sender and receiver ends in Figure 2. Then each receiver replies with its group attributes using TESLA secure mechanism in \( R_P(i) \), as shown in the second step of receiver end in Figure 2. In this phase, the data packet is shown in Figure 3. The MAC is calculated by \( K(i) \), the receiver’s initial \( GID \), and the receiver’s initial group attributes.

![Group Authentication Packets](image)

**Figure 3. Group Authentication Packets**

Group attributes are never transmitted between the server and receivers and only used to calculate the MAC value. Therefore, even if an adversary eavesdrops \( R_P \), it cannot get group certificate to fake as a group member. After waiting for a key disclosure delay, the receiver release \( K(l) \). Then the server uses released \( K(l) \) and its own group attributes to calculate the MAC value, as shown in the third step of both sender and receiver end in Figure 2. If the computed MAC equals to the MAC disclosed in \( R_P(i) \), the server considered the receiver is its group member; otherwise, the receiver is outside its group.

#### 3.3.1.3 Bootstrapping Parameter Transmission

After group authentication, the server sends bootstrapping packets to receivers as shown in the fourth step of sender end in Figure 2. Bootstrapping packets (\( B_P \)) is in the following format:

\[ B_P = (i, \ T_{int}, \ T_{begin}, \ F, \ F', \ d, \ K(i-d)) \]

where \( i \) denotes current time interval \( i \), \( T_{int} \) is the packet encryption interval duration that defines how long the packets will use same encryption key, \( T_{begin} \) describes the beginning of time interval \( i \) and \( d \) presents predefined key disclosure interval. Therefore, when a data packet coming, the receiver can roughly predict its key disclosure time and check the secure condition. \( F \) is a pseudo random function to calculate the keychain, \( F' \) is used to commit key generated in the keychain, and \( K(i-d) \) is a disclosed key committed the packet sending in time interval \( i-d \). Using above parameters, the receiver can verify released key and received data packet.

#### 3.3.2 Intra-group Communication

GSA enables group members verify received packet immediately and efficiently. Figure 4 shows the process of intra-group communications and we will describe it in the following.

![Intra-group Source Authentication Process](image)

**Figure 4. Intra-group Source Authentication Process**

### 3.3.2.1 Sender Operation

As shown in the first step of sender end in Figure 4, in the time interval \( i \), multicast sender sends data packets \( D_P(i) \) to its group members. Data packets \( D_P(i) \) are committed with the same key \( K(i) \). \( M(i) \) is the message the multicast sender want to send.

The sender uses committed \( K(i) \) to compute MAC of data packet and enclosed \( K(i) \) in the same packet. For better security, \( K(i) \) used in MAC computation is committed with a pre-determined hash function \( F' \). And released \( K(i) \) is encrypted with current group attributes (\( GID \) and mobility information), as shown in Figure 4. Therefore, even if an adversary intercepts the data packet \( D_P(i) \), it cannot get \( K(i) \) without inner group attributes. Inner group certificate is long-time valid and can be updated in vehicle check routine. But mobility information is dynamic changing and reflects real-time vehicle mobility information without statistical regularity. Thus, a malicious node cannot fake future messages by cracking old mobility information or predicting future mobility information.

#### 3.3.2.2 Receiver Operation

Once a data packet \( D_P(i) \) arrives, the group member receiver gathers its current mobility information e.g. speed, direction, and decrypts \( K(i) \) immediately using stored group certificate and current mobility information, as shown in the second step of the receiver end in Figure 4. Then it computes MAC with decrypted \( K(i) \) and verifies the equality of computed MAC and received MAC. As shown in the third and fourth steps of the receiver end in Figure 4, if they are equal, the receiver delivers data packet to the application immediately; otherwise, it drops the packet and sends warning message to the application. Because only group members have the knowledge of group certificate and mobility measurement method, non-group receivers cannot grab the release key and fake as the multicast server.

### 3.3.2.3 Group Membership Update

GSA protocol requires group members to perform group membership authentication periodically. During message transmissions, group members can join or leave the group at any time. In the setup phase, the server chooses the length of key chain \( K(N) \). Here, \( N \) is the life-time of the generated keychain. After running out of pre-produced keys, the server generates a new keychain and multicast nonce messages to receivers. When receiving nonce message, a receiver prepares new group authentication bootstrapping parameters and transmits committed group attributes to the server. The group membership update has the same steps as initial group authentication and adopts TESLA-Based scheme to ensure that group information is sent from genuine receivers.
After new round membership update, the sender disseminates encrypted GSA Intra-Group bootstrapping parameters to enable group members verify its multicast packets. For security concern, the length of key chain \( N \) for intra-group communication is often shorter than the one for inter-group communication.

3.3.3 Inter-group Communication

For Inter-Group communication, GSA adopts TESLA based security mechanism.

3.3.3.1 Sender Operation

After group membership authentication, the sender starts sending data packets to non-group receivers. In time interval \( i \), the sender multicasts data packets \( D_P(i) \) in the following format:

\[
D_P(i) = (M(i), K(i - d), MAC(F'(K(i)), M(i), K(i - d))
\]

\( D_P(i) \) discloses key \( K(i - d) \) which is a key used to compute MAC of packets sent in time interval \( i - d \). For secure concerning, the key to commit MAC is signed by a hash function \( F' \) which is known to receivers in communication initialization step. \( M(i) \) is unauthenticated message.

3.3.3.2 Receiver Operation

When a data packet \( D_P(i) \) arriving, the receiver first checks if the sender has not send out the key \( K(i) \). This is a necessary secure condition for Inter-Group TESLA based communication and it can be expressed in the following function:

\[
Arrivetime(D_P(i))+Key_Disclose_Delay(d*T_{lad}) < Sendtime(D_P(i+d))
\]

Here data packet \( D_P(i+d) \) will release \( K(i) \) whose sending time can be simply estimated by the starting time \( T_{begin} \) (the starting time of current time interval \( i \) ) and key disclosure time interval \( d \). The security condition requires key disclosure waiting time \( d \) to be longer than the network delay from the source to all the recipients.

If the security condition is not satisfied, the receiver drops the unsafe packet. Otherwise, the receiver stores unauthenticated message \( M(i) \) and its MAC value into a buffer. Then, it extracts disclosed \( K(i - d) \) and use it to compute an old authenticated \( K(j) (j < i - d) \) to verify \( K(i - d) \)'s reliability, just as TESLA did [1]. If the computed \( K(j) \) is equal to the previous authenticated \( K(j) \), the \( K(i-d) \) is authenticated. Then the receiver calculates MAC value of packet \( D_P(i - d) \) and compares with corresponding MAC in the buffer. If equal, \( M(i-d) \) is secure to delivery to the application; otherwise, \( M(i-d) \) will be dropped. In this way, every received message should be stored in buffer and wait for \( d \) time interval to be authenticated.

IV. IMPLEMENTATION AND EVALUATION

We implemented GSA in real network environment in order to accurately verify its function, evaluate the performance effect of different parameters and estimate computation and memory overhead.

We built an ad-hoc network with 7 laptops running Ubuntu Linux 9.10 and choose Optimized Link State Routing (OLSR) as the network routing protocol because they are open source and OLSR can support multicast communication using its plug-in. We installed OLSRD 0.5.6-r6 for unicast communication experiments and added basic multicast forwarding plug-in to do experiments for multicast transmissions. We implemented GSA in C with all of its functionality sitting in the application layer. We use UDP datagram to disseminate data packets. And TCP is deployed for reliable transmissions in group authentication and bootstrapping parameter distributing phases.

For sender setup, in our experiments, we followed parameter settings in traditional TESLA paper [1]. Here, time interval \( T_{lad} \) is 1s and the key disclosure delay is 2s. We use Hash-based Message Authentication Code (MAC) to construct MAC of data packets. The cryptographic hash function we use is MD5 with a 128 bit output [13]. We only choose the 80 most significant bits of the MD5 output in order to save space without losing security.

4.1 Average End-to-End Delay Evaluation

For Inter-Group communication, GSA scheme requires the minimum key disclosure delay to be larger than the message traveling time from the source to all the recipients. Therefore, we analysis the average network end-to-end delay in both intra-group and inter-group communications under variable network traffic density and message transmission hops in order to make optional choice on key disclosure delay.

From Figure 5 we observed that with transmission hops and GSA sessions increasing, network end-to-end delay becomes higher because of the larger queuing delay, transmission delay, propagation delay, and processing delay. For the comparatively small number of GSA sessions, network end-to-end delay increases slowly and smoothly with the increase of multicast hops. When the number of GSA sessions becomes larger, the end-to-end delay is increased sharply with multicast hops more than 3. We can choose the proper key disclosure delay depending on different number of GSA application sessions and the maximum transmission hops.

![Figure 5. Average End-to-End Delay vs. Number of Hops](image)

4.2 CPU and Memory Usage Evaluation

We also evaluate the computing consumption of CPU and memory of GSA protocol under different number of sessions. The CPU on our laptop is Mobile Intel Pentium M740 with 1.73GHz. And the size of laptop memory is 512MB. In Figure 6, we observed that for the session number is 5, GSA incurs CPU consumption of 2.56% and memory consumption of 0.5% and the maximum GSA sessions our server can support is 500. Figure 6 provides the reference point of how to choose supporting GSA session numbers based on the requirements of application.
4.3 Message Authentication Code (MAC) Computing Time of Multicast Server

We analyze the average MAC computing time in both intra-group and inter-group communications. In Figure 7, we observed that besides increasing trend of MAC computing time with more GSA sessions because of the consumption of CPU, there is an inflection point on MAC computing time curve. Here, the MAC average computing time increase smoothly when the GSA session number is below 300. When the number of sessions exceeds 300, MAC average computing time increased sharply. Figure 7 also provides the reference point of how to choose supporting GSA session numbers based on the requirements of application.

4.4 Performance Comparison of GSA and TESLA

We can download the TESLA’s source code from its project website and implement TESLA in the same scenario. In Table 1, we compare the time consumption of transmitting one packet using our proposed GSA scheme for intra-group communication and original TESLA scheme. The total time consumption consists sender’s MAC computation time, network end-to-end delay, and receiver’s verification delay. In our experiments, we use 2 seconds as the pre-defined TESLA session key disclosure delay. The difference of two schemes is the receivers’ data verification time. For TESLA protocol, the average verification time takes over 2 seconds, which takes up a large portion of the total time used for one packet transmission. While in GSA scheme, the average verification time for intra-group communication is significantly reduced to less than one millisecond. Because GSA adopt TESLA-based security mechanism for inter-group communication, the verification time will not be reduced significantly than other TESLA-based mechanism as mentioned in section 2. But for intra-group communication, the security processing time added by GSA only takes a little portion of total packet transmission time and is suitable for vehicle applications.

Table 1. Comparison of GSA and TESLA

<table>
<thead>
<tr>
<th></th>
<th>MAC Compute Time</th>
<th>End-to-End Delay</th>
<th>Verification Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESLA</td>
<td>11ms</td>
<td>186ms</td>
<td>2.3s</td>
</tr>
<tr>
<td>GSA</td>
<td>10ms</td>
<td>195ms</td>
<td>0.44ms</td>
</tr>
</tbody>
</table>

V. Conclusion

In this paper, we proposed a group based source authentication scheme (GSA) to achieve efficient and secure communication in VANETs. GSA has high efficiency in the applications where nodes have natural group attribute and share similar moving patterns. GSA also takes advantage of group moving pattern to authenticate information transmitted among group members, thus greatly reduces the verification waiting time generated by TESLA and keeps low packet overhead and little computation latency as well. Also, we built the Linux testbed to demonstrate GSA’s practicality to the real-world applications. The experiments results show that the GSA protocol reduce the receivers’ verification delay and is suitable for real vehicle applications.

REFERENCES